Faculty of Business, Economics and Information Systems
University of Passau

BACHELOR THESIS

Philipp Stockerl

Robust and Adaptive Path Planning for Autonomous Vehicles in Spatio-Temporal Cost Fields

Chair of

Business Decisions & Data Science
Business and Economics Focus Management Science
Operations and Supply Chain Management

Supervisor of the Thesis:	Prof. Dr. Marc Goerigk
Double Bachelor Programmes:	Business Administration and Economics, Information Systems

Passau, 23.01.2025

INFORMATION

Title:	Robust and Adaptive Path Planning for Autonomous Vehicles in Spatio-Temporal Cost Fields
Author:	Philipp Stockerl
Institute:	Chair of Business Decisions & Data Science, Business and Economics Focus Management Science/ Operations and Supply Chain Management
Supervisor:	Prof. Dr. Marc Goerigk

Keywords:	Optimization, Management Science Geostatistics, Modelling, Optimization under Uncertainty, Robotics, Path Planning

ABSTRACT

This thesis investigates minimum-cost path planning in spatio-temporal environments under uncertain edge costs. Our testbed is mobile-robot routing on directed graphs, with edge costs generated from synthetic spatio-temporal random fields via geostatistical methods. We compare two complementary approaches: (i) robust combinatorial optimization models for shortest paths, and (ii) an incremental shortest path heuristic that adapts after costs change. We address uncertainty both ex-ante (before executing a path) and ex-post (after cost shifts). We evaluate three options drawn from these two approaches and use the results to answer the research objectives and questions. Our research shows that...

TABLE OF CONTENTS

Notation

Symbol	Description
	Graphs and paths
$G = (V, E)$	Directed graph; (V, E) notation used in robust models.
$G = (S, E)$	Directed graph; (S, E) notation used in D* Lite.
V, S	Vertex sets (robust optimization vs. D* Lite grid cells).
E	Set of directed edges (arcs).
i, j	Node indices for edge endpoints $(i, j) \in E$.
s, t	Start and target nodes in robust models.
s_{start}, s_{goal}	Current start and goal vertices in D* Lite.
$\delta^+(\cdot)$, $\delta^-(\cdot)$	Successor and predecessor edge sets.
$\mathcal{X}$	Solution set (of feasible s–t paths).
x_e	Binary decision variable indicating whether edge e is used.
$L_{\min}$	Minimum number of edges on an s–t path (4-connected grid).
	Scenarios and costs
Ω, ω	Scenario index set and index.
$\mathcal{T}$, τ	Time index set and time index for STRF slices.
$\lvert\Omega\rvert$, $\lvert\mathcal{T}\rvert$	Number of scenarios and time steps.
c_e^ω	Cost of edge e in scenario ω.
$c(i, j)$	Traversal cost of edge (i, j) in the current scenario (D* Lite).
$\mathbf{c}^\omega$	Scenario cost vector.
$f_\tau(p, q)$	Scenario cost surface at time τ.
$Z(p, q, \tau)$	Spatio-temporal random field value.
	Discrete robust model
z	Worst-case path cost (discrete min–max model).
	Budgeted uncertainty model
$\underline{c}_e$	Nominal (average) edge cost.
d_e	Maximum deviation from nominal cost.
Γ	Budget of uncertainty.
π	Dual variable for the budget constraint.
ρ_e	Auxiliary deviation variable for edge e.
λ	Normalized budget factor, $\Gamma(\lambda) = \lambda\Gamma_{\max}$.
$\Gamma_{\max}$	Maximum budget, typically set to $L_{\min}$.

Continued on next page

Symbol	Description
	D* Lite
$g(s)$	Current shortest-path cost estimate from s to s_{goal}.
$\text{rhs}(s)$	One-step lookahead cost-to-go from s.
$h(i, j)$	Admissible heuristic estimate between vertices; also used as a metric distance in the geostatistical model (context by arguments).
$k(s) = (k_1(s), k_2(s))$	Lexicographic priority key for vertex s.
k_m	Heuristic shift variable due to start-vertex movement.
U	Priority queue of locally inconsistent vertices.
u	Vertex selected from U for expansion.
ComputeShortestPath()	Procedure that restores local consistency.
UpdateVertex(s)	Procedure that updates $\text{rhs}(s)$ and queue membership.
	Geostatistical/SRF model
$\mathbf{x} = (p, q, \tau)$	Coordinate vector in the 3D space–time domain.
p, q	Spatial coordinates along the x- and y-axes.
N_p, N_q	Grid dimensions in the spatial axes.
Δp, Δq, $\Delta\tau$	Coordinate increments in space and time.
ℓ_p, ℓ_q, ℓ_t	Spatial and temporal correlation lengths.
a_p, a_q, a_t	Anisotropy ratios.
σ^2	Field variance.
α	Stable-kernel exponent; also spatial scaling factor in experiments (context disambiguates).
β	Temporal scaling factor.
ν	Matérn smoothness parameter.
$\mathbf{k}_i$	Wave vectors in the spectral construction.
$Z_{1,i}, Z_{2,i}$	Standard normal coefficients in the spectral method.
$\gamma(r)$, $C(\cdot)$, $\rho(\cdot)$	Semi-variogram, covariance, and correlation functions.

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

Problem Setting Autonomous vehicle navigation admits a wide range of planning formulations, ranging from global route planning to local reactive control and hybrid approaches that combine multiple decision layers [KNK24, AK23]. Depending on the application, planners optimize objectives such as travel time, energy consumption, risk exposure, or mission goals under kinematic, safety, and environmental constraints [ASSM+22, GH23, ZM18]. Despite this diversity, many planning algorithms share a common structural core: they evaluate feasible paths on graph-based representations under additive cost models.

From a methodological perspective, these formulations can be viewed as instances of combinatorial optimization problems (COP) studied in operations research (OR). The shortest-path problem (SPP) provides the most elementary abstraction of this core and underlies a wide range of planning algorithms, including heuristic search, dynamic programming, and network flow formulations [WWW+25, GH24, QSW+23]. Even when planning problems extend beyond single-pair routing—for example, through timing constraints, risk measures, or vehicle dynamics—their algorithmic structure often still relies on modified shortest-path computations.

In real-world navigation tasks, however, the nominal assumptions of the classical SPP no longer hold. Traversal costs vary across space and time and are only partially observable at planning time. In unmanned aerial vehicle (UAV) applications, for instance, costs depend on environmental conditions such as wind fields, which directly affect travel time and operational risk along flight segments [RGRM24]. Forecasts and sensor data provide partial information, but future realizations remain uncertain and deviations from nominal predictions are unavoidable.

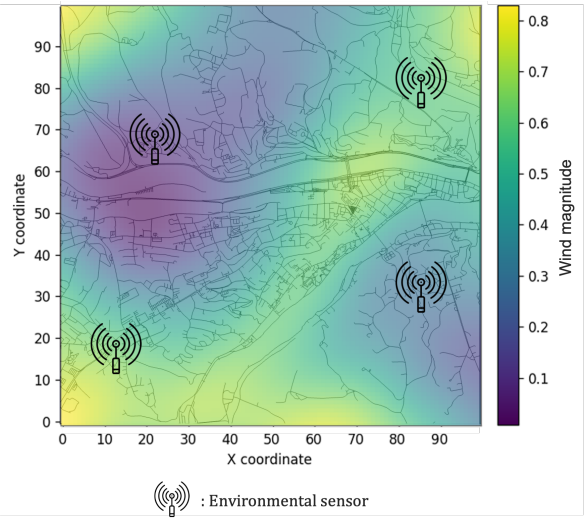
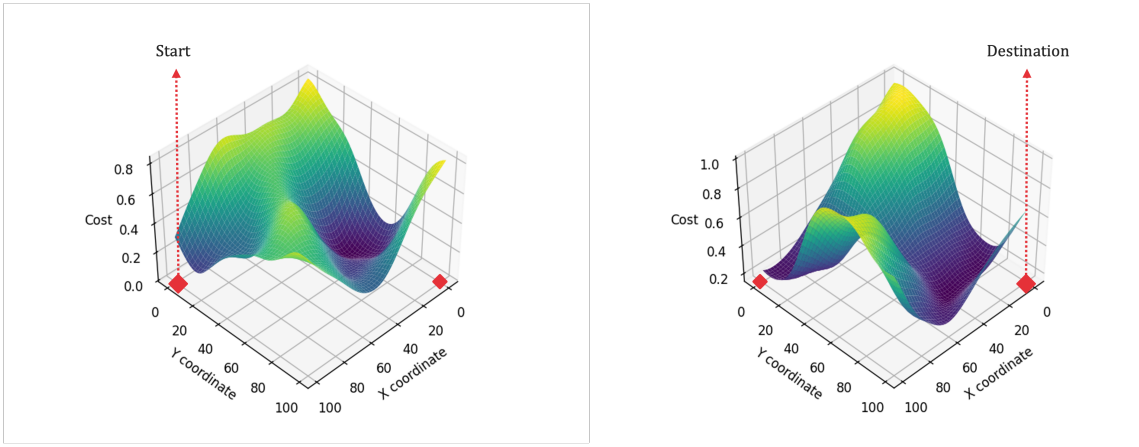


Figure 1.1: *Spatial environmental data with sensor locations and inferred wind intensity over a two-dimensional domain.*

Figure 1.1 shows how heterogeneous environmental information can be aggregated into spatial cost fields that serve as inputs to routing decisions. As costs evolve over time, the relative attractiveness of feasible paths changes rather than being blocked entirely [AS24], such that planning decisions based solely on nominal information may perform poorly or even fail. Figure 1.2 illustrates this effect by showing spatial cost realizations at two different time points over an identical graph structure. This motivates extensions of the classical shortest-path problem that explicitly account for uncertainty and temporal dynamics in traversal costs.



(a) *Time t* **(b)** *Later time $t + n$*

Figure 1.2: *Spatial cost realizations at two different time points over a fixed graph topology.*

Two dominant paradigms address shortest-path planning under such uncertainty by extending the classical model in fundamentally different ways:

Robust optimization (RO) addresses uncertainty *ex ante* by selecting complete paths prior to execution and protecting them against adverse cost realizations within explicitly defined uncertainty sets. Robust combinatorial shortest-path formulations optimize the worst-case path cost over such sets and increasingly derive uncertainty sets from data rather than expert choice [CDG19, GK25]. This approach adopts a global, strategic perspective on uncertainty and is particularly suited to planning problems in which route decisions must be fixed before execution and robust performance guarantees are required. Prominent application areas include vehicle routing problems (VRPs), where robustness is used to model application-specific uncertainty [RGRM24, ZZL23, YAB23]. Further extensions incorporate time-dependent decision making at multiple planning levels [GL21, ZLS+22] and address scalability challenges arising from large data sets [WWW+25]. Evaluation in this literature typically relies on scenario-based or bounded-deviation constructions to compare robustness across alternative formulations [CDG19]. These models provide the ex-ante reference solutions used for comparison in this work.

Incremental search-based planning addresses uncertainty *ex post* by revealing cost information during execution and repairing paths locally as changes occur. These

methods build on classical shortest-path and heuristic search principles, most notably Dijkstra's algorithm [Dij59] and A* [HNR68], and extend them to dynamic environments through algorithms such as D* [Ste94], LPA* [KLF04], and D* Lite [KL02]. By reusing previous search effort, incremental planners avoid full recomputation and enable efficient replanning when edge costs change [BDG+24]. Recent research focuses on refining these foundations to improve replanning efficiency, path quality, and robustness under frequent cost updates, for example by incorporating safety margins, kinematic constraints, or adaptive heuristics [WQL+21, WLJ+22, HHW24, LLZ+24, XXG+25]. As a result, incremental search produces globally consistent paths while emphasizing execution-time adaptivity in dynamic and partially known environments [KSDS21]. This paradigm is therefore particularly well suited to mobile robotic systems, where tight integration with onboard sensing and frequent environment updates is required.

In essence, robust optimization emphasizes strategic, global path planning by selecting a complete route at planning time and protecting it against worst-case cost realizations, often assuming centralized computation and limited adaptivity during execution. Incremental search-based methods likewise aim to compute globally consistent paths, but rely on locally grounded exploration and adaptation, trading increased online computation for the ability to react to newly revealed cost information. Although both paradigms address the same underlying routing task, they differ fundamentally in their information assumptions, degree of adaptivity, and evaluation criteria.

Research Objective The objective of this thesis is to develop and apply a controlled experimental framework that enables systematic comparisons between ex-ante robust planning and ex-post adaptive search under identical spatio-temporal uncertainty realizations. In addition, the thesis explores a hybrid planning approach in which robust optimization provides global, ex-ante guidance, while incremental search contributes local, ex-post adaptation during execution. Using this framework, we conduct a systematic experimental comparison of selected robust shortest-path formulations and an incremental replanning method under spatio-temporal cost uncertainty. Performance is evaluated with respect to solution quality, computational effort, and execution-time adaptivity in order to assess the practical implications of each paradigm for autonomous vehicle guidance.

Motivation A key motivation for this objective is that, despite extensive research in robust optimization and incremental search, there is little controlled, head-to-head comparison of these approaches under shared experimental conditions. Robust shortest-path formulations and incremental search methods are typically studied in isolation and rarely evaluated on identical graph structures, common spatio-temporal cost realizations, and comparable performance metrics. This lack of a unified experimental setting limits our understanding of their relative strengths, weaknesses, and practical

trade-offs for autonomous vehicle guidance on the same underlying cost fields.

