Efficient algorithms and representations for chance-constrained mixed constraint programming

by

Cheng Fang

Submitted to the Department of Aeronautics and Astronautics in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 2021

Author . . . .
Department of Aeronautics and Astronautics
August 17th, 2021

Certified by . . . .
Brian C. Williams
Professor
Thesis Supervisor

Accepted by . . . .
Jon How
Professor, Aeronautics and Astronautics, Chair, Graduate Program Committee

Efficient algorithms and representations for chance-constrained mixed constraint programming

by

Cheng Fang

Submitted to the Department of Aeronautics and Astronautics
on August 17th, 2021, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Resistance to adoption of autonomous systems in comes in part from the perceived unreliability of the systems. The concerns can be addressed by deploying decision making algorithms that operate in the presence of uncertainty.

The default approach is to optimise expected utility given probabilistic descriptions of uncertainty. However, such approaches become problematic when the cost of failure is difficult to define, for example when imposing constraints on remote science missions. Without well-defined costs of failure, it is difficult to balance the risks of failure against the rewards of success. This motivates an alternative approach, in which we define what it means to fail, and look for plans with the highest reward while limiting the probability of failure.

The alternative approach thus explicitly imposes a set of constraints required for success, and provide upper-bounds on the probability of violating such constraints. A chance-constrained mixed logical-linear program (CC-MLLP) is a natural formulation, allowing for the specification of linear and logical constraints, with probabilistic continuous variables. The formalism can be used to describe problems ranging from automonous underwater vehicle path planning, to network routing under uncertainty.

My thesis addresses shortcomings in current approaches to CC-MLLP. In particular, I focus on the problem of computation speed for solving CC-MLLPs, and the problem of accurate uncertainty representation.

While naive encodings of CC-MLLPs can be solved with generalised solvers, the solution time may be unreasonable. In this thesis, I study architectures to speed up solutions by partitioning CC-MLLPs into the discrete and continuous portions.

In order to provide faster solutions, I investigate methods for speeding up the solutions to the continuous chance-constrained linear programs. Further, by exploiting the new solution methods, I develop techniques for guiding the discrete decision making portion of the problem. The resulting algorithm achieves 10 times speed up over prior approaches on autonomous path planning benchmarks.

Lastly, current chance-constrained approaches require distributional descriptions of uncertainty. In this thesis, I consider the problem of deriving uncertainty bounds from data, which cover a required proportion of outcomes with a quantifiable amount of confidence. In particular, I provide bounds for real world scenarios which feature a finite number of executions. This is demonstrated on MBTA Red Line subway schedules.

Thesis Supervisor: Brian C. Williams
Title: Professor

Acknowledgments

This thesis provides closure to a very long journey. I have been very fortunate through out my experience in my graduate studies. I would like to acknowledge the support I have received from my peers, my thesis committee, and the community in general.

This thesis would not have been possible without the generosity of the NSF, DARPA, and Airbus. I would like to think that the advances made in this thesis are of interest not just to particular governments or corporations, but generally to people who would like some guarantees on the safety of automatically generated plans.

I would also like to thank my committee members and readers for their support and advice throughout. I would like to thank Howie Shrobe for his advice on what is important in a research project. I would like to thank Leslie Pack Kaelbling for her amazing ability to tease out the technical points of interest in any problem. I would like to thank Patrick Winston for his help in shaping the narrative at the outset of this thesis. Lastly, I would like to thank my adviser, Brian Williams, for his patient and rigorous review and discussions of ideas throughout my time at MIT.

I would also like to thank my colleagues at MERS, who have been such a huge part of my time at MIT. I would like to acknowledge Pedro for being an amazing sounding board for serious academic discussions and ridiculous puns, Steve for enduring antipodean humour, Andrew for all the PSTN and Rowan Atkinson chats, and Eric for carrying the infrastructure of the lab on his back. I would also like to thank Peng for being the best shipmate, and hope that Peng's Pingo off the coast of California will be an officially recognised landmark. I would also like to thank Jingkai for dealing with my nitpicking during our time together in the lab. I would also like to thank the postdocs who have come through MERS - Erez, Tiago, Christian, and Ashkan - each of whom has given me professional and academic advice. Thanks also to Hiro, who kick-started a lot of the chance-constrained work at MERS, and hosted me at JPL. Lastly, thanks to everyone at MERS over the years - David, Dan, James, Enrique, Szymon, Jonathan, Ben, Sang, Sylvia, Nikhil, Yuening, Cyrus, Zach, Marlyse, Sungkweon, Nick, Big Matt, Little Matt, Spencer, Jacbo and Allen - for putting up with me as social chair and picking terrible movies for movie nights. I hope my successor will carry on the tradition of Dominos and cringey movies in the MERS cave.

I would like to thank Anthony, Hang and Wei. Who knew sitting together on the bus to Target that one time during orientation would lead to such an enduring friendship? Also thanks to Anna, Anne, and Chris, for, I dunno, mateship and the ANZAC spirit in Boston. I would also like to thank Simon, Daniel, Daniel, and Qi, for the support they've given me across the seas.

It goes without saying that I owe everything to my mother Qin and my father Xuan. I am thankful for your love, trust, and support, and I know I couldn't be where I am today without your sacrifices.

I would also like to acknowledge Lele. For being lovely.

Contents

1 Introduction **17**

1.1 Challenges 19

1.1.1 Speed of solution 20

1.1.2 Uncertainty representation 20

1.2 Approach 20

1.2.1 Framework for CC-MLLP 21

1.2.2 Cutting planes for CC-LPs 21

1.2.3 Conflict Extraction for CC-MLLPs 22

1.2.4 Data-driven risk allocation 22

1.3 Summary 23

2 Background **25**

2.1 Efficient constraint programming 25

2.2 Chance-constrained programming 27

2.3 Robust uncertainty representation 28

3 Problem Description and Approach **31**

3.1 Problem Definition: CC-MLLP 31

3.2 Example CC-MLLPs 33

3.3 High Level Solution Approach 35

3.3.1 Efficient Subsolver 37

3.3.2 Pruning Using Relaxations 38

3.3.3 Conflict Guided Search 39

3.3.4 High Level Algorithm 39
3.3.5 Relationship to Prior Work 41
3.4 Summary 41

4 Cutting-planes for CC-LP 43
4.1 Introduction 44
4.2 Risk Allocation 47
4.3 Approach 48
4.3.1 Conflict-Based Risk Allocation 49
4.3.2 Related work 52
4.4 Experiments 54
4.5 Summary 59

5 Chance Constrained Mixed Logic Linear Program 61
5.1 Relaxed Subproblem 62
5.2 Conflict Extraction 63
5.2.1 Conflict Extraction for Relaxed CC-LP 63
5.2.2 Conflict Extraction for CC-LP using Cutting Planes 64
5.3 Correctness of Conflicts 67
5.4 Empirical Results 70
5.4.1 Simple Path Planning Benchmark 71
5.4.2 Scarborough Oceanography Benchmark 73
5.5 Summary 75

6 Data-driven Uncertainty Representation 77
6.1 Introduction 78
6.2 Problem formulation 79
6.3 Data-driven chance-constrained uncertainty sets 83
6.3.1 Bounds on underlying distribution 83
6.3.2 Bounds for finite executions 84
6.4 Experiments 87

6.4.1 Correctness of estimates 87
6.4.2 Effect on STNU scheduling 89
6.5 Summary 92

7 Conclusion 95
7.1 Future Work 96

List of Figures

3-1 Example AUV path planning over safe regions. The safe regions are shaded in blue. We can not travel directly between two points in r_1 and r_3, as demonstrated by the red path, because it may cut through an unsafe region. Instead, vehicles must transition between two regions by making a stop in r_2, which is the intersection of both r_1 and r_3. . . 34

3-2 Topology for the EdgeCT network. 35

3-3 Decomposition of CC-MLLP. 36

4-1 Example AUV bottom following mission. 45

4-2 Convergence of the quality of path for Map 17 in the the benchmark set. Utility is the sum of the depth of the vehicle, simulating a bottom following mission, such that a lower utility is better. 59

5-1 Simple 2D path planning benchmark. 71

5-2 Best candidate quality as a function of time for the simple benchmark problem, for a 20 time step plan. 74

5-3 Scarborough Oceanography map. 75

6-1 Subway scheduling example. 79

6-2 Probabilistic guarantee from bounds on quantiles. 81

List of Tables

4.1 Run time comparisons between CBRA and CRA for infeasible 3D AUV maps, with number of cuts found by CBRA. 57

4.2 Run times for feasible 3D AUV maps, for CBRA solution within 5% of the optimal, total time for optimality using the 5% solution as a starting point for CRA, and optimality using just CRA. 58

5.1 Time in seconds to prove optimality for the simple benchmark problem. 72

5.2 Number of nodes expanded for the simple benchmark problem. . . . 72

5.3 Number of full CCLPs solved for the simple benchmark problem. . . 72

5.4 Time in seconds to the first solution for the simple benchmark problem. 73

6.1 Table of empirical results for different traversal datasets. For each entry, the first number is the average number empirical proportion of traversals larger than the bound which should be smaller than the specified ϵ, and the second number is the empirical proportion of incorrect bounds which should be smaller than the specified α. The datasets cover the segments: 70063-70065 from Davis to Porter; 70065-70067 from Porter to Harvard; and 70067-70069 from Harvard to Central. . 90

6.2 Table of empirical results for traversal datasets. MBTA Redline segments: 70069-70071 from Central to Kendall; and 70071-70073 from Kendall to MGH. 91

6.3 Performance of Example 5 schedules optimised with upper bounds on traversal derived with different methods. High failure schedules are those which empirically fail more than 10% of the time, and thus do not meet the required risk bound. 92

Chapter 1

Introduction

Advances in hardware and algorithms have led to impressive developments in artificial intelligence (AI). High profile deployments of AI to solve challenging problems have inspired diverse industries to turn to automated systems. For example, aircraft construction, financial trading and medical procedures have adopted intelligent systems to varying degrees.

One of the strengths of automated systems lie in their ability for quantitative reasoning. In many applications, we may impose constraints on the desired output plans, reflecting collision avoidance, power constraints, and timing constraints. While manual decision making given such constraints is difficult, there are natural encodings of constrained problems for automated decision making systems. Such automated systems solve increasingly complex systems, providing highly optimised plans.

However, in the presence of uncertainty, optimal plans may be brittle. In the case of path planning for autonomous vehicles, the shortest distance plan may lead to the vehicle travelling too close to no-go zones. Given relatively small disturbances, the vehicle may then be lost in dangerous areas. In the case of network routing under uncertainty, the optimal plan may lead to placing a large amount of tasks on the network. When the network capacity fluctuates, important tasks may not receive the required quality of service, due to insufficient bandwidth on the network.

One critical aspect of real-world autonomous decision making is thus the ability to specify the conditions under which plans may be considered successful. We would like to specify constraint on the outcomes of following the plan set out by an autonomous agent. For example, in the AUV example, we would like to constrain the location of the vehicle to be above the seafloor for safety. In the network routing problem, we would like to require that the bandwidth reserved for high importance flows be sufficient.

Constraint programming is a paradigm for decision making with constraints on aspects of the solutions [47]. Constraint programming is widely applicable to autonomous systems, with applications in logistics problems [50], robotics [8], among others. However, simply being able to specify constraints and provide plans which satisfy the specifications is not enough. An autonomous agent must also be able to deal with the uncertainty inherent in the real world.

The traditional approach of planning under uncertainty focuses on optimising the performance of the plan, averaging over the uncertain outcomes. However, for applications in which the cost of failure is not well-defined, for example path planning for one-of-a-kind autonomous underwater vehicles, we would instead like to provide guarantees of the correctness of output plans. In the context of uncertainty, we may increase the trust that mission controllers place in autonomous planning capabilities by providing guarantees on the chance of success with respect to user specified constraints.

Intuitively, this is done by building additional safety margins into the automated plans. As humans, we naturally add safety buffers when we are acting in response to uncertainty. When driving, we have a minimum safe distance between us and the car ahead, in order to allow time to react to emergencies. When scheduling, we typically allocate additional time for travel, especially when we know the traffic conditions are uncertainty.

Humans thus achieve safety by allocating safety buffers to account for uncertainty. This can be adapted for automated decision making. The autonomous agent can quantify the amount of risk taken on for each decision, and make sure the risk incurred over the span of the plan is less than a specified tolerance. Such approaches, in which the probability of failure is bounded, is known as chance-constrained planning.

The constraint programming community have considered decision making under uncertainty with guarantees on the probability on success, in the family of approaches under stochastic constraint programming [58]. However, the stochasticity involved is primarily discrete. The work thus provides a good way of considering different modes of operation, for example deciding whether to deploy an AUV depending on the likelihood of interesting features in a region. However, the work is less applicable for continuous uncertainty, for example when the position of an AUV is affected by underwater currents.

In the constraint programming community, one way of capturing constraints over continuous and logical states is the mixed logical linear program (MLLP). The formalism allows different linear constraints to be applied when different logical conditions are satisfied. In this thesis, I extend the formalism by incorporating continuous probabilistic uncertainty in the linear constraints. I apply a further constraint on the probability of success, and define the chance-constrained mixed logical linear programs (CC-MLLP).

In this thesis, I advance the state of the art in chance-constrained mixed logical linear programming in order to address a lack of trust in automation. The primary challenges addressed are: 1) efficiency of solution;and 2) correctness of data representations.

I address these issues through two primary thrusts. I focus on efficient solutions to chance-constrained constraint programming, in order to allow such approaches to scale to real world problems. I also focus on efficient representations of uncertainty to be used with chance-constrained approaches, allowing data-driven descriptions of uncertainty with guarantees against over-fitting.

1.1 Challenges

While chance-constrained mixed logical linear programming is a promising approach, there are several challenges to address. Prominent concerns among these include: the speed of solution, and the correctness of the uncertainty representation.

1.1.1 Speed of solution

Current chance-constrained programming methods are typically cast as nonlinear optimisation problems, and processed with generalised solvers, for example IPOPT[57] or SNOPT [19]. Feasible problems are typically small. In order to extend the chance-constrained approach to real-world problems, for example in public transport scheduling or network resource allocation, we must find faster solution methods.

1.1.2 Uncertainty representation

As with all decision making under probabilistic uncertainty, current chance-constrained techniques rely on a proper representation of uncertainty through a reasonable probability distribution. However, deriving a distribution is problematic. Distributions are typically fitted from historical data and prior observations of the uncertain variables, and as with any distribution fitting approaches there are concerns with over-fitting. While there are results regarding the convergence of distributions derived from density estimation schemes to true distributions, these are asymptotic and do not provide guarantees of safety.

1.2 Approach

In this section, I will outline the steps needed to construct a decision making system which provides probabilistic guarantees of constraint satisfaction. I will start by considering two thrusts: efficient algorithms for conflict-directed mixed logical linear programming, and data-driven risk allocation. The two thrusts will respectively address the central challenges of fast computation and uncertainty representation.

The key elements of my work are as follows: 1) a decomposition framework for chance-constrained mixed logical linear programs; 2) an efficient subsolver for chance-constrained linear programs; 3) a set of conflict extraction procedures to coordinate the CC-LP subsolver a higher level search; and 4) methods for providing uncertainty sets covering required probability mass with confidence.

1.2.1 Framework for CC-MLLP

Chance-constrained mixed logical linear programs (CC-MLLPs) can be used to describe a variety of real-world applications, from cyber security to autonomous exploration. The problem formulations allow combinations of logical constraints and linear constraints, allowing us to describe different agent requirements and system dynamics which arise when different logical conditions are met.

However, with such flexibility comes difficulties in computation. The logical constraints introduce combinatorics into the solution process. Further, due to the presence of probabilistic uncertainty and the risk tolerance, even a purely continuous subproblem is nonlinear and thus difficult to optimise over.

In this thesis, I provide a framework for decomposing the CC-MLLP into discrete and continuous portions. The framework couples a discrete search with a dedicated solver for the continuous chance-constrained linear program subproblem. By coordinating the two elements, I am able to provide at least a 10 times speed up over prior approaches.

1.2.2 Cutting planes for CC-LPs

The first key element in the decomposition is an efficient solver for the continuous subproblem. Current approaches to CC-LPs formulate the problem as a convex optimisation, and apply off-the-shelf solvers. However, in our approach, we would like to quickly determine the feasibility of subproblems, and provide fast optimal solutions where possible.

By noting that the majority of constraints in CC-LPs are linear, we employ a cutting planes approach. The approach allows us to successively approximate the nonlinear chance constraint using a set of linear constraints. This allows us to leverage existing linear program solvers, which allow us to more quickly determine feasibility.

Once a problem is determined to be feasible, we are able to use the resulting first solution as a starting point for off-the-shelf nonlinear solvers. The feasible starting point allows quicker convergence to optimality for gradient-based solvers, resulting in faster optimal solutions.

1.2.3 Conflict Extraction for CC-MLLPs

The second key element in the decomposition is the use of previous subsolver results to guide search in the discrete portion of the problem.

Leveraging the successive linear approximations used to solve the CC-LP subproblem, we are able to identify conflicting linear constraints which prevent feasible solutions or higher quality solutions. We can then identify the logical conditions which give rise to these particular combinations of linear constraints.

By explicitly ruling out decisions which satisfy the identified logical conditions, the discrete search is able to avoid infeasible and suboptimal candidates by making sure the conflicting linear constraints never occur together. This allows the discrete decision making portion to skip over large amounts of computation when searching for the optimal solution.

1.2.4 Data-driven risk allocation

In this thrust, I will the address the problem of uncertainty representation. I will do this by constructing uncertainty sets from data rather than fitting distributions. Unlike prior work in robust programming, I will derive multiple uncertainty sets to be reasoned over during optimisation.

