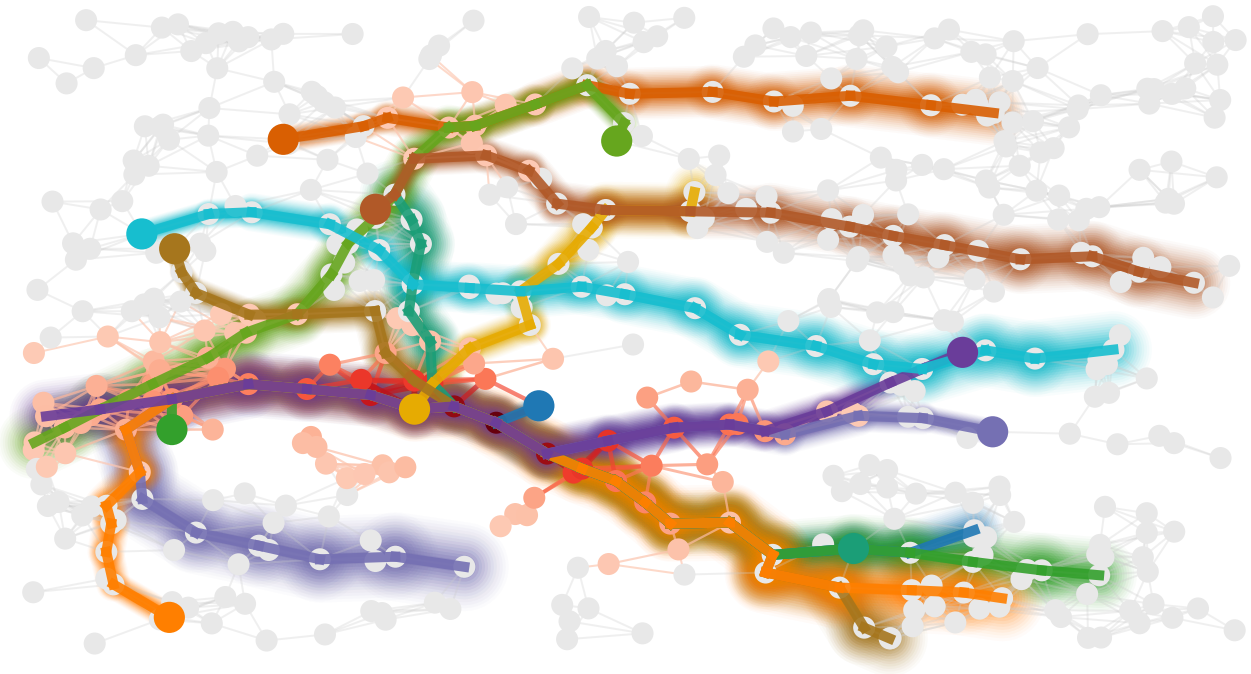

Probabilistically Robust Continuous-time Multi-Agent Path Finding

Dynamic Risk Allocation and Chance-Constrained Planning under Execution Uncertainty

Master's thesis in Systems, Control and Mechatronics

MOA JOHANNESSON
MAJED MAZEN

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Probabilistically Robust Continuous-time Multi-Agent Path Finding

Dynamic Risk Allocation and Chance-Constrained Planning under Execution Uncertainty

MOA JOHANNESSON
MAJED MAZEN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2026

Probabilistically Robust Continuous-time Multi-Agent Path Finding
Dynamic Risk Allocation and Chance-Constrained Planning under Execution Uncertainty
MOA JOHANNESSON
MAJED MAZEN

Supervisor: Alvin Combrink, Department of Electrical Engineering
Sabino Francesco Roselli, Department of Electrical Engineering
Sarmad Riazi, Kollmorgen - Autonomous Mobile Solutions
Examiner: Martin Fabian, Department of Electrical Engineering

Master's Thesis 2026
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The figure illustrates probabilistically robust continuous-time multi-agent pathfinding with Gaussian traversal-time uncertainty. Colored paths represent planned trajectories, while the surrounding variance fields capture accumulated timing uncertainty. Collision risk arises from overlapping time distributions and is highlighted in red. See Chapter 3 for details.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Probabilistically Robust Continuous-time Multi-Agent Path Finding
Dynamic Risk Allocation and Chance-Constrained Planning under Execution Uncertainty
Moa Johannesson
Majed Mazen
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Automated logistics and multi-robot systems use Multi-Agent Path Finding (MAPF) algorithms to coordinate collision-free routes. In real-world industrial deployments, execution uncertainties such as mechanical variations and friction inevitably cause delays. Current robust MAPF algorithms handle this either through conservative worst-case deterministic bounds (in both discrete and continuous time) or through probabilistic models limited to discrete-time grids. This thesis bridges that gap by introducing a probabilistically robust framework for continuous-time MAPF, which is essential for maintaining predictable throughput by enabling proactive scheduling instead of reactive halting.

To accurately model execution uncertainty, the proposed approach uses Gaussian distributions bounded by strict kinematic hardware limits. Instead of bounding maximum delays, the proposed approach models execution uncertainty using chance-constrained programming, evaluating the overlapping probability density of agent trajectories. The framework uses a modified Continuous Conflict-Based Search (CCBS) paired with Safe Interval Path Planning (SIPP), and evaluates several heuristic risk allocation strategies to dynamically distribute a user-defined global risk budget across local interactions.

The framework was evaluated on a standard benchmark environment, assessing metrics such as scalability, computational cost, risk utilization, and plan quality. Furthermore, Monte Carlo simulations were conducted to verify the robustness of the generated plans. The results show that the empirical execution success rate accurately matches the theoretical global risk calculated via the Union Bound. Ultimately, this confirms that the proposed framework successfully guarantees the collision risk of the system to be within the user-defined global risk budget, enabling a practical balance between schedule efficiency and system safety.

Keywords: Multi-Agent Path Finding, Continuous-Time Planning, Probabilistic Robustness, Traversal Time Uncertainty, Chance-Constrained Optimization, Conflict-Based Search, Safe Interval Path Planning.

Acknowledgements

We would like to thank our supervisors Alvin Combrink and Sabino Francesco Roselli this opportunity to work on this topic. A special thanks to Alvin for generously sharing your deep knowledge with us within the area of MAPF. Thank you for always being available and making this thesis so much more interesting than what we ever anticipated it to be. We would also like to thank Sarmad Riazi and Rasmus Åkerlund from Kollmorgen - Autonomous Mobile Solutions for coming with great insight on how to continue developing this thesis from an industrial point of view. Finally, a big thank you to our examiner Martin Fabian for helping us with everything.

Majed Mazen, Moa Johannesson, Gothenburg, May 2026

List of Acronyms

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

CBS	Conflict Based Search
CCBS	Continuous Conflict Based Search
CC-K-CBS	Chance-Constrained Kinodynamic Conflict Based Search
CCP	Chance-Constrained Planning
CDF	Cumulative Distribution Function
CT	Constraint Tree
MAPF	Multi-Agent Path Finding
MAPF_R	Continuous-time Multi-Agent Path Finding
PDF	Probability Density Function
RQ	Research Question
SIPP	Safe Interval Path Planning
CSIPP	Constrained Safe Interval Path Planning
SOC	Sum-of-Costs
MC	Monte Carlo

Nomenclature

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

Indices

i,j	Indices for agents or their corresponding actions
n	Index for final step or length of a single plan

Sets

$\mathcal{G}$	Graph representing a physical environment
$\mathcal{V}$	Set of all vertices within a graph
$\mathcal{E}$	Set of all edges within a graph
$\mathcal{M}$	Metric space through which agents navigate in MAPF_R
$\mathcal{A}$	Set of move actions in MAPF_R
Π	Set of single agent plans called a joint plan
CT	Constraint tree used in the high-level planning of CBS and its variants
I	Geometric unsafe time interval (general)
$I_{m,m'}$	Specific unsafe time interval for a move-move conflict
I_{mw}	Specific unsafe time interval for a move-wait conflict
I_d	Expanded unsafe interval including the safety margin d

Parameters

N	Total number of agents
s	Function mapping an agent to a starting vertex

g	Function mapping an agent to a goal vertex
k	Maximum number of delay time steps in k-Robust MAPF
T	Maximum number of delay time units in T-Robust MAPF$_R$
D	Duration characterizing a specific continuous-time action
p_d	Probability distribution of agent delays in p-Robust MAPF
p	Confidence threshold probability for a collision-free execution in p-Robust MAPF
δ	Maximum risk threshold for the probability of a collision in p-Robust MAPF
l	Distance between two vertices
z	Number of standard deviations
R_{target}	Global risk budget / maximum allowable risk
v_{max}	Maximum achievable kinematic speed of an agent
v_{nom}	Nominal cruising speed of an agent
t_{min}	Minimum physically feasible traversal duration for an edge
w_i	Deterministic waiting time duration for Agent i
M	Total number of Monte Carlo simulation samples

Variables

a	Action that maps current position to new position
a_i	Action for agent i
π	Sequence of actions called single agent plan
π_i	Single agent plan for agent i
v	Current vertex for an agent
v'	Subsequent vertex for an agent
Δ	Amount of time delay experienced by an agent
φ	Motion function that maps execution time to spatial coordinates
$P_{success}(\pi)$	Probability of a collision-free execution given a single agent plan
$P_{collision}$	Probability of a geometric collision between agents
C	Potential conflict based on delay distributions
η	Node in Conflict Based Tree
d	Safety margin for mean shifting
$r_{z^\star}$	Current selected conflict
R_{global}	Accumulated global system risk

D_e	Stochastic traversal duration along edge e
μ_e, σ_e^2	Mean and variance of the traversal duration along edge e
A_i	Stochastic arrival time of Agent i at a vertex v
T_i	Stochastic traversal start time for Agent i
ΔT	Relative timing difference between two agents' interacting action start times
ϵ	Excess risk that violates the global target ($R_{global} - R_{target}$)
d_{tree}	Current constraint tree depth of the high-level search node

Contents

List of Figures

List of Tables

1

Introduction

Multi-Agent Path Finding (MAPF) addresses the fundamental problem of computing collision-free paths for multiple agents moving from their current locations to designated goals. This problem has gained importance across diverse applications, such as warehouse and production automation, airport logistics, drone swarm operations, and video game character navigation [1, 2]. The practical deployment of MAPF faces a critical challenge: real-world execution rarely matches the deterministic assumptions of planning algorithms. Agent speeds vary due to mechanical differences, environmental conditions, or dynamic obstacles, and unexpected events cause temporary slowdowns or accelerations. These uncertainties can lead to deviations from the movement plan, potentially causing collisions.

Classical MAPF algorithms operate in discrete time [3], assuming agents move at regular time steps. This is an artificially imposed constraint that can lead to poor solutions. For instance, in grid-based environments where agents move with constant speed, discretization forces agent trajectories into rigid spatial states, preventing the planner from exploiting the full freedom of the physical workspace. To address the limitations of discretization, continuous-time algorithms have emerged [2, 4]. This gives solutions that are at least as good or better than those produced by discrete solvers. This is because continuous time is a relaxation of the problem from discrete time. Therefore, the solution quality has increased.

This thesis work, conducted at the Department of Electrical Engineering at Chalmers University of Technology in collaboration with Kollmorgen - Autonomous Mobile Solutions, develops probabilistic robustness techniques for continuous-time MAPF (MAPF_R) that incorporate traversal duration uncertainty. Rather than planning for worst-case delays, each agent movement is modeled by a probability distribution, allowing plans to be optimized for specified confidence levels while retaining the advantages of continuous-time planning.

The practical value of developing probabilistically robust continuous-time algorithms extends far beyond theoretical computer science; it directly impacts the efficiency and viability of modern automated logistics as mentioned previously. In real-world industrial environments, autonomous agents are equipped with low-level reactive safety systems, such as LiDAR scanners and physical bumpers, which ensure that

agents will not physically collide even if the high-level routing plan fails. However, relying on these low-level systems to resolve spatial conflicts is highly detrimental to system performance. When an agent unexpectedly halts to avoid a collision, it induces a localized bottleneck that can easily propagate into cascading delays or catastrophic deadlocks across the entire fleet.

By integrating probabilistic robustness into the high-level planning phase, the algorithm proactively accounts for stochastic delays, drastically reducing the frequency at which these low-level emergency systems are triggered. For industrial practitioners, this shift from reactive halting to proactive scheduling is critical. It guarantees a fluid, uninterrupted traffic flow, which yields highly predictable throughput, a metric of paramount importance in manufacturing and supply chain logistics. Furthermore, minimizing erratic stop-and-go behaviors reduces the mechanical wear and tear on the Autonomous Guided Vehicles (AGVs) and lowers overall energy consumption. On a broader societal scale, optimizing the efficiency and predictability of these massive autonomous networks reduces the carbon footprint of supply chain operations and lowers the logistical overhead embedded in the cost of consumer goods.

1.1 Aim

The primary aim of this thesis is to develop and evaluate a probabilistically robust planning framework for MAPF$_R$. The objective is to generate collision-free plans that satisfy a user-defined confidence level under stochastic traversal duration uncertainty. The work focuses on extending existing MAPF$_R$ formulations to account for execution variability while maintaining solution quality and computational tractability.

The project is carried out in collaboration with Kollmorgen - Autonomous Mobile Solutions, and aims to formulate a mathematical definition for probabilistic robustness in continuous time, develop an algorithm that can verify probabilistic constraints during the search process, and evaluate the trade-off between plan efficiency and robustness using simulation benchmarks.

1.2 Limitations

This thesis is subject to the following limitations:

- The study assumes centralized planning with perfect communication between agents. This is to isolate the planning problem itself and evaluate the proposed framework under complete system information. The work therefore does not address communication delays, data loss, or decentralized coordination.

- The uncertainty model is limited to Gaussian traversal duration distributions with a 3σ kinematic limitation. This choice was made because it allows for a simple mathematical formulation while still respecting physical speed limits,

and because no empirical data from industrial systems were available to support a more realistic uncertainty model. The framework, therefore, does not capture all forms of execution uncertainty encountered in industrial practice.

- The proposed framework was evaluated exclusively in simulation, since the thesis focused on algorithmic development and validation in a controlled environment. No hardware experiments were conducted. The reported results, therefore, reflect the adopted simulation model rather than full physical deployment conditions.

- The empirical evaluation was limited to a single benchmark setting, the obstacle-free empty-16-16 map [5]. This was to establish an initial validation on a standard benchmark environment. This restricts how strongly the results generalize to denser and more constrained environments.

1.3 Research Questions

The main research questions are:

- **RQ1:** How can traversal duration uncertainty be formally modeled within a MAPF$_R$ framework to ensure collision-free movement with a given confidence level?

- **RQ2:** How can solutions with a given confidence level be found for the MAPF$_R$ problem?

1.4 Methodology

This thesis uses a constructive methodology to design, implement, and evaluate a probabilistically robust solution for continuous-time Multi-Agent Path Finding MAPF$_R$ under uncertainty. The work combines theoretical analysis, algorithm design, implementation, and empirical evaluation. The overall approach is exploratory and iterative, with continuous interaction between literature review, problem formulation, probabilistic modeling, and algorithm design.

1.4.1 Research Approach and Problem Development

An extensive literature review was conducted to establish the theoretical foundation of the work and guide all major design decisions. The review covered classical MAPF, continuous-time extensions, robustness formulations, and formulations under execution uncertainty. The findings are detailed in Chapter 2.

A key observation was that all robustness formulations identified in the literature, such as k-robustness, T-robustness, and p-robustness are developed as extensions of Conflict-Based Search (CBS). This motivated the choice to base the work on a

CBS-derived framework, specifically Continuous Conflict-Based Search (CCBS), in order to remain aligned with previous work and enable meaningful comparisons in the future.

The early and intermediate phases of the project were highly iterative. Insights from the literature repeatedly led to reformulations of the problem and revisions of the probabilistic modeling assumptions. In this sense, the literature review was not a separate preparatory step but an active part of the development process. Problem formulation, modeling, and algorithm design evolved together as understanding of the limitations of existing approaches increased.

1.4.2 Algorithm Design and Modular Implementation

The algorithmic work focused on extending CCBS to incorporate probabilistic execution uncertainty. Existing implementations served as important reference points, including a C++ implementation of CCBS with Combrink's δ-branching extension and a Python implementation of Continuous Prioritized Lifelong Planning (CPLP) [6]. The final system was implemented in Python to enable rapid prototyping, flexible experimentation, and easier integration of probabilistic components.

The design process was highly exploratory. Several alternative directions were investigated before the final formulation was selected. These included extensions to the branching strategy, such as three-way branching, as well as formulations based on convex optimization and Approximate Bayesian Computation (ABC), described in Section **??**. These approaches were explored in an attempt to maintain a global notion of probabilistic robustness while preserving desirable algorithmic properties of CCBS. However, they were found to be either computationally impractical or incompatible with the structure of the algorithm, which motivated a shift in focus.

An early focus of the work, after deciding to modify CCBS, was to investigate whether probabilistic robustness could be incorporated while retaining the properties that make CCBS attractive. In particular, the work explored whether a global user-defined confidence level could be combined with the optimality guarantees of CCBS. Since no formulation was found that achieved both, the focus shifted toward constraining individual interactions between agents. This led to a formulation based on mean-shifting and explicit risk allocation, where local constraints are used to control risk within the planning process. The detailed modeling choices are presented in Chapter 3, and the resulting algorithmic framework is described in Chapter 4.

The implementation was developed using a modular architecture. This design was chosen for three reasons. First, it made it possible to update individual components throughout the project without requiring large changes to the entire system. Second, it allowed different parts of the implementation to be developed in parallel. Third, it increases the future usefulness of the work since individual components can be reused or extended independently.

The modular structure separates central functionalities such as environment generation, preprocessing, path planning, probabilistic modeling, risk evaluation, simulation, and result analysis. In parallel with the conceptual exploration, the implementation itself also went through multiple iterations aimed at improving computational performance. Data structures and internal processing strategies were repeatedly revised to reduce computational overhead and improve scalability. A detailed description of the software architecture and individual modules is provided in Section 4.3.

1.4.3 Evaluation and Validation

The developed approach was evaluated through a combination of incremental validation and simulation-based experiments. During the development process, individual components were first tested separately in order to verify that their behavior matched the intended formulation. In particular, the local risk model was initially tested on a small example graph after the local risk formulation had been established. This was done to validate the probabilistic interaction model before it was integrated into the full planning framework.

After the full algorithmic formulation had been established, the approach was evaluated through simulation using an established MAPF benchmark map. The evaluation focused on three aspects: solution quality, robustness, and computational efficiency. Robustness was measured as the probability of successful execution under the assumed uncertainty model, while computational efficiency was assessed through runtime and scalability. Monte Carlo simulation was used to estimate execution outcomes and validate the probabilistic behavior of the generated plans.

The purpose of the evaluation was to determine whether the proposed formulation provides a practical balance between robustness and computational performance, and to study how the different risk allocation strategies affect the resulting plans. A detailed description of the experimental setup, benchmark selection, and parameter settings is provided in Chapter 5.

1.4.4 Use of Generative AI

In accordance with Chalmers University guidelines, generative AI tools (NotebookLM, Gemini, and GitHub Copilot) were used to support the thesis development. Specifically, AI assisted in discovering relevant literature, evaluating the proposed mathematical models for potential weaknesses, and refining the language and structure of this report. For the algorithmic implementation, we (the authors) designed the core system architecture independently, while using AI in an iterative workflow for debugging, syntax generation, and translating concepts from existing C++ and Python frameworks.

All AI-generated output was critically reviewed, tested, and modified by us to ensure correctness and avoid bias or plagiarism. We retain full responsibility for the final system architecture, mathematical proofs, code functionality, and conclusions.

Furthermore, strict data integrity was maintained. No confidential information belonging to Kollmorgen - Autonomous Mobile Solutions was processed through these external platforms.

1.5 Contributions

The primary contributions of this thesis are as follows:

- **A formulation of a probabilistic continuous-time MAPF model** based on Gaussian traversal duration uncertainties, together with a 3σ kinematic limitation to ensure physically realistic traversal durations.
- **A derivation of an analytical local risk model** based on the Cumulative Distribution Function (CDF) for evaluating the probability of collision in continuous-time move-move and move-wait interactions.
- **A probabilistically robust extension to Continuous Conflict-Based Search (CCBS)** that dynamically expands unsafe time intervals (mean-shifting) to enforce statistical safety thresholds during the low-level SIPP search.
- **The design and implementation of multiple risk allocation strategies**, each distributing a global Union Bound risk budget across local conflicts.
- **An evaluation of the proposed algorithms** in simulated benchmark scenarios, highlighting the trade-offs between robustness, solution quality, and computational efficiency.

1.6 Thesis Outline

This thesis is divided into six chapters. The remainder of this thesis is structured as follows:

Chapter 2 establishes the theoretical foundation, reviewing continuous-time MAPF, Conflict-Based Search algorithms, and existing approaches to optimization under uncertainty to give an understanding for the need of a probabilistically robust framework.

Chapter 3 formalizes the problem environment and derives the mathematical models, specifically detailing how traversal duration uncertainty is quantified and how the local risk model evaluates potential conflicts.

Chapter 4 translates these mathematical models into algorithmic design, outlining the necessary modifications to the CCBS framework and introducing various strategies for allocating the global risk budget among agents.

Chapter 5 presents the experimental evaluation, analyzing the performance, optimality, and robustness of the proposed algorithms through simulated benchmark scenarios.

Chapter 6 discusses the implications of the results, concludes the thesis, and outlines potential directions for future research.

2

Theoretical Background

Multi-Agent Path Finding (MAPF) is the problem of planning paths for multiple agents such that each agent moves from its start position to its goal position without collisions. To understand the contribution of this work, it is necessary to review the established MAPF formulations, solution methods, and robustness concepts on which the thesis builds.

For several graph-based MAPF formulations, computing solutions that are optimal with respect to objectives such as makespan, total arrival time, and total distance is NP-hard [7], highlighting that MAPF is already challenging in deterministic settings.

Extending MAPF from discrete to continuous time adds complexity, since agents must be modeled as volumes moving along continuous trajectories rather than as point agents occupying vertices at discrete time steps [4]. When probabilistic execution uncertainty is also taken into account, the planning problem becomes more difficult, as the solver must reason about both continuous temporal variables and collision risk. Although the computational complexity of general MAPF$_R$ has not been established here, these challenges motivate the use of structured search methods together with heuristic or approximate risk-allocation strategies.

Optimal solutions can be found using *Conflict Based Search* (CBS) [8] in discrete time and *Continuous-time CBS* (CCBS) [9] in continuous time. However, real-world execution is rarely perfect. This has led to the development of robust formulations: *p-robustness* (probabilistic) for discrete time [10] and *T-robustness* (time-bounded) for continuous time [2].

Currently, there exists a gap in the literature: while probabilistic guarantees exist for discrete solvers, no such established formulation exists for continuous time. This section provides background necessary to bridge that gap.