Related work exists in global–local planning frameworks that combine long-horizon planning with local reactive control [GHF+24, LD25]. These methods typically decompose planning into a hierarchical structure, where a global planner computes a reference path and a local controller handles short-term disturbances and obstacle avoidance. While effective in practice, such approaches primarily focus on architectural integration and performance improvements within a single paradigm. Consequently, they do not provide a controlled comparison between ex-ante robustness and ex-post adaptivity under a common problem formulation, uncertainty model, and evaluation protocol.

Contribution The contributions of this thesis are fourfold: (i) the design of a reproducible experimental framework that enforces shared data, graph structures, and evaluation metrics across planning algorithms; (ii) a spatio-temporal cost generation process based on geostatistical random fields that are mapped consistently onto routing graphs; (iii) implementations of robust and incremental planning methods, including a hybrid guidance variant that combines ex-ante protection with ex-post adaptation; and (iv) a unified evaluation protocol for comparing solution cost, computational effort, and execution-time adaptivity. The experimental framework is implemented as a conceptual decision support system (DSS) that integrates spatio-temporal cost generation, robust combinatorial optimization models, and incremental search-based planning methods into a unified pipeline (Figure 1.3).

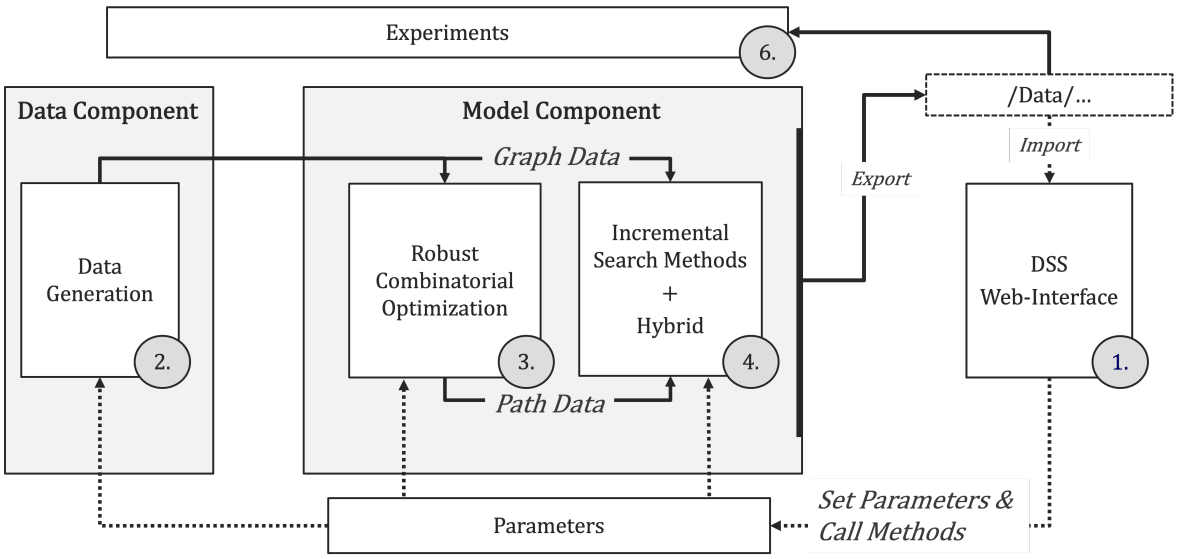


Figure 1.3: *Top-down architecture of the conceptual decision support system used for experimental comparison, integrating spatio-temporal cost generation, robust optimization, incremental search-based planning, and a hybrid guidance variant within a unified pipeline.*

CHAPTER 2

CONCEPTUAL FOUNDATIONS

2-1 Model and Data Driven Decision Support Systems

Decision Support Systems (DSS) are interactive, computer-based systems that support decision-making in complex and uncertain environments by integrating data, analytical models in user interfaces [FB22]. Rather than automating decisions, DSS structure decision-making by making assumptions explicit and enabling systematic comparison of alternatives.

At a functional level, DSS link data with analytical models. Input data are processed and transformed before being evaluated by models that represent the underlying decision problem. The resulting outputs—such as costs, feasibility indicators, or performance measures—are interpreted through visualization and interaction, forming the basis for model-driven decision support [Pow02].

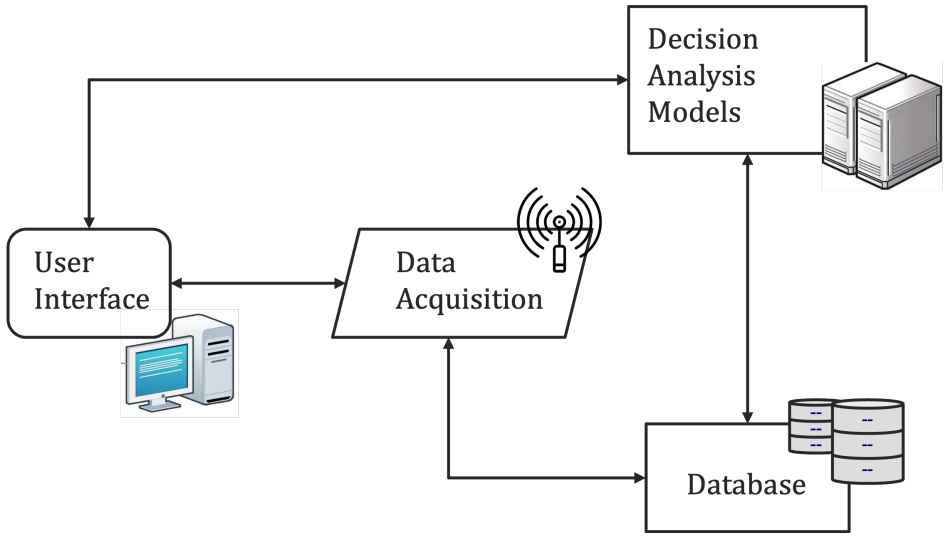


Figure 2.1: *Basic architecture of a model-driven decision support system*

DSS for Vehicle Routing Vehicle routing constitutes a prominent application area for decision support systems [RMA04, LRS21]. Routing decisions require selecting efficient paths while accounting for operational constraints, uncertain travel times, and

dynamic environmental conditions. As a result, vehicle routing DSS naturally integrate multiple analytical components.

A central role is played by model-driven DSS [PS07], which rely on formal models to represent routing decisions and evaluate alternatives. Such systems may incorporate deterministic optimization models, robust optimization models [GRT+16], heuristic algorithms, or simulation-based approaches [FIU+15]. Decision variables typically correspond to route choices or planning parameters, while performance measures include total cost, robustness, or computational effort. Environmental conditions act as exogenous inputs that influence outcomes but cannot be directly controlled.

Vehicle routing DSS often exhibit a strong data-driven component. When routing decisions depend on geographic or spatial information, this leads naturally to spatial DSS [Kee98], which integrate geographic representations such as maps or spatial cost fields. These representations enable decision makers to interpret routing decisions in their geographic context and to relate analytical results to real-world environments [UHV17]. In this thesis, spatial DSS concepts are combined with model-driven DSS to address routing problems under spatio-temporal uncertainty.

Model and Data Utilization The effectiveness of a DSS depends critically on the models and data it employs. Data quality, aggregation, and normalization directly influence the reliability of decision-relevant information, while the choice of analytical models determines which aspects of the decision problem can be analyzed and optimized. We employ a conceptual DSS for our autonomous vehicle routing problem as an interactive control surface that embeds a data component and model components.

2-2 Data Component: Geostatistical Data Generation

For the data component of our conceptual framework, we generate spatio-temporally correlated cost fields using geostatistical methods. These fields serve as controlled, reproducible inputs to all shortest-path models evaluated in this work. Since geostatistics is used purely as a data generation tool, we focus on implementation-relevant design choices rather than a detailed statistical treatment.

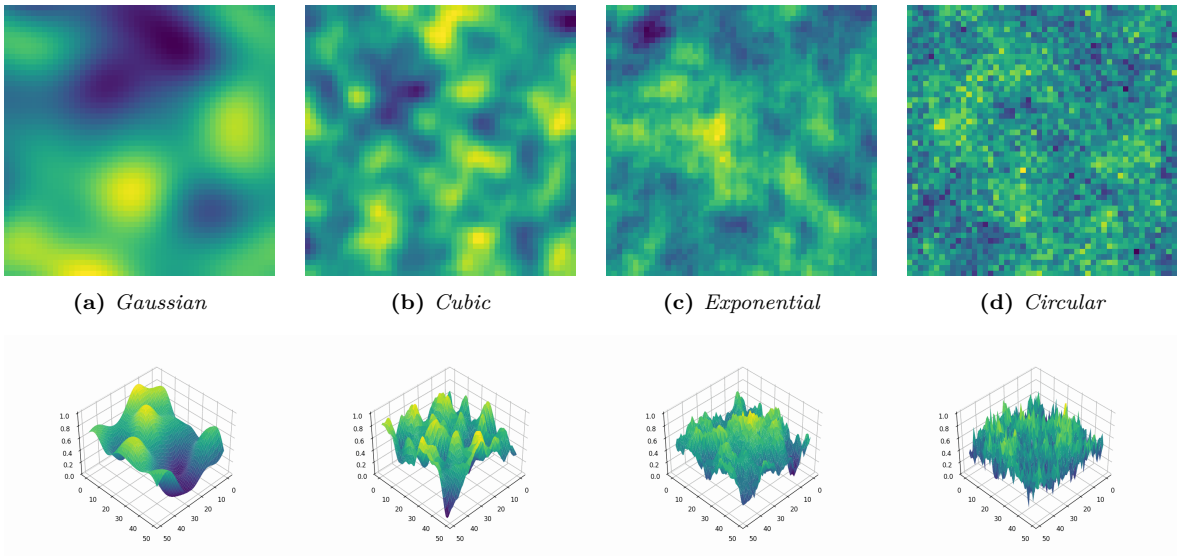
Using simulated data for model comparison is common practice when real-world data is scarce or difficult to control experimentally [MPW26, MSY+25]. Geostatistical frameworks provide a principled way to generate spatially and temporally correlated synthetic data while maintaining full control over variance, correlation length, and anisotropy [MSZH22]. In particular, we rely on the GSTools library, which offers reproducible implementations of second-order random field models widely used in

simulation studies [Rai19].

We model cost fields as realizations of a weakly stationary Gaussian random field defined on a three-dimensional space–time domain. Spatial and temporal dependence are introduced via a parametric covariance function, whose structure is fully determined by a normalized correlation kernel and a metric distance. Time is treated as an additional geometric dimension, resulting in temporally consistent spatial slices rather than independent scenarios.

GSTools provides a family of admissible correlation kernels with varying smoothness and compact support properties (see Table 6.1). While kernel choice influences local regularity, all kernels share a common structure in which dependence is governed by a single scalar distance. Directional anisotropy and spatio-temporal coupling are incorporated by defining this distance through a weighted metric.

Figure 2.2 illustrates the qualitative effect of different kernel choices under identical parameters. These visualizations demonstrate that kernel selection alone can induce substantially different spatial roughness, despite identical variance and correlation length. We further discuss kernel selection in Chapter 4.



(a) *Gaussian* **(b)** *Cubic* **(c)** *Exponential* **(d)** *Circular*

(e) *Gaussian 3D* **(f)** *Cubic 3D* **(g)** *Exponential 3D* **(h)** *Circular 3D*

Figure 2.2: *Difference of covariance models for same parameters visualized*

The resulting spatio-temporal random fields constitute the data-generating basis for all experiments. Each temporal slice represents a spatial cost realization, and temporal correlation ensures consistency across scenarios. By discretizing spatial slices into directed graphs, the continuous geostatistical model is transformed into time-indexed shortest-path instances, enabling a controlled comparison of robust and adaptive planning methods under identical uncertainty assumptions

2-3 Model Component: Shortest-Path Planning Under Uncertainty

This section establishes a common problem formulation for shortest-path planning that serves as a conceptual foundation for the two planning paradigms studied in this thesis. It serves as a backbone for the model component of our conceptual DSS. We begin with the classical shortest-path problem under nominal assumptions and then introduce cost uncertainty, highlighting how violations of these assumptions lead to fundamentally different planning logics.

Although both paradigms introduced in the introduction address the same underlying routing problem on an identical graph structure, they differ in how this problem is represented and solved. Following the taxonomy of Hart et al. [HNR68], we distinguish between optimization-based formulations and heuristic search–based methods as two complementary perspectives on shortest-path planning. This distinction provides the organizing lens for the remainder of this chapter.

Shortest-path problem formulation We consider a directed graph $G = (V, A)$, where V denotes the set of vertices and $A \subseteq V \times V$ the set of directed arcs. Each arc $e = (i, j) \in A$ is associated with a nonnegative traversal cost c_e. A path P from a start vertex $s \in V$ to a destination vertex $t \in V$ is defined as an ordered sequence of arcs connecting s to t. The single-pair shortest-path problem (SPSP) consists of finding a path $P^\star$ that minimizes the total path cost

$$C(P) = \sum_{e \in P} c_e.$$

Such graph structures naturally represent spatial or spatio-temporal cost fields derived from simulated or real-world data (Figures 1.1 and 1.2). The shortest-path problem is inherently combinatorial, as it requires selecting a feasible sequence of discrete arcs from a finite set.