Recall that we are interested in providing probabilistic guarantees, so that we can provide upper-bounds on the probability of failure when making decisions. One way of providing probabilistic guarantees to to construct robust sets which cover a set of outcome with a sufficient probability mass, and performing robust optimisation. This requires information about the lower and upper quantiles of uncontrollable variables.

For random variables with continuous outcomes, the key idea in my thesis will be to estimate the location of the quantiles. Due to the finite set of samples, the methods must be robust with respect to variations in sampling. I will thus also provide a *confidence* for estimates, a bound on the probability of estimates being incorrect.

Rather than directly fitting a cumulative distribution function from the data, I will find lower- and upper-bounds for the values of the quantiles, with the required confidence.

The approach contributes towards solving the challenges posed by chance-constrained programming by providing an explicit confidence on the estimates, the approach guards against inaccurate representation by being robust to sampling variations. This contrasts with traditional density estimation methods, which do not provide explicit guarantees of the accuracy of the distributions.

1.3 Summary

Despite advances in automated decision making, a lack of trust in the safety of derived plans is an major obstacle to adoption. Chance-constrained constraint optimisation is one way of addressing such concerns.

I have identified three main problems with current approaches to chance-constrained optimisation. Specifically:

1. Difficulties in scaling to larger problems due to slow computational speed;
2. Difficulties in handling programs over multiple types of constraint specifications; and
3. Difficulties in providing guarantees on the correctness of solutions, due to inappropriate uncertainty representation.

The remainder of the thesis is broken down as follows. In Chapter 2, I provide a review of the relevant literature. The chapter focuses on the work surrounding constraint programming, chance-constrained optimisation, and uncertainty representation.

In Chapter 3, I provide a formal definition of chance-constrained mixed logical linear program, as well as example encodings of problems in the formalism. I present the high level approach to solving CC-MLLPs. I outline how the problem is broken down into a high level search paired with a lower level subsolver. The high level search is responsible for the discrete decision making, while the lower level solver deals with the continuous portion of the problem. I describe efficient ways for the subsolver and the search to interact, and thus extract specifications on the lower level subsolver.

In Chapter 4, I describe the lower level solver in detail. I focus on how the use of cutting planes allows us to make successive linear approximations of the chance constraint, and can be solved efficiently by exploiting commercial solvers.

In Chapter 5, I describe the high level search in detail. I explain how the search incorporates the results of the lower level subsolver to guide the search away from unpromising branches, and how pruning is performed by solving relaxed versions of the continuous subproblems.

In Chapter 6, I consider the problem of uncertainty representation, and provide bounds on the outcome of uncertain variables with probabilistic guarantees. Two versions are provided: one for the true distribution, and one in which quality of service can be guaranteed even for a finite number of executions.

Chapter 2

Background

In this chapter we review prior work on which we build our proposed approach to chance-constrained constraint programming. We consider advances in efficient solution of mixed constraint systems, prior attempts at efficient chance-constrained programming, as well as recent developments in uncertainty representation for robust programming.

2.1 Efficient constraint programming

Constraint programming is concerned with finding feasible or optimal assignments to a set of variables, given a set of constraints between the variables. The variables may be finite domain, logical or continuous, and the constraints may be logical, linear, or nonlinear. A comprehensive survey of classical methods and key ideas is given in [48].

One of the key ideas in constraint programming is constraint propagation. When making an assignment to one variable in a constraint programming problem, we may consider the effect of the assignment on other variables. With constraint propagation, when a variable is assigned, restrictions are placed on the set of possible values for other variables due to constraints. Restrictions may be iteratively applied in the search for a full set of assignments, eliminating impossible combinations of assignments as the search progresses. However, propagating constraints is typically computationally expensive, especially when the set of constraints is large or when there is a mixture of different types of constraints.

The alternative is to generate candidate solutions, and then check against the set of all possible constraints, in a generate-and-check paradigm. Naively implemented, this guess-and-check approach may generate an arbitrarily long sequence of unacceptable candidates before coming up with the first feasible solution. When the verification of each candidate is computationally difficult, or when the space of candidates is large, an naive guess and check approach is impractical.

Recent advances in constraint programming have been concerned with improving the efficiency of algorithms by balancing the generate-and-test and the constraint propagation approaches. One line of investigation considers the discovery of important constraints, and the use of such constraints to guide the search for solutions. Usually only a subset of constraints are relevant to the search for solution. For example, consider a set of linear constraints. If solutions exist, the space of feasible assignments is a convex polytope, the intersection of half-planes defined by a subset of the linear constraints. The constraints not in this subset would be irrelevant and can be neglected when searching for a solution.

For some classes of constraints, the subset of relevant constraints may be discovered when testing candidate assignments. This is the key idea behind conflict-directed search. In [62, 29], conflicts or nogoods, combinations of assignments to a subset of variables which result in infeasibility are discovered by testing candidates. These constraints are then propagated when generating future assignments, leading to more promising candidates. Recent extensions have included methods of identifying constraints that must be satisfied by higher utility solutions [54, 33]. Moreover, conflict-directed search have been applied to different types of problems in scheduling, including robust scheduling and negotiating feasible mission parameters [60, 63].

A similar thrust, based on Bender's Decomposition, decomposes constraint programs into smaller subprograms to be solved in alternation by specialised solvers. The seminal work in [2, 18] described Bender's Decomposition for problems involving linear constraints, and its extension to nonlinear constraints. When each subprogram is solved, infeasibility and suboptimality constraints are discovered using duality from optimisation theory, and added to the set of constraints in other subprograms. This allows coordination between smaller subprograms, where each can be efficiently solved using specialised solvers. Recent generalisation to mixed logical constraint programs in [23] are shown to outperform state-of-the-art commercial software.

2.2 Chance-constrained programming

There are different sources of uncertainty in applications such as field robotics or large scale logistics. The timing of events, the positioning of autonomous agents, and the effects of actuators are uncertain parameters and variables which are not completely controllable by the autonomous system.

One approach is to maximise expected utility considering stochastic elements in the problem. Examples include the approaches based on Markov decision processes and its extensions [52, 27], which represents uncertainty with probability distributions. However, such approaches are problematic when the penalty for failure is not well defined. For example, due to the complex propagation of air traffic delays across multiple airports [17], the cost of failing to meet schedules for a single flight is difficult to estimate.

Chance-constrained programming was initially proposed as a paradigm in operations research [11, 12]. The problem was defined as maximisation of an objective function subject to probability of satisfying constraints over deterministic and probabilistic variables. Early work featured ideas familiar to more recent approaches, especially the focus on transformations of constraints into random variables, and solving the problem by determinising the random variables and mapping to well-known mathematical programming problems.

Chance-constrained programming has received recent interest in the planning and scheduling community. Applications included control planning for safe traversals, performed first through formulation of the problem as a convex program [6], and then through a heuristic hill-climbing technique for non-convex problems [44]. Other applications include scheduling under probabilistic uncertainty [16], which was extended to leverage efficient feasibility checking of temporal networks [60, 63]. In the wider community, chance-constrained programming has applications in such diverse fields as power management [61] and space systems architecture [37].

Additional theoretical work, relying on the theory of measures and moments [32], have been developed to provide convex formulations of chance-constrained programming. However, such approaches are limited, currently requiring bounded uncertainty for the probability distributions, and solving small-sized problems [25].

Within the constraint programming community, *stochastic programming* [58] has been a formalism for specifying constraints which must be satisfied with specified probability. However, much of the work has concentrated on probability distributions with discrete outcomes [53], and responding to uncertainty in stages. In contrast, my thesis concentrates on stochasticity in the continuous portion of the problem.

Interest in constraint programming with continuous random variables have led to a converge in the fields of study. For example, Probabilistic Continuous Constraint Satisfaction Problem (PCCSP) [10] calculate the probability of satisfying a set of numerical constraints, much like the method of moments approach [25], but using interval approximations instead.

2.3 Robust uncertainty representation

While chance-constrained programming has successfully provided compliance guarantees, there has been some criticism of the probabilistic representation of uncertainty.

Typically, density estimation is performed to find appropriate distributions to describe the uncertainty. Traditional methods are typically kernel-based [51], while more recent efforts include fitting to graphical models [31].

The resulting density functions are problematic for two reasons. First, while there has been literature exploring the error of the estimators [1], the guarantees are asymptotic and it is difficult to provide probabilistic guarantees on the correctness of any estimated density. Second, the resulting densities are typically non-convex, and thus difficult to incorporate into optimisation algorithms.

Robust optimisation was developed in response to concerns over the probabilistic representation [9]. In robust optimisation, uncertainty is represented with robust sets, giving bounds on the outcome of uncontrollable variables. Robust optimisation approaches look for assignments to decision variables which will satisfy all constraints, for any combination of outcomes described in the robust sets. Given the robust sets and constraints over controllable and uncontrollable variables, deterministic reformulations can be found by applying duality from optimisation.

Prior work in robust optimisation has concentrated on finding families of robust sets which lend themselves to optimisation [3]. For example, when a set of linear constraints are applied over a polygonal uncertainty set, the resulting deterministic reformulation is also a linear program. Robust programming is thus promising in terms of the tractability of the uncertainty representation. More recent work has been focusing on deriving uncertainty sets from data [4]. The approach uses hypothesis testing to derive tractable uncertainty sets, with probabilistic guarantees of coverage.

A weakness of current approaches in robust optimisation is over-conservatism. Typically, only one robust set is used, and decision variables are assigned given the fixed robust sets. However, different robust sets giving the same guarantees may lead to different utilities. Thus choosing an inappropriate robust set may lead to drastically lower utility. In chance-constrained optimisation, decision variables are assigned taking into account probability distributions for uncertainty. Intuitively, this corresponds to a simultaneous selection of robust sets and decision variable assignments, and leads to better utility. In general, chance-constrained methods tend to provide higher utility solutions, due to greater flexibility when choosing appropriate sets of uncertain outcomes.

Chapter 3

Problem Description and Approach

In this chapter, I introduce the chance-constrained mixed logic linear program (cc-MLLP). The cc-MLLP is motivated by real world applications, and I provide examples in cyber security and autonomous underwater exploration in this chapter.

The problem requires a mix of continuous and discrete decision making, the former introducing nonlinearities, and the latter introduces combinatoric search. In this chapter, I provide a high level outline of a decomposition approach in response. The proposed algorithm involves a high level search handling assignments to the logical variables, guided by the results returned by an efficient solver for the continuous portion of the problem.

3.1 Problem Definition: CC-MLLP

In constraint programming literature, the mixed logical linear program (MLLP) [22] is a standard formalism for deterministic decision making, allowing a combination of linear and logical constraints. The chance-constrained mixed logical linear program is developed as an extension, which allows specifications on the probability of success.

Problem 1 (Chance-constrained Mixed Logical Linear Program)**.** *Let $\langle \Omega, \mathcal{F}, \mathbb{P} \rangle$ be a probability space and $\{u_i : \Omega \to \mathbb{R}\}$ be a set of random variables. The chance-constrained mixed logical linear program is defined as:*

$$\min_{\mathbf{x}} \mathbf{c}^T\mathbf{x}$$
$$s.t. \Psi(\mathbf{p}) = \texttt{True}$$
$$\mathbb{P}(\neg \bigwedge_i C_i) \leq \Delta$$

where:

- $\mathbf{x} \in \mathbb{R}^{N_R}$ *is a vector of continuous decision variables.*
- $\mathbf{p} \in \{0,1\}^{N_P}$ *is vector of logical decision variables.*
- Ψ *is a logical formula over* $\mathbf{p}$*;*
- C_i *is of the form* $\Phi_i(\mathbf{p}) \Rightarrow \mathbf{A}_i\mathbf{x} + u_i \leq b_i$*;*
- Φ_i *logic formulae over decision variables* $\mathbf{p}$*;*
- $\mathbf{A}_i$ *are vectors of size* N_R *;*
- $b_i \in \mathbb{R}$ *are constants; and*
- $\Delta \in [0,1]$ *an upper bound on the probability of failure.*

For convenience, in the following discussion, we assume that for feasible problems, the optimal objective values are positive and bounded. This assumption could be relaxed to allow bounded but possibly negative optimal objective values by adding a constant term to the objective function.

The definition of the cc-MLLP consciously builds on the MLLP. Much like the MLLP formalism, we have the discrete part of the problem described by the logical formulae Ψ, and the continuous portions of the problem C_i.

However, the key distinction with the prior formulation is the introduction of the continuous random variables u_i in the continuous portion of the problem. These are use to describe uncertainty in our problems, for example positional uncertainty for AUV path planning, or uncertainty over available bandwidth for network routing.

In the presence of this uncertainty, we would like to provide some guarantee on the probability of success. Thus, we impose a chance constraint over the continuous portion of the problem, such that the probability of the continuous portion not being satisfied is bounded. Specifically, the probability of $\neg \bigwedge_i C_i$ has to be less than or equal to Δ.

While we accept all logical formulae, we assume without loss of generality that they are given in the equivalent conjunctive normal form [49]. Further, for convenience in the subsequent discussion, a linear constraint is said to be *implied* by a partial assignment to the logical variables if the associated logical formula is satisfied. Specifically, consider a constraint C_i of the form

$$\Phi_i(\mathbf{p}) \Rightarrow \mathbf{A}_i\mathbf{x} + u_i \leq b_i$$

If $\Phi_i(\mathbf{p}') = \texttt{True}$ for a partial assignment $\mathbf{p}'$ to $\mathbf{p}$ the set of logical variables, then $\mathbf{p}'$ *implies* the linear constraints $\mathbf{A}_i\mathbf{x} + u_i \leq b_i$.

3.2 Example CC-MLLPs

The cc-MLLP formulation occurs in many real world applications. We introduce two such examples to demonstrate how the encoding may be used. We begin with a chance-constrained autonomous underwater vehicle (AUV) path planning problem.

Example 1. *Consider a vehicle path planning problem, in which we plan waypoints for a vehicle which must stay within a sequence of safe polygonal regions. As before, we assume that the mean path of the vehicle can be approximated by a sequence of straight lines, while deviation from the mean paths can be described as random variables.*

Let $\mathbf{x}_t$ be the vehicle position at time t, and $\mathbf{u}_t$ the uncertainty at time t. For the continuous constraints, we have the linear dynamics as in the convex version of the problem. In addition, we may encode whether a vehicle in a region R^i at time t with the logical variable $r_t^i \in \{0,1\}$. We thus have constraints of the form $r_t^i \Rightarrow A_i(\mathbf{x}_t + \mathbf{u}_t) \leq b_i$, where A_i, b_i describe the region R_i. Note that $A_i(\mathbf{x}_t + \mathbf{u}_t) \leq b_i$ can be rewritten as $A_i\mathbf{x}_t + \mathbf{u}_t' \leq b_i$ as required for the CC-MLLP definition, where $\mathbf{u}_t' = A_i\mathbf{u}_t$.

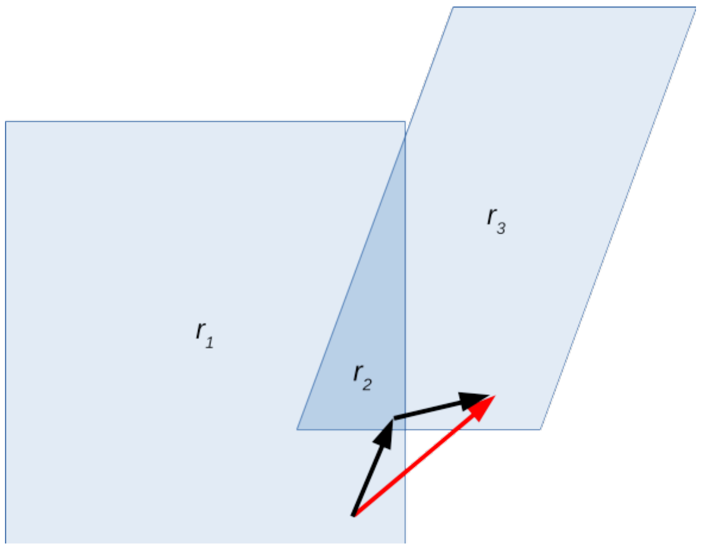

Figure 3-1: Example AUV path planning over safe regions. The safe regions are shaded in blue. We can not travel directly between two points in r_1 and r_3, as demonstrated by the red path, because it may cut through an unsafe region. Instead, vehicles must transition between two regions by making a stop in r_2, which is the intersection of both r_1 and r_3.

In addition, we require safe traversal between time steps - we do not wish to cut between unsafe regions in between time steps. This is illustrated in Figure 1. We must encode transitions between regions which intersect, by forcing the vehicle to make a stop in the intersection. For each original safe region, we may denote its set of neighbour as the safe region itself, and its intersections with all other regions. For each intersection, we denote its set of neighbours as the intersection itself, and the two regions which intersect. Then, we have logical constraints of the form $r_t^i \Rightarrow \bigvee_{j \in \texttt{Neigh}(i)} (r_{t-1}^j)$, *where* $\texttt{Neigh}(i)$ *denotes the neighbours of region* R^i.

The cc-MLLP can also be used to model a network routing problem with packet loss constraints. The following problem was taken from the DARPA EdgeCT cyber-security problem.

Example 2. *Figure 3-2 shows the topology for a network of computers. The circles denote network nodes, each of which is connected to several neighbours. For example, Node 0 has neighbours Nodes 1, 2, 6 and 7.*

Neighbouring nodes are connected by links, with a bounded amount of throughput, and a known cumulative loss. The problem asks us to route flows, for example voice over IP

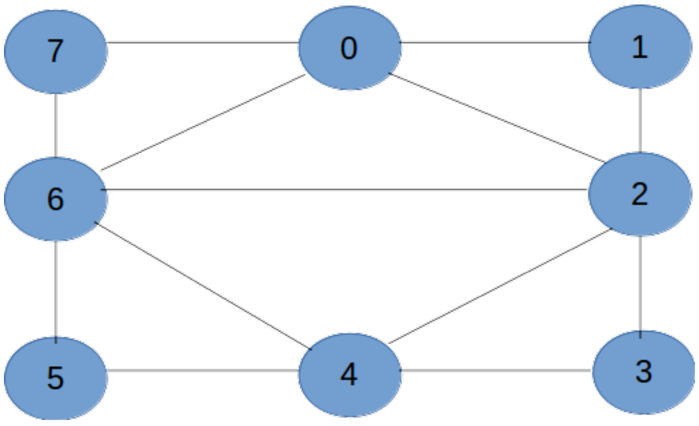

Figure 3-2: Topology for the EdgeCT network.