Section 2.1 defines the classical Multi-Agent Path Finding with its grid based planning and standard CBS algorithm. Section 2.2 replaces discrete steps with physical volumes and continuous time using CCBS and SIPP. Section 2.3 explains traditional worst-case safety buffers. Section 2.4 explains the mathematical logic of risk. Finally, Section 2.5 presents the concept of probabilistic MAPF$_R$.

2.1 Classical Multi-Agent Path Finding

The *classical MAPF* problem is defined in [11] with the assumptions that time is discrete and that agents are considered as points (not volumes). The input is defined by a tuple $\langle \mathcal{G}, s, g \rangle$ where (with N the number of agents):

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph with vertices $\mathcal{V}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$,
- $s : [1, \ldots, N] \to \mathcal{V}$ assigns a starting vertex to an agent, and
- $g : [1, \ldots, N] \to \mathcal{V}$ assigns a goal vertex to an agent.

Each agent is located at one of the vertices of the graph in each time step and has the ability to perform an action $a : \mathcal{V} \to \mathcal{V}$. An action a is a function that maps an agent's current vertex v to a new vertex v' in the next time step, such that $a(v) = v'$. There are two types of actions: *wait* and *move*. When an agent performs the *wait* action, it remains in its current vertex for an additional time step, such that $v = v'$. If an agent performs the *move* action, it moves from the current vertex v to an adjacent vertex v' in the next time step. In other words, it traverses an edge $(v, v') \in \mathcal{E}$.

A sequence of actions is called a *plan* and denoted π. For a plan to be valid, each action must start where the previous action ends, so that the actions can be executed sequentially from the agent's start vertex. All agents $i \in \{1, \ldots, N\}$ perform an individual plan $\pi_i = (a_1, a_2, \ldots, a_n)$ called a *single-agent plan*, with the length $|\pi_i| = n$ that can vary between agents and solutions. A solution is a *joint plan* $\Pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$ that contains a single-agent plan for each agent. The aim is to find a joint plan that avoids collisions between agents.

Conflicts describe situations that lead to collisions during execution. Two common types of conflicts that can occur are *edge conflicts* and *vertex conflicts*. An edge conflict arises when two agents traverse the same edge in the same time step, and a vertex conflict arises when two agents are planned to occupy the same vertex in the same time step. Some additional conflicts are mentioned in [11].

2.1.1 Conflict Based Search

Conflict Based Search (CBS) [8] is a widely used MAPF solver that consists of two levels of search. The high level of CBS searches a binary constraint tree (CT). Each node $\eta \in CT$ contains: **(1)** a set of all constraints in $\eta.constraints$ that are applied to each and every agent; **(2)** a single solution ($\eta.\pi$) that is *consistent* with all constraints ($\eta.constraints$); and **(3)** the total cost of the solution ($\eta.cost$). The high level performs a best-first search on the CT where nodes are ordered by their costs.

To generate a node in the CT, a given node η finds a shortest single-agent plan for

each agent from its start location to its goal location. This is done by the low level of CBS using A* [12] or Dijkstra's algorithm [13], and the single-agent plan must satisfy all constraints of node η imposed on the agent. The last step is to expand a node in the CT. Once CBS has chosen node η for expansion, it checks the solution $\eta.\pi$ for conflicts. Node η is a goal node if there are no conflicts and CBS returns its solution. If not, CBS splits node η in conflicts $\langle i, j, x, t\rangle$ by branching η into two children that inherit all parent constraints $\eta.constraints$ and a new constraint. The new constraint will either be $\langle i, x, t\rangle$ or $\langle j, x, t\rangle$ with x representing vertex v or edge e, and i, j defining the respective agent.

2.2 Continuous-time Multi-Agent Path Finding

The $MAPF_R$ problem defined by [4] extends the classical formulation of MAPF by replacing discrete time steps with a continuous timeline and modeling agents as arbitrary geometric shapes moving through a metric space $\mathcal{M}$. The problem input is extended with a set of *move actions* $\mathcal{A}$, where each action defines a specific trajectory between two vertices. Hence, MAPF_R is defined by the tuple $\langle \mathcal{G}, \mathcal{M}, s, g, \mathcal{A}\rangle$.

In classical grid-based MAPF, the discrete nature of the environment implicitly enforces a safe separation distance, as two agents cannot occupy the same cell simultaneously. However, in continuous space and time, representing agents as mathematical points would theoretically allow them to pass arbitrarily close to one another without triggering a spatial conflict. A way to avoid the problems is to give the agents volume, and model them as arbitrary geometric shapes moving through a metric space $\mathcal{M}$.

Unlike discrete steps, actions in MAPF_R have arbitrary durations. An action is defined as a tuple $a = \langle \varphi, D\rangle$, where D is the duration and $\varphi : [0, D] \to \mathcal{M}$ is a *motion function* mapping time to coordinates. As in classical MAPF, there are two types of actions: wait and move. Move actions define a trajectory (which may have an arbitrary shape) along an edge $(v, v') \in \mathcal{E}$ with potentially non-constant speeds. Wait actions keep an agent stationary at a vertex. Notably, the set of wait actions is infinite, as an agent may wait for any arbitrary duration.

A single-agent plan π_i is a sequence of m timed actions, denoted as $\pi_i = (\langle a_k, t_k\rangle)_{k=1}^{m}$, where each action must begin exactly where and when the previous action ended. Consequently, collisions are not defined by discrete graph vertices, but by continuous geometric overlap. A collision occurs if the volumetric shapes of two agents intersect in the metric space $\mathcal{M}$ at any time t during execution.

2.2.1 Classification of Conflicts in Continuous-time

In classical discrete-time MAPF, conflicts are typically categorized as either vertex conflicts or edge conflicts. However, in continuous-time $MAPF_R$, agents possess physical volumes, and actions have arbitrary durations. Therefore, a conflict is formally defined as occurring when, during the execution of their respective timed

actions, the intersection between two agents' physical volumes becomes non-zero. Because conflicts are more accurately classified by the kinematic state of the agents during this spatial overlap, we classify all potential collisions into two fundamental categories:

- **Move-Move Conflict:** This occurs when two agents simultaneously execute move actions, and their continuous spatial trajectories geometrically intersect during an overlapping time interval. Both agents are in motion at the time of the collision.

- **Move-Wait Conflict:** This arises when one agent is executing a wait action (remaining stationary at a specific vertex for a duration of time), while a second agent executes a move action that geometrically intersects the space occupied by the waiting agent during that same time period.

A third intuitive category, a *wait-wait* conflict, is where two stationary agents collide. The algorithm chooses not to consider this, and to understand why, we must consider the temporal sequence of actions.

By definition, a valid MAPF problem assumes that the initial start configurations of all agents are completely collision-free. Therefore, a wait-wait conflict cannot exist at $t = 0$. For a wait-wait conflict to occur at any time $t > 0$, two agents must simultaneously occupy overlapping spatial regions while both are executing wait actions. However, to transition from a collision-free initial state into this overlapping state, at least one of the agents must have executed a move action to arrive at the shared location.

If one agent's planned movement intersects with a location where another agent is already scheduled to wait, the timeline will contain a move-wait conflict. Alternatively, if both agents are planned to travel toward the shared location at the same time, the timeline will contain a move-move conflict. Because any generated plan that results in a wait-wait overlap inherently contains an earlier move-move or move-wait conflict, checking for wait-wait overlaps is redundant. By generating full plans and thereafter detecting collisions, the planner naturally eliminates the conditions that lead to a wait-wait state. Consequently, the planner only needs to detect and resolve move-move and move-wait interactions to guarantee safe paths.

2.2.2 Geometric Collision Detection

In continuous-time MAPF_R, determining exact unsafe time intervals requires calculating precisely when the physical volumes of two agents overlap in the metric space $\mathcal{M}$. The computational approach to this problem depends heavily on the kinematic complexity of the modeled agents.

To achieve high computational efficiency during the search process, solvers often rely on kinematic simplifications. A widely adopted simplifying assumption within

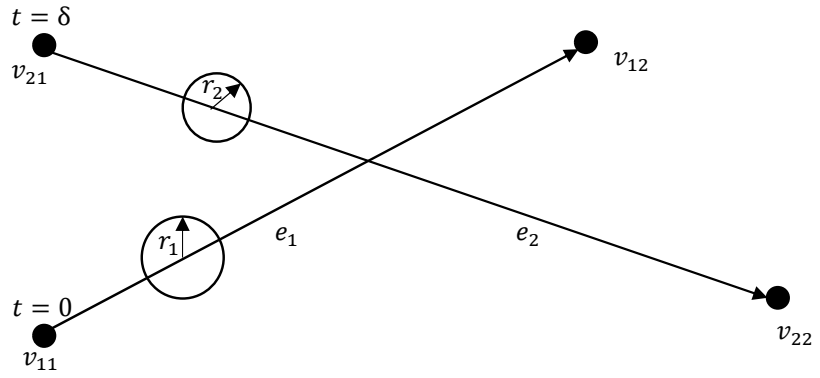

Figure 2.1: Two agents traversing $e_1 = (v_{11}, v_{12})$ at time $t = 0$ and $e_2 = (v_{21}, v_{22})$ at time $t = \delta$, respectively. Both agents are circular, with the respective radii r_1 and r_2. Adapted from Combrink [6]

However, the generalized MAPF$_R$ framework also accommodates real-world applications involving agents with complex, irregular footprints (e.g., autonomous forklifts) executing non-linear trajectories with variable acceleration. Under such general conditions, analytical closed-form equations are typically intractable. Instead, the system must rely on generalized computational geometric algorithms to detect continuous spatial overlaps [16]. Regardless of the underlying kinematic complexity, the fundamental objective of the collision detection process remains the same: identifying the precise interval I during which a spatial intersection occurs.

2.2.3 Continuous Conflict Based Search

To address the limitations of discrete time steps, *Continuous-time Conflict Based Search* (CCBS) [9] extends the CBS framework to support actions of arbitrary duration in a continuous timeline and to solve the MAPF$_R$ problem. In this setting, the low-level search typically uses *Safe Interval Path Planning* (SIPP) [17] to find paths that avoid forbidden time intervals rather than specific discrete time points. Conflicts are redefined as overlaps in the time intervals during which two agents occupy the same spatial region ($\mathcal{V}$ or $\mathcal{E}$), and constraints are represented as time ranges $[t_{start}, t_{end}]$ during which an agent is prohibited from a certain location. This is also a limitation for CCBS because it struggles in sparse environments. This is because a limited number of edges forces agents to plan on using the same routes, which generates a high number of conflicts. When agents lack alternative paths to

choose from to avoid these collisions, resolving the sheer volume of conflicts makes the problem much harder for the algorithm [4].

While CCBS has become a standard for MAPF_R, recent research has identified theoretical flaws in the original algorithm's soundness and completeness guarantees, meaning that it does not guarantee to find a solution if one exists, or return an optimal solution [18, 19]. To resolve this, a new version of CCBS was proposed featuring a novel branching rule (δ-BR) that guarantees to find an optimal solution within a finite time if one exists [19].

2.2.4 Safe Interval Path Planning

Safe Interval Path Planning (SIPP) [17] is a continuous-time path finding algorithm used in MAPF. It is an algorithm for finding a path for a single agent, while assuming all other agents are dynamic obstacles with a predetermined trajectory. Instead of searching through discrete time steps, it compresses continuous time into contiguous "safe intervals." This allows the algorithm to find collision-free paths for agents while avoiding the infinite search space that would otherwise result from continuous-time variables.

Originally, SIPP uses the trajectories of the dynamic obstacles to compute the safe intervals of the graph vertices. Since continuous-time uses time intervals as constraints rather than dynamic obstacles, SIPP has been modified by Andreychuk et al. [4] to a version called Constrained Safe Interval Path Planning (CSIPP).

CSIPP computes safe intervals from constraints given by CCBS. Because CSIPP operates as an optimal A^*-based search [12], its fundamental behavior when encountering any constraint is to minimize the overall sum-of-costs. The algorithm continuously evaluates whether it is cheaper to wait for an unsafe time interval to pass, or to route the agent through a faster, alternative path in the graph. Figure 2.2 demonstrates how CSIPP handles an unsafe interval I. The specific mechanics of how CCBS constraints are applied depend on whether they restrict a vertex or an edge.

2.2.4.1 Handling Vertex Constraints

When CCBS restricts an action for agent i from occupying a vertex by saying that it cannot occupy a vertex during a specific interval, it generates a constraint over a specific interval $[t_{start}, t_{end})$. In CSIPP, this single constraint splits the timeline for that vertex into two distinct safe intervals: one ending at t_{start} and another beginning at t_{end}.

If multiple constraints are imposed on the same vertex, the continuous timeline is segmented accordingly. For example, if two constraints $[t_1, t_1^u)$ and $[t_2, t_2^u)$ are applied to a vertex v such that $t_1 < t_1^u < t_2 < t_2^u$, the resulting safe intervals for v are computed as:

$$[0, t_1], \quad [t_1^u, t_2], \quad \text{and} \quad [t_2^u, \infty) \tag{2.1}$$

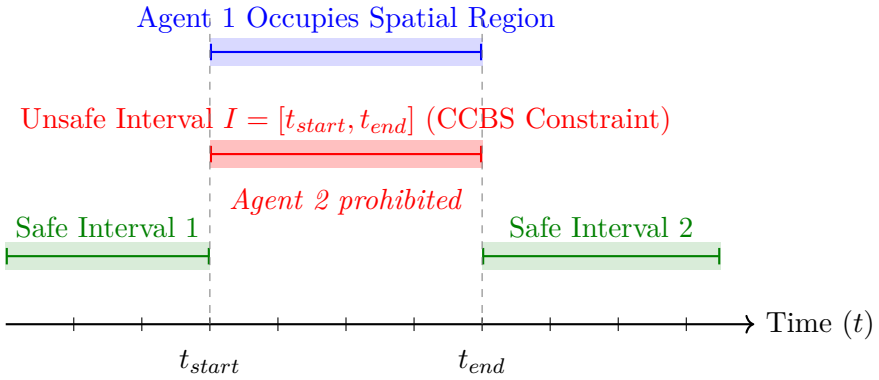

Figure 2.2: Timeline visualization of a CCBS constraint. Agent 1's occupancy of a spatial region creates an unsafe interval I (red). The low-level CSIPP algorithm inverts this high-level constraint to generate valid safe intervals (green) during which other agents are permitted to enter the region.

This allows CSIPP to plan around the forbidden intervals while maintaining a continuous timeline.

2.2.4.2 Handling Edge Constraints

When a CCBS constraint restricts a move action a_i — traversing an edge (v, v') — during an interval $[t, t^u)$, CSIPP does not slice the timeline at the starting vertex v. Instead, it modifies the execution of the action itself.

If the agent reaches vertex v but attempting to cross the edge would violate the constraint, CSIPP forces the agent to wait safely at vertex v if it cannot find another path that is faster. The agent waits for a duration of $t^u - t$ before initiating the spatial movement across the edge.

2.3 Robustness in Multi-Agent Path Finding

Robustness in MAPF refers to a solution's resilience against execution delays to avoid collisions. While traditional MAPF assumes deterministic execution, robust formulations ensure plans remain collision-free even when agents experience unexpected timing variations.

In the discrete-time formulation, a solution is defined as *k-robust* [20] if it can tolerate up to k time steps of delay for any subset of agents without resulting in a collision. Formally, a plan π is k-robust if it contains no k-delay conflicts. A k-delay conflict $\langle i, j, t \rangle$ occurs if there exists some $\Delta \in \{0, \ldots, k\}$ such that:

$$\pi_i(t) = \pi_j(t + \Delta) \tag{2.2}$$

This implies that agent j cannot occupy the same vertex v, nor traverse the same edge, that agent i occupied within the previous k time steps

For continuous-time scenarios, this concept is extended to *T-robustness* [2]. A continuous-time plan is *T-robust* if it remains collision-free under any series of delays where each agent's total delay does not exceed T time units. This is equivalent to ensuring no T-delay conflicts exist. A T-delay conflict occurs between two actions (i, t_i) and (j, t_j) if there exists a delay $\Delta \in [0, T]$ such that the agents collide when agent j executes its action at $t_j + \Delta$.

2.4 Optimization Under Uncertainty

As established in previous sections, transitioning from theoretical MAPF to physical deployments introduces inevitable uncertainty. To systematically resolve this uncertainty without defaulting to the rigid assumptions of k-robustness, the problem must be formulated within the broader mathematical domain of optimization under uncertainty. This domain is defined by two contrasting paradigms: Robust Optimization and Chance-Constrained Programming.

2.4.1 Robust Optimization (Worst-Case Planning)

Robust optimization approaches uncertainty deterministically. Instead of modeling the exact probability distribution of a deviation from the plan timing, it defines an uncertainty set $\mathcal{U}$. This is a bounded mathematical region containing all possible disturbances [21, 22].

The objective of a robust solver is to find a decision variable x that minimizes a cost function while remaining strictly feasible under the absolute worst-case scenario within that defined set. Formally, a generic robust optimization constraint is written as:

$$g(x, \zeta) \leq 0 \quad \forall \zeta \in \mathcal{U} \tag{2.3}$$

where ζ represents the uncertain parameters of all the possible values of $\mathcal{U}$, and g is the robustness constraint for a global robust optimization problem.

In the context of continuous-time MAPF, robust optimization equates to planning for the maximum possible delay at every edge. While this provides an absolute safety guarantee (such as the k-robustness or T-robustness paradigms), it is structurally blind to the statistical likelihood of those delays occurring.

2.4.2 Chance-Constrained Programming

To resolve the conservatism of worst-case planning, Stochastic Optimization treats the uncertain parameters not as bounded sets, but as random variables following specific probability distributions. Within this domain, the framework directly applicable to p-Robust MAPF$_R$ is *Chance-Constrained Programming*, originally introduced by Charnes and Cooper [23].

A chance constraint does not demand absolute feasibility under all possible, microscopic disturbances. Instead, it relaxes the worst-case requirement by allowing

the constraints to be violated with a small, mathematically bounded probability $\alpha \in (0, 1)$. The general mathematical formulation for a chance constraint is:

$$\mathbb{P}(g(x, \xi) \leq 0) \geq 1 - \alpha \tag{2.4}$$

where ξ is the vector of random variables, and $1 - \alpha$ represents the required safety confidence level.

By evaluating the probability density of the overlapping trajectories, the planner can precisely quantify the local risk of each interaction. Instead of asking if a collision is physically possible under extreme delay, the chance-constrained system asks if the probability of a collision is below the strict threshold α. By accepting a user-defined probability of failure, chance-constrained programming allows the planner to trim unnecessary waiting times. This preserves the optimality of the multi-agent schedule while maintaining rigorous, statistical safety guarantees [24].

2.5 Probabilistic Planning

While T-robustness provides strict safety in continuous domains, its worst-case assumptions heavily degrade efficiency. To apply the principles of Chance-Constrained Programming to multi-agent path finding, the paradigm must shift from deterministic bounds to probabilistic thresholds.

Previous research has successfully integrated chance constraints into MAPF environments. Algorithms such as CC-k-CBS [25] evaluate execution uncertainty probabilistically, ensuring the risk of a collision remains below a set threshold. However, these existing probabilistic frameworks are inherently bound to discrete time steps and grid-based spatial graphs, limiting their applicability to continuous, kinematically accurate trajectories.

To extend chance-constrained programming into the continuous-time domain, safety must be formulated as a statistical constraint rather than a temporal bound. A continuous-time probabilistically robust plan requires that the probability of a completely collision-free execution, $P_{success}(\pi)$, satisfies a user-defined confidence threshold $p \in [0, 1]$, meaning between 0-100%, and is defined as:

$$P_{success}(\pi) \geq p. \tag{2.5}$$

Unlike deterministic continuous robustness, this probabilistic formulation does not require the system to survive the maximum possible delay $\Delta \in [0, T]$. Instead, the system is permitted to accept routing solutions where temporal bounding distributions physically overlap, provided that the integrated mathematical risk of a collision remains below $1 - p$. By capping the required safety margin based on statistical likelihood rather than absolute worst-case bounds, probabilistic planning can safely ignore extreme-tail delays, preserving the optimality of the global schedule.

3

Problem Formulation and Modeling

While a solution to a MAPF$_R$ problem provides valid, collision-free planned trajectories, it relies on a perfect, deterministic execution. To address this limitation, this chapter extends the deterministic MAPF$_R$ formulation into a probabilistically robust model (p-Robust MAPF$_R$). Instead of assuming exact execution times, the model incorporates stochastic traversal durations and evaluates the resulting risk of collisions.

The chapter is structured as follows. Section 3.1 gives the problem formulation by starting from a deterministic environment and gradually incorporating uncertainty, leading to the definition of *potential conflicts* and *global risk*. Section 3.2 formalizes this approach by defining the mathematical models and assumptions used to quantify the *local risk* of these interactions and compute the resulting *collision probabilities*.

3.1 Problem Formulation

This section introduces the problem formulation for probabilistically robust MAPF$_R$. Beginning with a deterministic description of the environment and agent behavior, uncertainty in traversal durations is incorporated into the model. This leads to the definition of potential conflicts and their associated risks, which are then combined into a global measure of risk used to evaluate the safety of a joint plan.

3.1.1 Environment Modeling

Let the environment be defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where agents move between vertices along edges. An edge represents a transition between two vertices and corresponds to a continuous trajectory that may have an arbitrary shape.

Each agent $i \in [1 \dots N]$ is assigned a starting vertex $s_i \in \mathcal{V}$ and a goal vertex $g_i \in \mathcal{V}$. To move from its start to its goal, each agent follows a single agent plan that consists of a sequence of actions, where each action is either a movement along an edge or a wait at a vertex.

At the start of execution, the agents are assumed to begin from their initial positions at time $t = 0$, which is treated as a deterministic event.

3.1.2 Traversal Duration Uncertainty

In standard MAPF_R, the execution of a move action is assumed to take an exact, deterministic duration. However, to model the real world's uncertainties, the traversal duration of each edge is treated as a random variable rather than a fixed value. As an agent moves through the graph, the uncertainty associated with individual edge traversals accumulates, affecting the time at which the agent arrives at subsequent vertices.

As a result, the timing of an agent's execution can no longer be described deterministically, but must instead be represented probabilistically. The specific choice of distribution and its properties are defined in Section 3.2.

3.1.3 Arrival and Traversal Start Time

At each vertex, an agent may execute a wait action before continuing its movement. In this model, waiting is treated as a deterministic delay that shifts the timing of the agent's subsequent actions.

Consequently, the start time of each traversal is influenced by both the accumulated uncertainty of prior movements and any scheduled waiting time. This combined timing behavior forms the basis for evaluating interactions between agents in the presence of uncertainty.

3.1.4 Example

To illustrate how traversal duration uncertainty affects agent interactions, Figure 3.1 presents a simplified scenario involving two agents navigating a shared environment with no planned waiting time.

The first agent, which may travel by the vertices shown in blue, begins its path at the initial vertex. As defined by our model, this is a known event represented by $start_1$ at time $t = 0$. As the agent continues along its path, the uncertainty associated with each traversal accumulates, leading to increasing uncertainty in the agent's arrival time at subsequent vertices.