Nominal shortest-path problem In the classical (nominal) setting, traversal costs are assumed to be nonnegative, time-invariant, and fully known at planning time. Under these assumptions, both optimization-based and heuristic search–based approaches solve the same underlying combinatorial problem and are equivalent with respect to optimality, differing only in representation and solution strategy.

From an *optimization perspective*, the shortest-path problem is treated as a combinatorial optimization problem in which arcs constitute the ground set of decision elements. Following the general framework of combinatorial optimization [GH24], let

$S = A$ denote the finite ground set of directed arcs. Each arc $e \in A$ is associated with a nonnegative cost c_e, and a feasible solution corresponds to selecting a subset of arcs that forms an s–t path. Using binary decision variables $x_e \in \{0, 1\}$ to signal wether a edge is selected or not, the nominal shortest-path problem can be formulated as

Nominal Combinatorial Shortest-Path Problem

$$\min \quad \sum_{e \in A} c_e x_e \tag{2.1}$$

$$\text{s.t.} \quad \sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ji} = b_i \qquad \forall i \in V, \tag{2.2}$$

$$x_e \in \{0, 1\} \qquad \forall e \in A, \tag{2.3}$$

where $b_s = 1$, $b_t = -1$, and $b_i = 0$ for all $i \in V \setminus \{s, t\}$. The flow-balance constraints (2.2) define the feasible set $\mathcal{X} \subseteq \{0, 1\}^{|A|}$ and encode the combinatorial structure of valid s–t paths. Under nonnegative costs, any optimal solution corresponds to a simple path.

Heuristic search–based methods provide an algorithmic counterpart to this formulation. They interpret the graph as a discrete state space in which vertices represent states and arcs represent transitions. Paths are constructed implicitly through successive state expansions guided by cost estimates and heuristic information, rather than by explicitly selecting edges. The nominal shortest-path problem can equivalently be expressed as a discrete dynamic programming problem. Let $V^\star(v)$ denote the optimal cost-to-go from node v to the destination t. Under deterministic, time-invariant, and fully known arc costs, $V^\star$ satisfies Bellman's optimality equation

Bellman's Optimality Equation

$$V^\star(t) = 0, \qquad V^\star(v) = \min_{(v,u) \in A} \{c_{vu} + V^\star(u)\} \quad \forall v \in V \setminus \{t\}. \tag{2.4}$$

This recursion formalizes optimal substructure: every subpath of an optimal path is itself optimal. Heuristic search algorithms such as A* exploit this structure by constructing the optimal path implicitly. A* maintains cost-to-come estimates $g(v)$ and expands nodes in increasing order of

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

where $h(v)$ is an admissible and consistent heuristic estimate of the remaining cost from v to t. With an admissible, consistent heuristic, A* returns an optimal s–t

path. Thus, under nominal assumptions, optimization-based formulations and heuristic search methods solve the same combinatorial shortest-path problem and are equivalent with respect to optimality. A* serves as the nominal baseline; incremental methods such as LPA* and D* Lite reuse its heuristic structure and repair only affected nodes when edge costs change, rather than replanning from scratch (Table 2.1).

Conceptual Dimension	**Nominal Shortest Path**	**Ex-Ante Robust Optimization**	**Ex-Post Incremental Search**
Cost information	Deterministic, time-invariant, fully known	Uncertain but bounded via uncertainty sets	Revealed or updated during execution
Optimality principle	Bellman optimality holds globally	Bellman optimality generally fails under min–max objectives	Local consistency enforced incrementally
Representation	Explicit or implicit (equivalent)	Explicit edge-based decision variables	Implicit state-space exploration
Decision timing	Single planning step	Ex-ante path selection	Continuous replanning
Response to cost changes	Not applicable	No adaptation unless re-solved	Local repair
Primary objective	Minimum total cost	Worst-case protected cost	Adaptivity and replanning efficiency

Table 2.1: *Conceptual distinctions between nominal shortest-path planning, ex-ante robust optimization, and ex-post incremental search under cost uncertainty.*

Shortest-path planning under uncertainty In spatio-temporal routing environments, the nominal assumptions of time-invariant costs and full information no longer hold. We model uncertainty through time-varying traversal costs on a fixed graph. Let $k \in \{1, \dots, T\}$ index discrete planning or execution steps, and let c_e^k denote the cost of arc e at step k. The cost vector $c^k = (c_e^k)_{e \in A}$ represents a realization of a spatio-temporally correlated cost field. Conditional on a realization, costs remain deterministic and nonnegative, but future realizations are not known at planning time. At the time a path is planned, only partial information about future cost realizations is available, for example in the form of forecasts, bounds, or scenario samples. Additional cost information may become available during execution. Figure 2.3 illustrates how this uncertainty alters the decision context relative to the nominal setting. Uncertainty fundamentally affects the information structure and objective of the planning problem.

See [MAR+17, DP84] for additional SPP taxonomy. While each realized instance of the shortest-path problem satisfies Bellman's optimality principle individually, robust

objectives may couple decisions across time or scenarios. As a result, global optimal substructure need not hold with respect to the robust planning objective, even though the underlying graph structure is unchanged.

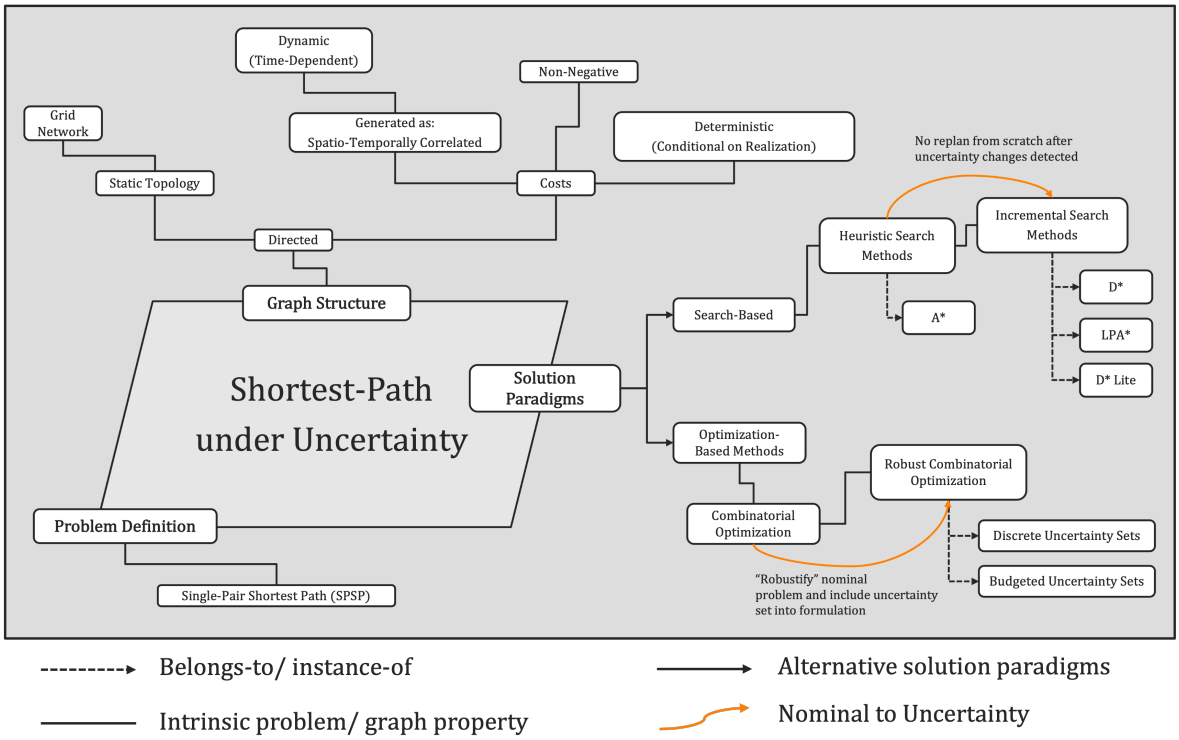


Figure 2.3: *Conceptual overview of the shortest-path problem under uncertainty. The underlying graph structure remains fixed, while traversal costs evolve over time. Robust combinatorial optimization and incremental search-based planning represent two distinct responses to uncertainty within the same problem setting.*

Two coherent planning paradigms emerge from this setting. Ex-ante robust optimization selects paths before execution and protects them against adverse cost realizations within prescribed uncertainty sets. Ex-post incremental search-based planning, in contrast, interleaves planning and execution by repairing paths locally as new cost information becomes available.

The following sections formalize these two paradigms in detail. Section 2-4 introduces robust combinatorial shortest-path models under ex-ante uncertainty, while Section 2-5 presents incremental search-based planning as an ex-post adaptive alternative.

2-4 Robust Combinatorial Optimization

Building on the nominal formulation for the shortest-path problem (Equation 2.3), we introduce uncertainty modeling and "robustify" our nominal problem. We refer generally to [GH24] for formulations in this section.

Robust Optimization and the Min–Max Principle In robust shortest-path planning, costs are assumed to lie in a prescribed uncertainty set $\mathcal{U} \subseteq \mathbb{R}^{|E|}_{\geq 0}$, and the planner selects a feasible path before the adversary selects a realization from $\mathcal{U}$ that maximizes its cost.

Formally, robust planning is based on the min–max criterion

$$\min_{x \in \mathcal{X}} \max_{\mathbf{c} \in \mathcal{U}} \sum_{(i,j) \in E} c_{ij} x_{ij},$$

where $\mathcal{X}$ denotes the set of feasible s–t paths defined by the flow-balance constraints (2.2). This criterion guarantees worst-case protection against all cost realizations contained in $\mathcal{U}$. Alternative robustness concepts exist; in this thesis we adopt the min–max formulation as a simple, widely used baseline for worst–case analysis.

In the following paragraphs, we introduce the uncertainty models used in this thesis. Their integration into fully specified shortest-path formulations is deferred to the Implementation chapter, where implementation details are discussed as well.

Uncertainty Sets The uncertainty set $\mathcal{U}$ specifies which realizations of edge costs are considered plausible and directly determines the level of robustness and conservatism of the resulting solution. In the context of shortest-path planning, each component of a cost vector corresponds to a single directed edge of the underlying graph.

In this thesis, uncertainty sets are constructed from the synthetic spatio-temporal random fields used throughout; each temporal slice yields a spatial cost scenario (2-2). Based on this representation, two complementary uncertainty models are derived and evaluated: discrete scenario-based uncertainty and continuous budgeted uncertainty.

Discrete Uncertainty Sets A discrete uncertainty set consists of a finite collection of cost scenarios

$$\mathcal{U} = \{\mathbf{c}^\omega \mid \omega \in \Omega\},$$

where each vector $\mathbf{c}^\omega$ represents a complete realization of edge costs over the graph.

As illustrated in Figure 2.4, discrete uncertainty represents uncertainty through a finite set of cost vectors corresponding to complete scenario realizations.

Discrete uncertainty arises directly from the spatio-temporal random field representation: each cost vector $\mathbf{c}^\omega$ is a temporal slice, so all edge costs within a scenario come from the same snapshot. Thus, $|\Omega|$ equals the number of

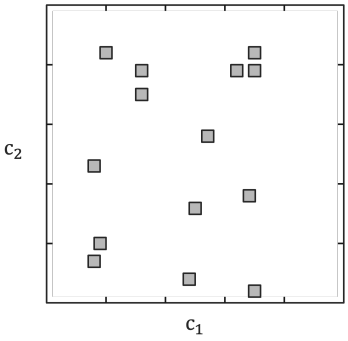


Figure 2.4: *Geometric interpretation of a discrete uncertainty set in a cost space.*

time steps, and robust optimization evaluates candidate paths against all temporal realizations.

For discrete uncertainty sets, the min–max problem can be reformulated using an epigraph formulation:

Epigraph Reformulation for Discrete Uncertainty

$$\begin{aligned} \min \quad & z && && (2.5)\\ \text{s.t.} \quad & z \geq \sum_{i\in[n]} c_i^{\omega} x_i && \forall \omega \in \Omega, && (2.6)\\ & x \in \mathcal{X}. && && (2.7) \end{aligned}$$

This reformulation replaces the inner maximization by a finite set of linear constraints and yields a deterministic mixed-integer optimization problem.

These modeling choices have important computational consequences. From a theoretical perspective, the discrete scenario-based min–max shortest-path problem is NP–hard in general, even when the nominal shortest-path problem is polynomially solvable. Hardness results already arise for small numbers of scenarios, and the problem becomes strongly NP–hard when the number of scenarios is part of the input [YY98, GH24]. Although pseudopolynomial dynamic programming algorithms exist for fixed scenario counts [ABV09], their runtime grows rapidly with the number of scenarios, which limits practical scalability.

The size of the epigraph reformulation grows linearly with the number of scenarios. For a graph with $|E|$ edges, $|V|$ vertices, and $|\Omega|$ scenarios, the resulting mixed-integer formulation contains $O(|E|)$ binary variables, $O(|V|)$ flow constraints, one additional continuous epigraph variable z, and $O(|\Omega|)$ robust constraints. Equivalently, the overall model size scales as $O(|E| + |V| + |\Omega|)$.