(VOIP) calls, from one node to another node on the network. For each link on the network, we must decide the amount of throughput we must reserve for each flow.

Each flow has a minimum amount of bandwidth. The uncertainty in the problem comes from the amount of bandwidth required by each flow. For example, the amount of VOIP calls varies day to day. The uncertain bandwidth requirements is as a random variable, BW^k, for flow k.

The real valued decision variables in this problem are the bandwidths allocated to each flow along each link, denoted $BW^k_{i,j}$ for flow k on the link from Node i to Node j, as well as BW^k_{alloc} the actual total amount of bandwidth allocated for each flow. For each flow k, we have the standard flow conservation constraints $\sum_{i \in \texttt{Neigh}(j)} BW^k_{i,j} = \sum_{i \in \texttt{Neigh}(j)} BW^k_{j,i}$ at nodes which are neither the source nor the sink. At the start node s of the flow k, we require $\sum_{j \in \texttt{Neigh}(s)} BW^k_{s,j} = BW^k_{alloc}$. At the destination node d of the flow k, we require $\sum_{j \in \texttt{Neigh}(d)} BW^k_{j,d} = BW^k_{alloc}$.

The logical part comes in when we decide whether to drop a flow. Let $\texttt{Drop}^k \in \{0,1\}$ be the logical variable deciding whether to drop flow k. For each flow k, we have $\texttt{Drop}^k \Rightarrow BW^k_{alloc} = 0$ and $\not{\texttt{D}}\texttt{rop}^k \Rightarrow BW^k_{alloc} - BW^k \geq 0$.

3.3 High Level Solution Approach

One of the common themes of constrained programming is a breakdown of a complex system of constraints into portions, each of which could be efficiently handled by dedicated subsolvers. In addition to the advantages gained by allowing each subsolver

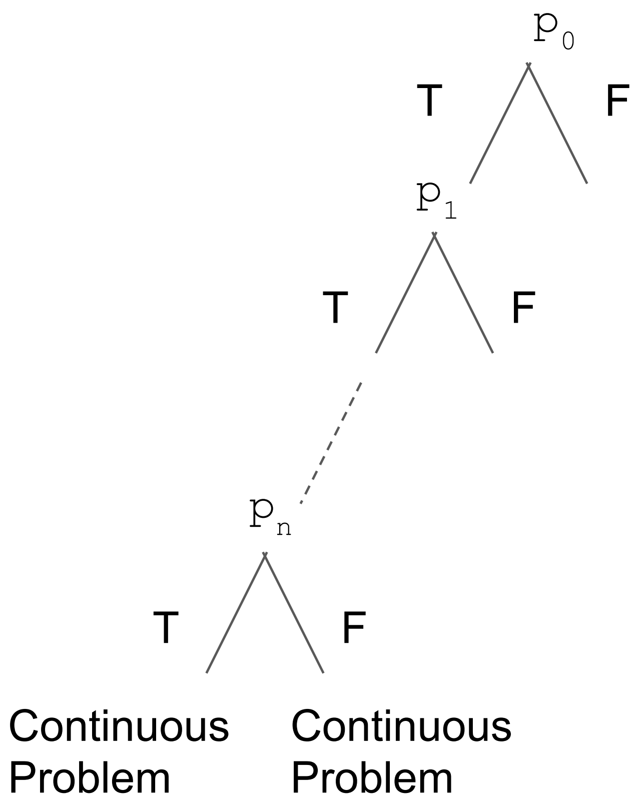

Figure 3-3: Decomposition of CC-MLLP.

to exploit the particular structure of the subproblems, a constraint programming approach would also identify how subsolvers can communicate. This allows computation to be reused, such that the results of solving one subproblem could be used to guide the computation when solving a different subproblem. I adopt this decomposition approach in my solution algorithm for CC-MLLPs.

A natural way to consider the CC-MLLP is to decompose the problem into the discrete and continuous components. This decomposition can be done by successively assigning the logical decision variables. Once all the logical variables are assigned, we arrive at purely continuous problems. This is visualised in Figure 3-3.

As logical variables are assigned, for every C_i in the continuous portion of the problem, the logical conditions Φ_i can be checked. If they are evaluated to be true, the implied linear constraints are imposed. Given a full set of assignments to the logical decision variables, we are able to evaluate all the logical formulae in the continuous part of the problem. Thus, for every C_i, we know whether the linear constraint involved will need to be satisfied. This results in optimisation program, in which the probability of jointly satisfying a set of linear constraints must be greater than a specified bound, a problem that has been studied in the literature [45].

The decomposition motivates three main ways of speeding up the solution process: 1) using a more efficient subsolver dedicated to solving the continuous subproblem; 2) evaluating easier subproblems corresponding to each partial assignment to logical variables, and pruning unpromising search branches; and 3) using the results from evaluating the full subproblems from full logical assignments and the relaxed subproblems from partial logical assignments as guides for search.

3.3.1 Efficient Subsolver

A full assignment to the logical variables represents a special case of the CC-MLLP, in which only linear constraints over continuous variables feature. As this will be the focus of our subsolver, we provide a formal definition of the purely continuous subproblem, known as the chance-constrained linear program (CC-LP).

Problem 2 (Chance-Constrained Linear Program)**.** *Let $\langle \Omega, \mathcal{F}, \mathbb{P} \rangle$ be a probability space and $\{u_i : \Omega \to \mathbb{R}\}$ be a set of random variables. A Chance-Constrained Linear Program is defined as*

$$\min_{\mathbf{x}} \mathbf{c}^T \mathbf{x} \tag{3.1}$$

$$s.t. \Pr(\mathbf{A}_i \mathbf{x} + u_i \leq b_i \quad \forall i) \geq 1 - \Delta \tag{3.2}$$

for:

- $\mathbf{x} \in \mathbb{R}^{N_R}$ *is a vector of continuous decision variables.*
- $\mathbf{A}_i$ *are vectors of size* N_R *;*
- $b_i \in \mathbb{R}$ *are constants; and*
- $\Delta \in [0, 1]$ *an upper bound on the probability of failure.*

An efficient subsolver for the CC-LP suproblem would greatly increase the speed of solution for a full CC-MLLP. Each full assignment to the discrete decision variables implies a different continuous subproblem. There are thus typically a large number of evaluations of continuous subproblems. The savings we would obtain from deploying a subsolver which quickly returns results would accumulate over the course of the search.

A chance constraint can be thought of as a constraint over the probability mass of outcomes for which the assignments to the decision variables satisfy a set of specifications. As distributions are often nonlinear, this introduces a nonlinear element to a problem which is otherwise composed only of linear constraints. I provide a detailed description of an efficient CC-LP solver in Chapter 4, based on successive linear approximations of the chance constraint. This allows us to leverage efficient off the shelf solvers.

3.3.2 Pruning Using Relaxations

In search, one common method of speeding up solutions is by pruning branches early, without evaluating a full subproblem. Given a partial set of assignments, we would like to quickly determine whether it would be promising to keep going down the branch. This is done by looking at relaxed problems, which are less difficult to solve than the full continuous problem, and yield optimistic estimates for the quality of solution provided if we continue on to a full assignment to the logical random variables.

In the case of CC-MLLPs, a partial set of assignments yields a relaxed CC-LP. The partial assignments means a subset of the conditions are satisfied for the linear constraints. As the more assignments are made, more conditions are satisfied, more linear constraints are imposed, and the CC-LP becomes more constrained. Given any partial assignment to the logical variables, the linear constraints implied by the partial assignments is a subset of the linear constraints implied by a full assignment to the logical variables. This subset of linear constraints thus constitute a relaxation.

A further relaxation can be done by relaxing the chance constraint. Again, the chance constraint introduces nonlinearity to the subproblem. We provide an optimistic relaxation of the chance constraint, determinising and linearising the subproblem, so that we are only solving linear programs when evaluating the relaxation for each partial assignment. This is elaborated in Chapter 5.

3.3.3 Conflict Guided Search

The third key technique to efficient solution processes is to extract guidance on which assignments to try, given previous computation. When solving both full subproblem from full assignments, and the relaxed subproblems from partial assignments, we are able to either prove infeasibility, or provide information about the quality of solutions possible given the logical assignments. We would like to use this information to guide further searches.

In the case of infeasible problems, search guidance can be extracted by considering the subsets of linear constraints which prevented a feasible solution. In the case of potentially suboptimal problems, search guidance can be extracted by looking at the subset of active linear constraints which prevent a better quality solution. In both cases, the logical conditions which imply the linear constraints can be extracted, and used to produce additional constraints to be added to Ψ, the discrete portion of the CC-MLLP. This process of conflict extraction is also detailed further in Chapter 5.

3.3.4 High Level Algorithm

My approach to solving CC-MLLPs combines an efficient CC-LP subsolver, pruning via relaxation, and conflict extraction. The high approach to the CC-MLLP is described in Algorithm 1.

The algorithm generates successive partial assignments to the logical variables via search, and solves the implied relaxed CC-LPs. When these relaxations are infeasible, or yield a utility function worse than the incumbent, we identify a subset of logical variable assignments which led to infeasibility or suboptimality. From these logical variable assignments, we construct resolution constraints over the logical variables which are then propagated to eliminate the infeasible or suboptimal part of the search

```
Input: Problem specified in Problem 1
Output: $\mathbf{x}, \mathbf{u}$
1  $Q \leftarrow$ InitQueue()
2  $\tilde{\mathbf{p}}, \tilde{\mathbf{x}} \leftarrow \{\}, \tilde{c} \leftarrow \infty$ ;            // initialise incumbent and incumbent cost
3  while $Q$ not empty do
4      candidate $\leftarrow$ Dequeue($Q$)
5      Solve relaxed CC-LP
6      if relaxed CC-LP infeasible or worse than incumbent then
7          Add resolution constraints, propagate for every element on queue
8      else
9          if candidate is a partial assignment then
10             children $\leftarrow$Expand(candidate)
11             Enqueue($Q$,children)
12         else
13             Solve CC-LP
14             Add we were able to quickly return results resolution constraints,
                 propagate for every element on queue
15             if CC-LP feasible, better than incumbent then
16                 $\tilde{\mathbf{p}} \leftarrow$ candidate
17                 $\tilde{\mathbf{x}} \leftarrow$ CC-LP solution variable assignments
18                 $\tilde{c} \leftarrow$ CC-LP solution cost
19             end
20         end
21     end
22 end
23 return $\tilde{\mathbf{p}}, \tilde{\mathbf{x}}$
```

Algorithm 1: Solving CC-MLLP.

space.

When a candidate is generated with full assignment to the logical variables, we solve the implied CC-LP. Similar to the relaxed CC-LP, we return resolution constraints for infeasibility and suboptimality. Note that when the CC-LP is feasible, we apply the resolution constraints even if the new candidate results in a solution better than the incumbent. The constraints are used to eliminate future candidates which have, at best, the same cost as the new candidate.

3.3.5 Relationship to Prior Work

The approach in this thesis builds on prior approaches in the literature. The NIRA algorithm [46] also solved mixed discrete-continuous chance-constrained program. NIRA concentrated on chance-constrained mixed integer-linear programs, with a similar decomposition of the problem into search and dedicated continuous solver. Further, NIRA introduced the idea of solving relaxed subproblems with partial assignments to the logical variables. However, NIRA did not try to extract conflicts from the subsolver to guide its discrete search component, and used an off-the-shelf solver for the continuous portion.

In deterministic problems, there have been work in extracting conflicts for disjunctive-linear programs [33], and mixed logic-linear programs [23]. The algorithms exploit duality in linear programs to identify subsets of linear constraints which prevent feasibility or constrain the subproblem from a better objective function. However, in the CC-MLLP context, the chance constraint introduces a nonlinear element to the continuous subproblems. This partially motivates our approach to the CC-LP as a sequence of linear linear approximations, as described in Chapter 4.

3.4 Summary

In this chapter, I have introduced the chance-constrained mixed logic-linear program, as well as the chance-constrained linear program, and provided motivating examples. I have outlined an approach to solving the CC-MLLP as a combination of high level search, guided by a dedicated CC-LP solver.

The CC-LP solver is central to the approach. It must be tractable, and it must summarise reasons for infeasibility or suboptimality in order to guide the higher level search. In Chapter 4, I provide a CC-LP solver which achieves significant speed ups over prior work. In Chapter 5, I show how the new CC-LP solver can be exploited to provide the necessary conflicts to guide the discrete search.

Chapter 4

Cutting-planes for CC-LP

Chance-constrained methods for decision making under uncertainty have proven successful for applications in which constraint violation is catastrophic. In particular, the idea of risk allocation has driven solution algorithms for problems featuring chance constraints over multiple constraints. For linear problems, the state-of-the-art risk allocation algorithm, Convex Risk Allocation (CRA) encodes the problem as a single nonlinear program; and 2) Iterative Risk Allocation (IRA), in which the problem is decomposed into a risk allocation master and a linear subprogram, and hill-climbing is used to iteratively improve utility. However, the former is difficult to scale as it involves solving large nonlinear programs, while the latter is incomplete and suboptimal.

In this chapter, we present Conflict-Based Risk Allocation (CBRA). We separate the problem into a determinised linear optimization and a risk allocation subproblems, and learn constraints approximating risk allocation as cutting planes. These cutting planes successively guide the generated risk allocations towards feasibility and optimality, give quick first solutions, and provide certificates of infeasibility.

We provide benchmarks comparing the new algorithm to CRA and IRA, on a set of underwater vehicle model predictive control benchmarks. The results demonstrate that IRA does not find solutions even in cases when CRA and CBRA quickly find solutions. We also show that CBRA is able to achieve significant speed ups over CRA, proving infeasibility up to 1000 times faster. Further, by finding a first solution using

CBRA, we are able to provide CRA with a good initial starting point, leading to up to 20 times speed ups in finding optimal solutions.

4.1 Introduction

Advances in hardware and algorithms have led to impressive developments in autonomous systems. High profile deployments of automated algorithms include aircraft construction, financial trading and medical procedures. However, one of the chief barriers to more widespread adoption of autonomous systems is the perceived lack of reliability in autonomous systems. For example, there is a hesitation to allow onboard path planning to be performed in NASA robotic missions to Mars. Mission controllers are more comfortable with hand-coded sequences of activities, the safety and correctness of which are manually checked. Such mistrust may be addressed if autonomous agents were able to provide guarantees of the correctness of output plans.

Chance-constrained programming is one such approach, which allows automated decision-making with guarantees on the probability of success and safety [11]. Consider an example autonomous underwater vehicle (AUV) problem as follows.

Example 3. *An AUV is on a bottom following mission, depicted as a side-on view in Figure 4-1. The mean position of the vehicle must stay as close to the bottom as possible during the duration of the traversal, while avoiding collision with the seafloor.*

The AUV starts at $\mathbf{x}_0 = [0, 5]$*, and after* n *time steps must end within a goal region, such that* $\mathbf{x}_n \in [n-1, n+1] \times [4, 6]$*. The vehicle is able to move 1m/s in both* x *and* y*, although there is an normally distributed actuation noise* $\omega_n \sim N(0, 0.05)$ *at each step. We require that the probability of collision with the seafloor or not arriving at the goal region is less than 1 in 20000.*

The example problem may be expressed as a chance-constrained linear program (CC-LP).

Example 4. *Let* $\mathbf{x}_t$ *be the 2-dimensional mean position of the vehicle at time step* t*.Note that the actuation noise are accumulated at each time step, and the uncertainty at time step* t *can written as* $\mathbf{y}_t \sim N(0, (0.05t)\mathbf{I}_2)$*, where* $\mathbf{I}_2$ *is the* 2×2 *identity matrix.*

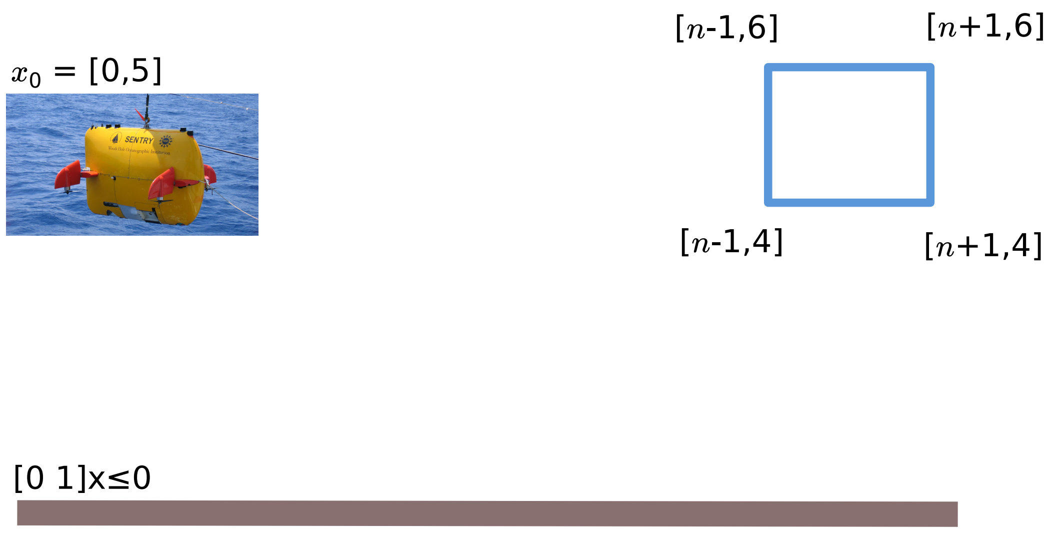

Figure 4-1: Example AUV bottom following mission.