In this example, the first agent can take two different paths, one being to the left, $(start_1, A, B, D)$, and the other to the right, $(start_1, A, C, D)$, leading to two different accumulated distributions at vertex D depending on which path is chosen. These distributions determine when the agent starts traversing edge 6 from vertex D. At the same time, a second agent which may travel by the vertices shown in orange navigates a crossing trajectory along edge 7 to vertex E and edge 8 (E, F). The intersection of these two paths creates a region in space where the agents may collide.

Due to the uncertainty in traversal durations, it is no longer possible to determine exactly when each agent reaches this region. Instead, there exists a range of possible arrival times during which the agents may overlap. This situation creates a possible interaction between the agents, where a collision may occur depending on their relative timing.

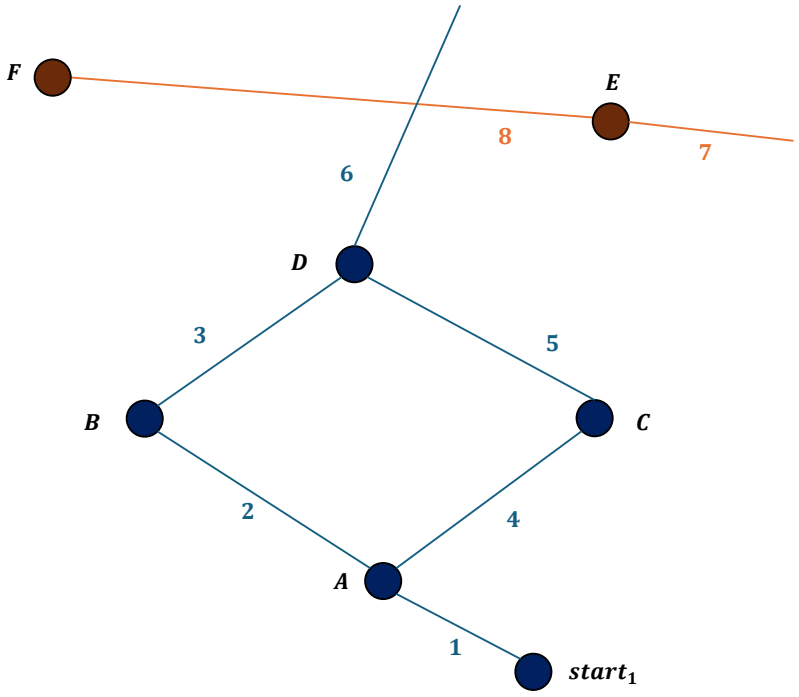

Figure 3.1: Snippet of a simple example graph showing one agent with two possible path choices shown in blue, one to the left and one to the right, and another agent with a crossing edge shown in orange.

3.1.5 From Deterministic to Stochastic Collisions

In a deterministic MAPF$_R$ setting, the occurrence of a collision is defined as a binary event. If two agents occupy overlapping regions of space at the same time, a collision occurs. Otherwise, the agents are considered collision-free.

However, when traversal durations are uncertain, this binary interpretation is no longer sufficient. Even if the planned trajectories of two agents intersect spatially, it is no longer possible to determine exactly when each agent will reach the intersection. Instead, their arrival times vary within a range of possible values.

As a result, a geometric intersection between two trajectories does not imply that a collision will certainly occur, but rather that a collision *may* occur depending on the agents' relative timing. Whether the agents actually collide depends on whether

their uncertain arrival times overlap in such a way that they occupy the same space simultaneously.

Consequently, collisions must be treated as probabilistic events rather than deterministic ones, which fundamentally changes the nature of collision detection. This shift from binary collisions to probabilistic interactions forms the basis for defining and evaluating potential conflicts in the following section.

3.1.6 Potential Conflict

As discussed in the previous section, uncertainty in traversal durations implies that collisions can no longer be treated as deterministic events. Instead, we define a *potential conflict*. We denote such a potential conflict by C_k.

A potential conflict occurs when the planned trajectories of two agents intersect in space, such that a collision may occur depending on their relative timing. Due to the uncertainty in execution, it is not possible to determine with certainty whether the agents will occupy the same space simultaneously.

In this approach, a potential conflict does not automatically invalidate a joint plan. Rather, it represents a situation in which a collision may occur, and must therefore be evaluated in terms of its associated risk.

3.1.7 Global Risk Accumulation

In a multi-agent system, multiple potential conflicts may arise between different pairs of agents. Each potential conflict C_k represents a situation in which a collision may occur, and can therefore be associated with a certain level of risk, denoted as $r_k = P(C_k)$.

To evaluate the safety of a joint plan Π, it is necessary to consider the combined effect of all such potential conflicts. A plan is considered to fail if at least one collision occurs during execution. Consequently, the probability of plan failure corresponds to the probability that at least one of the identified potential conflicts results in a collision.

Let $C_1, C_2, \ldots, C_n$ denote all potential conflicts in the system. The probability of plan failure can then be written as the union of these events:

$$P_{fail}(\Pi) = P(C_1 \cup C_2 \cup \cdots \cup C_n) \tag{3.1}$$

In general, computing this exact probability is difficult. The occurrence of different conflicts is not necessarily independent, as delays affecting one agent may influence its interaction with others. This creates dependencies between collision events, making the exact evaluation of the joint probability computationally intractable in a large-scale system.

To address this challenge, the approach applies Boole's Inequality, also known as the *Union Bound* [26]. The Union Bound states that the probability of the union of events is strictly less than or equal to the sum of their individual probabilities, regardless of whether the events are independent or highly correlated:

$$P(C_1 \cup C_2 \cup \cdots \cup C_n) \leq \sum_{k=1}^{n} P(C_k). \tag{3.2}$$

By applying this conservative upper bound, the global probability of failure is approximated by the simple summation of all local risks in the system:

$$P_{fail}(\Pi) \leq \sum_{k=1}^{n} r_k. \tag{3.3}$$

We define this accumulated quantity as the *global risk*, given by

$$R_{global} = \sum_{k=1}^{n} r_k. \tag{3.4}$$

This provides a simple and computationally efficient way to estimate the overall system risk by summing the individual risks of all potential conflicts. Since R_{global} is obtained through the Union Bound, it is not itself a probability and may exceed 1 when many conflicts contribute to the accumulated risk. Such values should be interpreted as indicating a more risk-prone system, not as implying that a collision is certain.

To ensure safe execution, the accumulated risk must remain below a user-defined threshold p, corresponding to a required probability of successful execution. Equivalently, a global risk budget can be defined as $R_{target} = 1 - p$. A plan is therefore considered acceptable if:

$$R_{global} \leq R_{target}. \tag{3.5}$$

This combined measure of risk allows the safety of a given plan to be evaluated with respect to the desired level of robustness.

3.1.8 Problem Definition

Based on the concepts introduced in the previous sections, the probabilistically robust MAPF_R problem can now be formally defined.

Let Π denote a joint plan consisting of single-agent plans for all agents. The execution of Π is subject to stochastic traversal durations, which induce uncertainty in the timing of agent movements and interactions. Waiting actions are treated as deterministic delays of fixed duration.

A plan is considered successful if no collisions occur during its execution. Let $P_{\text{success}}(\Pi)$ denote the probability that the execution of Π is collision-free. Equivalently, the probability of failure is given by $P_{\text{fail}}(\Pi)$, representing the probability that at least one collision occurs.

The objective is to find a joint plan Π such that the probability of successful execution satisfies a user-defined threshold p, such as:

$$P_{\text{success}}(\Pi) \geq p. \tag{3.6}$$

Equivalently,

$$P_{\text{fail}}(\Pi) \leq 1 - p. \tag{3.7}$$

This formulation corresponds to a chance-constrained version of the MAPF$_R$ problem, in contrast to robust formulations that enforce safety under all delays up to a specified deterministic time bound.

Using the formulation from Section 3.1.7, specifically the upper bound in Equation (3.3) and the definition in Equation (3.4), a sufficient condition for the chance-constrained problem formulation in Equation (3.7) is the global risk condition in Equation (3.5), i.e.,

$$R_{\text{global}} \leq R_{\text{target}},$$

where $R_{\text{target}} = 1 - p$ denotes the maximum allowable risk or the global risk budget.

The probabilistically robust MAPF$_R$ problem is therefore defined as finding a joint plan Π such that the accumulated risk over all potential conflicts remains within the specified global risk budget.

3.2 Mathematical Model and Assumptions

Following the problem formulation, we now introduce the mathematical models and assumptions used to quantify traversal uncertainty and evaluate collision risks between agents.

3.2.1 Traversal Probability Distribution

To model the uncertainty in traversal durations, each edge traversal is represented as a continuous random variable. This requires selecting an appropriate probability distribution that accurately captures execution variability while remaining computationally tractable.

Traversal durations are strictly non-negative, and may exhibit asymmetric behavior, making distributions such as the Gamma distribution [27] physically well-motivated. However, these distributions introduce significant complexity when uncertainties accumulate over multiple edges, since they lack simple closed-form expressions.

In contrast, the Gaussian distribution [28] provides an advantage. The sum of independent Gaussian random variables is itself Gaussian, allowing the accumulated traversal durations along a path to be computed efficiently using simple parameter addition. This property is particularly important in the context of multi-agent path planning, where many such computations must be performed during the search.

Each traversal duration is therefore modeled as a Gaussian random variable:

$$D_e \sim \mathcal{N}(\mu_e, \sigma_e^2), \tag{3.8}$$

where μ_e and σ_e^2 denote the mean and variance of the traversal duration along edge e. The implications and limitations of this choice, including its consistency with physical constraints, are discussed in the following section.

3.2.2 Physical Constraints on the Gaussian Model

While the mathematical properties of the Gaussian distribution provide crucial computational efficiency, it is not fully consistent with the physical properties of traversal durations. In particular, the Gaussian distribution has unbounded support, implying a non-zero probability of arbitrarily small or even negative traversal durations, which are physically impossible.

In real-world systems, traversal durations are constrained by the kinematic limits of the agents [29]. In particular, each agent has a maximum achievable speed v_{max}, which imposes a lower bound on the time required to traverse a given edge. For an edge of length l, the minimum physically feasible traversal duration t_{min} is given by:

$$t_{min} = \frac{l}{v_{max}} \tag{3.9}$$

To ensure that the Gaussian model respects this physical constraint, the distribution is restricted such that the overwhelming majority of its probability mass lies above $t_{\min}$. This is achieved by applying a statistical safety rule based on the standard deviation of the distribution, commonly referred to as the σ-rule [30]. In this work, we apply this principle using a 3σ bound. Specifically, we enforce that the lower bound of the distribution, defined as $\mu - 3\sigma$, satisfies

$$\mu - 3\sigma \geq t_{\min}. \tag{3.10}$$

This constraint can be rearranged to give an upper bound on the standard deviation:

$$\sigma \leq \frac{\mu - t_{\min}}{3}. \tag{3.11}$$

This provides a practical way to select the level of uncertainty, ensuring that the distribution remains consistent with the physical limits of the system. In practice, the small remaining probability mass outside the 3σ interval is considered negligible, and the distribution is effectively treated as bounded within this range.

3.2.2.1 Illustrative Example

The following example demonstrates how this constraint can be applied in practice. Consider an edge distance of $l = 10\,\mathrm{m}$. Assume the agents' nominal cruising speed is $v_{nom} = 1.0\,\mathrm{m/s}$, and their hardware-limited maximum speed is $v_{max} = 1.2\,\mathrm{m/s}$. Then the expected edge traversal duration will become:

$$\mu = \frac{l}{v_{nom}} = \frac{10\,\mathrm{m}}{1.0\,\mathrm{m/s}} = 10\,\mathrm{s},$$

which gives a minimum feasible traversal duration:

$$t_{min} = \frac{l}{v_{max}} = \frac{10}{1.2} \approx 8.33\,\text{s}.$$

Applying the constraint $\mu - 3\sigma \geq t_{\min}$ gives the standard deviation:

$$10.0 - 3\sigma \geq 8.33 \quad \Rightarrow \quad \sigma \leq 0.55\,\text{s}.$$

This ensures that more than 99.7% of the probability mass lies within physically feasible traversal durations while retaining a meaningful representation of uncertainty.

3.2.3 Local Risk Model

This section introduces the probabilistic approach used to compute the collision risk between two interacting agents. While the geometric conditions for a collision are deterministic, based on the pre-computed unsafe time intervals from Section 2.2.2, the execution timing of agent actions are uncertain, as traversal durations are stochastic while waiting durations remain deterministic.

To account for this, the local risk model treats arrival times as stochastic variables. These uncertainties propagate through traversal start times and relative timing differences, which are then used to compute the probability of collision.

3.2.3.1 Arrival and Traversal Start Time Distributions

Let A_1 and A_2 be random variables representing the arrival times of two agents at the vertices from which their subsequent actions are executed. Due to accumulated traversal duration uncertainty, these arrival times are modeled as Gaussian random variables:

$$A_1 \sim \mathcal{N}(\mu_{A_1}, \sigma_{A_1}^2), \quad A_2 \sim \mathcal{N}(\mu_{A_2}, \sigma_{A_2}^2),$$

where μ_{A_i} and $\sigma_{A_i}^2$ denote the mean and variance of the arrival time of agent i, resulting from accumulated traversal uncertainty along its path. We assume that traversal durations along different agents' paths are independent random variables.

Let $w_1 \geq 0$ and $w_2 \geq 0$ denote the deterministic waiting times before the subsequent action is executed. The traversal start time T_i for Agent i is then given by

$$T_i = A_i + w_i.$$

Adding a constant to a Gaussian random variable shifts its mean but leaves the variance unchanged:

$$T_1 \sim \mathcal{N}(\mu_{A_1} + w_1, \sigma_{A_1}^2), \quad T_2 \sim \mathcal{N}(\mu_{A_2} + w_2, \sigma_{A_2}^2).$$

For simplicity in the following notation, we define

$$\mu_i = \mu_{A_i} + w_i, \quad \sigma_i^2 = \sigma_{A_i}^2,$$

yielding

$$T_i \sim \mathcal{N}(\mu_i, \sigma_i^2).$$

The occurrence of a collision depends on the relative timing between the action start time of the agents. We denote this by

$$\Delta T.$$

The definition of ΔT depends on the interaction type and on the roles of the agents within that interaction.

Move–move interactions For interactions where both agents execute move actions, let T_1 and T_2 denote the times at which Agent 1 and Agent 2 start their respective move actions. Without loss of generality, Agent 1 is taken as the reference agent. The relative timing is then defined as the difference between the two move-action start times:

$$\Delta T = T_2 - T_1.$$

Since T_1 and T_2 are independent Gaussian random variables, their difference is also Gaussian distributed:

$$\Delta T \sim \mathcal{N}(\mu_{\Delta T}, \sigma_{\Delta T}^2),$$

where

$$\mu_{\Delta T} = \mu_2 - \mu_1, \quad \sigma_{\Delta T}^2 = \sigma_1^2 + \sigma_2^2.$$

The corresponding probability density function (PDF) is given by

$$f_{\Delta T}(t) = \frac{1}{\sigma_{\Delta T}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{t - \mu_{\Delta T}}{\sigma_{\Delta T}}\right)^2\right). \tag{3.12}$$

Move–wait interactions For interactions involving one waiting agent and one moving agent, the roles are defined by the interaction. Let Agent 1 be the waiting agent occupying the vertex, and Agent 2 the agent executing a traversal. Let A_1 denote the time at which Agent 1 arrives at the vertex and thereby starts its wait action, and let T_2 denote the time at which Agent 2 starts its move action. The relative timing is then defined as the difference between these two action start times:

$$\Delta T = T_2 - A_1.$$

Since T_2 and A_1 are independent Gaussian random variables, their difference is also Gaussian distributed:

$$\Delta T \sim \mathcal{N}(\mu_2 - \mu_{A_1}, \sigma_2^2 + \sigma_1^2).$$

3.2.3.2 Collision Probabilities

Using the relative timing distributions derived above, the probability of collision for each interaction type can now be computed by evaluating the probability that the timing difference falls within the corresponding unsafe interval.

Probability of Collision for Move-Move Conflicts For a conflict where both agents are moving, we can pre-compute the geometric time interval $I_{m,m'} = [\delta_{min}, \delta_{max}]$ such that if another agent begins executing its move m' at some time relative to the first agent within $I_{m,m'}$, a collision will occur. Thus, a collision happens if the traversal start time difference falls within this unsafe interval such that $\Delta T \in I_{m,m'}$.

The theoretical probability of a collision, $P_{collision}$, is the area under the PDF $f_{\Delta T}(t)$ over the interval $I_{m,m'}$:

$$P_{collision} = \int_{I_{m,m'}} f_{\Delta T}(t)dt = \int_{\delta_{min}}^{\delta_{max}} f_{\Delta T}(t)dt. \tag{3.13}$$

In practice, this integral is evaluated using the cumulative distribution function (CDF) [31] of the standard normal distribution, denoted by Φ:

$$P_{collision} = \Phi\left(\frac{\delta_{max} - \mu_{\Delta T}}{\sigma_{\Delta T}}\right) - \Phi\left(\frac{\delta_{min} - \mu_{\Delta T}}{\sigma_{\Delta T}}\right) \tag{3.14}$$

Probability of Collision for Move-Wait Conflicts In contrast to move-move interactions, a move-wait interaction involves an agent occupying a vertex over a fixed duration and another agent that executes a trajectory that geometrically intersects this vertex. Let Agent 1 be the waiting agent and Agent 2 be the moving agent. Agent 1 arrives at the vertex at $A_1 \sim \mathcal{N}(\mu_{a_1}, \sigma_{a_1}^2)$ and remains there for a deterministic wait duration w_1. The absolute time interval that Agent 1 occupies the conflict zone is then $[A_1, A_1 + w_1]$.

Agent 2 begins its move action at a stochastic traversal start time $T_2 \sim \mathcal{N}(\mu_{t_2}, \sigma_{t_2}^2)$. We can pre-compute the geometric time interval $\overline{I}^c = [\tau_{min}, \tau_{max}]$, which is the time relative to T_2 that Agent 2 passes through the conflict zone. The absolute time that Agent 2 occupies the conflict zone is then $[T_2 + \tau_{min}, T_2 + \tau_{max}]$.

A collision occurs if the agents occupy the same space simultaneously. This happens when the absolute time intervals $[A_1, A_1 + w_1]$ and $[T_2 + \tau_{min}, T_2 + \tau_{max}]$ overlap (as illustrated in Figure 3.2). For these two intervals to overlap, both of the following temporal conditions must be true:

- Agent 2 arrives at the conflict zone before Agent 1 has left:

$$T_2 + \tau_{min} < A_1 + w_1 \tag{3.15}$$

- Agent 1 begins waiting at the conflict zone before Agent 2 has left:

$$A_1 < T_2 + \tau_{max} \tag{3.16}$$

By rearranging these inequalities to evaluate the difference between the traversal start time of Agent 2 and the arrival time of Agent 1, defined as $\Delta T = T_2 - A_1$, we

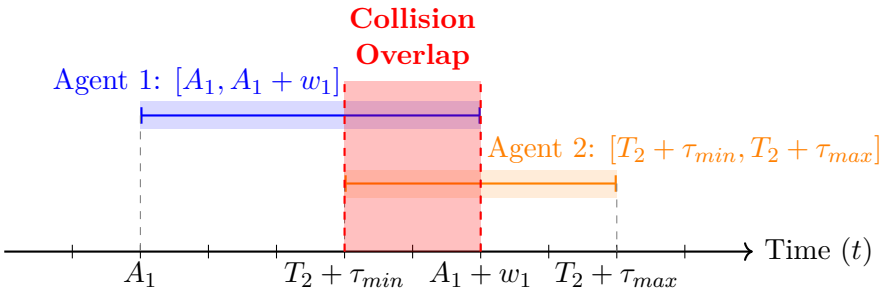

Figure 3.2: Timeline visualization of a move-wait conflict. A collision occurs because Agent 2 arrives ($T_2 + \tau_{min}$) before Agent 1 leaves ($A_1 + w_1$), and Agent 1 arrives (A_1) before Agent 2 leaves ($T_2 + \tau_{max}$).

obtain a move-wait specific unsafe interval I_{mw}. The lower bound of this interval comes from (3.16), and the upper bound from (3.15):

$$-\tau_{max} < T_2 - A_1 < w_1 - \tau_{min} \implies I_{mw} = [-\tau_{max}, w_1 - \tau_{min}] \tag{3.17}$$

Since T_2 and A_1 are independent normally distributed variables, their difference is $\Delta T \sim \mathcal{N}(\mu_{a_2} - \mu_{a_1}, \sigma_{a_2}^2 + \sigma_{a_1}^2)$. We can therefore compute the theoretical probability of collision $P_{collision}$ using the CDF method (3.14), substituting the geometric interval bounds of $I_{mm'}$ with I_{mw}.

4

Algorithmic Design

With the local risk model and probabilistic approach established in the previous chapter, the next challenge is translating these theoretical safety guarantees into an executable search algorithm.

Ideally, the objective would be to formulate this translation as a global constrained optimization problem, using an exact solver to perfectly optimize the distribution of risk across all agents and minimize total physical waiting time. However, finding an exact, polynomial-time solver for this continuous risk-allocation problem is difficult. Consequently, computing an absolute global optimum is practically impossible. To execute probabilistically robust planning within polynomial time, practical implementations require heuristic modifications and approximate allocation models.

This chapter outlines sufficient algorithmic adaptations developed to solve this problem. It details how CCBS is modified to manipulate constraints, and introduces heuristics designed to systematically allocate the global risk budget and resolve local conflicts. Section 4.1 explains how the constraint generation in CCBS is modified by artificially expanding unsafe time intervals, which forces the low-level planner to create a statistical safety buffer via mean-shifting. Section 4.2 breaks down the heuristic rules for selecting exactly which conflict to target, such as Highest Risk or Lowest Variance, and details the specific models used to calculate how much delay to apply to satisfy the global risk target.

4.1 Modification of CCBS

With the local risk model established, the next challenge is applying it within the approach of global path planning. To translate theoretical safety guarantees into executable paths, modifications to the underlying search algorithm are required. This section outlines the algorithmic adaptations to Continuous Conflict-Based Search (CCBS) that we have investigated during this thesis.

4.1.1 Mean-Shifting via Interval Expansion

To integrate probabilistic robustness guarantees without fundamentally altering the low-level planner, we investigate a modification to the constraint generation in CCBS. The core concept relies on utilizing the existing behavior of CSIPP by artificially expanding the unsafe time intervals.

For the considered interaction, let X_1 and X_2 denote the relevant action start times for the two agents, and define the relative timing as $\Delta T = X_2 - X_1$. Let $I = [\ell, u]$ denote the corresponding true unsafe interval, i.e., the interval of relative timings for which a collision occurs.

The interaction-specific definitions of X_1, X_2, ℓ, and u, introduced in Section 3.2.3, are summarized in Table 4.1.

Table 4.1: Interaction-specific definitions of the generic timing notation used in Section 4.1.1.