Continuous Budgeted Uncertainty Sets Continuous budgeted uncertainty summarizes cost variation across scenarios via nominal costs and bounded deviations.

Given nominal costs $\underline{c}_i$ and maximum deviations d_i, the budgeted uncertainty set restricts the total amount of deviation that may occur simultaneously across edges by means of a global budget parameter Γ.

$$\mathcal{U} = \left\{ \mathbf{c} \;\middle|\; c_i = \underline{c}_i + d_i \delta_i,\ 0 \leq \delta_i \leq 1,\ \sum_{i\in[n]} \delta_i \leq \Gamma \right\}. \tag{2.8}$$

In contrast to the discrete uncertainty set, the budgeted uncertainty set shown in Figure 2.5 defines a continuous polyhedral region around a nominal cost vector, where the total deviation across edges is limited by the budget parameter Γ.

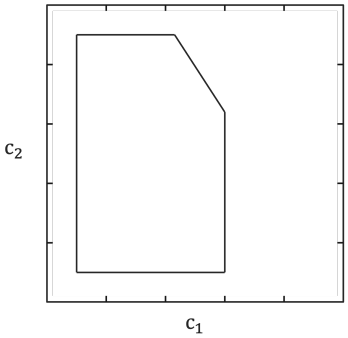


Figure 2.5: *Geometric interpretation of a continuous budgeted uncertainty set in cost space.*

For shortest-path planning, the budget parameter Γ admits an intuitive interpretation. It bounds the number or aggregate magnitude of edges along a path whose costs are allowed to deviate from their nominal values at the same time. Varying Γ interpolates between the nominal solution ($\Gamma = 0$) and full worst-case protection ($\Gamma \geq |E|$), yielding a controlled robustness–conservatism trade-off.

Following standard duality arguments, the inner maximization problem can be dualized, yielding the following robust counterpart:

Robust Min–Max Formulation for Budgeted Uncertainty

$$\min_{x,\,\pi,\,\boldsymbol{\rho}} \quad \sum_{i\in[n]} \underline{c}_i x_i + \Gamma\pi + \sum_{i\in[n]} \rho_i \tag{2.9}$$

$$\text{s.t.} \quad \pi + \rho_i \geq d_i x_i \qquad \forall i \in [n], \tag{2.10}$$

$$x \in \mathcal{X}, \tag{2.11}$$

$$\pi \geq 0, \quad \rho_i \geq 0 \qquad \forall i \in [n]. \tag{2.12}$$

This formulation preserves the combinatorial structure of the shortest-path problem while incorporating controlled worst-case protection against uncertain edge costs.

Bertsimas and Sim show that the inner maximization problem admits a compact dual reformulation that introduces one global variable and one auxiliary variable per uncertain coefficient. The resulting robust counterpart scales linearly in the number of edges and constraints. For shortest-path problems, this reformulation preserves polynomial solvability at the continuous level: whenever the nominal problem is solvable in polynomial time, the corresponding budgeted robust counterpart remains polynomially solvable as a linear optimization problem [BS04, BS03].

Importantly, this complexity guarantee applies to the continuous robust counterpart. Integrality enforcement and solver settings are discussed in Section 3-5.

Summary Table 2.2 summarizes the main computational properties of the two uncertainty models.

Aspect	Discrete Uncertainty	Budgeted Uncertainty
Uncertainty representation	Explicit finite scenarios	Nominal values with bounded deviations
Robust constraints	One per scenario	One per edge
Extra variables	One epigraph variable (z)	One global variable (π) and $\lvert E\rvert$ auxiliary variables (ρ_e)
Model size scaling	$O(\lvert E\rvert + \lvert V\rvert + \lvert\Omega\rvert)$	$O(\lvert E\rvert + \lvert V\rvert)$
Theoretical complexity	NP–hard in general [YY98, ABV09]	Polynomial (continuous relaxation) [BS04]
Worst-case modeling	Exact over scenarios	Aggregated worst-case envelope

Table 2.2: *Comparison of computational properties of discrete scenario-based and budgeted uncertainty min–max shortest-path models.*

This section established the theoretical foundation for ex-ante shortest-path planning under uncertainty. Robust combinatorial optimization models uncertainty explicitly at planning time and yields solutions with worst-case performance guarantees. While discrete uncertainty enumerates extreme realizations explicitly, budgeted uncertainty replaces enumeration by a continuous envelope that bounds coordinated deviations across edges. In the following section, this approach is contrasted with search-based path-planning algorithms that address uncertainty ex post through adaptive replanning during execution.

2-5 Incremental Search-Based Path-Planning

We introduced heuristic search methods for the nominal shortest-path problem in Section 2-3, where traversal costs are assumed to be static and fully known. Incremental search methods can be understood as extensions of heuristic search—most notably A*—to settings in which costs may change during execution. We refer mainly to [KL02] for this sections details about D* Lite.

While A* recomputes a shortest path from scratch when edge costs change, incremental search-based methods instead repair their internal search state. Rather than discarding previous results, they exploit the optimal substructure of shortest paths to reuse prior search effort [KLF04, Ste94]. These methods maintain lower-bound estimates on the remaining cost-to-go and update only those parts of the search space that are affected by newly revealed information. In this sense, incremental search constitutes a minimal relaxation of the nominal heuristic search framework: the underlying graph structure remains unchanged, Bellman-style optimality is preserved locally, and global optimality is recovered incrementally as traversal costs evolve during execution.

Conceptual View: Backward Search and Forward Execution D* Lite maintains a backward shortest-path tree rooted at the goal and incrementally updates this structure as traversal costs change. All search operations are performed backward from the goal, whereas execution proceeds forward from the agent's current position. Consequently, the agent does not plan forward directly but selects actions by querying the maintained backward cost-to-go information (see Figure 2.6 for visualization).

State Representation and Local Consistency Let $G = (S, E)$ be a directed graph with nonnegative edge costs. We consider a fixed goal vertex s_{goal} and an initial start vertex s_{init}. The vertex s_{start} denotes the current position of the mobile agent and changes as the agent moves during execution. For each vertex $s \in S$, D* Lite maintains a cost-to-go estimate $g(s)$ and a one-step lookahead value

$$\text{rhs}(s) = \begin{cases} 0, & \text{if } s = s_{\text{goal}}, \\ \min_{s' \in \text{Succ}(s)} \Big(c(s, s') + g(s') \Big), & \text{otherwise.} \end{cases}$$

The value $g(s)$ represents the currently stored estimate of the shortest-path cost from s to s_{goal}. It can be interpreted as a previously committed $\text{rhs}(s)$ value and may therefore become outdated when edge costs change or when the start vertex moves.

A vertex is said to be *locally consistent* if $g(s) = \text{rhs}(s)$. Inconsistencies indicate that the stored cost-to-go information at s no longer reflects the current environment. D* Lite restores optimality by incrementally re-establishing local consistency only for affected vertices.

Priority Queue and Lexicographic Keys All locally inconsistent vertices are stored in a priority queue U. The next vertex to expand (u) is the one with the smallest lexicographic key. Each vertex in U is assigned a key

$$k(s) = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} \min\{g(s), \text{rhs}(s)\} + h(s_{\text{start}}, s) + k_m \\ \min\{g(s), \text{rhs}(s)\} \end{bmatrix},$$

where the second component acts as a tie-breaker and the first component orders vertices by the best current estimate of total cost-to-go. The use of $\min\{g(s), \text{rhs}(s)\}$ ensures that both over- and under-consistent vertices are prioritized by their most reliable estimate, generalizing the f- and g-value ordering of A*.

Heuristic Shift under Start-Vertex Movement When the agent moves, the start vertex changes and the heuristic term appearing in each priority key changes accordingly, which would normally require reordering the entire priority queue. D* Lite avoids this overhead by introducing a heuristic shift variable k_m, which accumulates

the heuristic change between successive start positions and preserves the relative ordering of keys in the queue. This mechanism enables efficient incremental replanning without recomputing priorities from scratch. Koenig and Likhachev [KL02] also present a variant of D* Lite that operates without this heuristic shift; in this thesis, we focus on the optimized formulation that incorporates k_m.

Admissibility requires the heuristic to never overestimate true shortest-path costs. Consistency strengthens this condition by additionally enforcing a triangle inequality with respect to true shortest-path costs,

$$h(s, s'') \leq c^*(s, s') + h(s', s'') \quad \forall s, s', s'' \in V,$$

which ensures monotonic behavior along optimal paths. In D* Lite, consistency is crucial because heuristic values are reused across incremental repairs and start-vertex shifts; without it, the correctness of the priority queue ordering could not be guaranteed.

Computational Effort Incremental search methods trade full re-planning for localized updates. In the worst case, when cost changes are widespread, D* Lite may revisit a large portion of the graph and its work approaches that of a full A* search for the current start–goal pair. When changes are sparse, only a small set of vertices becomes locally inconsistent, so updates are confined to affected regions and prior search effort is reused. As a result, computational effort depends on graph size, heuristic quality, and the magnitude and frequency of cost updates.

Algorithmic Overview Figure 2.6 summarizes the algorithmic flow: backward propagation from s_{goal} maintains a consistent search tree, the agent executes forward from s_{start}, and any newly revealed costs trigger local inconsistency and subsequent repair through renewed backward propagation.

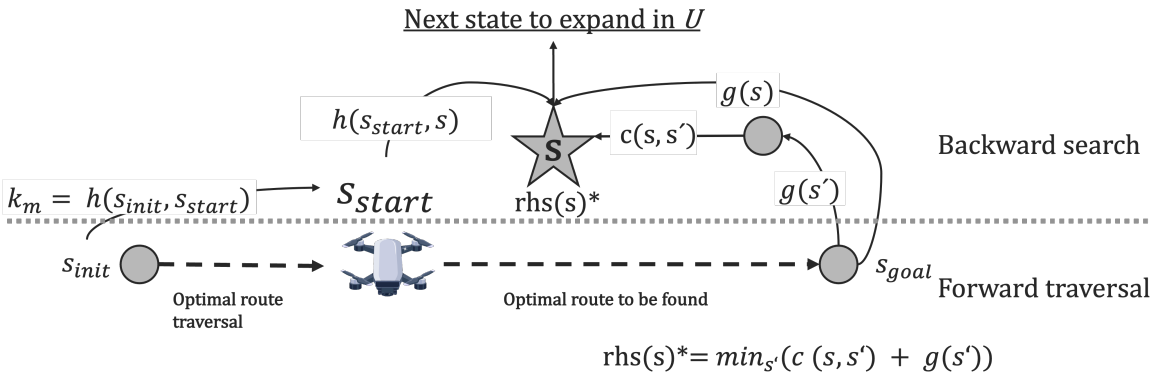


Figure 2.6: *Visualization of D* Lite internals, illustrating backward search, heuristic guidance, and forward path execution.*

Procedural Realization We now describe how the above invariants are enforced algorithmically in D* Lite, following the original pseudocode by Koenig and Likhachev [KL02]:

Priority Key Computation

CalculateKey ties the invariant definitions to the queue ordering by combining $\min\{g(s), \text{rhs}(s)\}$ with the heuristic estimate and the shift variable k_m. The heuristic appears only in this step, ensuring admissible guidance without violating correctness.

procedure CalculateKey(s)
{01"} return $[\min(g(s), rhs(s)) + h(s_{start}, s) + k_m; \min(g(s), rhs(s))]$;

Figure 2.7: *Priority key computation in CalculateKey [KL02].*

Initialization All g- and rhs-values are initialized to infinity. The goal vertex is assigned $\text{rhs}(s_{\text{goal}}) = 0$ and inserted as the only element in the priority queue. This initialization seeds the backward search from the goal and establishes the initial locally consistent state from which cost-to-go information is propagated.

procedure Initialize()
{02"} $U = \emptyset$;
{03"} $k_m = 0$;
{04"} for all $s \in S$ $rhs(s) = g(s) = \infty$;
{05"} $rhs(s_{goal}) = 0$;
{06"} U.Insert$(s_{goal}, [h(s_{start}, s_{goal}); 0])$;

Figure 2.8: *D* Lite initialization procedure [KL02].*

Local Consistency Maintenance

UpdateVertex recomputes the one-step lookahead value $\text{rhs}(s)$ and updates the membership of s in the priority queue U. Vertices that are locally inconsistent are inserted into or updated within U, while locally consistent vertices are removed. This procedure constitutes the fundamental mechanism by which D* Lite restores local consistency after changes in edge costs or state values.

procedure UpdateVertex(u)
{07"} if $(g(u) \neq rhs(u)$ AND $u \in U)$ U.Update$(u, CalculateKey(u))$;
{08"} else if $(g(u) \neq rhs(u)$ AND $u \notin U)$ U.Insert$(u, CalculateKey(u))$;
{09"} else if $(g(u) = rhs(u)$ AND $u \in U)$ U.Remove(u);