The linear constraints in the problem are as follows:

- *Actuation constraints:*

$$\begin{bmatrix}-1\\-1\end{bmatrix} \leq \mathbf{x}_{t+1} - \mathbf{x}_t \leq \begin{bmatrix}1\\1\end{bmatrix}, \ \forall t \in \{0, 1, ..., n-1\} \tag{4.1}$$

- *Seafloor collision avoidance constraints:*

$$\begin{bmatrix}0 & 1\end{bmatrix}(\mathbf{x}_t + \mathbf{y}_t) \geq 0, \ \forall t \in \{0, 1, ..., n\} \tag{4.2}$$

- *Goal region constraints:*

$$\begin{bmatrix}n-1\\4\end{bmatrix} \leq \mathbf{x}_n \leq \begin{bmatrix}n+1\\6\end{bmatrix} \tag{4.3}$$

- *Initial location constraint:*

$$\mathbf{x}_0 = \begin{bmatrix}0\\5\end{bmatrix} \tag{4.4}$$

Then, the joint CCLP can be written:

$$\min_{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n} \sum_{t=0}^{n} \begin{bmatrix}0 & 1\end{bmatrix} \mathbf{x}_t$$

$$\textit{s.t.} \ \Pr(\{\ \textit{(4.1), (4.2), (4.3), and (4.4) satified}\}) \geq 1 - 0.00005$$

In prior work, CC-LPs have been used to describe vehicle path planning [45], and power network planning [5], among other applications. In the classical definition, a chance constraint is associated with each constraint in the underlying problem. This is well-studied, and the CC-LP can be rewritten as a second order cone problem (SOCP) given radial distributions [9], while convex approximation results exist for other distributions [41].

However, for the example above, we would like to guarantee the probability of success over the entire system of constraints, and hence is a joint chance-constrained problem [38, 24]. As solving the exact joint chance-constrained problem is difficult, recent solution algorithms have relied on approximations. The trivial approximation, in which the probability of failure is evenly distributed over the each constraint, can be solved as a traditional chance-constrained problem. However, this may result in bad approximations. In Example 3, the trivial approximation may result in a problem such that no waypoints meeting the chance constraints are possible. Alternatively, evenly distributing the probability of failure may also lead to solutions with higher cost [7].

In the distributionally robust optimization (DRO) community, there has been extensive work based on reformulating the problem to an optimization based on conditional value at risk (CVar) [41, 13]. However, CVar approximations are loose even for chance constraints over a single constraint, given that they are derived from Markov's Inequality. Further, CVar approximations require choosing set of scaling parameters. While fixed scaling parameters lead to convex approximations, poor choices of scaling parameters may lead to infeasible or suboptimal approximations as with the trivial approximation. On the other hand, optimising over the scaling parameters is biconvex with no guarantees of global optimality [65].

This motivates an alternative approximation via *risk allocation*, in which the total risk is divided and allocated to each constraint. This approximation relies on the union bound, and is known as optimised Bonferroni approximation to the DRO community.

4.2 Risk Allocation

Solving the joint chance-constrained linear program is difficult, especially due to the need to evaluate the probability over a joint distribution, although [38] gives an exact method assuming independence of random variables. Without the independence assumption, a typical conservative approximation is performed using Boole's inequality [45, 7, 56, 20] to parcel out the risk to individual controls. The method is known as *risk allocation.*

Problem 3 (Risk allocation approximation for Joint CC-LPs). *The risk allocation approximation for a joint CC-LP is defined as*

$$\min_{\mathbf{x}} \mathbf{c}^T \mathbf{x} \tag{4.5}$$

$$s.t. \mathbf{A}_i \mathbf{x} + \epsilon_i \leq b_i \qquad \forall i \tag{4.6}$$

$$\sum_i (1 - \Pr(u_i \leq \epsilon_i)) \leq \Delta \tag{4.7}$$

for:

- $\mathbf{x}$*,* u_i*,* b_i*,* $\mathbf{c}$*,* $\mathbf{A}_i$*, and* $\Delta \in [0, 1]$ *as in Definition 2;*
- ϵ_i *decision variables, representing upper-bounds on random variables given risk allocations.*

Intuitively, we are allocating risk to find upper bounds ϵ_i on the outcomes of random variables u_i featured in each linear constraint. Given ϵ_i, Equation (4.6) ensures that $\mathbf{x}$ are chosen with a sufficient safety margin to satisfy the linear constraint given that upper bounds on non-deterministic portions of the problem.

Equation (4.7) requires that the sum of these risk allocations is less than the tolerated probability of failure Δ. This uses Boole's Inequality to conservatively approximate the probability of the scenario in which *any* of the random variables are greater than the corresponding upper-bounds.

The formulation has similarities to the individual chance-constrained problems. However, rather than having constant risk allocations, the risk allocations are cast as decision variables in our problem, and thus can not be solved using algorithms for problems with individual chance constraints. Given convex cumulative distribution functions, the approximate problem outlined in Definition 3 is a convex nonlinear program.

There are two prior algorithms for risk allocation: 1) Convex Risk Allocation (CRA), which provides an encoding of the problem as a convex nonlinear problem [7] and solves the problem with a convex nonlinear solver, for example SNOPT [19] or IPOPT [57]; and 2) Iterative Risk Allocation (IRA), which decomposes the problem into a risk allocation master and a deterministic subsolver and uses hill-climbing to iteratively improve on solutions [45].

However, IRA is known to be suboptimal and incomplete. On the other hand, CRA is typically slow to terminate for infeasible problems. Further, the time required for convergence to optimal solutions for feasible problems is highly dependent on the starting point. We would like to address these weaknesses through an algorithm which quickly either proves infeasibility or returns a good first solution.

In this chapter, we describe a cutting plane method [30] to quickly find first solutions or provide certificates of infeasibility. The key insight is that the chance constraint implicitly defines a feasible region for the deterministic subproblem. By decomposing the chance-constrained problem into the linear and risk allocation subproblems, we may iteratively generate solutions to the linear subproblem and check the resulting probability of failure. This procedure allows us to extract linear constraints at each iteration to approximate the chance constraint, and are used when generating subsequent solutions to the linear subproblem. We apply this method to quickly generate first solutions or prove infeasibility, resulting in significant speed ups in overall solution time.

4.3 Approach

In this section, we introduce Conflict-Based Risk Allocation (CBRA) for the Joint Chance-Constrained Linear Program. We decompose the problem into a linear programming subproblem, and a risk allocation subproblem. Candidate solutions are generated via linear programming, and checked against the chance constraint. Cutting planes are extracted to approximate the chance constraint as linear constraints, and added to the linear subproblem for subsequent candidates.

By generating promising candidates by solving a sequence of LPs, we are able to quickly find a first feasible solution if one exists. The cuts also allow us to provide optimality guarantees and certificates of infeasibility.

4.3.1 Conflict-Based Risk Allocation

The key observation in our work is that risk allocation implicitly imposes a nonlinear constraint over the solution space of the variables in the linear program. For any solution to the linear subproblem, we may evaluate the probability of violating constraints by considering the slack in each constraint. Rather than performing risk allocation explicitly, we may then evaluate each candidate solution to see whether the chance constraint is met. When the chance constraint is violated, we would like to generalise the reason for infeasibility, adding it as a constraint for subsequent candidates.

The procedure described above corresponds exactly to the cutting plane family of methods [30, 15, 42], applicable to convex nonlinear optimization problems. While the family of methods are slow to converge to optimality in practice, they have advantages over interior point methods and sequential quadratic methods. In particular, they do not require evaluations of the second derivatives at each iteration. Further, applying a cutting plane technique allows us to exploit the fact that all but one constraint is linear, and generate candidates using highly optimised linear solvers like Gurobi [21] or CPLEX [14].

In our approach, we first generate a solution candidate which is furthest from all linear constraints, ignoring the objective function. We then check whether the candidate meets the chance constraint. If not, we extract a cutting plane for the chance-constraint, adding it to the set of linear constraints for subsequent iterations. Otherwise, we generate another candidate optimising for the objective function, and update the lower- and upper-bounds. This is done iteratively until some termination condition on the lower- and upper-bounds have been met, or when the LPs are infeasible. Our approach is summarised in Algorithm 2. In subsequent discussions, we use the notation F_i for the cumulative density function of random variable u_i, and f_i for the probability density function.

```
Input: ${u_i}$, $\mathbf{c}$, ${b_i}$, ${\mathbf{A}_i}$, $\Delta$
Output: $\mathbf{x}$, $\epsilon$, $feasible?$, $\mathcal{C}$
1  $U \leftarrow \infty$, $L \leftarrow 0$, $\epsilon \leftarrow \mathbf{0}$, $\mathbf{x} \leftarrow \mathbf{0}$ ;
2  $\mathcal{C} \leftarrow \emptyset$;
3  while True do
4      // generate candidate
5      $[feasible?, \mathbf{x}] \leftarrow$ SolveChebLP(${b_i}$, ${\mathbf{A}_i}$, $\mathcal{C}$);
6      if $\neg feasible?$ then return;
7      $\epsilon_i \leftarrow d_i - \mathbf{A}_i\mathbf{x}, \forall i \in \{1, 2, ..., M\}$;
8      if $\sum_{i=1}^{M} 1 - F_i(\epsilon_i) > \Delta$ then
9          // extract cutting plane
10         $\mathcal{C} \leftarrow \mathcal{C} \cup \{$InfeasCut($\mathbf{x}$,${u_i}$, ${b_i}$, ${\mathbf{A}_i}$, $\Delta$)$\}$
11     else
12         // update lower- and upper-bounds
13         $[feasible?, \mathbf{x}'] \leftarrow$ SolveLP($\mathbf{c}$, ${b_i}$,${\mathbf{A}_i}$,$\mathcal{C}$);
14         $L \leftarrow \mathbf{c}^T\mathbf{x}$;
15         $\epsilon'_i \leftarrow b_i - \mathbf{A}_i\mathbf{x}', \forall i$;
16         if $\sum_i 1 - F_i(\epsilon'_i) \leq \Delta$ then
17             // tighten upper-bound
18             $[\mathbf{x}, \mathbf{x}'] \leftarrow$BiSearch($\mathbf{x}$, $\mathbf{x}'$,${u_i}$, $\mathbf{c}$, ${b_i}$, ${\mathbf{A}_i}$, $\Delta$)
19         else
20             $\mathbf{x} \leftarrow \mathbf{x}'$,$\epsilon \leftarrow \epsilon'$;
21             return
22         end
23         $U \leftarrow \min\{U, \mathbf{c}^T\mathbf{x}\}$;
24         $\epsilon_i \leftarrow d_i - \mathbf{A}_i\mathbf{x}, \forall i \in \{1, 2, ..., M\}$;
25         $\mathcal{C} \leftarrow \mathcal{C} \cup \{$InfeasCut($\mathbf{x}'$, ${b_i}$, ${\mathbf{A}_i}$, $\Delta$)$\}$;
26         $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathbf{c}^T\mathbf{x} \leq U\}$;
27         if Terminate($L, U$) then return;
28     end
29 end
```

Algorithm 2: Conflict-Based Risk Allocation (CBRA).

By substitution into standard results for cutting planes, for candidate ϵ' such that $\sum_{i=1}^{M} 1 - F_i(\epsilon'_i) > \Delta$, we can extract the linear constraint on future candidate ϵ:

$$\sum_{i=1}^{M} (1 - F_i(\epsilon'_i) + f_i(\epsilon'_i)(\epsilon_i - \epsilon'_i)) \leq \Delta \tag{4.8}$$

Since we require $\epsilon_i = b_i - \mathbf{A}_i\mathbf{x}$, `InfeasCut` thus returns a linear constraint on future candidates $\mathbf{x}$ such that:

$$-\mathbf{f}(\epsilon')^T \mathbf{A}\mathbf{x} \leq \Delta - \sum_{i=1}^{M} (1 - F_i(\epsilon_i') + f_i(\epsilon_i')\epsilon_i') \tag{4.9}$$

where $\mathbf{f}(\epsilon') = [f_1(\epsilon_1), f_2(\epsilon_2), ..., f_M(\epsilon_M)]^T$. From established results on cutting planes in convex programming, any $\mathbf{x}$ satisfying the chance constraint also satisfies the linear constraints describing each cutting plane. Thus the cuts do not eliminate candidates which satisfy the chance constraint.

As each cutting plane eliminates a part of the solution space containing the infeasible candidate, we would like to generate candidates at each iteration which are far from the boundaries defined by the existing linear constraints. We thus generate the first candidate using the well-known Chebyshev center [15], the center of the largest Euclidean ball which fits inside the space defined by the linear constraints. In Line 5, `SolveChebLP` solves the LP:

$$\begin{aligned}
&\min_{\mathbf{x}, r \geq 0} -r \\
s.t.&\mathbf{A}_i\mathbf{x} + r \leq b_i && \forall i \\
&\mathbf{g}^T\mathbf{x} + r \leq h && \forall \{\mathbf{g}^T\mathbf{x} \leq h\} \in \mathcal{C}
\end{aligned}$$

The constraints generated are thus far away from both the original constraints, as well as the cutting planes used to approximate the chance constraint. If the Chebyshev center LP is infeasible, we conclude that the original joint CC-LP is also infeasible, because the cutting planes do not eliminate a feasible solution if one exists.

If the Chebyshev center LP is feasible, we would like to update the lower- and upper-bounds for the objective function. `SolveLP` generates a candidate $\mathbf{x}'$ solving the LP:

$$\begin{aligned}
&\min_{\mathbf{x}} \mathbf{c}^T\mathbf{x} && \\
&s.t. \mathbf{A}_i\mathbf{x} \le b_i && \forall i \\
&\mathbf{g}^T\mathbf{x} \le h && \forall \{\mathbf{g}^T\mathbf{x} \le h\} \in \mathcal{C}
\end{aligned}$$

to optimistically estimate the lower-bound of the objective function. If the minimising $\mathbf{x}'$ satisfies the chance constraint, then we can exit, since we have found a candidate minimising the objective function and meeting the chance constraint.

If instead the minimising candidate does not satisfy the chance constraint, we use standard bisection search to find a better upper bound. While the procedure is optional and not part of standard cutting plane methods, its inclusion leads to better empirical performance, quickly reducing the upper-bound on the objective $\mathbf{c}^T\mathbf{x}$. `BiSearch` is summarised in Algorithm 3, for some specified N number of iterations.

```
  **Input:** $\mathbf{x}, \mathbf{x}'$, $\{u_i\}$, $\mathbf{c}$, $\{b_i\}$, $\{\mathbf{A}_i\}$, $\Delta$
  **Output:** $\mathbf{x}, \mathbf{x}'$
1 for $i \in \{1, ..., N\}$ do
2     $\mathbf{x}'' \leftarrow \frac{1}{2}(\mathbf{x} + \mathbf{x}')$;
3     $\epsilon''_i \leftarrow b_i - \mathbf{A}_i\mathbf{x}'', \forall i \in \{1, 2, ..., M\}$;
4     if $\sum_{i=1}^{M} 1 - F_i(\epsilon''_i) > \Delta$ then $\mathbf{x}' \leftarrow \mathbf{x}''$;
5     else $\mathbf{x} \leftarrow \mathbf{x}''$;
6 end
```

Algorithm 3: Bisection search for upper-bound.

For CBRA, we may impose a termination based on l and u, the lower- and upper-bounds for the objective. As noted, cutting plane methods tend to be slow to converge. In practice, rather than requiring $u - l$ to be smaller than some tolerance, in our benchmarks we find that terminating upon a first candidate which satisfies the chance-constraint provides a good first solution for subsequent optimization.

4.3.2 Related work

There exists other methods which also use divide the problem into the risk allocation and linear program subproblems. However, previous approaches concentrated on

Bender's Decomposition in the standard context of constraints systems featuring only random variables with finite support [35, 34]. Candidate risk allocation can thus be generated efficiently through search in a discrete space, and Bender's cuts extracted from the linear subproblem. However, as our problems feature random variables with continuous, possibly unbounded support, a risk allocation would need to be generated via nonlinear programming, and is thus computationally demanding.

Other iterative methods have also been proposed for chance-constrained problems with continuous random variables. Outer Approximation, an alternative cutting plane technique, is used in [5]. However, the prior work assumes that the probability of violating each constraint is pre-specified, as opposed to CBRA which explicitly optimises over the risk allocations. Crucially, this means the previous algorithm does not extract cutting plans approximating the chance constraint, as it is not solving a joint chance-constrained problem.

Iterative risk allocation (IRA) also decomposes the joint chance-constrained problem into a risk allocation subproblem and a linear subproblem [45]. IRA starts with an initial risk allocation, and improves the solution via hill climbing. However, IRA is not complete: while a risk allocation may exist, IRA will return no solution given an infeasible initial risk allocation. Further, IRA provides no optimality guarantees.

There have been recent work in Probabilistic Simple Temporal Networks (pSTN), which attempt to derive static assignments to execution times or an execution scheduling policy in the presence of probabilistic uncertainty [60, 64]. The approaches use a risk allocation master to generate candidate Simple Temporal Networks with Uncertainty, which feature set-bounded uncertainty and can be encoded as distance graphs and solved efficiently. Cuts are also returned to the risk allocation master in the form of negative cycles. We note that in the pSTN strong controllability case, in which we must find static execution times, the negative cycles correspond exactly to Bender's feasibility cuts. Our work, which also incorporate an objective function, can thus be considered a generalisation of the pSTN approaches. However, negative cycle detection on a distance graph using the Bellman-Ford algorithm is $O(nm)$, for n the number of variables and m the number of constraints, and we expect the pSTN algorithms to be faster on pSTN problems.