Interaction type	$\boldsymbol{X_1}$	$\boldsymbol{X_2}$	$\boldsymbol{\ell}$	$\boldsymbol{u}$
Move–move	T_1	T_2	$\delta_{\min}$	$\delta_{\max}$
Move–wait	A_1	T_2	$-\tau_{\max}$	$w_1 - \tau_{\min}$

In the original CCBS framework, conflicts are treated as deterministic. A valid joint plan therefore requires

$$\Delta T \notin [\ell, u]. \tag{4.1}$$

For the individual agents, this translates to the following time constraints, depending on which agent is constrained,

$$X_1 \notin [X_2 - u,\ X_2 - \ell], \tag{4.2}$$

$$X_2 \notin [X_1 + \ell,\ X_1 + u]. \tag{4.3}$$

However, introducing traversal duration uncertainty shifts the concept of a conflict from a binary collision to a potential conflict. A geometric overlap no longer guarantees a collision, but rather represents a probability mass $P_{collision}$ computed via our local risk model. An illustration of a potential conflict under uncertainty is shown in Figure 4.1.

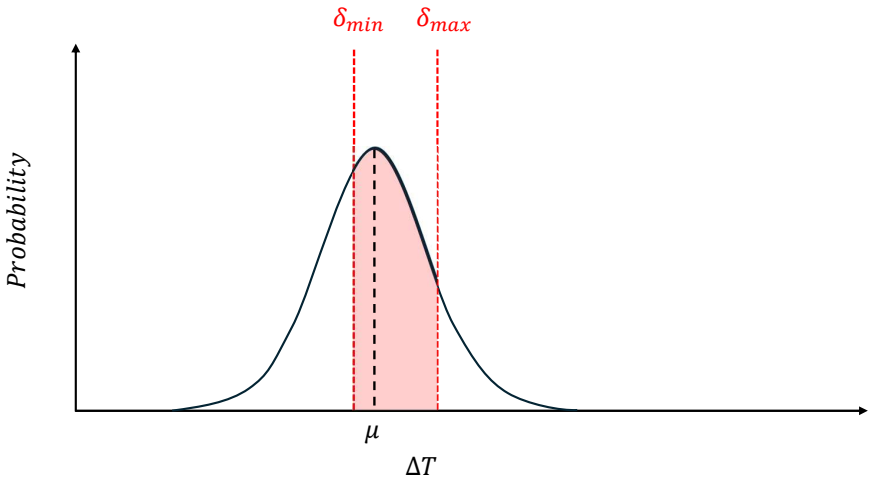

Figure 4.1: A possible collision risk in one interaction. The probability of a collision is represented by the red shaded area between the unsafe interval bounds, which in this example equals 30%.

This uncertainty makes it highly problematic to apply the exact geometric unsafe interval from (4.2) and (4.3) as a constraint. When unmodified, the low-level planner CSIPP treats the expected arrival times as deterministic points to optimize for the lowest sum of costs. Using its earliest-time selection strategy, CSIPP will schedule the constrained agent to arrive directly after the unsafe interval ends. This places the expected time difference at the boundary $\mu_{\Delta T} = u$, as shown in Figure 4.2, still leaving a significantly large local collision risk.

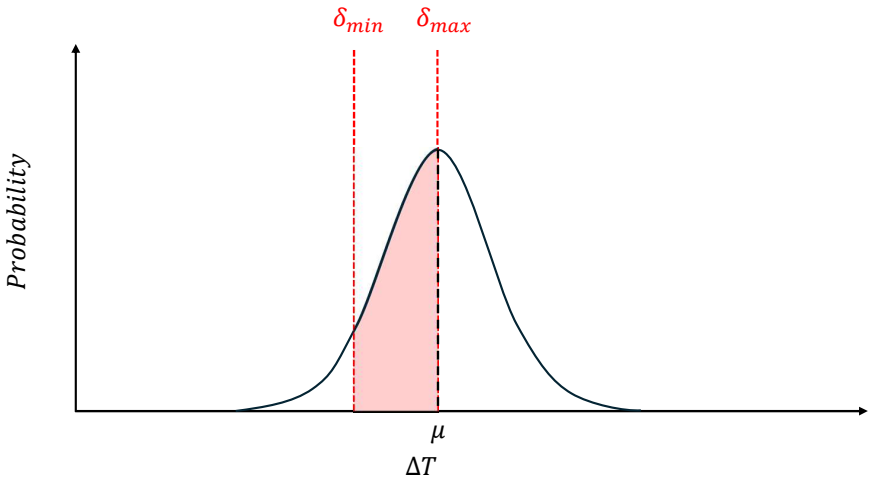

Figure 4.2: The collision risk in one interaction after passing the exact unsafe interval I as a constraint to CSIPP.

To decrease this local collision risk to an acceptable threshold p, the expected time difference $\mu_{\Delta T}$ must be shifted away from the unsafe interval. This mean-shift can be achieved by artificially expanding the constraints passed to CSIPP by a safety margin $d \geq 0$. The expanded unsafe interval I_d is defined as:

$$I_d = [\ell - d, u + d]. \tag{4.4}$$

By adding I_d as the new constraint instead, the CSIPP earliest-time selection will schedule the agents as early as possible or so that $\mu_{\Delta T}$ is placed directly after our increased unsafe interval at $u+d$ as illustrated in Figure 4.3. The updated constraints for the individual agents become:

$$X_1 \notin [X_2 - u - d, X_2 - \ell + d], \tag{4.5}$$

$$X_2 \notin [X_1 + \ell - d, X_1 + u + d]. \tag{4.6}$$

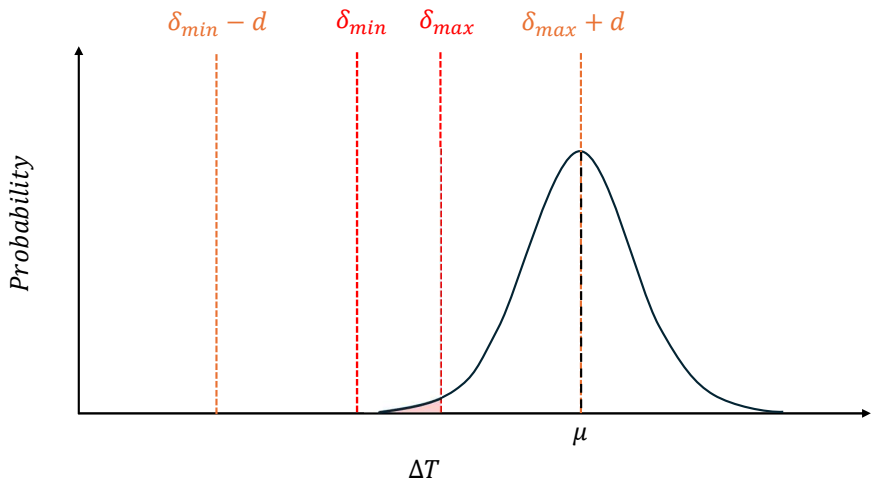

Figure 4.3: The collision risk in one interaction after passing an increased interval I_d as a constraint to CSIPP, resulting in a controlled mean-shift of the traversal start time distribution ΔT.

It is crucial that the safety margin d is also applied to the lower bound ℓ to guarantee the acceptable collision probability p. This handles scenarios where CSIPP schedules the agents to depart as early as possible and $\mu_{\Delta T} < \ell$. Without shifting the lower bound, the right tail of the distribution could still overlap the unsafe interval with a probability exceeding p. To force the required mean-shift, it is therefore necessary to also apply the margin d to the lower bound. Figure 4.4 demonstrates such a case.

The magnitude of the required safety margin d is determined by the acceptable collision probability p. Assuming the risk is shifted to the upper bound such that $\mu_{\Delta T} = u + d$, the exact risk within the true geometric interval I is calculated as:

$$P(\ell \leq \Delta T \leq u) = \Phi\left(-\frac{d}{\sigma_{\Delta T}}\right) - \Phi\left(\frac{\ell - (u+d)}{\sigma_{\Delta T}}\right) = p, \tag{4.7}$$

where Φ denotes the cumulative distribution function (CDF) of the standard normal distribution.

Since (4.7) lacks a closed-form analytical solution, the exact margin d must be derived using numerical methods. A more computationally efficient alternative during the tree search is to use a conservative tail-approximation. By disregarding the lower bound and defining the risk as the entire infinite tail $P(\Delta T \leq u) = p$, d can be explicitly calculated using the inverse CDF (quantile function):

$$d = \left|\Phi^{-1}(p)\right| \cdot \sigma_{\Delta T}. \tag{4.8}$$

This approximation guarantees that the probability of the time difference falling below u, and thus entering the true unsafe interval, is strictly bounded by p. A remaining challenge is determining how the desired local collision probability p should be distributed and chosen. We investigate different allocation strategies for p below in Section 4.2.

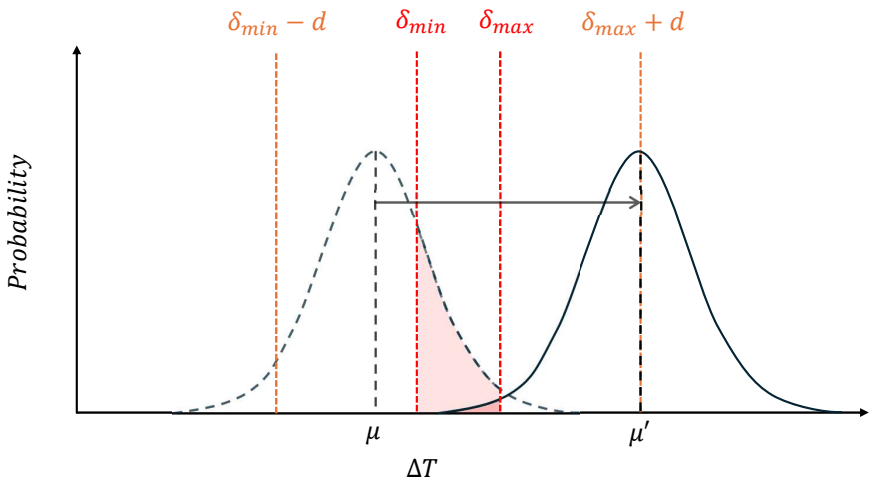

Figure 4.4: A scenario of a mean before the exact unsafe interval I that is then shifted after passing an increased interval I_d as a constraint to CSIPP. This shows the importance of increasing both bounds of the interval to achieve the controlled mean-shift.

4.1.2 Branch-Specific Constraint Generation

When a potential conflict is selected for mitigation, the high-level search creates two child nodes corresponding to the two general branch constraints in (4.5) and (4.6).

The exact constraint object passed to the low-level planner depends on the interaction type. Using the interaction-specific timing definitions introduced in Section 3.2.3 and summarized in Table 4.1, the resulting constraints are as follows.

For a move-move interaction, both branches are implemented as edge constraints, one for each agent's conflicting edge. The resulting constraints are:

$$\begin{aligned} c_1 &= (a_1, e_1, [\mu_{T_2} - \delta_{\max} - d,\ \mu_{T_2} - \delta_{\min} + d]), \\ c_2 &= (a_2, e_2, [\mu_{T_1} + \delta_{\min} - d,\ \mu_{T_1} + \delta_{\max} + d]), \end{aligned} \tag{4.9}$$

where e_1 and e_2 denote the conflicting edges.

For a move-wait interaction, one branch produces a vertex constraint for the waiting agent while the other produces an edge constraint for the moving agent,

$$\begin{aligned} c_{\text{wait}} &= (a_{\text{wait}}, v, [\mu_{T_2} - (w_1 - \tau_{\min}) - d,\ \mu_{T_2} - (-\tau_{\max}) + d]), \\ c_{\text{move}} &= (a_{\text{move}}, e, [\mu_{A_1} + (-\tau_{\max}) - d,\ \mu_{A_1} + (w_1 - \tau_{\min}) + d]). \end{aligned} \tag{4.10}$$

Thus, (4.5) and (4.6) define the general timing of the two branches, while the interaction type determines whether the resulting constraint is applied to a vertex or an edge.

4.2 Risk Allocation Strategies

When the high-level verifier determines that the accumulated global risk exceeds the allowable user-defined budget R_{target}, the planner must reduce the risk contributed by one or more local interactions. To bring the solution back within the allowable risk budget, the algorithm computes a new local target collision probability p for selected interactions, forcing the low-level CSIPP planner to introduce a corresponding mean-shift delay d.

To execute this systematically, the risk allocation approach developed in this thesis operates in two stages: *risk mitigation* and *risk allocation*. Furthermore, numerical fail-safes are incorporated to prevent search deadlocks.

4.2.1 Selecting Potential Conflicts for Risk Mitigation

Before computing *how much* risk to remove, the algorithm must determine *which* local conflict to mitigate. The system implements two distinct heuristics to select the target conflict from the list of active interactions:

- **Highest Risk:** This greedy heuristic targets the interaction with the largest individual probability of collision $P_{collision}$. By addressing the highest probable conflicts first, the algorithm aims to satisfy the global risk budget R_{target} with as few high-level tree branches as possible.

- **Lowest Variance:** This heuristic targets the interaction whose relative timing distribution ΔT has the smallest variance. Since the required shift d is proportional to the corresponding standard deviation $\sigma_{\Delta T}$, such conflicts typically require smaller delays to reduce their collision probability. This heuristic therefore prioritizes conflicts that are cheaper to mitigate in terms of schedule cost, favoring lower sum-of-costs over minimizing the number of high-level tree expansions.

Once a target conflict has been selected, the system employs one of several allocation models to compute a new local target probability p. For notational convenience in the following subsections, let $z^\star$ denote the conflict selected for mitigation in the current allocation step. These allocation models are *Sigma Limitation*, *Risk Budget Limitation*, *Shared Risk*, *Incremental Risk*, and *Proportional Risk*, which are detailed in the following subsections.

4.2.2 Sigma Limitation

The Sigma Limitation model enforces a static, conservative statistical bound on the selected interaction, independently of the current global risk state. Instead of dynamically calculating a shift to exactly satisfy the remaining risk budget, this model applies a fixed safety factor z, representing the number of standard deviations.

When a target conflict is selected, the algorithm calculates the acceptable local risk p by evaluating the tail probability of the standard normal distribution beyond the z-factor:

$$p = 1 - \Phi(z), \tag{4.11}$$

where $\Phi(z)$ is the Cumulative Distribution Function (CDF) of the standard normal distribution. In the implementation, p is additionally lower-bounded by Δ_{tol} as a numerical safeguard, since the inverse CDF used later to compute the corresponding delay according to Equation (4.8) becomes undefined as $p \to 0$.

The main advantage of the Sigma Limitation is computational stability. By enforcing a standardized 3σ or 4σ buffer, the model applies a conservative local risk bound without requiring continuous monitoring of the remaining global risk budget. However, because it enforces the same reduction regardless of how much risk reduction is actually needed, it may produce overly conservative plans by introducing unnecessary waiting delays and thereby increasing the global sum-of-costs.

4.2.3 Risk Budget Limitation

In contrast to the static nature of the Sigma Limitation model, the Risk Budget Limitation model is dynamic. Its core principle is to reduce exactly the amount of risk required for the system to satisfy the user-defined global budget R_{target}.

Using the global risk R_{global} defined in Equation (3.4), the algorithm first computes

the excess risk ϵ that violates the global target:

$$\epsilon = R_{global} - R_{target}. \tag{4.12}$$

Once a target conflict is selected, the algorithm attempts to absorb this entire excess by deducting it directly from the target conflict's current local risk, $r_{z^\star}$. The new target probability p is then given by:

$$p = \max(\Delta_{tol},\ r_{z^\star} - \epsilon). \tag{4.13}$$

The main advantage of this model is that it attempts to remove only the excess risk required to satisfy the global target. In this sense, it aims to introduce only as much delay as is needed to bring the accumulated risk back within the allowable budget, thereby preserving the efficiency of the agents' paths when possible.

If the reduction at the selected conflict is not sufficient in a single step, the CCBS solver may need to branch repeatedly, with the remaining excess risk handled through subsequent conflict selections.

4.2.4 Shared Risk

The Shared Risk model operates on the principle of uniform distribution. Instead of attempting to calculate exact excess risk, it divides the global risk budget R_{target} equally among all currently active conflicts in the system.

Let Z be the total number of active, unresolved conflicts identified by the high-level solver. When a target conflict is selected, its local target risk p is set to:

$$p = \frac{R_{target}}{Z}. \tag{4.14}$$

This model is computationally simple and ensures that no single interaction is assigned a disproportionate share of the available risk budget. However, it treats all conflicts as mathematically equivalent. In practice, a uniform local risk allocation may be inefficient, since highly uncertain interactions generally require larger temporal shifts to achieve the same probability reduction. Consequently, this model may produce suboptimal sum-of-costs compared to models that account for conflict-specific uncertainty.

4.2.5 Incremental Risk

The Incremental Risk model incrementally tightens the allowed local risk target with increasing search depth, thereby discouraging prolonged exploration of the same branch. It dynamically shrinks the allowed local risk target based on the depth of the CCBS constraint tree.

Let d_{tree} represent the current depth of the high-level search node, and let $\alpha \in (0, 1)$ be a user-defined exponential decay factor. The target probability p assigned to the

selected conflict is calculated as:

$$p = R_{target} \cdot \alpha^{d_{tree}}. \tag{4.15}$$

At the root of the search tree ($d_{tree} = 1$), the constraint is relatively mild. However, if the CCBS algorithm continues to generate deeper child nodes, the target probability p decreases exponentially. This results in increasingly restrictive constraints, which can raise the cost of the current branch and thereby encourage the A^*-based high-level solver to explore alternative routing options.

4.2.6 Proportional Risk

The Proportional Risk model recognizes that the effort required to reduce risk is not uniform across all conflicts. For Gaussian relative timing distributions, conflicts with lower uncertainty generally require smaller shifts to achieve the same reduction in collision probability than conflicts with higher uncertainty.

To better preserve the global sum-of-costs, this model distributes the global risk budget R_{target} proportionally according to the standard deviation of each active conflict. Let $\sigma_z = \sigma_{\Delta T_z}$ denote the standard deviation of the relative timing distribution for conflict z. The new target probability p for the selected conflict $z^\star$ is then calculated as:

$$p = R_{target} \cdot \frac{\sigma_{z^\star}}{\sum_{z=1}^{Z} \sigma_z}. \tag{4.16}$$

Using this ratio, the system assigns lower target probabilities to conflicts with lower uncertainty, since these conflicts typically require smaller shifts. Conflicts with higher uncertainty are assigned less restrictive targets, which reduces the risk of unnecessarily large schedule modifications.

If $\sigma_z = 0$ for all active conflicts, the proportional expression becomes undefined because the denominator is zero. In that case, the implementation falls back to the Shared Risk model and assigns the local risk target p according to Equation (4.14). This fallback is a separate design choice used for the deterministic case, rather than the result of the proportional formula itself.

4.2.7 Algorithmic Fail-safes

Calculating a new target risk p can occasionally produce ineffective constraints. For instance, an allocation formula may return a target risk p that is greater than or equal to the conflict's current risk $r_{z^\star}$, or the resulting risk reduction may be smaller than a prescribed numerical tolerance Δ_{tol}. Passing such constraints to CSIPP would fail to improve the global risk R_{global} and might cause the high-level search to repeatedly revisit the same unresolved conflict without making progress.

To avoid this, the approach incorporates a dynamic exclusion mechanism. If the chosen allocation model fails to yield a meaningful reduction in risk, i.e., at least

Δ_{tol}, the corresponding conflict is placed on an exclusion list, and the selection heuristic targets the next best available candidate. If all active conflicts have been considered and placed on the exclusion list, the fail-safe applies a fallback reduction to the final candidate.

In the implementation, this is done by decreasing the candidate's target risk by Δ_{tol}, subject to the lower bound $p \geq \Delta_{tol}$. Accordingly, the fallback target is set to

$$p = \max(\Delta_{tol},\, r_{z^\star} - \Delta_{tol}). \tag{4.17}$$

This mechanism is intended as a practical safeguard against search stagnation. Since the conflict-selection heuristics, the allocation models, and the fallback rule are all heuristic components, no formal completeness claim is made for the risk allocation procedure.

4.3 Software Structure and Implementation

This section describes how the proposed planning framework was realized in software. Figure 4.5 presents the overall workflow of the implementation, while Table 4.2 summarizes the main files in the codebase. The following paragraphs then describe the role of each component in the order in which it appears in the workflow.

Table 4.2: Main files in the implementation and their primary roles.

File	Primary role
`main.py`	Full workflow orchestration.
`ProbabilisticCCBS.py`	High-level constraint tree search and global risk verification.
`ConstraintGenerator.py`	Computes the safety margin and generates constraints.
`Risk_Models.py`	Selects conflicts and assigns new local risk targets.
`LocalRiskModel.py`	Evaluates local collision probability using the CDF-based model.
`SIPP.py`	CSIPP with accumulated traversal-time uncertainty.
`EnvironmentLib.py`	Generates instances and assigns traversal-time distributions.
`GraphPreProcessing.py`	Computes unsafe intervals and conflict preprocessing data.
`LineSegmentIntersections.py`	Performs geometric intersection calculations for preprocessing.
`MCSimulation.py`	Runs batch Monte Carlo validation of saved plans.
`ResultStatisticsLib.py`	Aggregates and visualizes evaluation results.

At the top level, the full workflow is coordinated through `main.py`. This module handles instance loading, graph preprocessing, planner execution, simulation, and benchmark evaluation. It therefore acts as the entry point of the system and connects the planning pipeline to the experimental evaluation presented in Chapter 5.

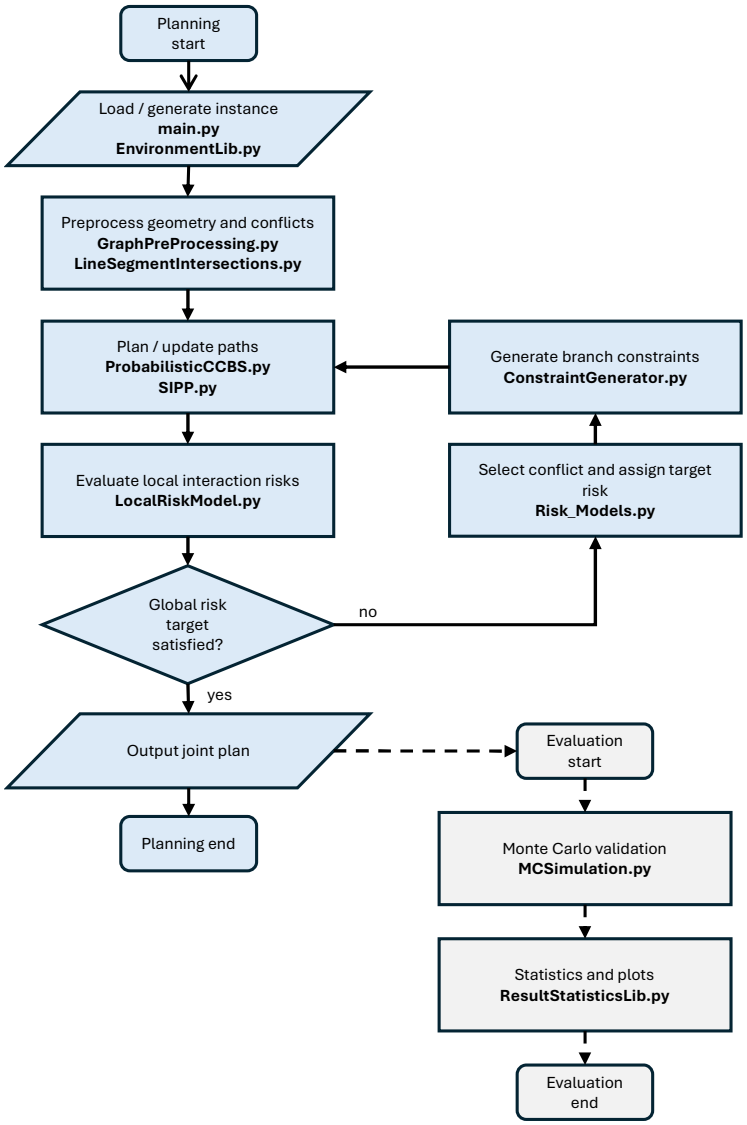

Figure 4.5: Flowchart of the main files in the implementation and their functions.