Figure 2.9: *Local consistency maintenance via UpdateVertex [KL02].*

Backward Search and Consistency Repair

ComputeShortestPath repeatedly expands the smallest key in U until the current start vertex is locally consistent and no better key remains. Because changes propagate through predecessor relationships, the repair step is a backward search from s_{goal}. This procedure is re-entered whenever cost changes introduce new inconsistencies.

```
procedure ComputeShortestPath()
{10"} while (U.TopKey() < CalculateKey(s_start) OR rhs(s_start) > g(s_start))
{11"}   u = U.Top();
{12"}   k_old = U.TopKey();
{13"}   k_new = CalculateKey(u);
{14"}   if(k_old < k_new)
{15"}     U.Update(u, k_new);
{16"}   else if (g(u) > rhs(u))
{17"}     g(u) = rhs(u);
{18"}     U.Remove(u);
{19"}     for all s ∈ Pred(u)
{20"}       if (s ≠ s_goal) rhs(s) = min(rhs(s), c(s, u) + g(u));
{21"}       UpdateVertex(s);
{22"}   else
{23"}     g_old = g(u);
{24"}     g(u) = ∞;
{25"}     for all s ∈ Pred(u) ∪ {u}
{26"}       if (rhs(s) = c(s, u) + g_old)
{27"}         if (s ≠ s_goal) rhs(s) = min_{s'∈Succ(s)}(c(s, s') + g(s'));
{28"}       UpdateVertex(s);
```

Figure 2.10: *Backward consistency repair in ComputeShortestPath [KL02].*

Interleaving Planning and Execution

The main loop alternates between greedy forward execution and incremental repair. At each step, the agent selects the successor that minimizes $c(s, s') + g(s')$, detects changes in traversal costs, updates affected vertices, and invokes *ComputeShortestPath* as needed to re-establish local consistency. This interleaving of execution and replanning constitutes the core mechanism of D* Lite and provides a natural integration point for the hybrid extensions considered in this thesis.

```
procedure Main()
{29"} s_last = s_start;
{30"} Initialize();
{31"} ComputeShortestPath();
{32"} while (s_start ≠ s_goal)
{33"}   /* if (g(s_start) = ∞) then there is no known path */
{34"}   s_start = arg min_{s'∈Succ(s_start)}(c(s_start, s') + g(s'));
{35"}   Move to s_start;
{36"}   Scan graph for changed edge costs;
{37"}   if any edge costs changed
{38"}     k_m = k_m + h(s_last, s_start);
{39"}     s_last = s_start;
{40"}     for all directed edges (u, v) with changed edge costs
{41"}       c_old = c(u, v);
{42"}       Update the edge cost c(u, v);
{43"}       if (c_old > c(u, v))
{44"}         if (u ≠ s_goal) rhs(u) = min(rhs(u), c(u, v) + g(v));
{45"}       else if (rhs(u) = c_old + g(v))
{46"}         if (u ≠ s_goal) rhs(u) = min_{s'∈Succ(u)}(c(u, s') + g(s'));
{47"}       UpdateVertex(u);
{48"}     ComputeShortestPath();
```

Figure 2.11: *Interleaved planning and execution in Main [KL02].*

CHAPTER 3

IMPLEMENTATION DETAILS

We discuss our experimental framework implementation in this chapter. For details about the programming environment see Chapter 6.

3-1 Decision Support Interface

The experimental pipeline is implemented as a lightweight, model- and data-driven decision support system (DSS) that orchestrates data generation, model execution, and result visualization. It provides a unified control interface for executing all experiments under identical parameter settings and for producing directly comparable outputs (Figure 1.3). The DSS is implemented as a `Streamlit` application, which eliminates the need for a dedicated backend framework and facilitates rapid prototyping. While the system is primarily intended to be run locally, a deployed instance is also provided to demonstrate portability, albeit with restricted access due to licensing restrictions.

3-2 Data and Graph Generation

This section links the implementation in the STGRF generator to the formulations in the appendix (Chapter 6). Each step below states the code action and its mathematical counterpart. We use the `GSTools` and `NetworkX` libraries.

Step 1: Covariance model and kernel selection A kernel from `GSTools` (e.g., Gaussian, Matérn, Stable) (see Table 6.1) is selected and parameterized by variance σ^2, length scale ℓ, anisotropy ratios a_i, and (if applicable) kernel-specific parameters ν or α. In the implementation this corresponds to

`Kernel(spatial_dim=3, temporal=False, var=..., len_scale=..., anis=...)`.

This instantiates the covariance model $C(r)$ in (6.2) with $\mathrm{cor}(\cdot)$ from (6.3). The anisotropy ratios define effective correlation lengths $\ell_i = \ell a_i$ and enter the metric distance in (6.5), which in turn defines the spatio-temporal covariance in (6.6). Time is treated as a third spatial dimension (spatial dimension $d = 3$), consistent with $\mathbf{x} = (x, y, \tau)$.

Step 2: Structured space–time grid. The implementation constructs the coordinate arrays

$$x = (0, \Delta, \ldots, (N-1)\Delta), \quad y = (0, \Delta, \ldots, (N-1)\Delta), \quad \tau = (0, \ldots, |\mathcal{T}| - 1),$$

where Δ is the cell size and N is the number of grid points per axis (`grid_size` in the implementation). These arrays define the structured grid on which the field is evaluated, and each grid point corresponds to a coordinate vector $\mathbf{x} = (x, y, \tau)$ in the formulation of $Z(\mathbf{x})$. Let

$$Z(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d,$$

denote a real-valued random field indexed over a d-dimensional domain.

Step 3: Spectral random field generation. Given the covariance model, `GSTools` constructs a stationary Gaussian random field via its spectral randomization method. The call `SRF(model, seed).structured([x, y, t])`, where `t` is the scenario index array τ, produces samples $Z(\mathbf{x})$ on the structured grid, implementing the generation in (6.7) for the covariance in (6.6).

Step 4: Global normalization. To ensure nonnegative and comparable costs across scenarios, the sampled three-dimensional field is rescaled according to

$$\tilde{Z} = \frac{Z - \min Z}{\max Z - \min Z},$$

where the minimum and maximum are taken over the entire space–time grid. This transformation maps all costs to the unit interval [0,1]. The implications of this normalization for experimental results are discussed in Chapter **??**.

Step 5: Scenario slices and graph mapping. For each scenario index τ, the 2D slice $f_\tau(x, y) := Z(x, y, \tau)$ is extracted as the spatial cost surface. The routing graph is generated as a directed 4-connected grid using `NetworkX`; each grid point (x, y) is mapped to a node id $i = xN + y$. Each directed edge (i, j) receives the mean of its incident node values,

$$c(i, j) = \tfrac{1}{2}\Big(f_\tau(x_i, y_i) + f_\tau(x_j, y_j)\Big),$$

yielding a scenario-specific edge-cost graph derived directly from the realization $Z(\mathbf{x})$.

Step 6: Export for downstream models. The implementation writes a global node file and one edge file per scenario, as well as the per-scenario field slices, enabling downstream shortest-path and robust optimization models to consume the same spatio-temporal realizations.

3-3 Discrete Uncertainty Min–Max Model

This section instantiates the discrete scenario-based min–max formulation introduced in Section 2-4.

The directed graph $G = (V, E)$ is constructed from `nodes.csv` and `scenario_000/edges.csv`, which fixes a consistent edge index set shared across all scenarios (Section 2-2).

For each scenario $\omega \in \Omega$, the corresponding edge-cost vector $\mathbf{c}^\omega$ is read from `scenario_`ω`/edges.csv` and aligned to this common indexing.

The mixed-integer linear program is formulated using binary edge-selection variables x_e and a continuous epigraph variable z. Path feasibility is enforced via the flow-balance constraints (2.2), while robustness with respect to discrete uncertainty is captured through the scenario-wise constraints $z \geq \sum_{e \in E} c_e^\omega x_e$ for all $\omega \in \Omega$. No additional modeling assumptions are introduced at the implementation level. The resulting MILP is solved using Gurobi with the solver configuration described in Section 3-5.

Discrete-Scenario Min–Max Shortest-Path Model

$$\min_{x,\,z} \quad z \tag{3.1}$$

$$\text{s.t.} \quad z \;\geq\; \sum_{e \in E} c_e^\omega \, x_e \qquad \forall \omega \in \Omega, \tag{3.2}$$

$$\sum_{e \in \delta^+(v)} x_e \;-\; \sum_{e \in \delta^-(v)} x_e \;=\; \begin{cases} 1, & v = s, \\ -1, & v = t, \\ 0, & \text{otherwise}, \end{cases} \qquad \forall v \in V, \tag{3.3}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E, \tag{3.4}$$

$$z \in \mathbb{R}. \tag{3.5}$$

The solution represents an s–t path that is feasible across all scenarios and minimizes the worst-case realized traversal cost. Theoretical complexity and model-size properties of the discrete min–max formulation are discussed in Section **??**.

3-4 Budgeted Uncertainty Min–Max Model

This section instantiates the budgeted uncertainty formulation introduced in Section 2-4. In contrast to the discrete scenario-based model, scenario-dependent edge costs are first aggregated in a preprocessing step to obtain a nominal cost and an associated maximum deviation for each edge. Let Ω denote the set of cost scenarios and let c_e^ω be the cost of edge e under scenario $\omega \in \Omega$.

Scenario costs are aggregated using (3.7)–(3.8), while preserving the same edge indexing as in the discrete uncertainty model. The resulting nominal costs $\underline{c}_e$ and deviations d_e are treated as fixed parameters in the robust optimization model. The robust counterpart is formulated using binary edge-selection variables x_e, a global deviation variable π, and auxiliary variables ρ_e. Path feasibility is enforced via the flow-balance constraints (2.2), while robustness is captured by the budgeted constraints $\pi + \rho_e \geq d_e x_e$ for all $e \in E$. The uncertainty budget Γ is treated as an experimental parameter, and solver limits and integrality settings follow Section 3-5.

The nominal cost $\underline{c}_e$ is defined either as the minimum observed cost (3.6) or as the average cost across all scenarios (3.7). The deviation d_e captures the maximum adverse deviation from the nominal value and is computed as (3.8).

$$\underline{c}_e^{\min} = \min_{\omega \in \Omega} c_e^\omega, \tag{3.6}$$

$$\underline{c}_e^{\text{avg}} = \frac{1}{|\Omega|} \sum_{\omega \in \Omega} c_e^\omega, \tag{3.7}$$

$$d_e = \max_{\omega \in \Omega} c_e^\omega - \underline{c}_e. \tag{3.8}$$

These quantities define a budgeted uncertainty representation in which the realized cost of each edge may increase from its nominal value by at most d_e, subject to a global uncertainty budget. In preliminary experiments, using the minimum-based nominal cost (3.6) resulted in an overly optimistic abstraction of the underlying cost field when compared to the remaining planning paradigms. Therefore, all subsequent experiments employ the average-based nominal cost (3.7) in combination with the deviation definition in (3.8).

Budgeted-Uncertainty Min–Max Robust Shortest-Path

$$\min_{x,\,\pi,\,\rho} \quad \sum_{e\in E} \underline{c}_e\, x_e \;+\; \Gamma\,\pi \;+\; \sum_{e\in E} \rho_e \tag{3.9}$$

$$\text{s.t.} \quad \pi + \rho_e \;\geq\; d_e\, x_e \qquad \forall e \in E, \tag{3.10}$$

$$\sum_{e\in\delta^+(v)} x_e \;-\; \sum_{e\in\delta^-(v)} x_e \;=\; \begin{cases} 1, & v = s, \\ -1, & v = t, \\ 0, & \text{otherwise}, \end{cases} \qquad \forall v \in V, \tag{3.11}$$

$$x_e \in \{0,1\} \qquad \forall e \in E, \tag{3.12}$$

$$\pi \geq 0,\; \rho_e \geq 0 \qquad \forall e \in E. \tag{3.13}$$

This formulation yields an s–t path that is robust with respect to the budgeted uncertainty set. Theoretical complexity results are summarized in Section **??**, while solver settings and integrality enforcement are described in Section 3-5.

3-5 Solver Settings

Both robust models are implemented as mixed-integer linear programs and solved using the `Gurobi` optimizer via its Python interface. Unless stated otherwise, all runs use a wall-clock time limit of 60 seconds and a relative MIP optimality gap of 2%, with presolve routines, cutting planes, and branching strategies left at solver defaults. Binary edge-selection variables enforce path integrality; the budgeted deviation constraints do not preserve total unimodularity in general, so the linear relaxation can be fractional. Gurobi solves the MILPs via branch-and-bound; reported runtimes correspond to solver wall-clock time measured on identical hardware.

3-6 D* Lite

We implement D* Lite in accordance with the reference pseudocode as described in Chapter 2, Section 2-5. All theoretical properties and correctness guarantees of D* Lite are discussed in Chapter 2.

Modeling Differences. Compared to the reference formulation by Koenig and Likhachev [KL02], the following modeling and implementation choices are adopted:

- a 4-connected grid graph instead of 8-connectivity; consequently, an admissible Manhattan-distance heuristic is used rather than a Euclidean heuristic,
- initial edge costs are derived from the first STRF realization rather than uniform default values,
- edges are never blocked; unfavorable regions are modeled exclusively via increased traversal costs,
- the priority queue is implemented using the `heapq` library. Queue updates rely on lazy deletion, i.e., outdated entries are discarded upon extraction rather than removed explicitly. This implementation choice preserves the algorithmic behavior and correctness of D* Lite while simplifying queue management.