4.4 Experiments

In this sections, we compare empirical results for IRA, CRA and CBRA. We used SNOPT 7.2 as the nonlinear solver for CRA, and Gurobi 6.51 as the linear solver for the IRA and CBRA linear subproblems.

In order to directly address the motivating problem of linear AUV path planning under uncertainty, we constructed a number of benchmark 3-dimensional seafloors. For each instance, we uniformly sampled a number of points in 3 dimensional space, and constructed a convex hull.

We modeled the dynamics and observation of the AUV using parameters from a January 2018 Woods Hole Oceanographic Institute (WHOI) mission at Hawaii. The maximum velocity was 1m/s in each direction, with a 0.7m/s standard deviation representing the current. In addition, we allowed localization via Ultra-Short BaseLine (USBL) every 60 seconds, with 7m standard deviation in the observation noise. We also required that $z \leq 0$ for the vehicle at every time step: the vehicle must remain in the water.

Path planning problems were defined over the generated maps. For each map, we required velocities over 120s, 240s, and 600s, such that the vehicle ended within a rectangular $50m \times 50m$ column, with no restrictions on the final depth. The probability of collision or failing to reach the goal was required to be less than 10%, and we minimise the sum of the depths at each time step.

For a problem with n faces and m time steps, there are $3m$ variables for vehicle state, $mn+4$ variables for risk allocation for collision avoidance and goal satisfaction. There are $mn+4$ constraints for obstacle avoidance and goal satisfaction, and $6(m-1)$ constraints to describe the velocity restrictions, in addition to a single chance constraint. For example, for a problem with 72 faces and 600 time steps, we would have 45004 variables, and 46798 constraints.

We encoded CRA as described in Definition 3, with initial variables assignments set to 0, and a final tolerance of 10^{-6} For CBRA, we chose to perform 10 bisections in `BiSearch`. We terminate CBRA and return a solution when $\frac{u-l}{\max\{|u|,|l|\}} \leq 0.05$. We thus return a solution with objective within 5% of the optimum with CBRA. For convergence to optimality with the required tolerance, we use the returned CBRA solution as a starting point for CRA.

We first generated 3D maps by sampling $x, y \in [-200, 800]$, and $z \in [0, -10]$. This was expected to create maps for which no plans satisfying the chance-constraints could be found. We did not run IRA on these maps, because IRA assumes the existence of a solution.

The results are summarised in Table 4.1. Of the 90 cases, CBRA performs better in 77 instances. Note further that for many of the problems in which CBRA is faster, it is able to prove infeasibility orders of magnitude more quickly than CRA, especially for the 600 time step instances. For example, for Map 26 with 600 time steps, CBRA was able to prove infeasibility in 13.87 seconds with 6 cuts, whereas CRA needed 8638.55 seconds. Where CRA is faster, the solution times for the two algorithms are typically within the same order of magnitude.

We also considered the same maps, but with depth scaled such that $z \in [0, -50]$. These are problem instances where a solution should exist. We performed a modified version of IRA. As the method would terminate if the initial even risk allocation were infeasible, we allowed for risk to be randomly allocated if the initial risk allocation were infeasible. Random risk allocation was repeated until a feasible risk allocation were found. Even with the adjustment, IRA was unable to find feasible solutions within a 3600 minute time limit for any of the problems.

Empirically, we found that CBRA tended to have difficulty converging with the condition that $u - l \leq 10^{-6}$: a large number of cuts were generated which did not improve the upper bound in later iterations. However, by terminating when a first chance-constrained candidate has been found, we were able to speed up solutions by first finding a feasible solution with CBRA, and then using the risk allocation and vehicle positions as a starting point for CRA. This leverages the fact that CBRA typically quickly finds a first feasible solution if one exists, and good starting points typically allow SNOPT to converge faster.

Table 4.2 summarises the results, showing the times for CBRA to return a first solution satisfying the chance-constraint, the number of cuts needed, the total time for running CBRA to find a starting point and then running CRA, and the time taken if we ran CRA without the CBRA solution as a starting point. Purely running CRA was slower in 84 of the 90 instances. Again, we generally see order of magnitude improvements in computation time for the problems with larger number of time steps. As expected, the problems in which CRA performed better were those in which CBRA had trouble converging, requiring a large number of cuts.

Noting the use case, we see that the introduction of CBRA is important as it allows chance-constrained path planning to scale and provide real-time controls for the vehicles. Notably, for problems with a planning horizon of 240 seconds, we were able to find controls for all problems in under 240 seconds with combined CBRA+CRA, whereas we were only able to do so for 11 out of 30 problems using CRA. For problems with a planning horizon of 600 seconds, we are able to find optimal solutions by combining CBRA and CRA in 17 of 30 instances, while no instances were solved in under 600 seconds by running CRA alone.

In the above results, we have used CBRA to provide an initial seed solution, and solved problems to optimality using a monolithic encoding. This is because CBRA, while quick to find an initial solution, has trouble converging to optimality. As an example we may consider the maintained lower- and upper-bounds on the utility as we run CBRA over a longer time.

As an example, we may consider the lower- and upper-bounds for CBRA on Map 17, the map described by the largest number of linear constraints, planning for 120 time steps. The convergence is given in Figure 4-2. The CBRA approach was run for 3600 seconds. We see that there is a steady decrease in the upperbound, as we update the upper-bound whenever a new chance-constrained candidate is found.

However, note that bounds only seem to converge after around 500 seconds. This is significantly slower than the results in Table 4.2. Running CRA provided an optimal path in 202.33 seconds, whereas running CBRA for a first solution and subsequently

Map #	Faces	Steps	# Cuts	CBRA (s)	CRA (s)
		120	1	**0.00**	0.58
0	35	240	1	**0.01**	2.43
		600	1	**0.02**	21.13
		120	294	13.87	**12.76**
1	57	240	1	**0.01**	4.54
		600	1	**0.05**	30.61
		120	316	18.13	**13.15**
2	57	240	245	**25.89**	29.91
		600	13	**20.94**	586.22
		120	1	**0.00**	0.63
3	36	240	1	**0.01**	2.89
		600	1	**0.02**	23.08
		120	305	**12.80**	19.76
4	49	240	227	**31.59**	34.40
		600	1	**0.02**	32.46
		120	338	18.83	**17.11**
5	55	240	1	**0.02**	4.34
		600	1	**0.03**	27.97
		120	272	**7.20**	19.30
6	36	240	52	**22.18**	86.98
		600	47	**75.92**	699.00
		120	326	20.46	**19.49**
7	57	240	1	**0.01**	3.85
		600	1	**0.02**	27.04
		120	386	23.86	**17.80**
8	50	240	279	**40.19**	75.26
		600	15	**13.18**	2259.48
		120	1	**0.01**	0.80
9	48	240	1	**0.01**	3.00
		600	1	**0.02**	19.55
		120	325	**13.43**	23.31
10	42	240	54	**22.76**	158.45
		600	12	**13.83**	1098.21
		120	382	**31.79**	37.05
11	76	240	289	**46.52**	1204.17
		600	11	**17.08**	348.45
		120	1	**0.00**	0.59
12	37	240	1	**0.01**	2.40
		600	1	**0.02**	15.18
		120	466	43.94	**35.32**
13	71	240	345	**57.05**	202.78
		600	285	**143.42**	3265.55
		120	353	**25.36**	33.17
14	72	240	279	**35.68**	291.94
		600	35	**50.30**	1659.40
		120	1	**0.00**	0.67
15	37	240	1	**0.01**	2.72
		600	1	**0.02**	19.64
		120	1	**0.01**	1.20
16	60	240	1	**0.02**	4.62
		600	1	**0.04**	36.17
		120	367	36.55	**27.13**
17	83	240	1	**0.02**	9.12
		600	1	**0.06**	62.74
		120	288	**7.70**	18.14
18	34	240	208	18.98	**15.29**
		600	1	**0.02**	12.49
		120	1	**0.01**	1.35
19	58	240	1	**0.01**	4.90
		600	1	**0.04**	33.80
		120	348	24.72	**22.27**
20	67	240	268	**47.63**	50.25
		600	47	**61.30**	853.39
		120	1	**0.00**	0.71
21	37	240	1	**0.01**	2.61
		600	1	**0.02**	15.50
		120	250	**14.38**	16.48
22	68	240	189	**54.27**	85.77
		600	1	**0.05**	36.88
		120	311	**24.69**	43.51
23	77	240	229	**32.48**	235.36
		600	183	**140.94**	4675.05
		120	399	18.60	**2.25**
24	41	240	282	**22.51**	157.19
		600	1	**0.02**	19.98
		120	319	**15.20**	21.55
25	50	240	246	**64.16**	172.39
		600	15	**17.00**	41.97
		120	450	33.26	**27.04**
26	62	240	344	**46.46**	65.35
		600	6	**13.87**	8638.55
		120	1	**0.00**	0.69
27	35	240	1	**0.01**	2.14
		600	1	**0.02**	15.51
		120	330	16.50	**10.37**
28	48	240	1	**0.01**	3.19
		600	235	462.50	**358.84**
		120	201	**10.65**	21.47
29	71	240	148	**20.89**	102.46
		600	15	**29.20**	619.08

Table 4.1: Run time comparisons between CBRA and CRA for infeasible 3D AUV maps, with number of cuts found by CBRA.

Map #	Faces	Steps	CBRA (s)	# Cuts	CBRA+CRA (s)	CRA (s)
0	35	120	3.66	27	**31.74**	35.23
		240	13.47	69	**56.11**	295.21
		600	21.82	8	**261.59**	1825.22
1	57	120	4.75	25	**32.09**	65.10
		240	17.17	62	**61.81**	307.14
		600	704.13	935	**819.83**	2764.71
2	57	120	5.13	26	**35.01**	56.84
		240	18.76	78	**74.35**	252.66
		600	53.95	44	**530.94**	3726.68
3	36	120	3.67	24	**10.83**	40.23
		240	15.80	71	**35.70**	183.01
		600	29.73	27	**930.77**	1474.76
4	49	120	4.63	26	53.81	**49.46**
		240	17.08	61	**37.94**	259.84
		600	334.69	613	**476.21**	3158.51
5	55	120	6.99	30	**19.32**	84.56
		240	22.54	86	**42.14**	322.92
		600	59.92	70	**164.83**	2586.02
6	36	120	3.52	25	**15.04**	40.98
		240	14.84	67	**38.55**	217.22
		600	332.20	556	**413.36**	1048.81
7	57	120	5.66	27	**17.61**	54.52
		240	20.27	81	**53.57**	262.44
		600	94.70	193	**197.37**	4639.95
8	50	120	5.25	32	**12.81**	66.78
		240	20.61	78	**36.05**	192.94
		600	753.78	1059	**824.00**	2107.08
9	48	120	4.64	30	**14.81**	94.01
		240	21.21	87	**45.45**	284.89
		600	59.70	118	**154.66**	1719.39
10	42	120	3.71	25	**21.52**	53.45
		240	17.82	53	**37.38**	175.64
		600	450.30	697	**539.42**	3083.50
11	76	120	6.94	25	**48.99**	119.48
		240	26.14	82	**71.89**	665.69
		600	899.72	954	1043.37	**612.37**
12	37	120	4.83	29	**12.97**	34.68
		240	13.94	68	**29.42**	178.80
		600	1213.22	1314	**1506.90**	3141.87
13	71	120	6.76	25	**42.19**	94.87
		240	27.02	80	**62.37**	591.80
		600	175.19	241	**411.60**	1035.14
14	72	120	6.98	26	**36.29**	130.32
		240	31.33	98	**53.89**	542.03
		600	807.65	814	**944.56**	1375.46
15	37	120	3.85	28	**15.85**	30.09
		240	12.60	63	**46.94**	158.75
		600	249.71	575	**404.41**	1335.90
16	60	120	6.74	28	**18.25**	69.76
		240	16.87	75	**34.10**	793.40
		600	2299.96	2093	2375.18	**2361.94**
17	83	120	8.15	27	**23.36**	202.33
		240	30.27	102	**96.66**	892.04
		600	97.11	12	**580.83**	988.05
18	34	120	3.55	27	**28.48**	31.97
		240	12.47	77	**63.35**	99.89
		600	26.14	7	**296.38**	1535.66
19	58	120	5.57	28	**26.39**	79.86
		240	21.97	89	**59.36**	330.96
		600	124.51	220	**435.23**	2557.80
20	67	120	6.52	27	**19.18**	81.20
		240	23.57	104	**43.91**	316.23
		600	640.56	789	**739.81**	774.49
21	37	120	4.53	24	**14.07**	37.58
		240	15.94	66	136.63	**135.42**
		600	32.77	41	**222.93**	1789.12
22	68	120	6.32	35	**21.76**	116.65
		240	48.19	15	**126.08**	368.90
		600	802.84	841	1085.12	**884.00**
23	77	120	8.26	32	**41.37**	113.82
		240	29.07	93	**56.27**	600.41
		600	1339.11	936	1461.75	**727.80**
24	41	120	3.83	23	**17.57**	74.60
		240	17.31	67	**33.99**	174.67
		600	124.43	198	**203.62**	2511.03
25	50	120	5.45	25	**34.77**	75.75
		240	26.30	95	**38.07**	243.66
		600	1639.54	1447	**1723.15**	4085.27
26	62	120	5.94	25	**25.72**	84.19
		240	21.75	66	**39.82**	387.78
		600	79.14	73	**1753.15**	2117.43
27	35	120	3.28	27	**11.24**	30.04
		240	15.50	71	**30.48**	176.77
		600	35.57	59	**494.75**	1163.20
28	48	120	4.97	29	**20.18**	87.58
		240	18.25	78	**52.03**	207.04
		600	677.90	932	**764.59**	2504.45
29	71	120	7.03	31	**14.83**	124.16
		240	31.57	134	**84.28**	415.13
		600	346.98	357	**477.35**	2184.00

Table 4.2: Run times for feasible 3D AUV maps, for CBRA solution within 5% of the optimal, total time for optimality using the 5% solution as a starting point for CRA, and optimality using just CRA.

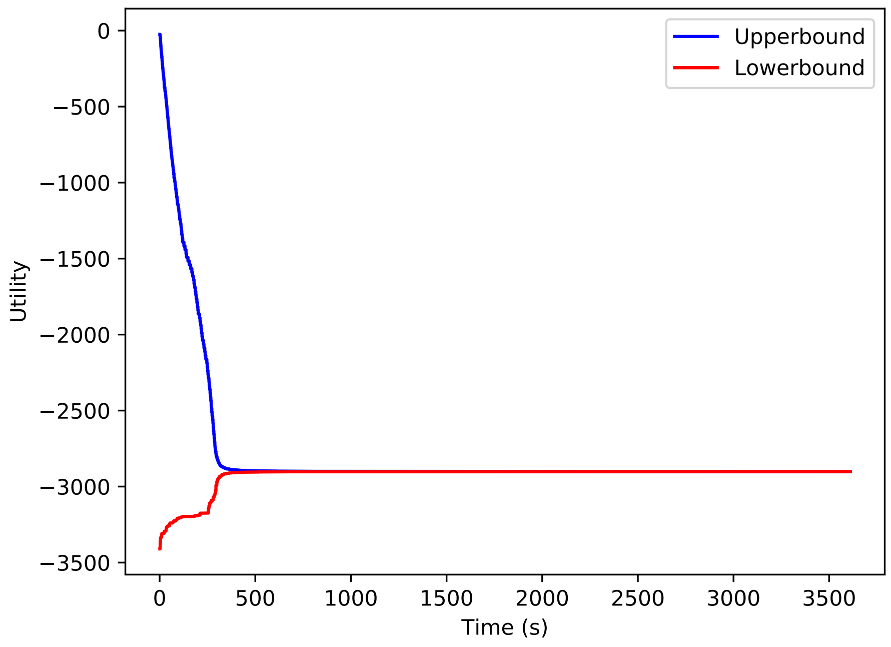

Figure 4-2: Convergence of the quality of path for Map 17 in the the benchmark set. Utility is the sum of the depth of the vehicle, simulating a bottom following mission, such that a lower utility is better.

solving with CRA provided an optimal solution in 23.36 seconds.

If we considered the optimality gap as a proportion of the maintained bounds, $\frac{u-l}{\max\{|u|,|l|\}}$, we find that in this instance, the bounds are within 20% of each other after 271.09 seconds, 10% after 290.10 seconds, 5% 300.46 seconds, 1% after 363.39 seconds. This provides an empirical demonstration of the difficulties with convergence.

4.5 Summary

Chance-constrained programming has been successful in applications where constraints must be satisfied despite stochasticity in the problem, from autonomous vehicle controls to planning for power flow infrastructure. Joint chance constraints are seen as intractable, and the risk allocation approximation has been popular. While Convex Risk Allocation encodes the problem as a single convex program, the solution time required is often prohibitive.

In this chapter, we have proposed a new algorithm, Conflict-Based Risk Allocation (CBRA). We decompose the problem into a linear subproblem and a chance constraint subproblem. We are thus able to solve the linear subproblem with highly optimised commercial solvers. We learn linear constraints approximating the chance constraint in the space of deterministic variables, using cutting planes to inform the master risk allocation problem. This allows us to quickly find first solutions or prove that the problem is infeasible.

We have shown empirically, using a set of vehicle path planning problems, that CBRA is often able to prove infeasibility orders of magnitude faster than CRA. We also show that CBRA is often able to find a good solution very quickly. Using the first solution from CBRA as a starting point and then optimising with CRA, we are also able to find optimal solution much more quickly than just running CRA alone. Crucially, this enables us to perform real-time risk-bounded model predictive control.