The environment representation is generated through `EnvironmentLib.py`, which creates one-shot planning instances and benchmark instances, and assigns Gaussian traversal-time distributions to edges according to the probabilistic model introduced in Chapter 3. For the benchmark evaluation, the same module is also used to convert benchmark maps into the graph format required by the planner. Before planning begins, the graph is preprocessed using `GraphPreProcessing.py`, which computes deterministic unsafe intervals and conflict-related geometric information. This preprocessing step relies on `LineSegmentIntersections.py` for the underlying geometric intersection computations. The preprocessing is taken directly from Combrink's CPLP implementation [6]

The core of the planning system is implemented in `ProbabilisticCCBS.py`, which extends Continuous Conflict-Based Search with probabilistic reasoning. This module maintains the high-level constraint tree, stores the current set of constraints,

tracks the current paths and their cost, computes the accumulated global risk, and decides whether further mitigation is required. In this sense, `ProbabilisticCCBS.py` acts as the central coordinating component of the planner.

The local probabilistic reasoning is separated from the high-level search logic. The module `LocalRiskModel.py` evaluates the collision probability of an individual interaction by computing the probability mass inside the corresponding unsafe interval using the cumulative distribution function derived in Section 3.2.3. The resulting local interaction risks are then used by the high-level planner to compute the accumulated global risk according to the formulation in Section 3.1.7. If the global risk target is not satisfied, the planner enters the mitigation loop.

The mitigation loop consists of two main steps. First, `Risk_Models.py` selects the conflict to mitigate and assigns a new local probability target according to the chosen risk mitigation heuristic and risk allocation model from Section 4.2. Second, `ConstraintGenerator.py` uses this target probability to compute the required safety margin and construct the corresponding branch-specific constraints according to the interval-expansion formulation in Section 4.1.2. These constraints are then passed back to the planner for replanning.

Low-level replanning is performed by `SIPP.py`, which extends CSIPP by propagating accumulated traversal-time uncertainty along the path. In addition to computing collision-free paths under vertex and edge constraints, this module keeps track of the planned mean arrival time and accumulated standard deviation at each step. This ensures consistency between the low-level search and the probabilistic timing model used in the risk evaluation.

Once a joint plan has been generated, post-planning evaluation is handled separately from the planning process. `MCSimulation.py` is used for Monte Carlo validation of saved plans by reconstructing the environment and repeatedly simulating stochastic execution. Finally, `ResultStatisticsLib.py` aggregates, processes, and visualizes the planning and simulation results presented in Chapter 5.

Overall, the modular structure separates environment generation, geometric preprocessing, probabilistic risk evaluation, conflict mitigation, low-level path planning, and post-planning evaluation into distinct components. This improves extensibility for future work.

5

Evaluation and Results

This chapter presents the experimental evaluation of the proposed probabilistically robust continuous-time MAPF approach. It is structured to first validate the underlying mathematical models in detail before assessing the algorithmic performance at scale. Section 5.1 demonstrates the application and accuracy of the local risk model using an isolated multi-agent interaction. Section 5.2 details the comprehensive benchmark testing, outlining the experimental setup before evaluating the approach across three primary metrics: computational scalability, the trade-off between schedule optimality and global risk, and the empirical validation of the theoretical safety bounds through Monte Carlo simulations.

5.1 Local Collision Risk in Small Example

To demonstrate the application of the local risk model, we evaluate the potential conflict introduced in the example from Figure 3.1. In this scenario, Agent 1 in blue has two possible paths to reach vertex D, a left path via vertex B and a right path via vertex C. At the same time, Agent 2 approaches the conflict zone via vertex E.

To calculate the exact risk, we assign the agents a nominal speed of 50.0 units/s and a physical collision radius of 15.0 units. Let us look at Figure 3.1. Depending on the chosen path, Agent 1 accumulates different levels of uncertainty. Based on the individual edge variances defined in this scenario shown in Table 5.1 and that we assume no planned waiting time, the left path gives Agent 1 a traversal start time distribution with a lower variance $T_{1_{left}} \sim \mathcal{N}(6.66, 0.25)$, while the right path gives a higher variance $T_{1_{right}} \sim \mathcal{N}(6.66, 1.65)$. Agent 2 starts traversing the intersecting edge with the distribution $T_2 \sim \mathcal{N}(8.00, 0.50)$.

Figure 5.1 shows the probability density function of the traversal start time difference, $\Delta T = T_2 - T_1$, for both scenarios. The narrow green curve represents the left path with lower variance, and the wider purple curve represents the right path with higher variance. The geometric unsafe interval $I_{m,m'} = [\delta_{min}, \delta_{max}]$, is indicated by the vertical red dashed lines. The shaded areas inside this interval show the theoretical probability of a collision calculated from (3.14).

Table 5.1: Properties in The Example Illustrated in Figure 3.1.

Edge Label	Trajectory	Distance (d)	Traversal Duration Distribution ($\mathcal{N}(\mu, \sigma)$)
1	$(start_1, A)$	100.00	$D_1 \sim \mathcal{N}(2.00, 0.05)$
2	(A, B)	116.62	$D_2 \sim \mathcal{N}(2.33, 0.10)$
3	(B, D)	116.62	$D_3 \sim \mathcal{N}(2.33, 0.10)$
4	(A, C)	116.62	$D_4 \sim \mathcal{N}(2.33, 0.80)$
5	(C, D)	116.62	$D_5 \sim \mathcal{N}(2.33, 0.80)$
6	(D, end_1)	300.00	$D_6 \sim \mathcal{N}(6.00, 0.10)$
7	$(start_2, E)$	-	$D_7 \sim \mathcal{N}(8.00, 0.00)$
8	(E, F)	300.00	$D_8 \sim \mathcal{N}(6.00, 050)$

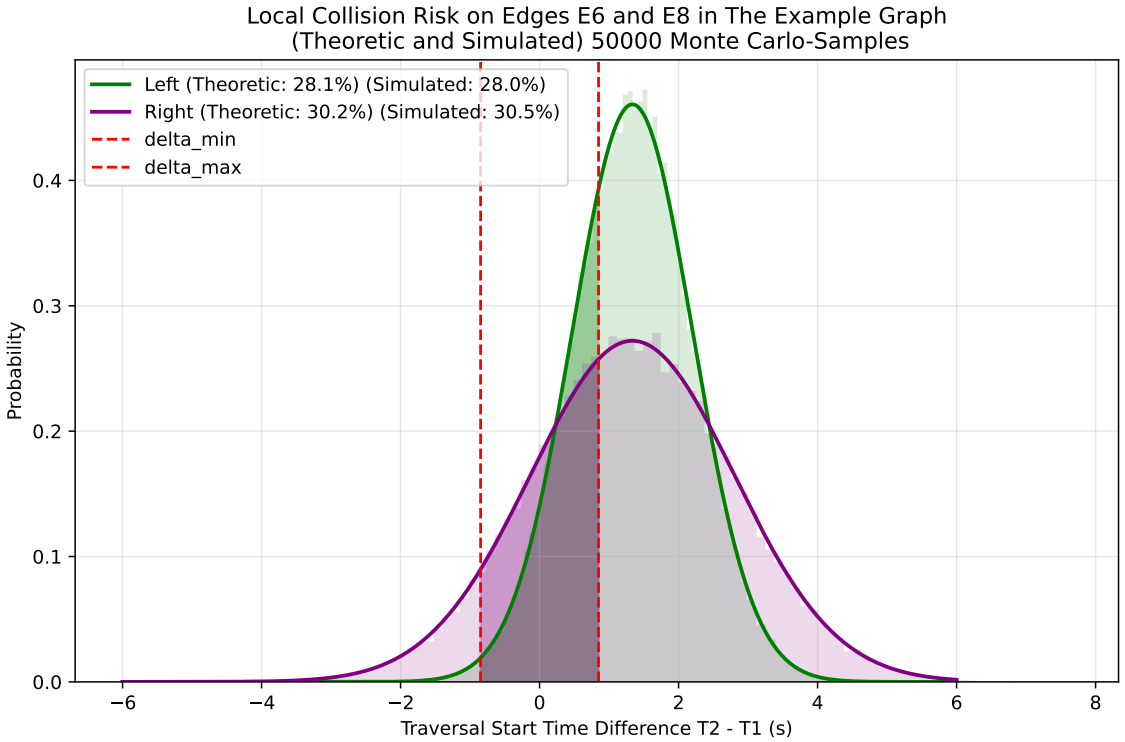

Figure 5.1: Local risk model compared to MC-simulation in interaction from the example graph in Figure 3.1.

To validate the analytical model, a Monte Carlo simulation of $M = 50000$ trials computed the empirical collision risk $\hat{P}_{collision}$ using the indicator function $\mathbb{1}_{I_{m,m'}}(x)$:

$$\mathbb{1}_{I_{m,m'}}(x) \begin{cases} 1 & \text{if } x \in I_{m,m'} \\ 0 & \text{else} \end{cases} \tag{5.1}$$

$$\hat{P}_{collision} = \frac{1}{M} \sum_{k=1}^{M} \mathbb{1}_{I_{m,m'}}(\Delta T_k) \tag{5.2}$$

where ΔT_k is the sampled traversal duration difference in trial k. As Figure 5.1 demonstrates, the theoretical risks align well with these simulated results. This confirms that the analytical CDF is highly accurate.

5.2 Benchmark Testing and Evaluation

Having validated the accuracy of the local risk model within an isolated interaction, this section scales the evaluation to a comprehensive multi-agent environment. The benchmark testing is designed to assess the practical viability of the probabilistically robust approach under various amount of agents and risk constraints. The evaluation is structured into four parts: Section 5.2.1 defines the physical and computational parameters of the simulated environment. Section 5.2.2 analyzes the scalability of the algorithms and the computational overhead introduced by the continuous-time chance constraints. Section 5.2.3 examines the theoretical solution quality. Finally, Section 5.2.4 provides empirical validation of the generated plans through extensive Monte Carlo simulations, verifying that the physical execution success rates rigorously respect the theoretical risk bounds.

5.2.1 Experimental Setup

The experiments were conducted on the `empty-16-16` map from the MovingAI benchmark suite [5]. This environment was modeled as an 8-connected (2^3) grid, allowing agents to move both orthogonally (distance 1.0) and diagonally (distance $\sqrt{2}$). This specific open-space map was chosen because it is well-suited for the underlying CCBS framework that struggles with sparse environments. Furthermore, using this grid aligns our evaluation with established continuous-time MAPF literature [9], enabling comparison with the original CCBS algorithm.

The proposed probabilistically robust approach was evaluated on 25 scenarios taken from the `empty-16-16-random.scen` scenario file in the MovingAI benchmark suite [5]. These scenarios correspond to randomized start–goal assignments on the `empty-16-16` map. For each scenario, the number of agents was incrementally scaled up to a maximum of 50 agents.

Following the configuration of the original CCBS evaluation [9], agents were modeled as geometric disks with a physical collision radius of $\sqrt{2}/4$. For the unit-grid geometry used in this benchmark setting, this radius is a standard choice in continuous-time MAPF, since it allows agents to pass safely along adjacent parallel or orthogonal lanes while still forcing a collision in diagonal pass-through situations. The nominal cruising speed (v_{nom}) was set to 1.0 units/s, and the maximum speed (v_{max}) was capped at 1.2 units/s.

Since the benchmark maps and scenario files do not specify traversal-duration uncertainty, stochastic traversal-duration distributions were assigned to the graph edges during environment generation. To ensure that these distributions remained consistent with the physical speed limit, the standard deviation of each edge distribution was bounded using the 3σ rule detailed in Section 3.2.2. This affects only the traversal-duration model associated with each edge, not the topology of the benchmark graph itself.

A planning timeout of 60 seconds was enforced for the high-level search. This timeout is extended compared to the original deterministic CCBS evaluation [9] that had a timeout of 30 seconds, due to the computational load introduced by the continuous-time chance constraints. The probabilistic approach was evaluated under three distinct user-defined global risk budgets ($R_{target} \in \{0.01, 0.05, 0.10\}$), testing all combinations of the proposed risk allocation models and risk mitigation heuristics showed in Table 5.2.

To validate the theoretical safety bounds guaranteed by the planning algorithms ($1 - R_{global}$), the generated joint plans were executed in a simulated Monte Carlo environment with 50,000 samples each. During each simulation run, the traversal duration for every individual edge executed by the agents was independently sampled from the exact same constrained Gaussian distributions. The entire execution was then evaluated for any collisions. This empirical testing allows for a direct comparison between the theoretical global risk R_{global} computed by the Union Bound defined in (3.4) and the actual execution success rate of the system.

All algorithmic planning and Monte Carlo simulations were run on Intel 8 Cores i7 6700K, 4.0 GHZ, 32GB RAM running Ubuntu-24.04.03 LTS.

Table 5.2: All evaluated combinations of risk allocation models and risk mitigation heuristics, and their assigned legend color and style in result visualization.

Risk Mitigation Heuristic	**Incremental**	**Proportional**	**Risk Budget**	**Shared Risk**	**Sigma**
Highest Risk	solid	solid	solid	solid	solid
Lowest Variance	dashed	dashed	dashed	dashed	dashed

5.2.2 Scalability and Computational Performance

Figure 5.2 illustrates the empirical scalability of the proposed algorithms, measured by their success rate in finding a valid plan within the given time limit. Across all evaluated settings, the success rate begins to decline already at relatively small agent counts and becomes generally low from around 15 agents onward. The algorithms manage to find plans for up to 20 agents, while for the strictest risk budget, $R_{target} = 0.01$, the maximum observed solved instance contains 19 agents.

When comparing the two risk mitigation heuristics, the highest risk heuristic generally maintains a slightly higher success rate than the lowest variance heuristic, whose curves tend to lie below the corresponding highest risk curves. This separation is most visible for the most generous risk budget, $R_{target} = 0.10$, whereas the difference between $R_{target} = 0.05$ and $R_{target} = 0.01$ is less pronounced.

Overall, the probabilistic approach scales less effectively than the original deterministic CCBS algorithm, which solves instances with up to 32 agents in the corresponding grid setting. In the same evaluation, the best-performing enhanced CCBS variant reaches up to 38 agents [9].

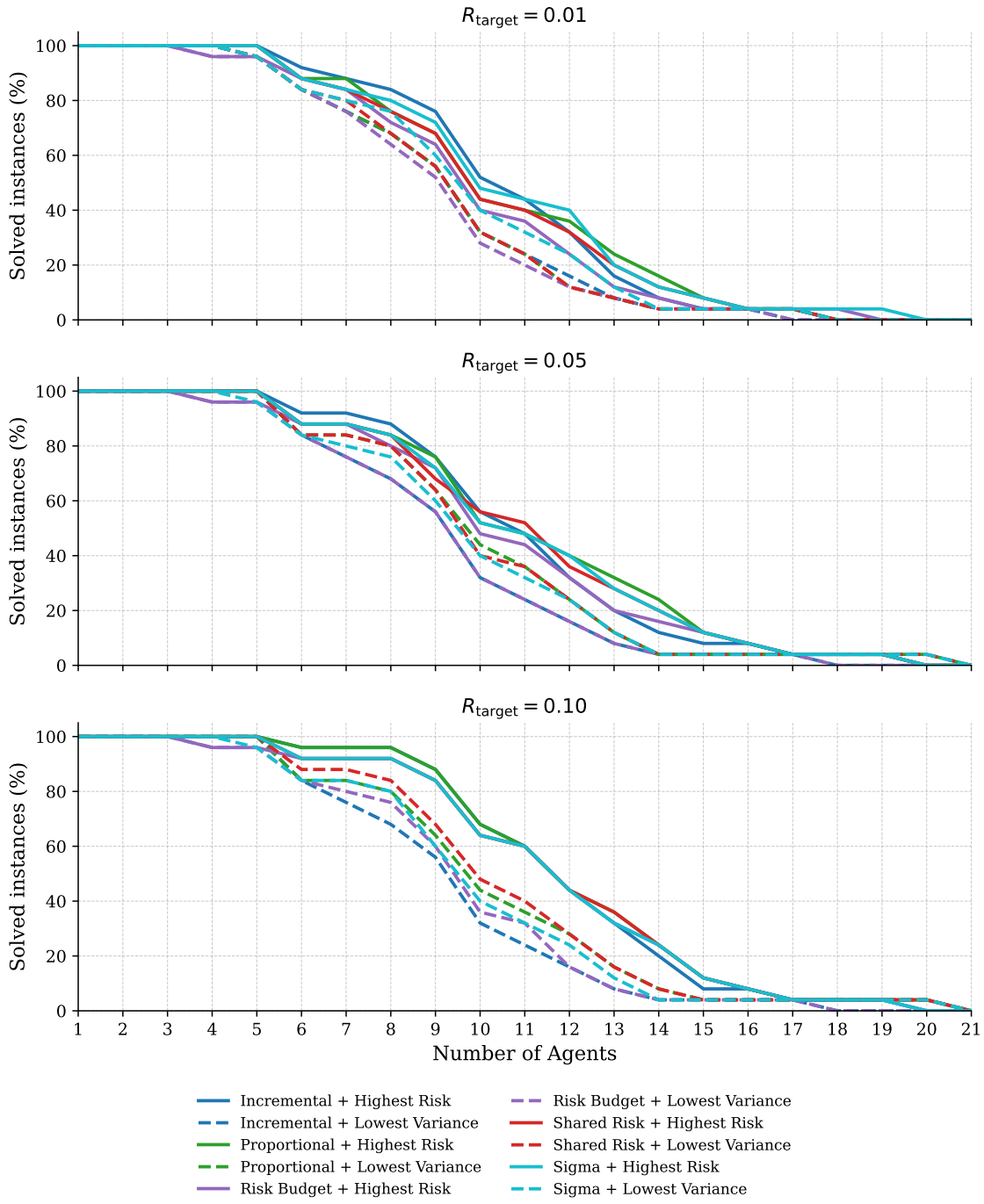

Figure 5.2: Success rate (plan found) during planning for different risk budgets R_{target}.

The computational effort required to solve the benchmark instances is shown in Figures 5.3 and 5.4, which present cactus plots for planning time and expanded search nodes, respectively. In these plots, each curve shows how much computational effort is required to solve an increasing number of instances. Across all evaluated settings, the curves become progressively steeper toward the right-hand side, indicating that the remaining hardest solved instances require disproportionately more time and search effort.

Furthermore, the figures indicate that the computational time scales inversely with the global risk budget. Evaluating the system at a lower R_{target} results in a higher number of expanded nodes and increased planning time, which corresponds to a decrease in the total number of solved instances within the given time limit.

Comparing the specific configurations reveals differences in computational efficiency. The lowest variance risk mitigation heuristic is consistently more costly than the highest risk heuristic. In the plots, the lowest variance curves are positioned higher and further to the left, indicating that the heuristic requires more time and node expansions to solve fewer instances. Among the risk allocation models, the shared risk model frequently performs well, whereas the incremental risk model generally yields worse computational scalability.

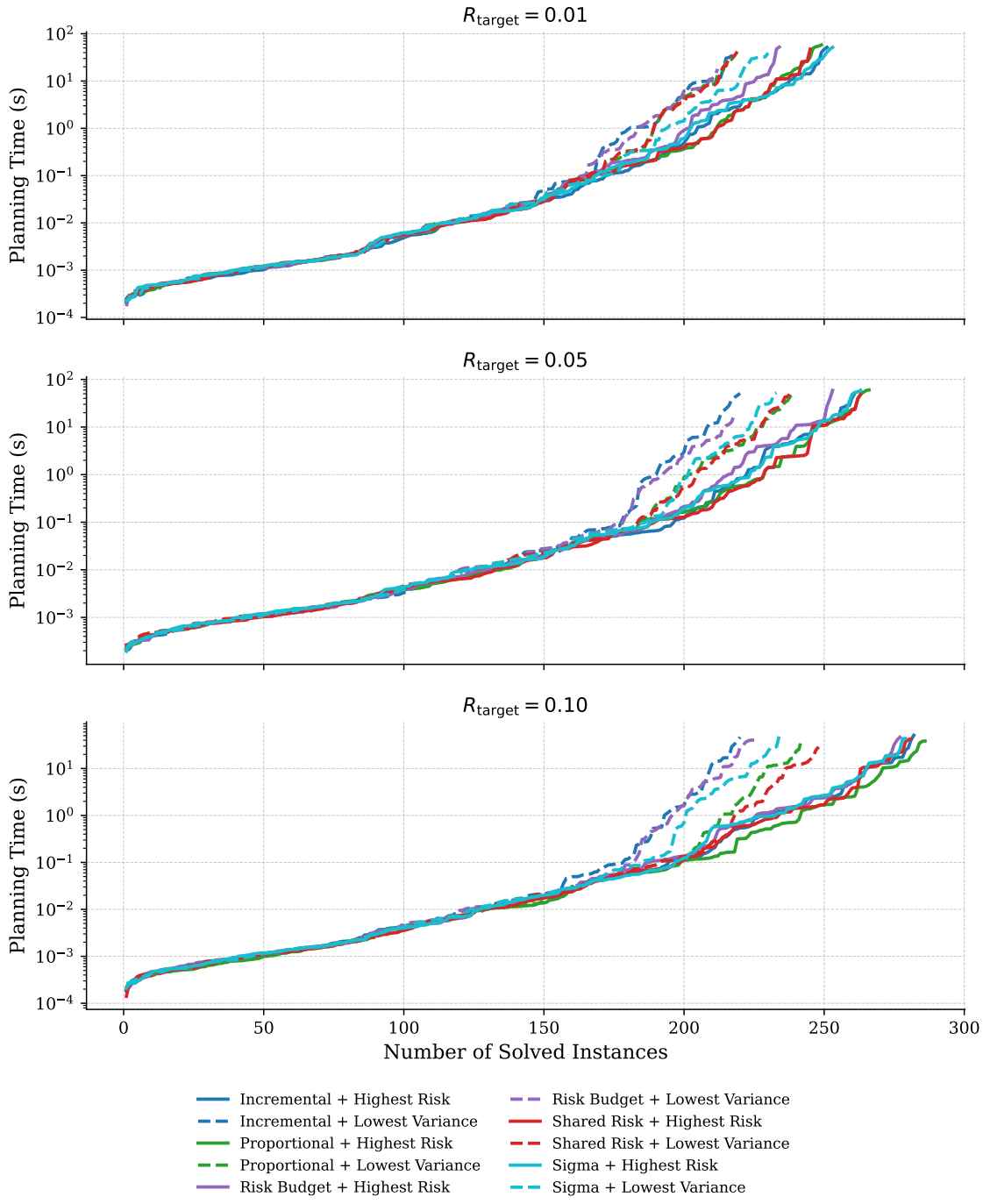

Figure 5.3: Cactus plots of planning time for three global risk budgets R_{target}. Each curve shows the cumulative number of instances solved within a given planning time.