These choices ensure structural consistency with the robust optimization models and isolate uncertainty effects to edge costs.

Preprocessing and Graph Construction. Prior to executing D* Lite, the graph structure and associated cost data are constructed from the CSV inputs. Vertices are loaded from `nodes.csv` and mapped to planar coordinates `node_id` $\mapsto (x, y)$. Directed edges and their initial traversal costs are read from `scenario_000/edges.csv`, from which adjacency lists of successors and predecessors are constructed. All vertices are included in the graph, including those without outgoing edges.

For each subsequent scenario directory `scenario_`ω with $\omega \geq 1$, the corresponding `edges.csv` file is parsed and stored as a complete list of edge–cost triples (u, v, c). Scenario files are stored as complete edge–cost lists and are processed on demand during execution as part of the scenario-based cost evolution mechanism.

If a robust offline path is provided, it is loaded as an ordered sequence of node identifiers and used to construct guidance beacons as described above. Otherwise, the algorithm proceeds without global guidance.

Scenario-Based Cost Evolution. The original D* Lite algorithm assumes static edge costs, while allowing cost changes to be revealed during execution. However, the reference pseudocode by Koenig and Likhachev [KL02] (lines {37–38} in the *Main()* procedure) does not prescribe a concrete mechanism for detecting or generating edge-cost updates; such changes are treated as exogenous events. This mechanism is referred to as *scenario-based cost evolution* in the remainder of this section.

To operationalize this assumption under spatio-temporal uncertainty, traversal costs are modeled as a sequence of time-indexed cost maps derived from spatio-temporal random field (STRF) realizations. Each realization defines a complete edge-cost assignment that remains fixed over a finite execution interval.

The base graph is initialized using the first realization (`scenario_000`). A deterministic update schedule controls when subsequent cost maps become active. Let

$$\texttt{steps\_per\_layer} = \left\lceil \frac{|x_s - x_t| + |y_s - y_t|}{|\Omega|} \right\rceil,$$

where $|x_s - x_t| + |y_s - y_t|$ denotes the Manhattan distance between start and goal, and $|\Omega|$ is the number of available scenarios. After each forward move, a step counter is incremented; when it reaches `steps_per_layer`, the active scenario index is advanced (capped at the final scenario) and the counter is reset. Because the graph is initialized with `scenario_000` while the active scenario index starts at 1, the updates corresponding to `scenario_001` are applied immediately after the first movement step; subsequent scenario changes occur every `steps_per_layer` forward moves. Thus, the initial backward search is computed on `scenario_000`, but the first update step switches to `scenario_001` immediately after the first forward move, so the initial search primarily guides that first action and seeds the subsequent incremental replanning.

Coupling Robust Planning and Incremental Execution. When a robust offline path is available, it is used exclusively as coarse guidance for incremental execution. Let the robust path be given as an ordered sequence of nodes. Removing the start and goal yields an interior node list of length n. A maximum number of guidance points (beacons) is specified as an experimental parameter. We set

$$c = \min\{\texttt{beacon\_cap}, n\},$$

partition the interior list into c contiguous segments, and select the midpoint node of each segment. The selected nodes form the beacon sequence, with at most c beacons.

If at least one beacon is present, the first beacon is used as the initial goal; otherwise, the final destination node is used. At any time, exactly one beacon is active. Whenever a new beacon becomes active—either by schedule or upon reaching the cur-

rent beacon—the D* Lite search state is fully reinitialized: all g- and rhs-values are reset, the priority queue is cleared, and the new goal vertex is inserted before invoking *ComputeShortestPath()*. This design intentionally discards cross-goal reuse in order to isolate the effect of global guidance from local replanning.

Beacon activation follows two rules. First, when the agent reaches the currently active beacon, the next beacon in the sequence is activated immediately. Second, beacons may be activated periodically in response to scenario changes. Let $L = \texttt{max\_scenario} + 1$ denote the total number of scenario layers and let B be the number of beacons. The activation interval is defined as $\lceil L/B \rceil$, and a beacon is activated when

$$(\texttt{current\_scenario} - 1) \bmod \lceil L/B \rceil = 0,$$

with at most one activation per scenario layer. Once all beacons have been exhausted, the final goal is restored and the algorithm proceeds without further global guidance.

CHAPTER 4

EXPERIMENTAL RESULTS

In our experiments, global path planners are evaluated with respect to solution quality and computational effort.

For each configuration, results are aggregated over runs using identical seeds across algorithms. We report the median realized costs and median runtime. Uncertainty is summarized via a non-paramteric bootstrap 95% confidence interval of the median cost. Our CI is estimated by repeatedly resampling observed costs with replacement 1,000 times. Each resample is the same size as the original sample and we compute the median for each resample. We then take the 2.5th and 97.5th percentiles of those 1,000 medians as the 95% CI.

This combination provides a statistically sound and interpretable summary of algorithmic performance under stochastic spatio-temporal uncertainty.

Two-layer experiments:

- layer: run extremes to find out which parameters yield the most promising comparison results
- layer: re-run promising parameters in greater variability

Research Questions To achieve the stated research objective, this thesis addresses the following research questions through experimental evaluation:

RQ1: How do ex-ante robust optimization and ex-post adaptive search-based planning methods compare in terms of path quality and realized traversal cost under identical spatio-temporal cost uncertainty?

RQ2: What are the computational trade-offs between robust optimization models and incremental search methods when applied to shortest-path planning under uncertainty?

RQ3: How does execution-time adaptivity influence the performance of planning approaches in environments with dynamically evolving traversal costs?

RQ4: Can a hybrid planning approach that combines robust ex-ante guidance with incremental ex-post replanning improve performance compared to using either paradigm in isolation?

Computational Effort In line with Section **??**, runtime is analyzed with respect to the scenario count $|\Omega|$ for discrete uncertainty and the uncertainty budget Γ for budgeted uncertainty. We report median solver runtimes alongside cost statistics in the results below.

4-1 Baseline

These are the baseline parameters we test our models with and compare to. To keep our experiments reasonable and comparable, interesting parameters will be changed with the remaining parameters unchanged.

Table 4.1: *Baseline experimental configuration*

Parameter	**Symbol/ Description**	**Value**
Grid size	$N_p \times N_q$	20×20
Anisotropy	(a_p, a_q, a_t)	$(1.0,\ 1.0,\ 1.0)$
Number of scenarios	$\lvert\mathcal{T}\rvert$	10
Spatial scaling factor	α	0.25
Temporal scaling factor	β	0.30
Gamma scaling factor	λ	0.10
Spatial length scales	(ℓ_p, ℓ_q)	$(\alpha N_p,\ \alpha N_q)$
Temporal length scale	ℓ_t	$\beta\lvert\mathcal{T}\rvert$
Variance	σ^2	1.0
Robustness budget	Γ	$\Gamma(\lambda) = \lambda\Gamma_{\max}$
Covariance model	–	Gaussian
Seed range	–	1–100
Milestone cap (D* Lite)	–	10

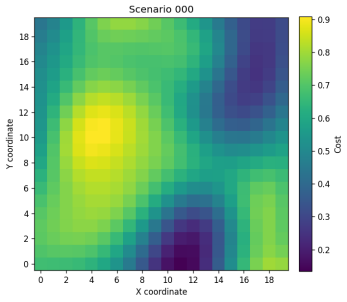
Algorithm	Cost (median)	95% CI	Runtime (ms, median)
Discrete Uncertainty	17.19	[16.55, 17.98]	287.55
Budgeted Uncertainty	17.04	[16.37, 17.35]	85.43
D* Lite	18.05	[17.38, 18.92]	230.96
D* Lite w. discrete guidance	16.28	[15.33, 17.00]	231.38
D* Lite w. budgeted guidance	15.33	[14.57, 16.12]	231.29

Table 4.2: *Performance comparison under the baseline configuration.*

Length Scale To ensure comparability across instance sizes, correlation lengths are defined relative to the extent of the modeled domain. Throughout all experiments, the spatial discretization is kept fixed, i.e., the cell size satisfies $\Delta p = \Delta q = 1$, and correlation lengths are therefore expressed directly in grid units. Spatial correlation lengths are scaled proportionally with grid size according to

$$\ell_p = \alpha N_p, \qquad \ell_q = \alpha N_q,$$

where the scaling constant α is calibrated from a reference configuration. This construction keeps the ratio ℓ/N constant and thereby preserves the relative spatial smoothness of the generated cost fields across grid resolutions.



Many real-world planning and routing decisions must be made under conditions of uncertainty and dynamic change. Traversal costs may depend on environmental factors such as weather conditions, terrain properties, or sensor noise and may vary over both space and time.
This illustration shows how heterogeneous environmental sensor data can be spatially aggregated into a continuous cost field that serves as input for graph-based routing models.

Figure 4.1: *Visualization of the baseline cost field for the first time frame.*

Temporal correlation is defined relative to the scenario horizon $|\mathcal{T}|$ (with fixed time step). When $|\mathcal{T}|$ is fixed (e.g., $|\mathcal{T}| = 10$), the temporal correlation length ℓ_t is kept constant; if $|\mathcal{T}|$ changes, we set

$$\ell_t = \beta|\mathcal{T}|,$$

with β calibrated analogously, so that the ratio $\ell_t/|\mathcal{T}|$ remains constant.

The scaling constants are calibrated from the baseline configuration, yielding $\alpha = \ell_p^0/N_p^0 = \ell_q^0/N_q^0 = 0.25$ and $\beta = \ell_t^0/|\mathcal{T}|^0 = 0.3$.

Uncertainty Budget Since the budget parameter Γ is continuous in our implementation, we parameterize robustness via a normalized factor $\lambda \in [0, 1]$ and set

$$\Gamma(\lambda) = \lambda\Gamma_{\max}, \qquad \Gamma_{\max} := L_{\min} = (N_p - 1) + (N_q - 1),$$

where $L_{\min}$ is the minimum number of edges on an s–t path in the 4-connected grid. Thus, $\lambda = 0$ yields the nominal model, while $\lambda = 1$ corresponds to full conservatism. We use $\lambda = 0.1$ as the baseline robustness level.

4-2 Experiment 1: Urban vs. Regional Scale

Parameter	Baseline	Tested values
Grid Size	20×20	$\{50 \times 50,\ 75 \times 75,\ 100 \times 100\}$

Table 4.3: *Experiment 1: Parameter variations relative to the baseline configuration*

By increasing the overall grid size, we also scale our spatial length scale (l_x, l_y) as well as our uncertainty budget Γ.

Table 4.4: *Performance comparison across grid sizes*

	Algorithm	Cost (median)	95% CI	Runtime (ms, median)
$20 \times 20^*$	Discrete Uncertainty	17.19	[16.55, 17.98]	287.55
	Budgeted Uncertainty	17.04	[16.37, 17.35]	85.43
	D* Lite	18.05	[17.38, 18.92]	230.96
	D* Lite w. Discrete guidance	16.28	[15.33, 17.00]	231.38
	D* Lite w. Budgeted guidance	15.33	[14.57, 16.12]	231.29
50×50	Discrete Uncertainty	44.61	[42.92, 45.97]	3160.54
	Budgeted Uncertainty	43.83	[42.71, 45.00]	3989.21
	D* Lite	47.10	[44.77, 48.91]	2011.42
	D* Lite w. Discrete guidance	42.21	[40.16, 44.06]	1951.31
	D* Lite w. Budgeted guidance	39.45	[37.56, 41.49]	1943.66
75×75	Discrete Uncertainty	68.07	[65.09, 69.42]	10354.67
	Budgeted Uncertainty	66.27	[64.08, 67.79]	41848.94
	D* Lite	70.81	[67.90, 74.09]	5985.83
	D* Lite w. Discrete guidance	63.25	[60.74, 65.59]	5873.51
	D* Lite w. Budgeted guidance	59.39	[56.82, 62.16]	5862.06
100×100	Discrete Uncertainty	90.90	[87.12, 92.82]	26568.95
	Budgeted Uncertainty	88.91	[85.00, 90.72]	61120.00
	D* Lite	94.80	[90.50, 100.33]	14442.30
	D* Lite w. Discrete guidance	85.25	[81.03, 87.82]	14037.08
	D* Lite w. Budgeted guidance	79.48	[75.90, 82.99]	14346.51

$*$ Baseline configuration.

4-3 Experiment 2: Short vs. Long Exposure

In the short–long exposure experiment we vary the number of scenarios $|\mathcal{T}|$ while keeping the temporal length scale ℓ_t fixed.