While we have used cutting planes to quickly detect infeasibility, and to provide a good starting point for interior point convex optimisation, we can further exploit the technique for mixed discrete-continuous problems. Specifically, because we have a sequence of linear optimisations, we can easily identify the sources of infeasibility in terms of linear constraints which taken together exclude solutions. We may also look at the active linear constraints at optimal solutions to identify the subset which may prevent better solutions to be found. This is exploited in the next chapter.

Chapter 5

Chance Constrained Mixed Logic Linear Program

In Chapter 3, I introduced a high level algorithm for solving CC-MLLPs. The algorithm requires a tractable subsolver for CC-LPs, described in Chapter 4. However, further efficiency can be gained at the search level. Computation can be reduced if we were able to restrict search down unpromising combinations of assignments to the logical variables.

In this chapter, I describe the relaxations solved given partial assignments, which allow for early pruning. I also describe how additional logical constraints are extracted to guide discrete search, in the form of conflicts which summarise why certain portions of the search space are not promising.

The approaches exploit linearity in the relaxed subproblems, as well as our approach to full CC-LPs using successive linear approximations. I prove the correctness of the conflicts extracted. Empirical evaluations show that the conflict directed method is able to achieve at least an order of magnitude speed up in finding the first solution, and to prove optimality.

5.1 Relaxed Subproblem

In the main algorithm, a relaxed CC-LP is solved whenever a new assignment to one of the logical variables is made. This results in a set of implied linear constraints. We thus have a relaxed CC-LP, in which we consider only the implied constraints.

As in NIRA [46], we perform a further full-risk relaxation, in which we allocate the entirety of the risk bound to each random variable. This is a relaxation because the risk allocated to each random variable can only be less than or equal to the risk bound when solving for the actual CC-LP.

```
Input: Partial assignments $\mathbf{p}'$ to logical decision variables, real-valued decision
       variables $\mathbf{x}$, random variables $\{u_i\}$, constraints $\{C_i\}$, incumbent objective $U$,
       risk bound $\Delta$
Output: FeasOpt? whether the relaxation is infeasible or suboptimal
1 $u_i^+ \leftarrow$ the upper $\Delta$-quantile for each random variable $u_i$
2 SolveLP($\min_{\mathbf{x}} c(\mathbf{x})$, $st$ $\{\mathbf{A}_i\mathbf{x} + u_i^+ \leq \mathbf{b}_i\}$ $\forall i$, $\Psi_i(\mathbf{p}') = $ True)
3 return FeasOpt?
```

Algorithm 4: Relaxed CC-LP.

The procedure is given in Algorithm 4. We first find the upper Δ-quantile for each random variable. Then, for each constraint in the continuous portion of the problem, if the logical formulae is evaluated to be true, we add the corresponding linear constraint to the LP relaxation, replacing the random variable with the corresponding quantile.

For any full assignment to the logical decision variables which contain the partial assignments $\mathbf{p}'$, the solution space of the full CC-LP will be smaller. Again, this is due to two reasons:

1. the set of linear constraints in the full CC-LP will be a superset of the linear constraints in the relaxation; and

2. the risk allocated to each random variable must be less than or equal to Δ.

Thus, if the relaxation is infeasible, then the full CC-LP will also be infeasible. Similarly, if the relaxation is feasible, then the objective obtained is an optimistic bound on that of the full CC-LP.

5.2 Conflict Extraction

One of the keys for the proposed decomposition algorithm for CC-MLLPs is the conflict extraction.As the algorithm performs computation on the continuous portion of the problem, we would like to extract information about the subproblems. In particular, we would like to understand which of the linear constraints were responsible for ruling out all feasible solution, or eliminating solutions with better objective values.

If we were able to identify these conflicting linear constraints, we can then trace back to the associated logical variable assignments. This would allow us to guide the discrete search, by avoiding combinations of assignments leading to unpromising parts of the search space.

We first outline the approach used for the relaxed CC-LP, and then extend the approach to the full CC-LP.

5.2.1 Conflict Extraction for Relaxed CC-LP

The resulting CC-LP thus becomes an LP, which can be solved using off-the-shelf solvers. If the relaxed CC-LP is infeasible, we know that the full CC-LP is also infeasible, and we may extract infeasibility conflicts by looking at irreducible infeasible sets. Alternatively, if we observe optimality, we may look at the active constraints to find optimality conflicts if the relaxed CC-LP has a worse objective function than the incumbent.

The procedure is given in Algorithm 5. It takes as input the set of implied linear constraints as well as the associated logical formulae, the incumbent objective, and a risk bound. It returns whether the relaxed CC-LP is feasible and better than the incumbent, and any conflicts if not.

We first construct the relaxation as in Algorithm 4, by finding the upper Δ-quantiles of the random variables, and considering only the constraints implied by the partial logical assignments.

We then call an off-the-shelve LP solver for the relaxed CC-LP, and consider the solution. If the problem is infeasible, then we may identify the irreducible infeasible

```
Input: Partial assignments $\mathbf{p}'$ to logical decision variables, real-valued decision
       variables $\mathbf{x}$, random variables $\{u_i\}$, constraints $\{C_i\}$, incumbent objective $U$,
       risk bound $\Delta$
Output: FeasOpt? whether the relaxation is infeasible or suboptimal, $\Psi'$ the set of
        conflicts if so
1  $u_i^+ \leftarrow$ the upper $\Delta$-quantile for each random variable $u_i$
2  SolveLP($\min_{\mathbf{x}} c(\mathbf{x})$, st $\{\mathbf{A}_i\mathbf{x} + u_i^+ \leq \mathbf{b}_i\}\ \forall i,\ \Psi_i(\mathbf{p}') =$ True)
3  if LP infeasible then
4      $\Psi' \leftarrow \bigwedge_j \Phi_j$ for all $j$ such that $\{\mathbf{A}_j\mathbf{x} + u_j^+ \leq \mathbf{b}_j\}$ is in the irreducible infeasible set
5      FeasOpt? $\leftarrow$ False
6  else if LP feasible but objective worse than incumbent $U$ then
7      $\Psi' \leftarrow \bigwedge_j \Phi_j$ for all $j$ such that $\{\mathbf{A}_j\mathbf{x} + u_j^+ \leq \mathbf{b}_i\}$ is active at the solution
8      FeasOpt? $\leftarrow$ False
9  else
10     FeasOpt? $\leftarrow$ True, $\Psi' \leftarrow \emptyset$
11 end
12 return FeasOpt?, $\Psi'$
```

Algorithm 5: Conflict extraction from Relaxed CC-LP.

set of constraints. These linear constraints meant that no solution could be found. We would like to prevent these linear constraints from being implied in all future candidates when searching for assignments to the logical variables.

We thus collect the associated logical formulae for each linear constraint. When the conjunction of these logical formulae evaluate to `True`, the same set of linear constraints will be present, and thus no solution can be found. We may thus return the conjunction of the logical formulae as a conflict.

Similarly, when the LP solver returns a solution which is optimal, but with an objective value worse than the incumbent, we may consider the set of active linear constraints. The set of active constraint prevent the objective value from getting any better, and we would like to avoid these linear constraints in future candidates. Similar to the above, we thus return the conjunction of the associated logical formulae as a conflict.

5.2.2 Conflict Extraction for CC-LP using Cutting Planes

We now consider the extraction of conflicts from the full CC-LP, building on the ideas presented in conflict extraction from the relaxed CC-LP. While the CC-LP with the union bound approximation is a nonlinear convex program, we may still employ similar techniques to identify sets of logical statements which must be negated in order to generate better candidates.

The key idea is that we may incrementally approximate the convex nonlinear union bound using a sequence of cutting planes. These are linear constraints which restrict the feasible region, which approximate the nonlinear constraint. In Chapter 4, we have demonstrated that the application of cutting planes allows us to quickly identify infeasible solutions, and provide good initialisation for nonlinear solvers. We now demonstrate how cutting planes may also be used to identify conflicts in a CC-MLLP setting. The procedure is given in Algorithm 6.

As we are solving the full CC-LP, we explicitly create decision variables for upper- and lower-bounds of random variables, and impose a union bound constraint. The procedure is similar to that described in Chapter 4. We again begin with an empty set of cutting plane constraints C'. We then enter a loop and iteratives add cutting planes.

At each iteration, we first solve an LP with the cutting planes and the original linear constraints, generating solutions that are as far away from the constraint boundaries as possible. This is done through the standard cutting planes method of finding the Chebyshev Centre, by maximising a decision variable r representing the distance of the solution to the boundaries.

While our previous procedure in Chapter 4 would exit if the LP is infeasible, here we also return an infeasibility conflict using the same methods as the relaxed CC-LP conflict extraction in Algorithm 5.

If the LP returns a solution, we may evaluate the risk of the solution using the solution values for lower- and upper-bounds of the random variables. If the risk is greater than the risk bound Δ, we construct cutting planes as in Chapter 4, and add them to the set of constraint in C'. The LP is the solved again with the newly added constraints.

We exit the cutting planes phase of the procedure when a candidate meeting the risk bound is found. As for convex CC-LPs, we use the solution values for $\mathbf{x}$

```
Input: Assignments $\mathbf{p}'$ to logical decision variables, real-valued decision variables $\mathbf{x}$,
       random variables $\{u_i\}$ such that $u_i$ has PDF $f_i$ and CDF $F_i$, constraints
       $\{C_i\}$, incumbent objective $U$, risk bound $\Delta$
Output: FeasOpt? whether the relaxation is infeasible or suboptimal, $\Psi'$ the set of
        conflicts if so
1  $u_i^+ \leftarrow$ decision variables for the upper-bounds of each random variable $u_i$
2  $C' \leftarrow \emptyset$ ;                    /* Set of cutting planes linear inequalities */
3  while True do
4      SolveLP($\max_{\mathbf{x},\{u_i^+\},r} r$, st $\{\mathbf{A}_i\mathbf{x} + u_i^+ + r \leq b_i\}$ $\forall i, \Psi_i(\mathbf{p}') =$ True, and for $C'$)
5      if LP infeasible then
6          $\Psi' \leftarrow \bigwedge_j \Phi_j$ for all $j$ such that $\mathbf{A}_j\mathbf{x} + u_j^+ + r \leq b_j$ is in the irreducible
            infeasible set
7          FeasOpt? $\leftarrow$ False
8          return FeasOpt?, $\Psi'$
9      end
10     $\mathbf{v}^+ \leftarrow$ the LP solution values for $\mathbf{u}^+$
11     Risk $\leftarrow \sum_{v_i \in \mathbf{v}^+} 1 - F_i(v_i)$
12     if Risk $> \Delta$ then
13         $\mathbf{f}^+ \leftarrow [f_i(v_i)]^T_{v_i \in \mathbf{v}^+}$
14         $C' \leftarrow C' \cup \{-\mathbf{f}^+\mathbf{u}^+ + r \leq \Delta -$ Risk $- \mathbf{f}^+\mathbf{v}^+\}$;
15     else
16         Break
17     end
18 end
19 Solve CC-LP with nonlinear solver;
20 if CC-LP feasible but objective worse than incumbent $U$ then
21     $\Psi' \leftarrow \bigwedge_j \Phi_j$ for all $j$ such that $\mathbf{A}_j\mathbf{x} + u_j^+ \leq b_j$ is active at the solution
22     FeasOpt? $\leftarrow$ False
23 else
24     FeasOpt? $\leftarrow$ True, $\Psi' \leftarrow \emptyset$
25 end
26 return FeasOpt?, $\Psi'$
```

Algorithm 6: Conflict extraction from Full CC-LP.

and the upper-bounds of the random variables as an initial point for an off-the-shelf nonlinear solver. Once a solution has been found, we examine the objective value. If the objective is better than the incumbent, we exit noting that the new candidate is better than the incumbent. Otherwise, we extract a suboptimality conflict by considering the active constraints, similar to the way described for the relaxed CC-LP conflict extraction in Algorithm 5.

5.3 Correctness of Conflicts

The algorithm derives its efficiency from the ability to prune large parts of the search space using the conflicts. However, it remains to be shown that the algorithm does not prune combinations of logical variables which lead to CC-LPs which are feasible and better than the incumbent. In this subsection, I present results which guarantee the correctness of the conflicts. The proofs of correctness for the conflicts extracted from the full CC-LP is presented first. The proofs of correctness for relaxed CC-LPs follow, drawing on similar ideas to those used in the results for conflict extraction from full CC-LPs.

In the following discussions, I use the following notation:

- $\tilde{L}$ the set of linear constraints returned as part of conflict extraction;
- $\mathbf{U}$ the set of random variables for a subproblem; and
- $\tilde{\mathbf{U}}$ the set of random variables featuring in $\tilde{L}$.

I first show that the feasibility conflicts only prune parts of the search space which contain infeasible CC-LPs. The proof procedure can then be followed to show that the optimality conflicts only prune parts of the search space with CC-LPs containing worse utility.

Theorem 1 (Correctness of feasibility conflicts for full CC-LPs)**.** *Let $\tilde{L}$ the set of linear constraints returned as feasibility conflicts from the cutting planes methods for some CC-LP with risk bound Δ. Then any CC-LP with $\tilde{L}$ as a subset of its linear constraints will also be infeasible.*

Proof. The cutting planes method provide an approximation to the union bound approximation of the chance constraint $\sum_{u \in \mathbf{U}} F(u \leq d) \leq \Delta$. If $\tilde{L}$ is the set of linear constraints returned, it means these linear constraints feature in the irreducible infeasible subset. There is thus no solution to a system of constraints which feature both $\tilde{L}$ and a linear *relaxation* of the chance constraints.

Thus the system of constraints

$$\sum_{u \in \mathbf{U}} P(u \leq d) \leq \Delta \quad | \quad \tilde{L} \tag{5.1}$$

has no solution.

Note that, for any random variable not featured in $\tilde{L}$, there are no constraints on the risk bounds, as $\tilde{L}$ are the only constraints in the above system apart from the chance constraint. Thus, we may essentially choose the risk allocations for such random variables to be zero in the above system. The above system of constraints thus reduces to

$$\sum_{u \in \tilde{\mathbf{U}}} P(u \leq d) \leq \Delta \quad | \quad \tilde{L} \tag{5.2}$$

where $\tilde{\mathbf{U}}$ is the set of random variables featuring in $\tilde{L}$.

Since this reduced system of equations is infeasible, any CC-LP which has $\tilde{L}$ as a subset of its linear constraints will also be infeasible.

□

The key insight for the above proof lies in the fact that we can write a relaxed subproblem, with just the linear constraints returned in the feasibility conflicts, and a chance constraint defined only over the random variables featured in the linear constraints. This can be shown to be infeasible, meaning that any other CC-LP containing these linear constraints will also be infeasible.

This result proves that using the feasibility conflicts, the conflict directed algorithm only eliminates CC-LPs which will be infeasible. Again, the conflicts are summaries of assignments to the logical constraints which lead to combinations of linear constraints infeasible given the risk bound. By avoiding these logical assignments, we will never obtain CC-LPs containing the same linear constraints.

I now provide a similar result for optimality conflicts. The proof procedure is similar, also making use of a reduced system of constraints featuring only the linear constraints.

Theorem 2 (Correctness of suboptimality conflicts for full CC-LPs)**.** *Let $\tilde{L}$ the set of linear constraints returned in suboptimality conflicts from the cutting planes methods for some CC-LP with risk bound Δ, with a resulting cost C. Then any CC-LP with $\tilde{L}$ as a subset of its linear constraints will also have cost lower-bounded by U.*

Proof. Optimality conflicts are extracted as the active constraints at the optimal solution. Thus we know that, for $f(x)$ the cost function in the CC-LP, the numerical program:

$$\min f(x) \quad | \quad \tilde{L}, \sum_{u \in \mathbf{U}} P(u \leq d) \leq \Delta \tag{5.3}$$

has optimal cost C.

Note that, for any random variable not featured in $\tilde{L}$, there are no constraints on the risk bounds, as $\tilde{L}$ are the only constraints in the above system apart from the chance constraint. Thus, we may essentially choose the risk allocations for such random variables to be zero in the above system. The numerical program thus reduces to

$$\min f(x) \quad | \quad \tilde{L}, \sum_{u \in \mathbf{U}} P(u \leq d) \leq \Delta \tag{5.4}$$

where $\tilde{\mathbf{U}}$ is the set of random variables featuring in $\tilde{L}$. This program thus also has cost C

Consider an CC-LP with $\tilde{L}$ as a subset of its linear constraints. Any additional linear constraints would place additional restrictions on the feasible set of solutions, leading to a higher cost. Thus any CC-LP containing $\tilde{L}$ will have a cost lower-bounded by C □

The above result shows that the optimality conflicts are correct, and only eliminate logical assignments which lead to CC-LPs with worse cost.

I now present correctness results for the conflicts extracted from relaxed CC-LPs. The key to the results is that the relaxed CC-LPs have, by construction, a larger feasible solution space than any full CC-LPs expanding on the set of partial logical assignments. This is because the relaxations contain relaxed versions of the chance constraint.

Theorem 3 (Correctness of conflicts for relaxed CC-LPs)**.** *Let $\tilde{L}$ the set of linear constraints returned as feasibility conflicts from a relaxed CC-LP with risk bound Δ. Then any CC-LP with $\tilde{L}$ as a subset of its linear constraints will also be infeasible.*

Let $\tilde{L}$ the set of linear constraints returned as suboptimality conflicts from a relaxed CC-LP with risk bound Δ. Then any CC-LP with $\tilde{L}$ as a subset of its linear constraints will also be suboptimal.

Proof. Suppose a relaxed CC-LP is infeasible, with $\tilde{L}$ the set of linear constraints identified as feasibility conflicts. Then, by construction of the relaxed CC-LP, $\tilde{L}$ is a set of linear constraints with a relaxed chance constraint. Any full CC-LP containing the corresponding linear constraints will also need to satisfy the full chance constraint. The full CC-LP will thus not be feasible, as the full chance constraint is more restrictive than the relaxation.