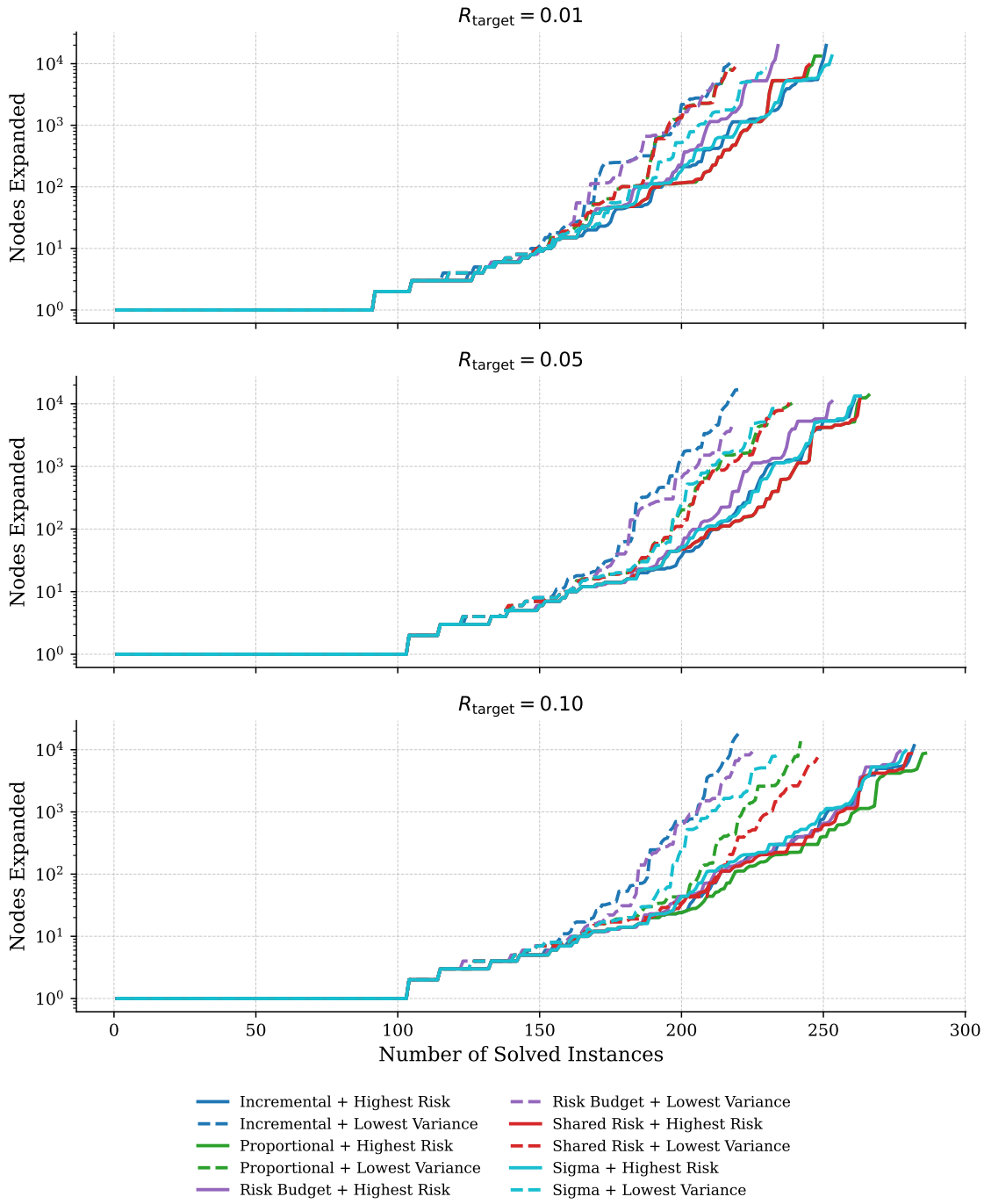

Figure 5.4: Cactus plots of expanded search nodes for three global risk budgets R_{target}. Each curve shows the cumulative number of instances solved within a given number of node expansions.

5.2.3 Solution Quality and Risk Trade-off

Figures 5.5 and 5.6 show the theoretical solution quality of the generated plans in terms of suboptimality ratio and makespan, respectively. In both figures, the lines represent the mean values across solved instances for each agent count, while the shaded regions indicate one standard deviation across scenarios. Here, sum-of-costs denotes the total planned path cost, obtained by summing the planned mean arrival times at the goals of all agents.

Starting with the suboptimality ratio in Figure 5.5, a value of 1 corresponds to the root-path baseline. Across all evaluated settings, the ratio remains very close to 1, indicating that the final solutions are generally only slightly more costly than the corresponding root paths. The highest observed mean value is approximately 1.01, meaning that the sum-of-costs increases by only about 1% in the worst observed case.

The lowest variance heuristic generally remains closer to 1 than the highest risk heuristic, indicating slightly better solution quality in terms of sum-of-costs. For the two less restrictive risk budgets, the curves stay at or very near the baseline over a broader range of agent counts, whereas for the strictest budget this behavior is mainly visible only at very small agent counts. Beyond 13 agents, all curves at some point move back toward 1. This should be interpreted with caution, however, since the shaded standard deviation also becomes very small in this region, suggesting that only a small number of relatively similar solved instances remain.

Figure 5.6 shows the corresponding makespan results, where makespan denotes the planned time at which the last agent reaches its goal. The makespan curves show similar overall shapes across the three risk budgets, and the different methods follow similar qualitative trends as the number of agents increases. The shaded regions are wider at lower agent counts, reflecting greater variation across solved scenarios, and tend to shrink at higher agent counts as fewer solved instances remain. In general, the lowest variance variants tend to produce slightly higher makespan than the corresponding highest risk variants, although the differences remain modest. Local fluctuations are visible in the mid-range of agent counts, but no strong systematic separation between the risk budgets is apparent.

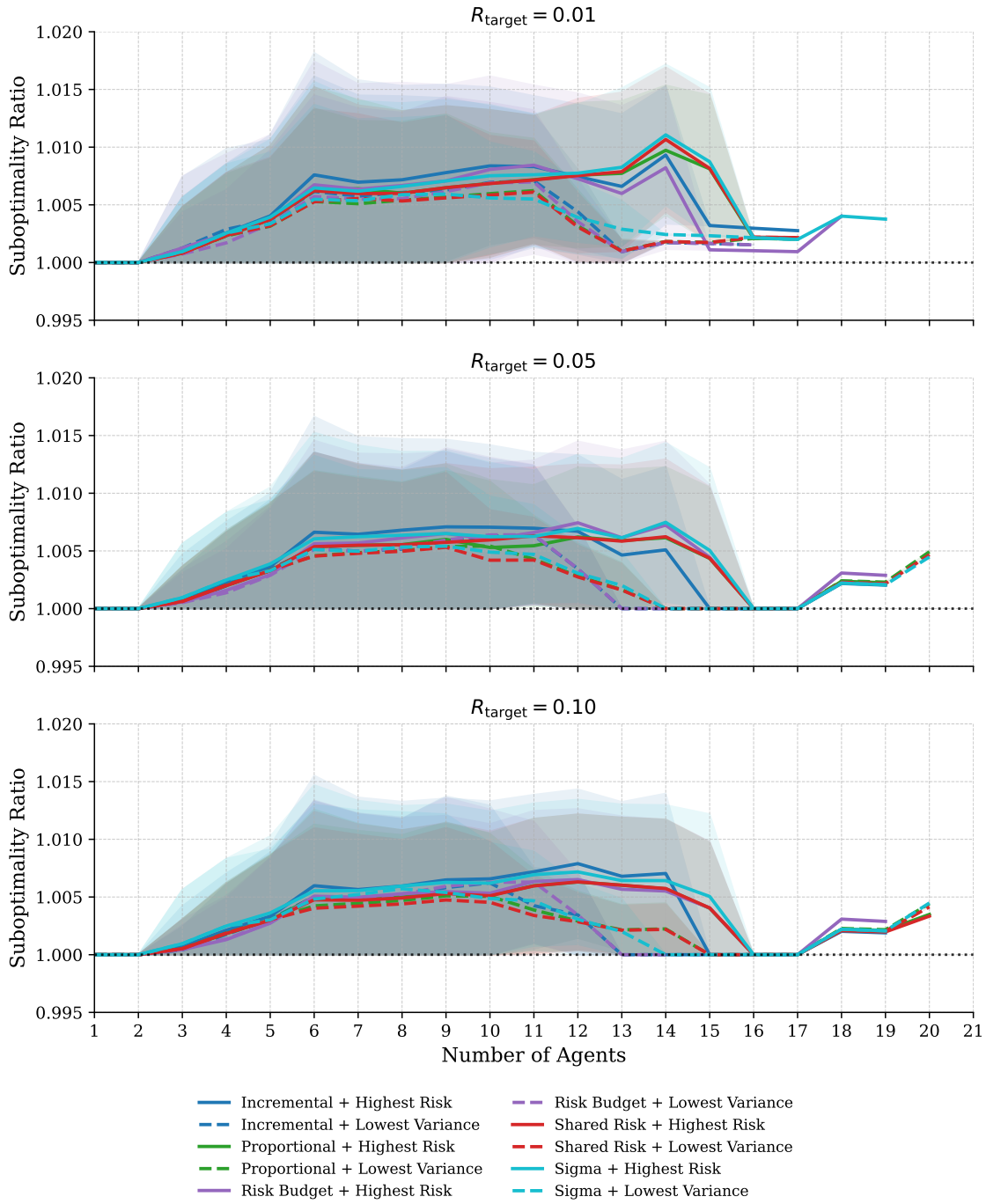

Figure 5.5: Suboptimality ratio for three global risk budgets R_{target}. Each subplot shows the mean ratio across solved instances, while the shaded region indicates one standard deviation across scenarios. The dashed horizontal line at 1 marks the root-path baseline.

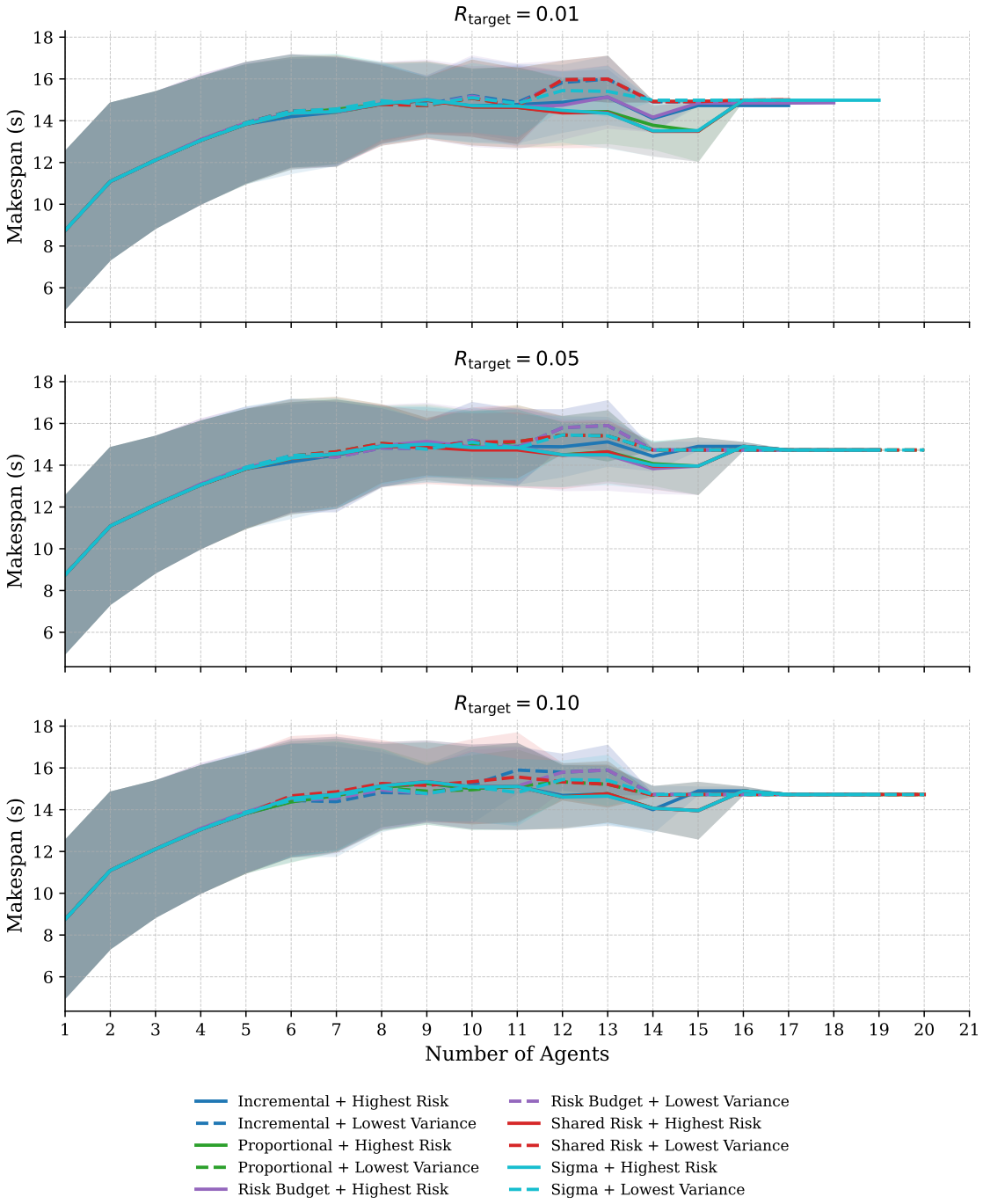

Figure 5.6: Makespan for three global risk budgets R_{target}. Each subplot shows the mean makespan across solved instances, while the shaded region indicates one standard deviation across scenarios.

Figure 5.7 shows the normalized risk utilization for the solved instances, defined as R_{global}/R_{target}. In each subplot, the lines represent the mean utilization across solved instances for each agent count, while the shaded regions indicate one standard deviation across scenarios. A value of 0 corresponds to no accumulated global risk, while a value of 1 corresponds to full utilization of the available risk budget. The dashed red line marks this full-budget threshold.

Across all three risk budgets, the normalized utilization generally increases as the number of agents grows, indicating that denser multi-agent plans tend to consume a larger fraction of the available risk budget. The curves for $R_{target} = 0.05$ appear comparatively smooth, whereas the settings $R_{target} = 0.01$ and $R_{target} = 0.10$ exhibit more local fluctuations. A consistent trend across all three subplots is that the lowest variance variants tend to use a smaller fraction of the available risk budget than the corresponding highest risk variants.

Another notable observation is that none of the evaluated configurations reaches full budget utilization. Even in the most demanding cases, the normalized risk remains below 1. The configuration that comes closest is the risk budget limitation model combined with the highest risk heuristic under $R_{target} = 0.01$, but it still remains below the target line. This suggests that the proposed approach is generally conservative in how much of the available global risk budget it actually uses.

Finally, Figure 5.8 illustrates the cost of safety through a Pareto-style comparison between the global risk budget R_{target} and the resulting suboptimality ratio. Across all evaluated configurations, the increase in suboptimality remains small. From the most generous budget to the most restrictive budget considered here, the change appears to stay below 0.1%, and is visually closer to about 0.05% for most of the models. This indicates that, although stricter safety requirements do increase path cost, the resulting penalty remains modest within the tested range.

At the same time, a consistent ordering is visible across the evaluated models. Configurations using the highest risk heuristic generally yield higher suboptimality ratios than the corresponding lowest variance configurations. Among the allocation models, the shared risk and proportional risk models tend to produce the lowest suboptimality ratios, with risk budget limitation often performing similarly, whereas incremental risk and sigma limitation tend to produce the highest suboptimality ratios.

As the risk budget becomes less restrictive, the suboptimality ratio decreases for all configurations, but the slopes of the curves are generally shallow. The difference between $R_{target} = 0.10$ and $R_{target} = 0.05$ is particularly small, while the decrease becomes somewhat more visible between $R_{target} = 0.05$ and $R_{target} = 0.01$. Overall, this suggests that, within the evaluated setup, selecting conflicts with lower variance tends to yield a more favorable trade-off between safety and path cost, and that the total cost of enforcing stricter safety remains limited.

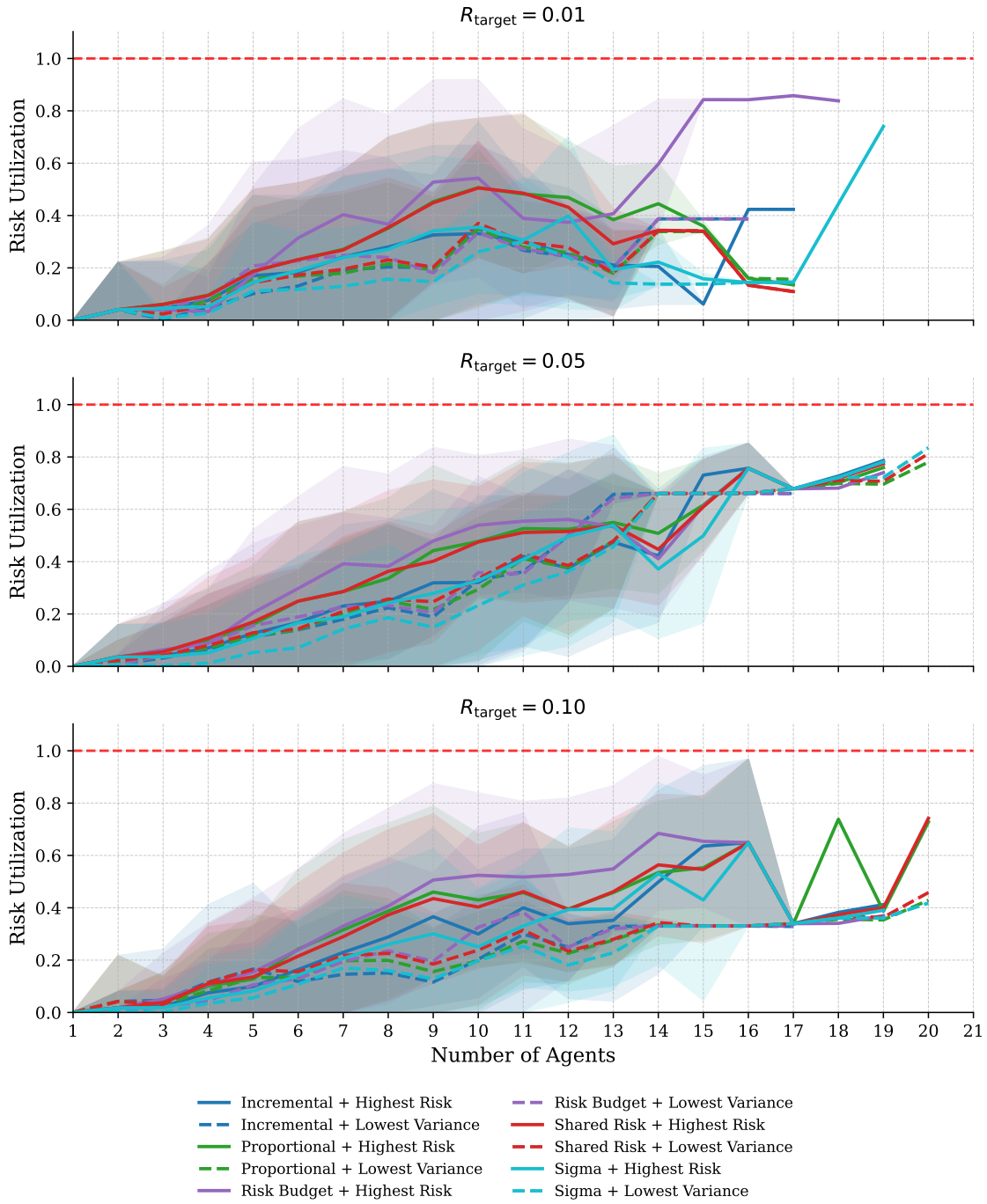

Figure 5.7: Normalized risk utilization for three global risk budgets R_{target}. Each subplot shows the mean value of R_{global}/R_{target} across solved instances, while the shaded region indicates one standard deviation across scenarios. The dashed red line at 1 marks full utilization of the available risk budget.

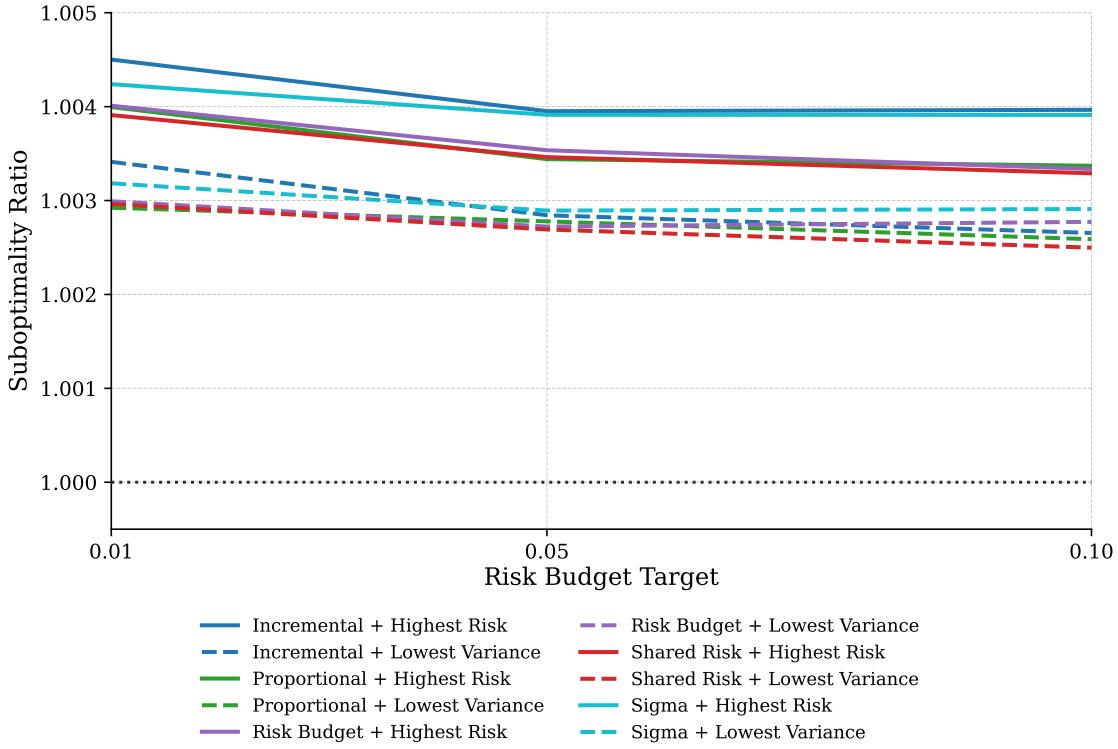

Figure 5.8: Pareto-style comparison between the global risk budget R_{target} and the suboptimality ratio, defined as SOC / ideal SOC. Lower values indicate solutions closer to the root-path baseline. Across all evaluated configurations, the variation remains small, indicating that the cost of stricter safety is modest within the tested risk-budget range.