Parameter	**Baseline**	**Tested values**
Number of Scenarios	10	$\{1, 5, 20, 50, 100\}$

Table 4.5: *Experiment 2: Parameter variations relative to the baseline configuration*

Table 4.6: *Performance comparison across exposure levels* $|\mathcal{T}|$

	Algorithm	Cost (median)	95% CI	Runtime (ms, median)
$\|\mathcal{T}\| = 1$	Discrete Uncertainty	12.35	[11.58, 13.07]	80.74
	Budgeted Uncertainty	12.35	[11.58, 13.11]	39.52
	D* Lite	13.59	[12.64, 14.80]	40.46
	D* Lite w. Discrete guidance	12.35	[11.52, 13.11]	40.67
	D* Lite w. Budgeted guidance	12.35	[11.54, 13.10]	40.64
$\|\mathcal{T}\| = 5$	Discrete Uncertainty	15.18	[14.11, 15.97]	167.74
	Budgeted Uncertainty	14.70	[14.17, 15.37]	74.12
	D* Lite	15.94	[15.17, 17.22]	123.71
	D* Lite w. Discrete guidance	14.56	[13.75, 15.10]	123.69
	D* Lite w. Budgeted guidance	13.78	[12.71, 14.35]	123.77
$\|\mathcal{T}\| = 10^*$	Discrete Uncertainty	17.19	[16.55, 17.98]	287.55
	Budgeted Uncertainty	17.04	[16.37, 17.35]	85.43
	D* Lite	18.05	[17.38, 18.92]	230.96
	D* Lite w. Discrete guidance	16.28	[15.33, 17.00]	231.38
	D* Lite w. Budgeted guidance	15.33	[14.57, 16.12]	231.29
$\|\mathcal{T}\| = 50$	Discrete Uncertainty	18.94	[18.57, 19.91]	1406.50
	Budgeted Uncertainty	19.12	[18.82, 19.58]	167.20
	D* Lite	18.99	[18.14, 19.57]	1064.51
	D* Lite w. Discrete guidance	17.64	[17.36, 17.95]	1066.09
	D* Lite w. Budgeted guidance	17.40	[16.90, 18.06]	1065.92
$\|\mathcal{T}\| = 100$	Discrete Uncertainty	18.84	[18.17, 19.59]	2839.01
	Budgeted Uncertainty	19.80	[19.53, 20.18]	278.15
	D* Lite	19.28	[18.58, 19.93]	2011.47
	D* Lite w. Discrete guidance	19.00	[17.90, 19.56]	1997.88
	D* Lite w. Budgeted guidance	18.53	[17.49, 18.97]	2000.87

$*$ Baseline configuration.

Although the budgeted uncertainty model admits a polynomial-size reformulation, its LP relaxation is significantly weaker than that of the discrete min–max formulation. The global uncertainty budget introduces coupling across all edges, allowing the relaxation to distribute deviations fractionally over many arcs. As a result, the solver faces substantially more fractional variables at the root node, leading to slower convergence in practice. In contrast, the discrete uncertainty model yields tighter relaxations

despite its theoretical NP-hardness, explaining its superior empirical runtime.

4-4 Experiment 3: Rough vs. Smooth Terrain

By varying the spatial scaling factor α, we change the spatial length scales and thus the terrain roughness.

Parameter	**Baseline**	**Tested values**
Spatial scaling factor α	0.25	$\{0.10, 0.50\}$

Table 4.7: *Experiment 4: Parameter variations relative to the baseline configuration*

Table 4.8: *Performance comparison across spatial scaling factors α*

	Algorithm	Cost (median)	95% CI	Runtime (ms, median)
$\alpha = 0.10$	Discrete Uncertainty	16.47	[16.03, 16.90]	281.89
	Budgeted Uncertainty	16.52	[16.15, 17.00]	77.05
	D* Lite	18.46	[18.11, 19.10]	231.20
	D* Lite w. Discrete guidance	15.49	[15.18, 16.03]	230.87
	D* Lite w. Budgeted guidance	15.28	[14.80, 16.00]	230.76
$\alpha = 0.25^*$	Discrete Uncertainty	17.19	[16.55, 17.98]	287.55
	Budgeted Uncertainty	17.04	[16.37, 17.35]	85.43
	D* Lite	18.05	[17.38, 18.92]	230.96
	D* Lite w. Discrete guidance	16.28	[15.33, 17.00]	231.38
	D* Lite w. Budgeted guidance	15.33	[14.57, 16.12]	231.29
$\alpha = 0.50$	Discrete Uncertainty	16.91	[15.75, 18.56]	271.63
	Budgeted Uncertainty	17.45	[16.90, 18.31]	83.14
	D* Lite	18.46	[16.22, 19.84]	228.03
	D* Lite w. Discrete guidance	16.75	[14.83, 17.98]	227.85
	D* Lite w. Budgeted guidance	14.97	[14.36, 16.68]	228.19

$*$ Baseline configuration.

4-5 Experiment 4: Temporal Stability

By varying the temporal scaling factor β, we change the temporal length scale and thus the stability across scenarios.

Parameter	**Baseline**	**Tested values**
Temporal scaling factor β	0.30	$\{0.10, 0.60\}$

Table 4.9: *Experiment 6: Parameter variations relative to the baseline configuration*

Table 4.10: *Performance comparison across temporal scaling factors β*

	Algorithm	Cost (median)	95% CI	Runtime (ms, median)
$\beta = 0.10$	Discrete Uncertainty	18.69	[17.80, 19.55]	324.63
	Budgeted Uncertainty	18.47	[18.11, 19.00]	105.77
	D* Lite	18.97	[18.26, 19.93]	237.79
	D* Lite w. Discrete guidance	17.45	[16.53, 18.08]	237.79
	D* Lite w. Budgeted guidance	16.35	[15.82, 16.99]	237.63
$\beta = 0.30^*$	Discrete Uncertainty	17.19	[16.55, 17.98]	287.55
	Budgeted Uncertainty	17.04	[16.37, 17.35]	85.43
	D* Lite	18.05	[17.38, 18.92]	230.96
	D* Lite w. Discrete guidance	16.28	[15.33, 17.00]	231.38
	D* Lite w. Budgeted guidance	15.33	[14.57, 16.12]	231.29
$\beta = 0.60$	Discrete Uncertainty	15.32	[14.53, 16.13]	284.08
	Budgeted Uncertainty	14.89	[14.42, 15.33]	88.66
	D* Lite	16.09	[15.41, 17.25]	235.29
	D* Lite w. Discrete guidance	14.72	[13.83, 15.22]	235.18
	D* Lite w. Budgeted guidance	13.85	[12.98, 14.50]	234.96

$*$ Baseline configuration.

4-6 Experiment 5: Uncertainty Budgeted

By varying the robustness budget scaling factor λ, we change the uncertainty budget $\Gamma = \lambda \Gamma_{\max}$.

Parameter	Baseline	Tested values
Gamma scaling factor λ	0.10	$\{0, 0.25, 0.50, 1.00\}$

Table 4.11: *Experiment 7: Parameter variations relative to the baseline configuration*

Table 4.12: *Performance comparison across budget levels λ*

	Algorithm	Cost (median)	95% CI	Runtime (ms, median)
$\lambda = 0$	Discrete Uncertainty	17.19	[16.60, 17.97]	291.86
	Budgeted Uncertainty	↓ 15.77 ↓	[15.18, 16.10]	↓ 49.30 ↓
	D* Lite	18.05	[17.38, 18.92]	232.22
	D* Lite w. Discrete guidance	16.28	[15.41, 17.00]	232.12
	D* Lite w. Budgeted guidance	15.24	[14.57, 15.95]	231.65
$\lambda = 0.10^*$	Discrete Uncertainty	17.19	[16.55, 17.98]	287.55
	Budgeted Uncertainty	17.04	[16.37, 17.35]	85.43
	D* Lite	18.05	[17.38, 18.92]	230.96
	D* Lite w. Discrete guidance	16.28	[15.33, 17.00]	231.38
	D* Lite w. Budgeted guidance	15.33	[14.57, 16.12]	231.29
$\lambda = 0$	Discrete Uncertainty	17.19	[16.60, 17.96]	287.01
	Budgeted Uncertainty	↑ 18.25 ↑	[17.74, 18.75]	↑ 433.59 ↑
	D* Lite	18.05	[17.38, 18.90]	235.27
	D* Lite w. Discrete guidance	16.28	[15.33, 17.05]	234.16
	D* Lite w. Budgeted guidance	15.59	[14.63, 16.28]	233.42
$\lambda = 0.50$	Discrete Uncertainty	17.19	[16.60, 17.98]	315.02
	Budgeted Uncertainty	19.73	[19.31, 20.32]	3415.74
	D* Lite	18.05	[17.38, 18.90]	313.06
	D* Lite w. Discrete guidance	16.28	[15.42, 17.05]	268.97
	D* Lite w. Budgeted guidance	15.75	[15.14, 16.53]	254.03
$\lambda = 1.00$	Discrete Uncertainty	17.19	[16.55, 17.97]	323.56
	Budgeted Uncertainty	21.64	[21.00, 22.02]	45773.08
	D* Lite	18.05	[17.36, 18.92]	333.87
	D* Lite w. Discrete guidance	16.28	[15.32, 17.07]	273.82
	D* Lite w. Budgeted guidance	15.75	[15.15, 16.59]	257.00

$*$ Baseline configuration.

4-7 Experiment 6: Number of Beacons

In this experiment, we study the effect of milestone-based replanning frequency by varying the number of beacons used during execution. Increasing the number of beacons corresponds to more frequent replanning opportunities along the path.

Parameter	**Baseline**	**Tested values**
Number of beacons	5	$\{1, 5, 20\}$

Table 4.13: *Experiment 8: Parameter variations relative to the baseline configuration*

Table 4.14: *Performance comparison across beacon counts*

	Algorithm	Cost (median)	95% CI	Runtime (ms, median)
1 Beacon	Discrete Uncertainty	17.19	[16.60, 17.98]	282.84
	Budgeted Uncertainty	17.04	[16.46, 17.35]	85.14
	D* Lite	18.05	[17.38, 18.90]	229.52
	D* Lite w. Discrete guidance	16.80	[16.04, 17.65]	229.42
	D* Lite w. Budgeted guidance	16.35	[15.81, 17.35]	229.04
5* Beacons	Discrete Uncertainty	17.19	[16.48, 17.97]	276.43
	Budgeted Uncertainty	17.04	[16.40, 17.34]	83.11
	D* Lite	18.05	[17.36, 18.90]	226.48
	D* Lite w. Discrete guidance	15.92	[15.01, 16.72]	224.72
	D* Lite w. Budgeted guidance	15.39	[14.54, 16.28]	224.59
20 Becons	Discrete Uncertainty	17.19	[16.60, 17.97]	279.12
	Budgeted Uncertainty	17.04	[16.40, 17.35]	83.22
	D* Lite	18.05	[17.38, 18.88]	227.46
	D* Lite w. Discrete guidance	16.45	[15.51, 17.14]	227.22
	D* Lite w. Budgeted guidance	15.57	[14.64, 16.29]	226.85

$*$ Baseline configuration.

CHAPTER 5

CONCLUSION

Our experiments show that combining robust combinatorial optimization handling uncertainty via uncertainty sets can be used for mobile robot path finding to a certain degree.

One possible adaptation that we find worth looking into in future research is robust optimization for smaller scenarios/ windows. For example, imagine an evironment with environmental sensors placed on ”hotspots”. We could map sensor data as costs on an unstructured grid and use this real time data as possible waypoints for a coupled-D*Lite approach. A robust combinatorial optimization method could be used beforehand to make out the best possible route using these possible waypoints to give a rough estimate route for D* Lite to then traverse through.

BIBLIOGRAPHY

[ABV09] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438, September 2009.

[AK23] Jaafar Ahmed Abdulsaheb and Dheyaa Jasim Kadhim. Classical and Heuristic Approaches for Mobile Robot Path Planning: A Survey. *Robotics*, 12(4):93, June 2023.

[AS24] Elnaz Azizi and Abbas Seifi. Shortest path network interdiction with incomplete information: A robust optimization approach. *Annals of Operations Research*, 335(2):727–759, April 2024.

[ASSM+22] Amylia Ait Saadi, Assia Soukane, Yassine Meraihi, Asma Benmessaoud Gabis, Seyedali Mirjalili, and Amar Ramdane-Cherif. UAV Path Planning Using Optimization Approaches: A Survey. *Archives of Computational Methods in Engineering*, 29(6):4233–4284, October 2022.

[BDG+24] Prathamesh Bagad, Prachi Dahatonde, Rushikesh Gagare, Ishwari Athare, and Gauri Ghule. Optimizing Pathfinding in Dynamic Environments: A Comparative Study of A* and D-Lite Algorithms. In *2024 IEEE Pune Section International Conference (PuneCon)*, pages 1–11, Pune, India, December 2024. IEEE.

[BS03] Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98(1-3):49–71, September 2003.

[BS04] Dimitris Bertsimas and Melvyn Sim. The Price of Robustness. *Operations Research*, 52(1):35–53, February 2004.

[CDG19] André Chassein, Trivikram Dokka, and Marc Goerigk. Algorithms and uncertainty sets for data-driven robust shortest path problems. *European Journal of Operational Research*, 274(2):671–686, April 2019.

[Dij59] E W Dijkstra. A note on two problems in connexion with graphs. 1959.

[DP84] Narsingh Deo and Chi-Yin Pang. Shortest-path algorithms: Taxonomy and annotation. *Networks*, 14(2):275–323, June 1984.

[FB22] Joey G. Fernando and Myelinda Baldelovar. Decision Support System: Overview, Different Types and Elements. *TechnoareteTransactions on Intelligent Data Mining and Knowledge Discovery*, 2(2), May 2022.