Suppose a relaxed CC-LP is suboptimal, with $\tilde{L}$ the set of linear constraints identified as suboptimality conflicts. Similar to the above argument, by construction of the relaxed CC-LP, $\tilde{L}$ is a set of linear constraints with a relaxed chance constraint. Any full CC-LP containing the corresponding linear constraints will also need to satisfy the full chance constraint. The full CC-LP will have a worse objective value, again due to the fact that the full chance constraint is more restrictive than the relaxation.

□

5.4 Empirical Results

In this section, we present empirical evaluations of the new conflict-directed algorithm on benchmark problems, and compare them against existing nonconvex risk allocation

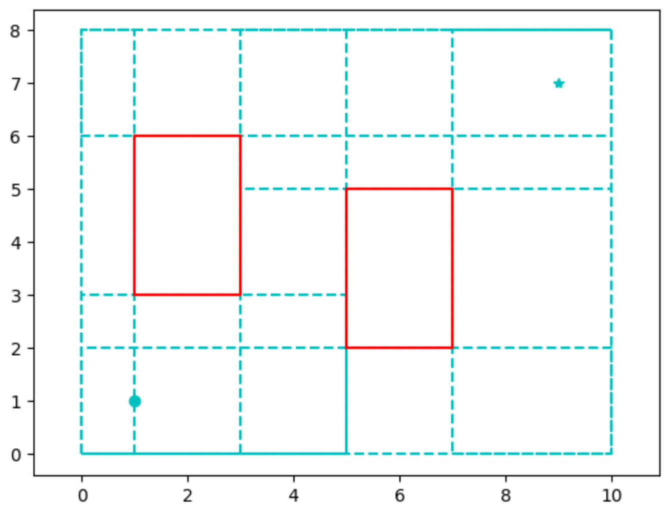

Figure 5-1: Simple 2D path planning benchmark.

approaches.

5.4.1 Simple Path Planning Benchmark

The key performance properties of the algorithm was first demonstrated with the simple benchmarking problem, as shown in Figure 5-1. The vehicle must travel from [1,1] to [9,7]. Planning was performed for varying number of time steps to demonstrated the scalability of the algorithm. We compared our conflict-directed algorithm against NIRA from [46], except swapping out the subsolver from SNOPT to our faster cutting planes based CCLP. We planned with simple linear dynamics, and a risk bound of 0.2.

As seen in Table 5.1, CBRA has significant time savings over NIRA, when considering the time to arrive prove optimality by exhausting search candidates. Crucially, if we were to provide control with 1 second time steps, CBRA would be able to provide real-time control for horizons of up to 18 seconds, whereas NIRA would have trouble planning over a 10 second horizon.

Our key claim was that we would observe faster computation times because the conflict-directed algorithm would be able to effectively prune the search space. The

Time Steps	8	10	12	14	16	18	20
NIRA	4.64	12.54	28.14	65.19	125.09	248.20	521.35
CBRA	**1.20**	**2.45**	**3.66**	**7.08**	**11.30**	**17.18**	**53.49**

Table 5.1: Time in seconds to prove optimality for the simple benchmark problem.

Time Steps	8	10	12	14	16	18	20
NIRA	338	871	1955	4015	7527	13465	23988
CBRA	**96**	**171**	**259**	**430**	**667**	**897**	**1968**

Table 5.2: Number of nodes expanded for the simple benchmark problem.

conflicts should rule out parts of the search space, drastically reducing the number of expansions.

As seen in Table 5.2, the number of nodes expanded by CBRA is far fewer than that required by NIRA. This becomes more significant as the number of time steps increase. Even though the number of distinct logical combinations increases exponentially, at each time step the vehicle must choose from the same set of regions. Thus, a conflict denoting a sequence of regions becomes more powerful as the number of time steps increase - the same conflict now prunes an exponentially larger number of possible trajectories.

One key motivation for the pruning was to avoid expensive evaluations of full CCLPs. At each partial solution, a fixed risk relaxation is solved as a linear program. However, a full CCLP requires a sequence of LPs to be solved to prove feasibility, and then possibly a nonlinear convex optimisation if the CCLP is feasible. The results are given in Table 5.3. We again observe a similar amount of time saving of CBRA over NIRA.

We also note that both NIRA and CBRA are able to be used as anytime al-

Time Steps	8	10	12	14	16	18	20
NIRA	13	30	66	136	293	730	2257
CBRA	**5**	**13**	**19**	**31**	**41**	**59**	**153**

Table 5.3: Number of full CCLPs solved for the simple benchmark problem.

Time Steps	8	10	12	14	16	18	20
NIRA	**0.08**	**0.12**	1.35	3.06	5.29	9.62	15.24
CBRA	0.10	0.15	**0.49**	**0.70**	**0.99**	**1.58**	**2.21**

Table 5.4: Time in seconds to the first solution for the simple benchmark problem.

gorithms. We may thus consider the time to first solution of the algorithms. The comparison is given in Table 5.4.

Interestingly, NIRA is quicker to first solutions for smaller number of time steps. There are two reasons for this. First, CBRA expends additional computation to find conflicts when an infeasible fixed-risk relaxation occurs, because it must calculate the irreducible infeasible subset. Second, there might be a slight computation expenditure used when propagating on conflicts. However, these additional computation is overshadowed for long time step plans, in which the additional computation to find and propagate on conflicts is far outweighed by the number of pruned partial CC-LPs before a first solution.

As CBRA only begins pruning suboptimal solutions after the first solution is found, the quality of first solutions is the same for both algorithms - the first feasible full CC-LP found is identical. However, we may consider the improvements of the best candidate solutions with time. This is shown in Figure 5-2. In addition to being able to quickly find a first solution, CBRA is able to very quickly find the best candidate, in about 10 seconds. By contrast, NIRA takes about 100 seconds to come up with the same candidate.

5.4.2 Scarborough Oceanography Benchmark

The algorithms were run on a benchmark problem inspired by a WHOI AUV mission. The map is given in Figure 5-3. The regions outlined in red were obstacles, and the free space was divided into the regions outlined in black. The vehicle was required to traverse from [-531, 2565] to [984, -2974] within 20 time steps, with a 10% risk bound. We compared the run time of the new dCBRA algorithm against NIRA as described in [46], with a planning time out of 3 hours. Both algorithms used CBRA as solvers

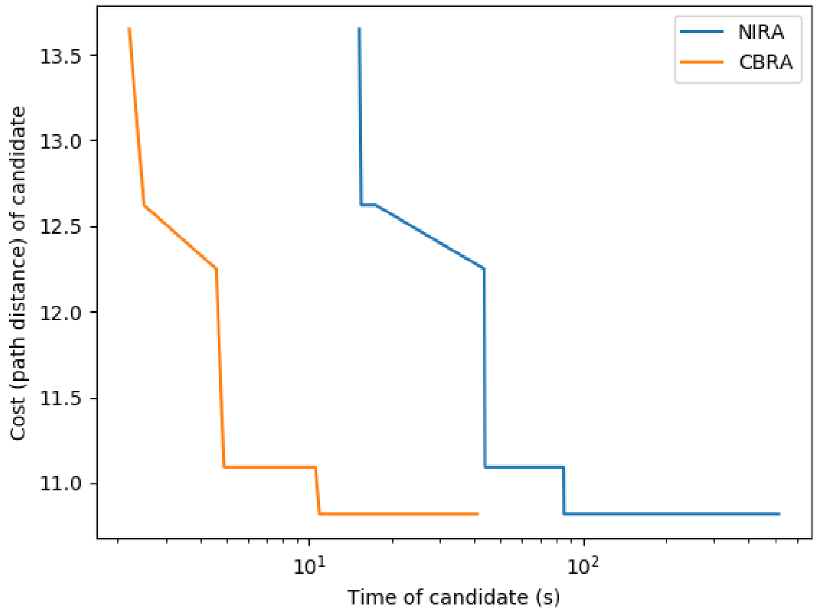

Figure 5-2: Best candidate quality as a function of time for the simple benchmark problem, for a 20 time step plan.

for the CC-LP subproblems. This was done in order to provide a fair comparison, such that dCBRA did not enjoy speedups due to a faster subsolver.

The dCBRA algorithm was able to solve the problem to optimality within 2522.22 seconds, while NIRA was unable to finish in time. This improvement is observed because NIRA was only pruning branches based on a fixed risk relaxation, while dCBRA was able to prune large sections of the search space using conflicts derived from the CBRA.

The use of infeasibility conflicts allowed dCBRA to learn which linear constraints were in conflict whenever an infeasible relaxation or full candidate was found. As such, dCBRA obtain a first solution in 3.5 seconds, while it took NIRA 21.77 seconds - dCBRA was able to generalise the infeasible partial candidates and skip over similar candidates for a first solution.

Beyond the time to the first solution, we also observe that dCBRA expanded 110881 candidates, of which 596 were full candidates. In contrast, NIRA expanded 377435 candidates, of which 25635 were full candidates. The lower total number of

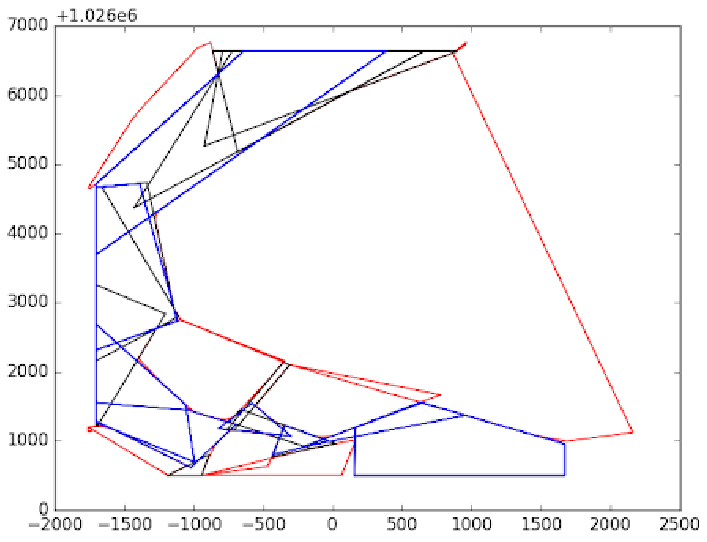

Figure 5-3: Scarborough Oceanography map.

candidates expanded is directly responsible for the much lower run time. Further, note the ratio of full candidates to total candidates explored by dCBRA was two orders of magnitude lower. This means that the use of conflicts was effective in reducing the number of evaluations of full CCLPs, each of which would have been computationally expensive.

Finally, the best solution found using conflicts to guide search had distance 2875382.13. In contrast, the best solution at termination for NIRA had distance 2982142.31. Thus, at termination, NIRA was yet to find the optimal solution.

5.5 Summary

In this chapter, I have introduced the conflict extraction methods central to the proposed algorithm for CC-MLLPs. The conflict extraction techniques allow us to identify subsets of linear constraints which prevent feasible solutions, or rule out better quality solutions. Given the conflicting linear constraints, we can trace back to the logical conditions which entail the linear constraints. These conditions thus form the conflicts we require in our discrete search.

I have provided proofs for the correctness of the derived conflicts. Empirically, I have demonstrated the effectiveness of conflict guided search for CC-MLLPs. I have shown that orders of magnitude improvements in speeds can be gained, as the conflicts allow us to eliminate large sections of the search space, reducing the number of expansions in search needed to first solution and to optimality.

The focus of the thesis has thus far been on CC-MLLPs with explicitly stated distributions. However, in real world applications, the true distributions may not be available, and must be estimated from data. In the next Chapter, we provide methods for deriving correct bounds for uncertainty from data, which allow us to provide chance-constrained solutions with quantifiable confidence.

Chapter 6

Data-driven Uncertainty Representation

Chance-constrained programs aim to account for uncertainty while providing guarantees on the probability of success. Chance-constrained approaches typically rely on good probabilistic models to provide guarantees, because inaccurate distributions to represent uncertainty may lead to decisions incurring excessive risk. While models may be derived from sampled data using existing density estimation approaches, there is little guarantee on the correctness of the resulting model, especially with respect to variance in sampling.

In this chapter, we propose an alternative to probability distributions in the particular context of chance-constrained scheduling. Instead, we derive set-bounded uncertainty from data, with additional quantitative guarantees on the correctness of the bounds with respect to variance in sampling. In relation to the general CC-MLLPs, these set-bounds could be considered analogous to the results of risk allocation - we find bounded sets which contain a sufficiently large probability mass, and make sure that our decisions have enough leeway to cover all outcomes within the bounds.

In addition to guarantees of correctness, the derived set-bounded uncertainty allows us to apply well-explored algorithms for scheduling Simple Temporal Networks with Uncertainty. We provide two such methods, one for estimating uncertainty bounds with respect to the true distribution, and one for estimating uncertainty bounds given a finite number of future executions. We provide proofs of soundness, as well as empirical comparisons of our methods with kernel density estimation in a subway scheduling domain.

6.1 Introduction

Scheduling under uncertainty is a problem with real-world applications, including planetary rover [37] and human-robot interaction [28]. The Simple Temporal Problem with Uncertainty (STNU) framework [55] has been a particularly successful formalism for scheduling given set-bounds for uncertain durations, with recent algorithmic breakthroughs providing efficient scheduling algorithms [39, 43]. However, sometimes it is impossible to cover the entire range of possible outcomes for uncontrollable durations, either due to possibly unbounded uncertainty, or due to tight timing requirements. In such cases, a chance-constrained approach may be adopted, in which scheduling only accounts for some range of outcomes covering a required probability mass [16, 60].

Existing chance-constrained approaches typically rely on reasonable probability distributions to represent uncertainty. When distributions are fitted from historical data, for example with kernel density estimation (KDE), there are concerns with over-fitting. When such models are used in chance-constrained programming, the resulting decisions may not provide correct probabilistic guarantees due to incorrect assessments of the likelihood of outcomes. Inaccuracies in distributions thus invalidate the key feature of chance-constrained programming, the ability to provide guarantees on succcess.

In this chapter, we concentrate on providing accurate models for chance-constrained scheduling. Instead of relying on distributions fitted from data, we propose using bounded sets which cover required probability of outcomes, with probabilistic guarantees on the correctness of the bounds given to variances in sampling. The resulting bounded sets are exactly those used in the STNU literature, allowing us to provide chance-constrained schedules while leveraging efficient STNU algorithms. In this

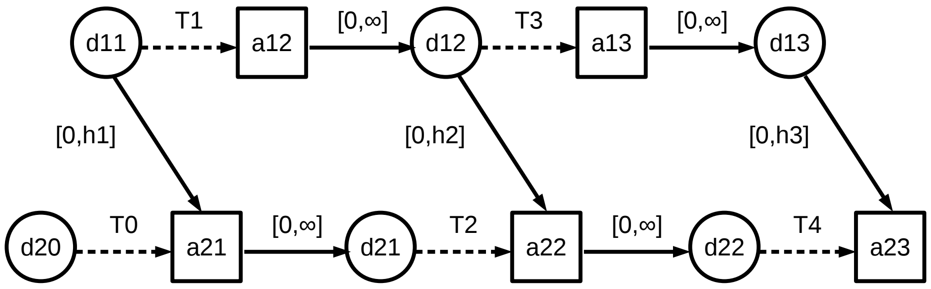

Figure 6-1: Subway scheduling example.

chapter, we present two such methods for constructing the models, one estimating bounds on the true distribution and one estimating bounds for a finite number of future executions. We present theoretical proofs of correctness, as well as empirical comparison of the correctness of the new methods with KDE, using a set of subway traversal data. We also compare the conservatism incurred by adopting such methods through a subway scheduling problem.

6.2 Problem formulation

We begin by offering a motivating example of a chance-constrained problem, in which we are not given distributional representations of uncertainty. The example serves to illustrate the key insight in our proposed approach. We do not need to reason over the actual distribution to guarantee the probability of success - we only need to extract upper and lower bounds on the possible outcomes. In doing so, we are estimating nonparametric statistics, and can provide further probabilistic guarantees on the correctness of the upper and lower bounds.

Example 5. *Consider an example problem as in Figure 6-1, in which trains on the Boston subway system must be scheduled. There are two consecutive trains, and we would like to find the departure times of trains to minimise the headway between the trains, such that the delay between the departure of the first train and the arrival of the second is small.*

The events d_{11}, d_{12}, and d_{13} represent the departure times of the first train from the stations Central, Kendall and MGH respectively, while d_{20}, d_{21}, and d_{22} represent *the departure times of the second train from Harvard, Central, and Kendall. The arrival times for the first train, a_{12} and a_{13} at Kendall and MGH, as well as the arrival times for the second train, a_{21}, a_{22} and a_{23} at Central, Kendall and MGH, are uncontrollable and dependent on varying conditions. There are five traversals which take an uncertain amount of time: T_0 the traversal from Harvard to Central, T_1 and T_2 traversals between Central and Kendall, and T_3 and T_4 traversals between Kendall and MGH. While we are not given the probability distributions for the duration of each traversal, we have access to historical data[36]. A common metric of the quality of a timetable is the headway, or the time between the departure of one train and the arrival of the next, and is represented by the upper-bound variables h_1, h_2, h_3. The total headway is represented by the sum $h_1 + h_2 + h_3$, the objective function to be minimized.*

This problem is an example of a chance-constrained probabilistic simple temporal problem (cc-pSTP) [16], although the traversal distributions are unknown. Given the available data, one solution is to directly fit a distribution, for example using KDE. However, there are two complications: 1) we do not know the shape of the underlying distribution; and 2) given that there is only a finite number of samples, the data may not be representative of the actual distribution.

Note that in practice, for chance-constrained scheduling with cc-pSTPs, we only use the distributions for uncertain durations to generate bounded representation of uncertainty. The problem is usually solved by constructing bounded sets of outcomes for uncertain durations, and appealing to results for simple temporal problems with uncertainty (STNUs).