5.2.4 Empirical Validation

For empirical validation the respective global plans were tested using Monte Carlo (MC) simulations in order to test whether their empirical execution behavior matched the probabilistic safety guarantees derived during planning. It is important to distinguish between two different notions of success in this context. Figure 5.2 reports the fraction of benchmark instances for which a plan was successfully generated within the time limit. By contrast, Figure 5.9 reports the empirical collision-free execution rate of those plans that were actually generated and subsequently simulated. The empirical figures therefore do not include planning failures, but only the subset of instances for which a complete plan was available. This explains why empirical results can still be shown at agent counts where the planning success rate is already low.

Figure 5.9 shows that the empirical collision-free execution rate remains above the required success threshold $1-R_{target}$ for all evaluated configurations. In other words, once a plan is generated, its simulated execution remains within the prescribed risk budget. This observation is reinforced by Figure 5.10, which shows the normalized empirical failure utilization,

$$\frac{\hat{P}_{\text{fail}}}{R_{target}},$$

where $\hat{P}_{\text{fail}}$ denotes the observed failure rate in the Monte Carlo simulation.

For all tested configurations, this quantity remains below 1, meaning that the observed empirical failure rate stays within the allowable failure budget. Another observation is that Figure 5.10 closely mirrors the theoretical normalized risk-utilization plot in Figure 5.7. The empirical curves follow the same overall ordering and trends, but often lie slightly below the theoretical ones, indicating that the simulated executions tend to use no more, and often slightly less, of the available risk budget than the conservative planning estimate suggests.

This consistency also extends to the qualitative behavior of the heuristics and allocation models. The lowest variance heuristic frequently yields slightly higher empirical success rates and tends to use a smaller fraction of the available risk budget than the corresponding highest risk variants, indicating somewhat more conservative behavior. At the same time, the differences remain small, and all evaluated configurations satisfy the required safety threshold with margin.

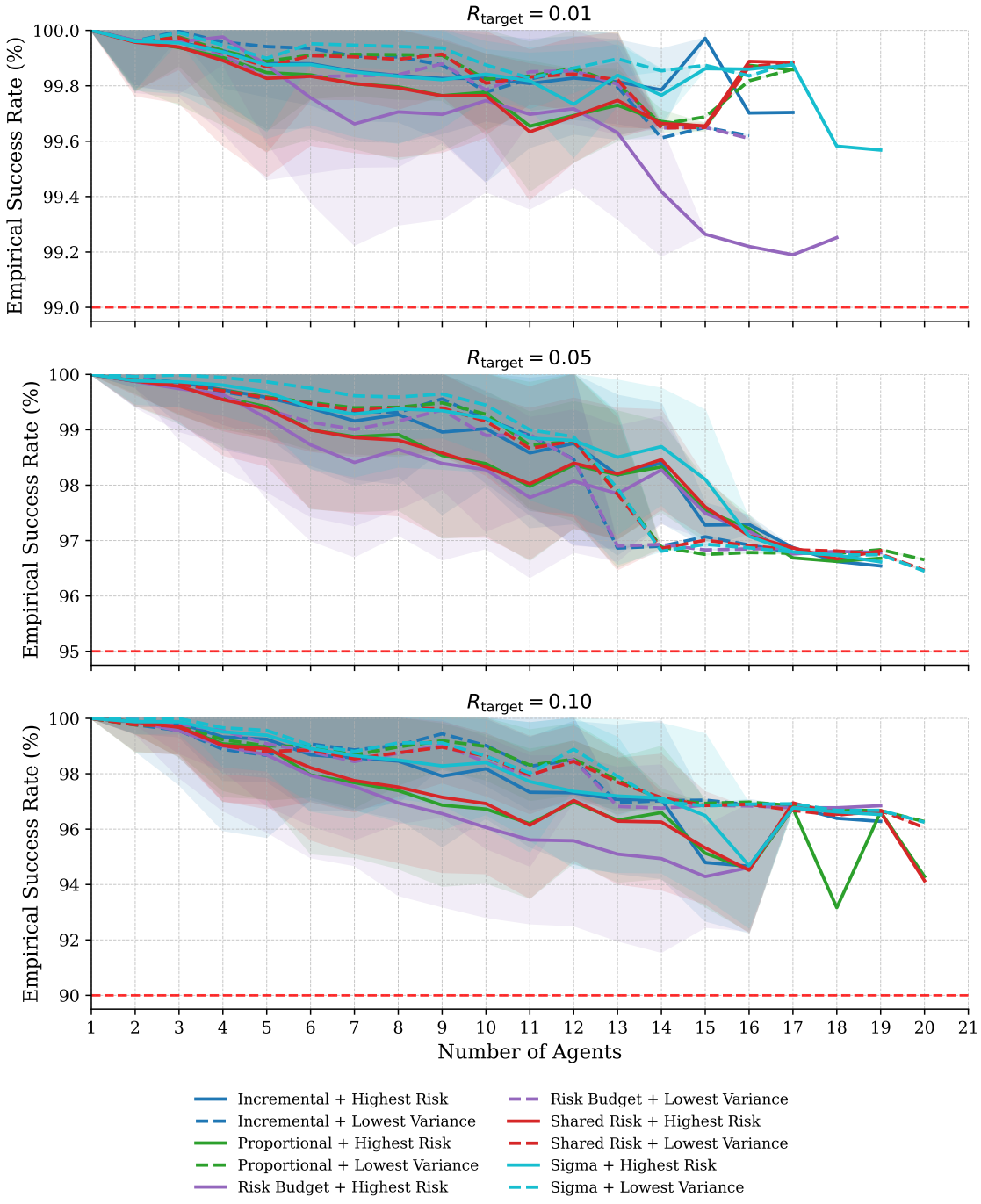

Figure 5.9: Empirical success rate from Monte Carlo simulation for three global risk budgets R_{target}. Each subplot shows the mean collision-free execution rate across solved plans, while the shaded region indicates one standard deviation across scenarios. The dashed red line corresponds to the required success threshold $1 - R_{target}$.

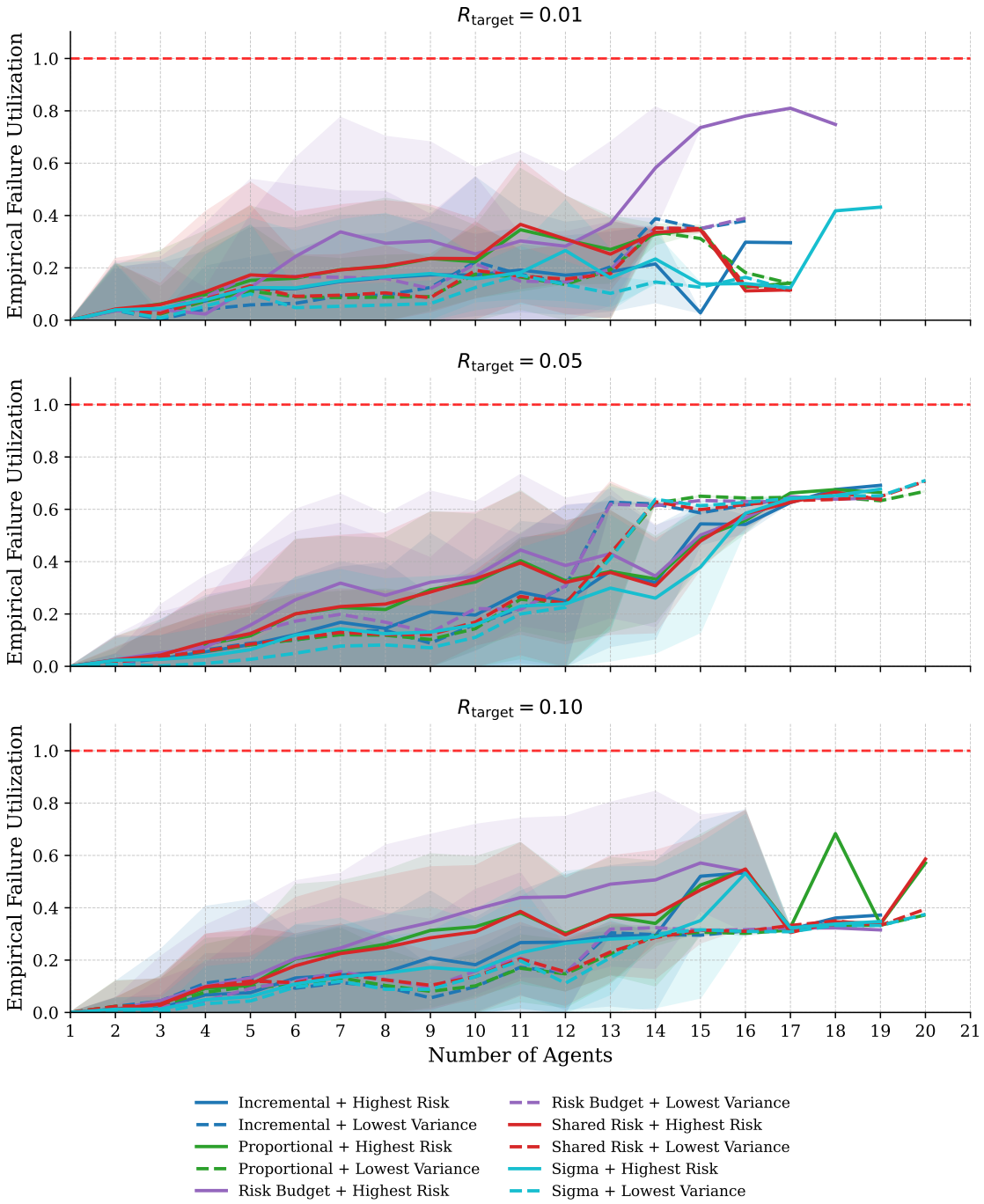

Figure 5.10: Normalized empirical failure utilization for three global risk budgets R_{target}. Each subplot shows the mean value of the observed failure rate divided by R_{target}, while the shaded region indicates one standard deviation across scenarios. The dashed red line at 1 marks full utilization of the allowable failure budget.

Turning to solution quality, Figures 5.11 and 5.12 show the empirical counterparts of the theoretical metrics in Figures 5.5 and 5.6. The empirical suboptimality ratio remains close to the corresponding theoretical curves, and the empirical makespan follows the same overall qualitative trends. This is consistent with the fact that the Monte Carlo simulations sample from the same constrained Gaussian traversal-duration model used during planning. The empirical execution therefore fluctuates around the planned mean behavior rather than introducing a fundamentally different duration model.

The same broad ordering observed in planning remains visible in simulation. The lowest variance configurations generally retain slightly better solution quality in terms of both sum-of-costs and makespan, although the differences are small in absolute magnitude. Similarly, the allocation models that performed better in the theoretical evaluation, such as Shared Risk, remain among the stronger empirical performers, while the more conservative models, such as Incremental, continue to incur slightly higher path cost. Thus, the empirical results suggest that the qualitative trade-offs observed during planning persist under stochastic execution.

Because the empirical and theoretical solution-quality curves are visually very close, Figures 5.13 and 5.14 show the relative differences directly. These plots confirm that the deviations remain small across all evaluated settings. In particular, the empirical sum-of-costs and makespan stay very close to their planned counterparts, which supports the interpretation that the probabilistic planner captures the dominant timing behavior of the executed solutions well. The remaining differences can be understood as stochastic variation around the planned mean arrival times rather than as a systematic mismatch between planning and execution.

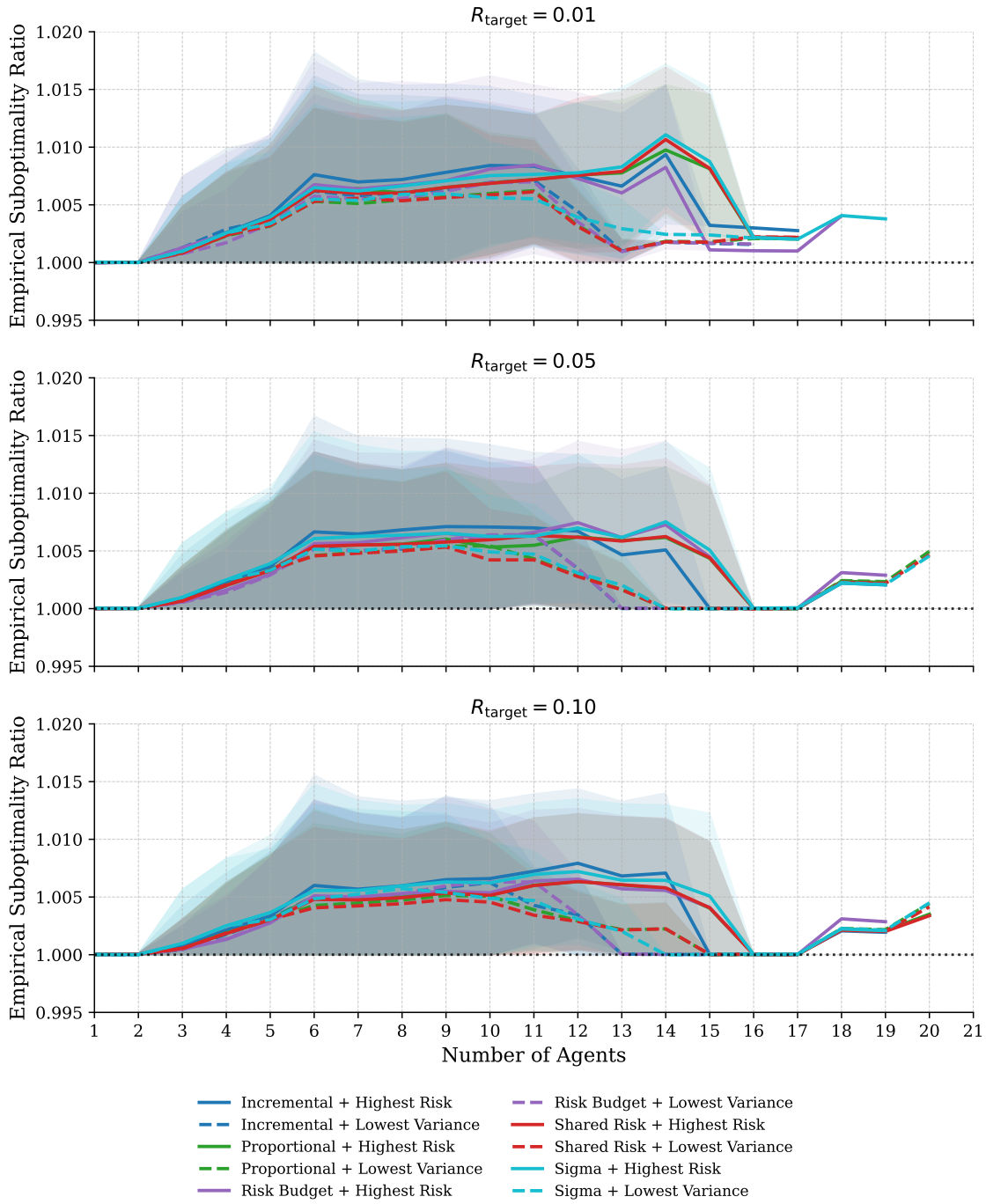

Figure 5.11: Empirical suboptimality ratio for three global risk budgets R_{target}. Each subplot shows the mean ratio between simulated SOC and the corresponding root-path baseline, while the shaded region indicates one standard deviation across scenarios. The dashed horizontal line at 1 marks the root-path baseline.

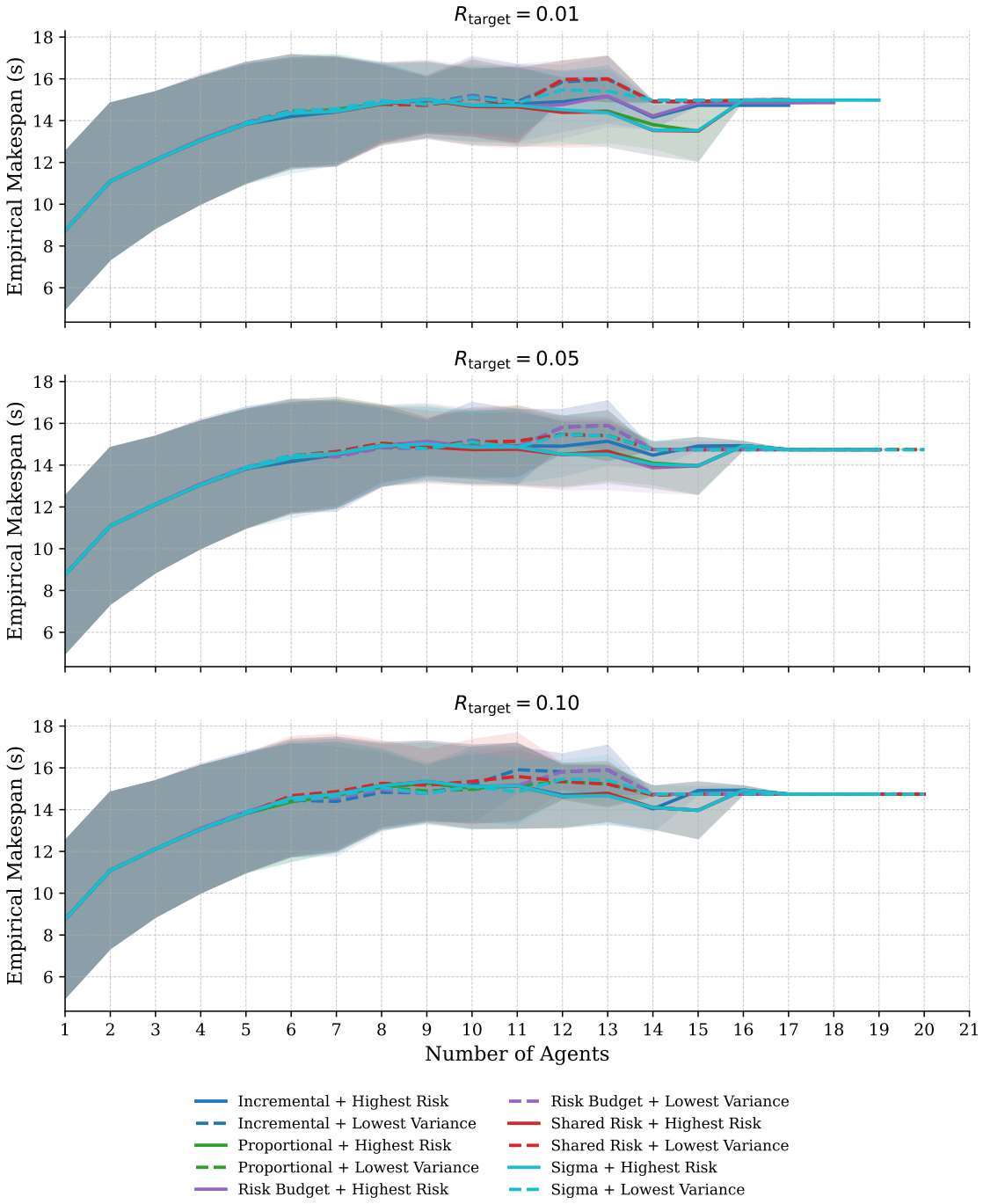

Figure 5.12: Empirical makespan for three global risk budgets R_{target}. Each subplot shows the mean simulated makespan across solved plans, while the shaded region indicates one standard deviation across scenarios.

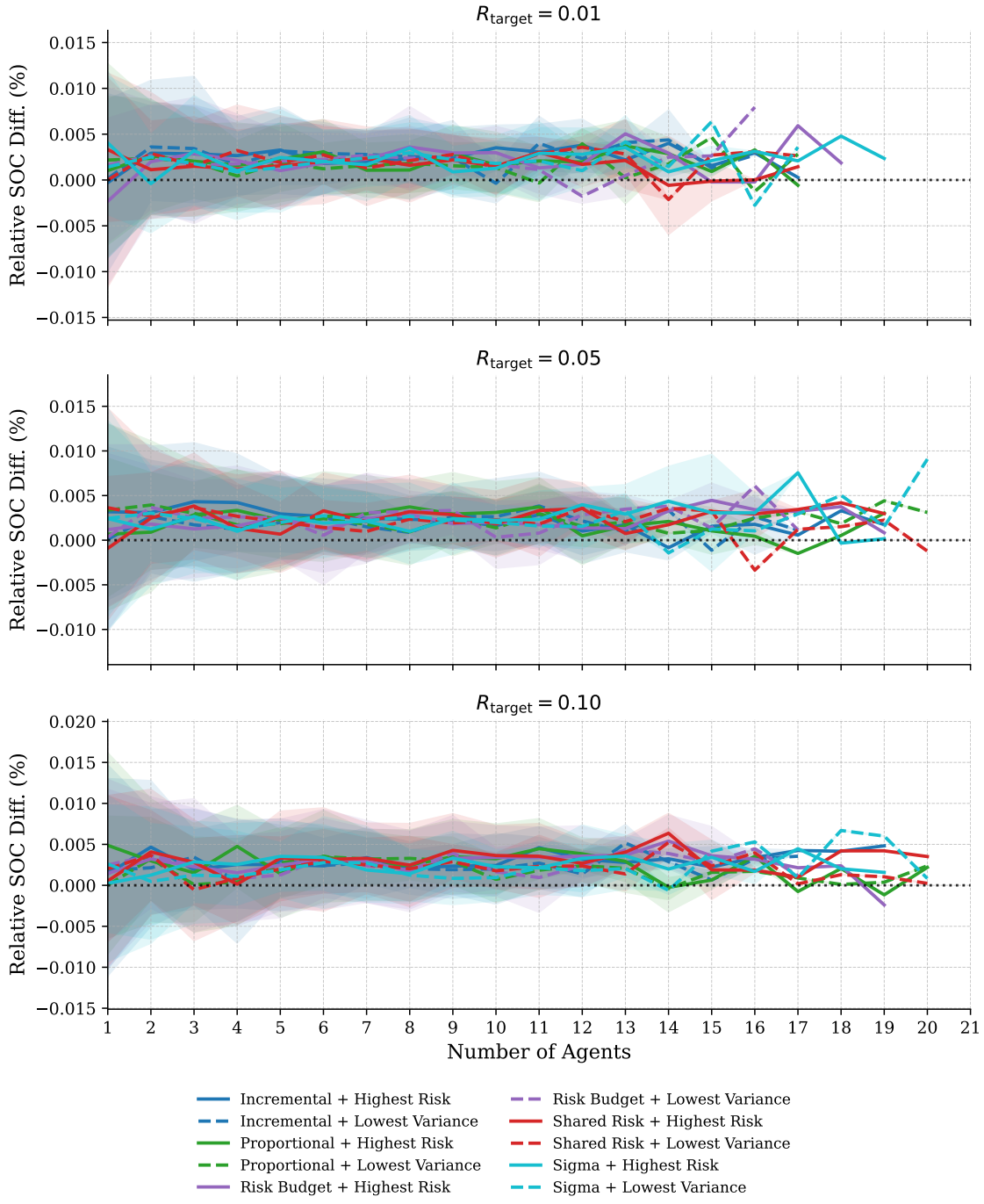

Figure 5.13: Relative difference between empirical and theoretical sum-of-costs for three global risk budgets R_{target}. Each subplot shows the mean value of $(\text{SOC}_{\text{emp}} - \text{SOC}_{\text{plan}})/\text{SOC}_{\text{plan}}$ across solved instances, expressed in percent. The shaded region indicates one standard deviation across scenarios. The dotted horizontal line at 0 indicates exact agreement.