[FIU+15] M.P. Fanti, G. Iacobellis, W. Ukovich, V. Boschian, G. Georgoulas, and C. Stylios. A simulation based Decision Support System for logistics management. *Journal of Computational Science*, 10:86–96, September 2015.

[GH23] Gopi Gugan and Anwar Haque. Path Planning for Autonomous Drones: Challenges and Future Directions. *Drones*, 7(3):169, February 2023.

[GH24] Marc Goerigk and Michael Hartisch. *An Introduction to Robust Combinatorial Optimization: Concepts, Models and Algorithms for Decision Making under Uncertainty*, volume 361 of *International Series in Operations Research & Management Science*. Springer Nature Switzerland, Cham, 2024.

[GHF+24] Yang Gao, Qidong Han, Shuo Feng, Zhen Wang, Teng Meng, and Jingshuai Yang. Improvement and Fusion of D*Lite Algorithm and Dynamic Window Approach for Path Planning in Complex Environments. *Machines*, 12(8):525, August 2024.

[GK25] Marc Goerigk and Jannis Kurtz. Data-driven prediction of relevant scenarios for robust combinatorial optimization. *Computers & Operations Research*, 174:106886, February 2025.

[GL21] Marc Goerigk and Stefan Lendl. Robust Combinatorial Optimization with Locally Budgeted Uncertainty. *Open Journal of Mathematical Optimization*, 2:1–18, May 2021.

[GRT+16] Chrysanthos E. Gounaris, Panagiotis P. Repoussis, Christos D. Tarantilis, Wolfram Wiesemann, and Christodoulos A. Floudas. An Adaptive Memory Programming Framework for the Robust Capacitated Vehicle Routing Problem. *Transportation Science*, 50(4):1239–1260, November 2016.

[HHW24] Yong He, Ticheng Hou, and Mingran Wang. A new method for unmanned aerial vehicle path planning in complex environments. *Scientific Reports*, 14(1):9257, April 2024.

[HNR68] Peter Hart, Nils Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[Kee98] Peter B Keenan. Spatial decision support systems for vehicle routing. *Decision Support Systems*, 22(1):65–71, January 1998.

[KL02] Sven Koenig and Maxim Likhachev. D* Lite. *AAAI-02 Proceedings*, 2002.

[KLF04] Sven Koenig, Maxim Likhachev, and David Furcy. Lifelong Planning A∗. *Artificial Intelligence*, 155(1-2):93–146, May 2004.

[KNK24] Kornél Katona, Husam A. Neamah, and Péter Korondi. Obstacle Avoidance and Path Planning Methods for Autonomous Navigation of Mobile Robot. *Sensors*, 24(11):3573, June 2024.

[KSDS21] Karthik Karur, Nitin Sharma, Chinmay Dharmatti, and Joshua E. Siegel. A Survey of Path Planning Algorithms for Mobile Robots. *Vehicles*, 3(3):448–468, August 2021.

[LD25] Yaping Lu and Chen Da. Global and local path planning of robots combining ACO and dynamic window algorithm. *Scientific Reports*, 15(1):9452, March 2025.

[LLZ+24] Xiaomei Li, Ye Lu, Xiaoyu Zhao, Xiong Deng, and Zhijiang Xie. Path planning for intelligent vehicles based on improved D* Lite. *The Journal of Supercomputing*, 80(1):1294–1330, January 2024.

[LRS21] Philippe Lacomme, Gwénaël Rault, and Marc Sevaux. Integrated decision support system for rich vehicle routing problems. *Expert Systems with Applications*, 178:114998, September 2021.

[MAR+17] Amgad Madkour, Walid G. Aref, Faizan Ur Rehman, Mohamed Abdur Rahman, and Saleh Basalamah. A Survey of Shortest-Path Algorithms, May 2017.

[MPW26] Dmytro Matsypura, Yu Pan, and Hanzhao Wang. Learning Shortest Paths When Data is Scarce, January 2026.

[MSY+25] Lingkai Meng, Yu Shao, Long Yuan, Longbin Lai, Peng Cheng, Xue Li, Wenyuan Yu, Wenjie Zhang, Xuemin Lin, and Jingren Zhou. Revisiting Graph Analytics Benchmark. *Proceedings of the ACM on Management of Data*, 3(3):1–28, June 2025.

[MSZH22] Sebastian Müller, Lennart Schüler, Alraune Zech, and Falk Heße. GSTools v1.3: A toolbox for geostatistical modelling in Python. *Geoscientific Model Development*, 15(7):3161–3182, April 2022.

[Pow02] Daniel J. Power. *Decision Support Systems: Concepts and Resources for Managers*. Quorum Books, Westport, Conn., 1. publ edition, 2002.

[PS07] Daniel J. Power and Ramesh Sharda. Model-driven decision support systems: Concepts and research directions. *Decision Support Systems*, 43(3):1044–1061, April 2007.

[QSW+23] Hongwei Qin, Shiliang Shao, Ting Wang, Xiaotian Yu, Yi Jiang, and Zonghan Cao. Review of Autonomous Path Planning Algorithms for Mobile Robots. *Drones*, 7(3):211, March 2023.

[Rai19] Juste Raimbault. Second-order control of complex systems with correlated synthetic data. *Complex Adaptive Systems Modeling*, 7(1):4, December 2019.

[RGRM24] Mohsen Roytvand Ghiasvand, Donya Rahmani, and Mohammad Moshref-Javadi. Data-driven robust optimization for a multi-trip truck-drone routing problem. *Expert Systems with Applications*, 241:122485, May 2024.

[RMA04] Rubén Ruiz, Concepción Maroto, and Javier Alcaraz. A decision support system for a real vehicle routing problem. *European Journal of Operational Research*, 153(3):593–606, March 2004.

[Ste94] Anthony Stentz. The D* Algorithm for Real-Time Planning of Optimal Traverses. 1994.

[UHV17] Marlin W. Ulmer, Leonard Heilig, and Stefan Voß. On the Value and Challenge of Real-Time Information in Dynamic Dispatching of Service Vehicles. *Business & Information Systems Engineering*, 59(3):161–171, June 2017.

[WLJ+22] Huanwei Wang, Shangjie Lou, Jing Jing, Yisen Wang, Wei Liu, and Tieming Liu. The EBS-A* algorithm: An improved A* algorithm for path planning. *PLOS ONE*, 17(2):e0263841, February 2022.

[WQL+21] Huanwei Wang, Xuyan Qi, Shangjie Lou, Jing Jing, Hongqi He, and Wei Liu. An Efficient and Robust Improved A* Algorithm for Path Planning. *Symmetry*, 13(11):2213, November 2021.

[WWW+25] Xuan Wu, Di Wang, Lijie Wen, Yubin Xiao, Chunguo Wu, Yuesong Wu, Chaoyu Yu, Douglas L. Maskell, and You Zhou. Neural Combinatorial Optimization Algorithms for Solving Vehicle Routing Problems: A Comprehensive Survey with Perspectives, April 2025.

[XXG+25] Longyan Xu, Mao Xi, Ren Gao, Ziheng Ye, and Zaihan He. Dynamic path planning of UAV with least inflection point based on adaptive neighborhood A* algorithm and multi-strategy fusion. *Scientific Reports*, 15(1):8563, March 2025.

[YAB23] Vincent F. Yu, Pham Tuan Anh, and Roberto Baldacci. A robust optimization approach for the vehicle routing problem with cross-docking under demand uncertainty. *Transportation Research Part E: Logistics and Transportation Review*, 173:103106, May 2023.

[YY98] Gang Yu and Jian Yang. On the Robust Shortest Path Problem. *Computers & Operations Research*, 25(6):457–468, June 1998.

[ZLS+22] Yunfan Zhang, Feng Liu, Yifan Su, Yue Chen, Zhaojian Wang, and Joao P. S. Catalao. Two-Stage Robust Optimization Under Decision Dependent Uncertainty. *IEEE/CAA Journal of Automatica Sinica*, 9(7):1295–1306, July 2022.

[ZM18] Mohd. Nayab Zafar and J.C. Mohanta. Methodology for Path Planning and Optimization of Mobile Robots: A Review. *Procedia Computer Science*, 133:141–152, 2018.

[ZZL23] Aobei Zhang, Ying Zhang, and Yanqiu Liu. Robust Optimization of Electric Vehicle Paths in Uncertain Environments. *IEEE Access*, 11:124240–124251, 2023.

CHAPTER 6

APPENDIX

6-1 Geostatistical Formulations

$$\gamma(r) = \sigma^2 \left(1 - \mathrm{cor}\left(s \cdot \frac{r}{\ell}\right)\right) + n, \tag{6.1}$$

$$C(r) = \sigma^2 \ \mathrm{cor}\left(s \cdot \frac{r}{\ell}\right), \tag{6.2}$$

$$\rho(r) = \mathrm{cor}\left(s \cdot \frac{r}{\ell}\right). \tag{6.3}$$

$$h = \sqrt{\sum_{i=1}^{d} \left(\frac{r_i}{\ell_i}\right)^2}, \tag{6.4}$$

$$h(x, y, \tau) = \sqrt{\left(\frac{\Delta p}{\ell_p}\right)^2 + \left(\frac{\Delta q}{\ell_q}\right)^2 + \left(\frac{\Delta \tau}{\ell_t}\right)^2}. \tag{6.5}$$

$$C_m(\mathbf{r}, \Delta\tau) = C\left(\sqrt{\sum_{i=1}^{2} \left(\frac{r_i}{\ell_i}\right)^2 + \left(\frac{\Delta \tau}{\ell_t}\right)^2}\right), \tag{6.6}$$

where $\mathbf{r} = (x - x', y - y')$ and $\Delta\tau = \tau - \tau'$.

$$U(\mathbf{x}) = \sqrt{\frac{\sigma^2}{N}} \sum_{i=1}^{N} \left(Z_{1,i} \cos(\mathbf{k}_i \cdot \mathbf{x}) + Z_{2,i} \sin(\mathbf{k}_i \cdot \mathbf{x})\right), \tag{6.7}$$

where $Z_{1,i}, Z_{2,i} \sim \mathcal{N}(0, 1)$ are independent standard normal variables and $\mathbf{k}_i \in \mathbb{R}^3$ are wave vectors sampled from the spectral density associated with the covariance model 6.2.

Table 6.1: *Predefined covariance models in GSTools [MSZH22].*

Model	**cor(h)**	**Source**
Gaussian	$\exp(-h^2)$	Webster and Oliver (2007)
Exponential	$\exp(-h)$	Webster and Oliver (2007)
Stable	$\exp(-h^\alpha)$	Wackernagel (2003)
Matérn	$\frac{2^{1-\nu}}{\Gamma(\nu)}\,(\sqrt{\nu}\,h)^\nu \cdot K_\nu(\sqrt{\nu}\,h)$	Rasmussen and Williams (2005)
Rational	$\left(1+\frac{h^2}{\alpha}\right)^{-\alpha}$	Rasmussen and Williams (2005)
Cubic	$\left(1-7h^2+\frac{35}{4}h^3-\frac{7}{2}h^5+\frac{3}{4}h^7\right) \quad (h<1)$	Chilès and Delfiner (2012)
Linear	$(1-h) \quad (h<1)$	Webster and Oliver (2007)
Circular	$\frac{2}{\pi}\left(\cos^{-1}(h)-h\sqrt{1-h^2}\right) \quad (h<1)$	Webster and Oliver (2007)
Spherical	$\left(1-\frac{3}{2}h+\frac{1}{2}h^3\right) \quad (h<1)$	Webster and Oliver (2007)
HyperSpherical	$\left(1-h\cdot\frac{{}_2F_1\left(\frac{1}{2},-\frac{d-1}{2},\frac{3}{2},h^2\right)}{{}_2F_1\left(\frac{1}{2},-\frac{d-1}{2},\frac{3}{2},1\right)}\right) \quad (h<1)$	Matérn (1960)
SuperSpherical	$\left(1-h\cdot\frac{{}_2F_1\left(\frac{1}{2},-\nu,\frac{3}{2},h^2\right)}{{}_2F_1\left(\frac{1}{2},-\nu,\frac{3}{2},1\right)}\right) \quad (h<1)$	Matérn (1960)
JBessel	$\Gamma(\nu+1)\cdot\left(\frac{h}{2}\right)^{-\nu}\cdot J_\nu(h)$	Chilès and Delfiner (2012)
TPLSimple	$(1-h)^\nu \quad (h<1)$	Wendland (1995)
TPLGaussian	$H\cdot E_{1+H}(h^2)$	Di Federico and Neuman (1997)
TPLExponential	$2H\cdot E_{1+2H}(h)$	Di Federico and Neuman (1997)
TPLStable	$\frac{2H}{\alpha}\cdot E_{1+\frac{2H}{\alpha}}(h^\alpha)$	Müller et al. (2021a)

Formulas including the subscript $(h<1)$ are piecewise-defined functions being constantly zero for $h>1$. Here, h is the non-dimensional distance, d is the dimension, $\Gamma(x)$ is the gamma function, $K_\nu(x)$ is the modified Bessel function of the second kind, $J_\nu(x)$ is the Bessel function of the first kind, ${}_2F_1(a,b,c,x)$ is the ordinary hypergeometric function, and $E_\nu(x)$ is the exponential integral function (Abramowitz et al., 1972). All other variables are shape parameters of the respective models.