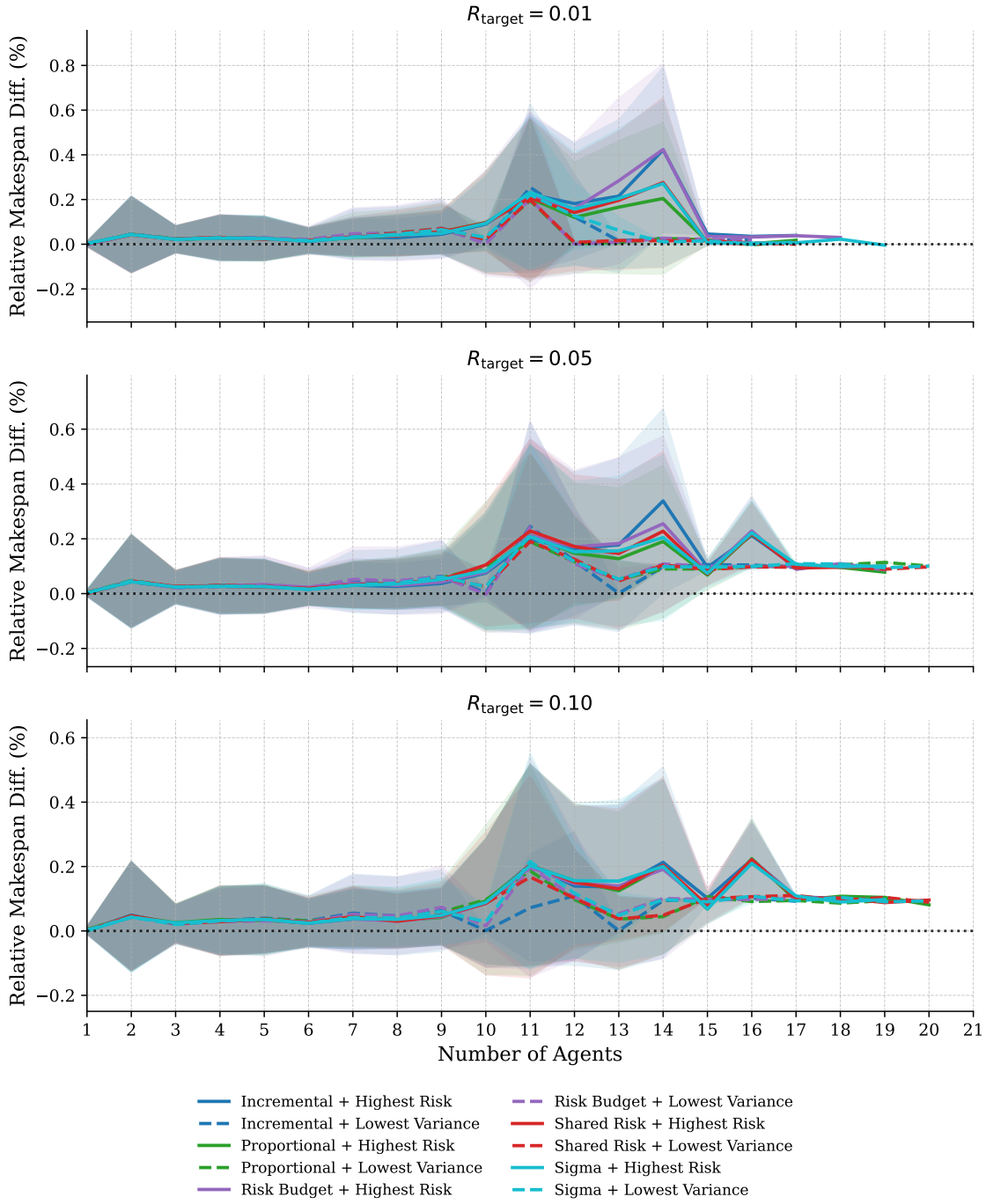

Figure 5.14: Relative difference between empirical and theoretical makespan for three global risk budgets R_{target}. Each subplot shows the mean value of $(\text{Makespan}_{\text{emp}} - \text{Makespan}_{\text{plan}})/\text{Makespan}_{\text{plan}}$ across solved instances, expressed in percent. The shaded region indicates one standard deviation across scenarios. The dotted horizontal line at 0 indicates exact agreement.

6

Discussion and Conclusion

This chapter synthesizes the findings from the experimental evaluation and discusses their implications within the broader context of probabilistically robust continuous-time Multi-Agent Path Finding. It discusses the mechanical and statistical reasons behind the observed algorithm behaviors. Section 6.1 analyzes the trade-offs between schedule optimality and computational tractability driven by the risk mitigation heuristics. Section 6.2 addresses the divergence between theoretical risk bounds and empirical execution safety. Section 6.3 evaluates the algorithmic efficiency, and unexpected behaviors of the various risk allocation models. Section 6.4 contextualizes these findings within the constraints of practical industrial deployments. Finally, Section 6.5 outlines necessary directions for future research, and Section 6.6 concludes the thesis.

6.1 Lowest Variance vs. Highest Risk

The experimental results indicate that the choice of risk mitigation heuristic—specifically, lowest variance versus highest risk—has a more substantial impact on overall system performance than the specific risk allocation model.

The lowest variance heuristic consistently demonstrates a lower sub optimality ratio in the Pareto-style comparison in Figure 5.8. However, prioritizing the lowest variance seems to force the algorithm to expand significantly more search nodes when fewer number of instances have been solved, as illustrated in Figure 5.4. This heuristic leads to a steeper decline in scalability and a lower overall success rate within the 60 seconds timeout limit (Figure 5.2).

An important divergence is also observed between the sub optimality ratio (representing the global Sum-of-Costs) and the makespan, detailed in Figure 5.5 and Figure 5.6. The lowest variance approach optimizes the overall SOC more effectively at the expense of a slightly extended makespan for the slowest agent. Because the standard practice within CCBS literature evaluates path efficiency primarily through the SOC, the lowest variance heuristic remains the superior choice for theoretical solution quality, despite its computational cost. If the timeout limit had been extended beyond 60 seconds, the results would possibly have been even more

in favor of the lowest variance heuristic. Though this was not tested, it seems plausible since the lowest variance heuristic does not target the intersections with the biggest risks. Therefore, when choosing an intersection, that delay will not reduce the global risk as much, and it will require CCBS to branch much more.

6.2 Theoretical vs. Empirical Risk

The empirical validation via Monte Carlo simulations demonstrates a strong structural alignment with the theoretical global risk bounds. As expected, the theoretical risk computed via the Union Bound serves as a strict, conservative upper limit. The simulated, true collision risk is consistently lower across all evaluated scenarios (Figure 5.9). This divergence is because of the Union Bound which is what makes our approach robust. It approximates global risk by additively summing the marginal probabilities of local conflicts. However, during actual execution, collision risks are correlated. A delay resolving one conflict often naturally prevents a subsequent interaction entirely. By neglecting these joint probabilities, the theoretical model functions as a deliberately pessimistic upper bound, ensuring the true empirical failure rate consistently remains below the calculated limit.

The data indicates a substantial gap between the accumulated Union Bound risk and the user-defined R_{target}, particularly in instances with fewer agents, as shown in the risk utilization plots (Figure 5.7). This under utilization of the risk budget is largely attributed to the sparsity of the empty-16-16 benchmark map. The open spatial environment naturally limits the frequency of intersecting trajectories, resulting in inherently low collision probabilities before any risk mitigation is actively applied.

6.3 Comparison of Risk Allocation Models

Analyzing the specific risk allocation models reveals several counter-intuitive behaviors. Tightening the global risk target R_{target} from 0.10 to 0.01 restricts the number of solved instances somewhat (Figure 5.2), particularly for the models using highest risk. Yet the degradation in actual solution quality remains surprisingly marginal in the simulations shown in Figure 5.13. This can be mathematically attributed to the shape of the Gaussian distribution's tail. Satisfying a strict 0.01 risk threshold compared to a 0.10 threshold requires moving significantly further along the time-axis. Consequently, the low-level solver must apply a proportionally large local time delay to the specific agents involved in the conflict to secure those final few percentages of theoretical safety. However, while this delay is substantial relative to the local interaction, the absolute physical time added is exceedingly small when compared to the total execution time of the agents' complete trajectories. When this localized penalty is added to SOC, its overall impact becomes negligible. This explains why the successfully generated plans show nearly identical sub optimality ratios despite the algorithm operating under a stricter risk budget.

Furthermore, the Risk Budget Limitation model (Section 4.2.3) utilizes the allow-

able risk the best, as initially hypothesized. However, we thought it would use it much more, and this dynamic utilization did not translate to a substantially lower SOC when compared to other models (Figure 5.13). Also, the Sigma Limitation model (Section 4.2.2) demonstrated stability and performed competitively alongside the dynamic allocation models, which was hypothesized to be overly conservative and computationally slow. We believe that the Sigma Limitation model competes because of its rigid constraint generation. By applying a static buffer to targeted conflicts, it efficiently prunes the search space, forcing the algorithm to decisively abandon highly trafficked routes in favor of alternative paths.

The strong correlation between theoretical (Figure 5.5) and simulated solution quality (Figure 5.11) validates the algorithmic design choice to evaluate the SOC using the expected traversal duration (the mean) during the search process.

However, an anomaly in Figure 5.11 that can be seen between 13–17 agents shows that certain robust solutions occasionally achieve the exact same SOC as the initial root node, accompanied by a smaller standard deviation after 13 agents. This anomaly is heavily influenced by a survivorship bias introduced by the 60-second planning timeout. In highly trafficked scenarios, the algorithm fails to resolve complex conflicts within the given time, meaning that the only instances successfully returning a plan are those where the randomized start and goal positions fortuitously align to allow near-optimal routing. Consequently, the sub optimality ratio reflects only these statistically "lucky" scenarios, converging to 1.0 and eliminating the variance because there are no other solutions.

6.4 Practical Applicability and Timeout Constraints

The computational overhead introduced by continuous-time chance constraints necessitated an extended 60-second planning timeout, doubling the 30-second benchmark standard used in the evaluation of deterministic CCBS [9]. The primary justification for probabilistically robust planning is that proactively guaranteeing a high success rate eliminates the need for reactive online replanning. However, in real-world industrial deployments, autonomous systems may encounter dynamic, unmodeled disturbances (such as unexpected obstacles or hardware failures) that still require rapid replanning. In such scenarios, the extended computation times required by this framework present a practical limitation for applications.

6.5 Future Work

While this thesis successfully establishes a functional approach for probabilistically robust continuous-time MAPF, several suggestions remain for both empirical validation and theoretical advancement. The primary directions for future research are categorized into extended benchmarking and algorithmic optimization.

6.5.1 Extended Empirical Evaluation

To further validate the proposed algorithms, future research should expand the empirical evaluation across a broader set of benchmark environments. The current simulations were conducted on an obstacle-free grid, which inherently limits the frequency of complex, multi-agent bottlenecks. Testing the algorithms on constrained maps featuring narrow corridors and forced intersections would provide deeper insights into how the dynamic risk allocation models perform under severe spatial restrictions.

Additionally, while this evaluation focused on an internal comparison of risk mitigation heuristics, future work must rigorously benchmark the proposed probabilistic models against established external baselines. Comparing these models directly against standard deterministic CCBS and worst-case T-Robust formulations would explicitly quantify the efficiency gains achieved by using chance constraints over absolute deterministic bounds.

Finally, a dedicated sensitivity analysis regarding the planning timeout limit is essential. Investigating the system's performance under stricter operational constraints (e.g., 30 seconds) versus extended planning horizons (e.g., 90 to 120 seconds) would better define the scalability bottleneck and determine the computational threshold required to resolve dense traffic scenarios.

6.5.2 Algorithmic Optimality via 3-Way Branching

Beyond empirical benchmarking, the most significant theoretical suggestion for future development lies in resolving the mathematical optimality of the search process. The heuristic risk allocation models proposed in this thesis successfully generate safe plans, but they sacrifice formal optimality guarantees due to the non-convexity of continuous-time chance constraints. A promising direction is the development of an optimal solver utilizing a continuous 3-way branching scheme, inspired by discrete p-Robust CBS [10].

In this proposed branching scheme, the high-level solver would not attempt to immediately resolve the delays of a conflict. Instead, when a probabilistic conflict is detected between two agents, the algorithm would split the search tree into three distinct branches:

1. Agent 1 is prohibited from crossing the intersection during a specified interval, I_1.

2. Agent 2 is prohibited from crossing the intersection during a specified interval, I_2.

3. A positive constraint that actively accepts the spatial interaction, enforcing that the agents' relative traversal durations must remain within the acceptable statistical tail of the risk distribution ($\Delta T \leq \delta_{max} - d$).

The innovation lies in the third branch. Rather than optimizing this branch locally, the search would continue to expand, treating the accepted risk not as a fixed delay, but as an open, bounded interval. Because the acceptable risk is defined by the tail of the Gaussian cumulative distribution function (CDF), the condition established by the third branch could form a convex polytope. Geometrically, the intersection of these bounds forms a continuous, multidimensional feasible region.

As the search tree deepens and accumulates these third-branch constraints for every interaction across the map, it constructs a global system of inequalities. Once the topological sequence of the agents is fully locked by the search tree, the previously intractable, non-convex path finding problem collapses into a strictly convex geometric space.

Consequently, finding the optimal traversal durations within this convex region is no longer an NP-hard combinatorial task. The accumulated intervals formulate a linear programming problem, hopefully meaning that the optimal SOC that satisfies the global risk target R_{target} can be mathematically solved in polynomial time. While constructing the complete topological tree introduces computational overhead, this method presents a pathway to bridging the gap between probabilistically robust planning and optimality in continuous time.

6.6 Conclusion

The primary objective of this thesis was to bridge the gap between theoretical continuous-time Multi-Agent Path Finding (MAPF$_R$) and the uncertainties of real-world industrial deployments. By shifting from deterministic worst-case bounds to a probabilistically robust framework, this research successfully demonstrates how autonomous systems can maintain both schedule optimality and rigorous safety guarantees.

This thesis answers the two foundational research questions. Regarding the formal modeling of traversal duration uncertainty (RQ1), the approach models individual edge traversals as Gaussian distributions, bounded by a 3σ kinematic limit to ensure realistic values. As these uncertainties accumulate along an agent's path, the system isolates potential conflicts and computes the relative arrival time difference (ΔT) between agents. The local risk of collision is then evaluated by calculating the probability mass of ΔT falling within the deterministic geometric unsafe interval using the Gaussian Cumulative Distribution Function (CDF).

To find solutions that satisfy a user-defined confidence level (RQ2), the framework uses the Union Bound to safely accumulate these local risks into a global system risk metric. The Continuous Conflict-Based Search (CCBS) algorithm was subsequently modified to enforce this global risk budget via dynamic mean-shifting. By using risk mitigation heuristics and risk allocation methods (such as Risk Budget Limitation and Sigma Limitation), the system assigns a local acceptable risk threshold p to specific interactions. Applying the inverse CDF translates this probability into a

precise delay d, expanding the constraint intervals passed to the low-level CSIPP planner to safely separate the agents' trajectories.

The empirical validation via Monte Carlo simulations shows that this probabilistic approach works. The simulated execution failure rates consistently respected the theoretical Union Bound guarantees. However, the evaluation also highlights an unavoidable trade-off between schedule optimality and algorithmic tractability. Resolving continuous-time chance constraints via heuristic mean-shifting causes computational bottlenecks for the high-level solver with many agents.

Despite the current computational limitations of the high-level search tree, the theoretical framework demonstrates industrial readiness. By mathematically bounding the risk of collision rather than planning for impossible worst-case extremes, the proposed framework guarantees a predictable throughput, establishing a viable foundation for the next generation of automated logistics.

Bibliography

[1] J.-M. Alkazzi and K. Okumura, "A comprehensive review on leveraging machine learning for multi-agent path finding," *IEEE Access*, vol. 12, pp. 57 390–57 409, 2024. DOI: 10.1109/ACCESS.2024.3392305.

[2] W. J. Tan, X. Tang, and W. Cai, "Robust multi-agent pathfinding with continuous time," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 34, 2024, pp. 570–578.

[3] Wikipedia contributors, *Multi-agent pathfinding — Wikipedia, the free encyclopedia*, [Online; accessed 6-June-2026], 2025. [Online]. Available: https://en.wikipedia.org/wiki/Multi-agent_pathfinding.

[4] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," *Artificial Intelligence*, vol. 305, p. 103 662, 2022. DOI: 10.1016/j.artint.2022.103662.

[5] N. R. Sturtevant, "Benchmarks for grid-based pathfinding," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 144–148, 2012, Dataset available at: https://movingai.com/benchmarks/mapf/index.html.

[6] A. Combrink, "Advances in multi-agent path finding: Scalable lifelong planning and optimal continuous-time solutions," Licentiate thesis, Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden, 2025. [Online]. Available: https://research.chalmers.se/publication/549458.

[7] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 27, 2013, pp. 1443–1449.

[8] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015, ISSN: 0004-3702. DOI: 10.1016/j.artint.2014.11.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370214001386.

[9] A. Andreychuk, K. Yakovlev, E. Boyarski, and R. Stern, "Improving continuous-time conflict based search," *Proceedings of the International Symposium on Combinatorial Search*, vol. 12, pp. 145–146, 2021. DOI: 10.1609/socs.v12i1.18564.

[10] D. Atzmon, R. Stern, A. Felner, N. R. Sturtevant, and S. Koenig, "Probabilistic robust multi-agent path finding," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 29–37.

[11] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, R. Barták, and E. Boyarski, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, AAAI Press, Sep. 2021, pp. 151–158. [Online]. Available: https://ojs.aaai.org/index.php/SOCS/article/view/18510.

[12] GeeksforGeeks. "A* search algorithm." Accessed: 2026-02-06. [Online]. Available: https://www.geeksforgeeks.org/a-search-algorithm/.

[13] GeeksforGeeks. "Dijkstra's algorithm." Accessed: 2026-02-06. [Online]. Available: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/.

[14] K. Kasaura, M. Nishimura, and R. Yonetani, "Prioritized safe interval path planning for multi-agent pathfinding with continuous time on 2d roadmaps," in *IEEE Robotics and Automation Letters*, 2022.

[15] T. T. Walker and N. R. Sturtevant, "Collision detection for agents in multi-agent pathfinding," in *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, 2023.

[16] C. Ericson, *Real-Time Collision Detection*. Boca Raton, FL: CRC Press, 2005.

[17] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 5628–5635. DOI: 10.1109/ICRA.2011.5980306.

[18] A. Li, Z. Chen, D. Harabor, and M. Vered, *Cbs with continuous-time revisit*, 2025. arXiv: 2501.07744 [cs.MA]. [Online]. Available: https://arxiv.org/abs/2501.07744.

[19] A. Combrink, S. F. Roselli, and M. Fabian, *Optimal multi-agent path finding in continuous time*, 2025. arXiv: 2508.16410 [cs.MA]. [Online]. Available: https://arxiv.org/abs/2508.16410.

[20] D. Atzmon, R. Stern, A. Felner, G. Wagner, R. Barták, and N.-F. Zhou, "Robust multi-agent path finding and executing," *Journal of Artificial Intelligence Research*, vol. 67, pp. 549–579, 2020.

[21] D. Bertsimas and M. Sim, "The price of robustness," *Operations research*, vol. 52, no. 1, pp. 35–53, 2004.

[22] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust optimization*. Princeton university press, 2009.

[23] A. Charnes and W. W. Cooper, "Chance-constrained programming," *Management science*, vol. 6, no. 1, pp. 73–79, 1959.

[24] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, 2011.

[25] A. Theurkauf, J. Kottinger, N. Ahmed, and M. Lahijanian, "Chance-constrained multi-robot motion planning under gaussian uncertainties," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 835–842, Dec. 2023. DOI: 10.1109/

LRA.2023.3337700. [Online]. Available: https://arxiv.org/abs/2303.11476.

[26] Wikipedia contributors, *Boole's inequality — Wikipedia, the free encyclopedia*, [Online; accessed 12-March-2026], 2026. [Online]. Available: https://en.wikipedia.org/wiki/Boole%27s_inequality.

[27] Wikipedia contributors, *Gamma distribution — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Gamma_distribution, [Online; accessed 23-March-2026], 2026.

[28] Wikipedia contributors, *Normal distribution — Wikipedia, the free encyclopedia*, [Online; accessed 22-June-2026], 2026. [Online]. Available: https://en.wikipedia.org/wiki/Normal_distribution.

[29] "Industrial trucks — Safety requirements and verification — Part 4: Driverless industrial trucks and their systems," International Organization for Standardization, Geneva, Switzerland, Standard, Jun. 2023. [Online]. Available: https://www.iso.org/obp/ui/en/#iso:std:iso:3691:-4:ed-2:v1:en.

[30] Wikipedia contributors, *68–95–99.7 rule — Wikipedia, the free encyclopedia*, [Online; accessed 22-June-2026], 2026. [Online]. Available: https://en.wikipedia.org/wiki/68%E2%80%9395%E2%80%9399.7_rule.

[31] GeeksforGeeks, *Cumulative distribution function*, https://www.geeksforgeeks.org/engineering-mathematics/cumulative-distribution-function/, Last Updated: 23 Jul 2025, Accessed: 21 Jun 2026, 2025.

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

CHALMERS
UNIVERSITY OF TECHNOLOGY