The STNU formalism uses a set-bounded representation of uncertainty for uncontrollable durations [55], referred to in the STNU literature as *contingent constraints.* The standard solution method for cc-pSTPs construct contingent constraints which cover a required probability of outcomes. This construction generates STNUs candidates, which are checked for the existence of static and dynamic execution policies which satisfy any combination of outcomes for the contingent constraints [40, 39, 43].

For example, we may wish to plan using an upper bound on the traversal duration

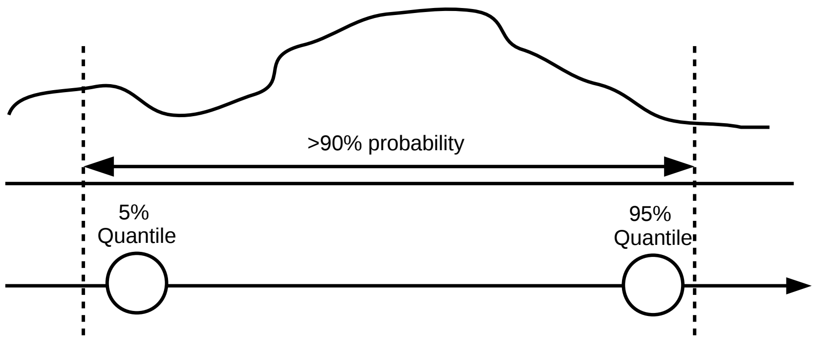

Figure 6-2: Probabilistic guarantee from bounds on quantiles.

from Harvard to Central such that 99% of future traversals will be shorter. Training using standard Gaussian KDE as implemented in Scipy [26] on 8000 observations of the traversal time from Harvard to Central, and testing against 40000 observations, we found that the returned upper-bounds are underestimates in 2220 out of 5000 trials. Over 40% of time, the upper-bounds from KDE would be incorrect, and we may return plans which do not provide the required probabilistic guarantee.

This motivates an alternative approach when we are not given the true distributions, but we are only given sampled data. Instead of constructing a distribution, we can directly construct from sampled data upper and lower bounds for uncertain durations which cover the required probability mass. We should also provide guarantees on the probability of the upper and lower bounds covering less probability mass than is required. An example is given in Figure 6-2, in which with 99% confidence the 5% quantile traversal is greater than 30 seconds and the 95% quantile is less than 180 seconds. Then, with 99% confidence the traversal takes between 30 seconds to 180 seconds with probability at least 90%. Our problem of finding a set-bounded representation of uncertainty is thus formally defined as follows.

Problem 4 (Chance-constrained bounded sets with confidence guarantees)**.** *Let* $<\Omega, \mathcal{F}, \mathbb{P}>$ *be a probability space and* $X : \Omega \to \mathbb{R}$ *a random variable, with cumulative distribution function* F.

Let b^l_ϵ *be the lower* ϵ*-quantile of* X*, where* $F(b^l_\epsilon) = \epsilon$*. Given* ϵ*, for* confidence

$1-\alpha$, $\alpha \in [0,1]$, *we must find lower bound* l_ϵ *such that*

$$P(l_\epsilon > b^l_\epsilon) \leq \alpha \tag{6.1}$$

Similarly, let b^u_ϵ *be the upper* ϵ*-quantile of* X *such that* $F(b^u_\epsilon) = 1-\epsilon$*, for* $\epsilon \in [0,1]$*. For* confidence $1-\alpha$, $\alpha \in [0,1]$, *we must find upper bound* u_ϵ *such that*

$$P(u_\epsilon < b^u_\epsilon) \leq \alpha \tag{6.2}$$

Recall that we need lower- and upper-bounds so that the probability mass outside the bounds is limited. Specifically for lower-bounds, we wish to find $l_\epsilon \in \mathbb{R}$ and associated ϵ such that $P(X \leq l_\epsilon) \leq \epsilon$. Similarly for upper-bounds, we wish to find $u_\epsilon \in \mathbb{R}$ and associated ϵ such that $P(X \geq u_\epsilon) \leq \epsilon$.

An equivalent statement would be that we wish to find $l_\epsilon \leq b^l_\epsilon$ and $u_\epsilon \geq b^u_\epsilon$ for different combinations of ϵ. While the tightest bounds would be the quantiles themselves, these are not known. The key idea is to instead choose l_ϵ and u_ϵ from the sample points.

In the presence of variations in sampling, we need a further probabilistic guarantee on the correctness of these bounds. This gives us a way of being robust to sampling: we can require the probability of underestimating or over estimating ϵ-bounds to be less than some small $\alpha \in [0,1]$.

We note that the solution to the above problem will find lower and upper bounds on the actual underlying distribution. However, the empirical success rate is evaluated on a finite number of future observations from the underlying distribution. The empirical proportion of successes is thus also subject to variations in sampling. For chance-constrained problems in which we are tasked with providing guarantees on rate of success for a finite number of future executions, we have the finite sample version of the problem as follows.

Problem 5 (Chance-constrained bounded sets for finite executions with confidence guarantees)**.** *Let* $< \Omega, \mathcal{F}, \mathbb{P} >$ *be a probability space and* $X : \Omega \to \mathbb{R}$ *a random* *variable. Suppose that we will also draw a set of M future samples of X, with ordered set $\mathcal{D}' = \{d'_1, d'_2, ..., d'_M\}$, where $d'_i \leq d'_j$ for $i < j$.*

Given $m \in \{1, ..., M\}$, for confidence $1-\alpha$, $\alpha \in [0,1]$, *we must find lower bound l'_m such that*

$$P(l'_m > d'_m) \leq \alpha \tag{6.3}$$

Similarly, for confidence $1-\alpha$, $\alpha \in [0,1]$, *we must find upper bound u'_m such that*

$$P(u'_m < d'_{M-m}) \leq \alpha \tag{6.4}$$

In the following section we describe how to construct lower and upper bounds with the confidence guarantees from past samples.

6.3 Data-driven chance-constrained uncertainty sets

We propose to construct the uncertainty bounds as defined in Problems 4 and 5 by picking out promising observed samples. Given an ordered set of samples from the underlying distribution, we can derive the likelihood of the nth sample being underestimates or overestimates for quantiles of the underlying distribution using results from order statistics. Similarly, we can derive the probability of the nth observed sample being less than or greater than the mth future sample using counting.

In subsequent discussion in this section, assume that we have $\mathcal{D} = \{d_1, d_2, ..., d_N\}$, a set of ordered samples of X.

6.3.1 Bounds on underlying distribution

We begin with results for uncertainty bounds for the underlying distribution, before considering the finite case.

Theorem 4. *For nth ordered sample d_n out of N samples sorted in ascending order:*

- $P(d_n \leq b^u_\epsilon) = \sum_{i=n}^{N} \binom{N}{i} [1-\epsilon]^i [\epsilon]^{N-i}$

- $P(d_n \geq b^l_\epsilon) = \sum_{i=N-n+1}^{N} \binom{N}{i} [1-\epsilon]^i [\epsilon]^{N-i}$

where $B(*; a, b)$ is the cumulative density for the $Beta(a, b)$ distribution.

Proof. From order statistics, for finite x and the n-th ordered sample d_n:

$$P(d_n \leq x) = \sum_{i=n}^{N} \binom{N}{i} [F(x)]^i [1 - F(x)]^{N-i}$$

Substituting $F(x) = \epsilon$, we have:

$$P(d_n \leq b^u_\epsilon) = \sum_{i=n}^{N} \binom{N}{i} [1-\epsilon]^i [\epsilon]^{N-i}$$

$$P(d_n \geq b^l_\epsilon) = \sum_{i=N-n+1}^{N} \binom{N}{i} [1-\epsilon]^i [\epsilon]^{N-i}$$

as required

□

The above results give us an easy way of constructing lower and upper bounds given ϵ and α. We may simply iterate through the list of ordered samples until we find the appropriate n. In the case of samples of continuous random variables, we may do so using bisection search. In the case of samples which are discrete, with multiple samples taking on the same value, we may use the following procedure.

Note that the above procedure can be used for both lower and upper bounds. For lower bounds, the samples should be sorted in ascending order. Conversely, for upper bounds, the samples should be sorted in descending order.

6.3.2 Bounds for finite executions

The previous result allows us to find bounds for the underlying distribution. However, as previously noted, the above result is inappropriate for finite future executions. Intuitively, the above result only accounts for variance in the historical data. It does

```
Data: $\epsilon, \alpha \in [0,1]$, sorted samples $\mathcal{D}$
Result: $b_\epsilon$
1  $d_{prev} \leftarrow d_1$, $n \leftarrow 1$;
2  while $n < N$ do
3      while $d_{prev} = d_n$ do
4          $n \leftarrow n + 1$
5      end
6      if $B(1-\epsilon, n, N-n+1) > \alpha$ then
7          break
8      end
9      $d_{prev} \leftarrow d_n$
10 end
11 $b_\epsilon \leftarrow d_{prev}$
```

Algorithm 7: Finding ϵ-bound of distribution with confidence $1 - \alpha$

not account for outliers which may be observed during a finite number of actual executions. For finite future executions, we have the result as follows.

Theorem 5. *For nth sample d_n out of N previous samples in ascending order, it is larger than mth out of M future observations with probability*

$$\binom{M}{m} \sum_{k=0}^{n-1} \binom{N}{k} \binom{N+M}{m+k}^{-1} \frac{m}{m+k}$$

Proof. Suppose the future observations were also sorted in ascending order, and given as $\mathcal{D}' = \{d'_1, d'_2, ..., d'_M\}$. The above probability is equivalent to $P(d_n > d'_m)$.

Note that $d_n > d'_m$ is equivalent to d'_m being greater than up to $n-1$ elements from the set $\mathcal{D}$. Summing up the probability over the disjoint scenarios, we have:

$$P(d_n > d'_m) = \sum_{k=0}^{n-1} P(d'_m \text{ larger than only } k \text{ samples})$$

Consider when d'_m is larger than only k samples. This is the same as arranging a combined set of $\mathcal{D} \bigcup \mathcal{D}'$, so that the $m+k$th element is in the set $\mathcal{D}'$. The first $m+k$ elements consist of m from the set of future observations, k from the past samples, and the $m+k$th element must be from the future observations. The remaining elements from the combined set can be arranged in any order.

There are $\binom{M}{m}$ ways of choosing m of the future observations, and $\binom{N}{k}$ of choosing

k of the previous samples. From the set of m future observations, there are m ways of choosing an element for the $m+k$th element of the combined set. There are $(m+k-1)!$ ways of arranging the first $m+k-1$ element, and $(M+N-m-k)!$ ways of arranging the remaining elements. This must be normalised over a total of $(N+M)!$ ways of arranging all the elements.

Combining the above, we have

$$\begin{aligned}
&P(d_n > d'_m)\\
=&\sum_{k=0}^{n-1}\binom{M}{m}\binom{N}{k}\frac{(m+k-1)!m(M+N-m-k)!}{(N+M)!}\\
=&\binom{M}{m}\sum_{k=0}^{n-1}\binom{N}{k}\binom{N+M}{m+k}^{-1}\frac{m}{m+k}
\end{aligned}$$

as required. □

A direct application of the above result allows us to construct a lower bound for finite execution, as required in Problem 5. Note that a similar argument based on counting can be made to derive the result for the finite execution upper bound.

Theorem 6. *For nth sample d_n out of N previous samples in descending order, it is smaller than mth out of M future observations with probability*

$$\binom{M}{m}\sum_{k=0}^{n-1}\binom{N}{k}\binom{N+M}{m+k}^{-1}\frac{m}{m+k}$$

The proof is the similar to that for the lower bound, substituting descending order for ascending order in the original proof.

Using the above results, we may construct lower and upper bounds to provide guarantees for a finite number of future executions. Similar to bound estimate for distributions, for continuous samples we would use bisection search. For discrete samples, we have the following procedure.

Again, the lower bound can be found with input samples sorted in ascending order, and the upper bound can be found with input samples sorted in descending order.

```
Data: $m, M, \alpha \in [0,1]$, sorted samples $\mathcal{D}$
Result: $b_\epsilon$
1  $d_{prev} \leftarrow d_1$, $n \leftarrow 0$;
2  $\alpha', a \leftarrow \binom{M}{m}\binom{N}{n}\binom{N+M}{m+n}^{-1}(\frac{m}{m+n})$, ;
3  while $n < N$ do
4      $n \leftarrow n + 1$;
5      $a \leftarrow a(\frac{N-n+1}{n})(\frac{m+n-1}{M+N-m-n+1})$;
6      $\alpha' \leftarrow \alpha' + a$;
7      if $d_{prev} \neq d_n$ then
8          if $\alpha' > \alpha$ then
9              break
10         end
11         $d_{prev} \leftarrow d_n$
12     end
13 end
14 $b_\epsilon \leftarrow d_{prev}$
```
Algorithm 8: Finding bound for m of M future observations with confidence $1-\alpha$

6.4 Experiments

The results in the previous section provide theoretical justification for our methods for constructing chance-constrained bounds with confidence. In this section, we present empirical validation of the correctness of the constructed bounds, with comparisons to bounds derived from KDE, using traversal data from the Red Line in the MBTA subway system.

In the first subsection, we examine the correctness of bounds for a number of different traversals, with varying ϵ bounds and $1-\alpha$ confidences.

In the second subsection, we return to the motivating example, and perform empirical comparisons for the correctness of schedules resulting from bounds on distribution, bounds for finite execution, and KDE. We also present empirical results on the conservatism in terms of differences in headway.

6.4.1 Correctness of estimates

In this section, we provide an empirical comparison of the correctness of bounds derived on the underlying distribution, bounds for finite execution, and bounds derived from KDE.

We conducted the experiment for five sets of traversal data. In each set, we wanted to find upper bounds on traversal time, varying ϵ and α parameters. For each set of traversal data, and with each combination of parameters, we randomly chose 8000 data points as training samples, derived bounds using the varying methods, and found the proportion of traversals which took longer than the derived bounds from a random set of 40000 other data points. For each data set and each combination of parameters, the process was repeated 5000 times.

The results are summarised in Table 6.1 and Table 6.2. Each entry has two numbers. The first number is the average proportion of traversals which was longer than the derived upper bounds. This gives an idea of how closely the bounds matched the desired probability covered, as specified by the ϵ parameter. The second number is the proportion of bounds which were incorrect - that is, for a bound u derived with a given ϵ, the proportion of traversals in the test set greater than u was greater than ϵ. This gives an idea of how frequently the chance constraint was incorrect due to variances in sampling, and should be compared with the corresponding α parameter.

The results confirm that, in a large number of instances, the bounds derived from KDE are incorrect. On average the KDE bounds are correct with respect to the probability covered as specified with the ϵ parameters. However, for a number of datasets and for a number of choices of ϵ, a significant proportion of the time the derived bounds are incorrect. For example, if we asked for an upper bound such that only 1% of future traversals were longer, KDE is wrong for traversals 70063-70065, 70069-70071, and 70071-70073 over 40% of the time.

This is contrasted with the performance of the data-driven methods with confidence guarantees. For bounds on underlying distribution and bounds for finite execution, we see that the averages are correct with respect to the probability covered as specified with the ϵ parameters.

The main difference between the two data-driven chance-constrained methods is in the correctness of the confidence guarantees. For example, for the 1% upper bound on the 70063-70065 and 70069-70071 traversals, the proportions of distribution bounds which were incorrect were significantly higher than that allowed by the α parameter.

In contrast, by explicitly accounting for variance in future observations as well as sampling, the proportion of bounds for finite execution which were incorrect were close to that allowed with the α parameter.

Of particular interest is the fact that, for some traversals, KDE was also significantly more conservative than the new data-driven methods. For example, for traversal 70069-70071, when we asked for u such that up to 10% of future traversals would take longer, the average proportion of test traversals which were longer than those found by KDE was 6.92%. This contrasts with the confidence based methods, which returned averages greater than 8.5%. In this case, there is a good chance KDE would overestimate the required traversal time, possibly leading to worse utility.

6.4.2 Effect on STNU scheduling

In this section, we return to the motivating scenario in Example 5. We would like to schedule departures time for two consecutive trains. In addition to the constraints previously outlined, we impose the chance constraint that the probability of the schedule being infeasible is less than 10%. As we are constructing the schedule based on real world data, we also want a 90% confidence in the correctness of the chance-constraint.

For simplicity, we assume the the random variables describing the traversal times are independent. Distributing the risk evenly, and noting that we have 5 instances of uncertain traversal durations, we find that we must find upper bounds for each traversal with $\epsilon = 1 - .9^{1/5}$.

In guaranteeing a 90% confidence in the correctness of the chance-constraint, we divide the risk evenly between the correctness of the ϵ upper bounds for the three uncertain traveral durations. Thus, we have $\alpha = 1 - .9^{1/3}$ for each traversal.

The traversal T0 from Harvard to Central corresponds to the dataset 70067-70069. The traversals T1 and T2 from Central to Kendall corresponds to the dataset 70069-70071. The traversals T3 and T4 from Kendall to MGH corresponds to the dataset 70071-70073.

For each traversal, we found upper bounds using the ϵ and α parameters above. This gave us set-bounded uncertainty for traversal durations, and thus allowed a

		KDE	Distribution			Finite execution		
			$\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.05$
70063-70065	$\epsilon = 0.01$	9.76e-03; 4.52e-01	6.72e-03; 1.60e-03	7.44e-03; 1.80e-02	8.21e-03; 5.08e-02	6.46e-03; 1.00e-03	7.19e-03; 9.00e-03	7.96e-03; 4.52e-02
	$\epsilon = 0.05$	3.35e-02; 0.00e+00	3.95e-02; 0.00e+00	4.15e-02; 0.00e+00	4.27e-02; 3.40e-03	3.87e-02; 0.00e+00	4.09e-02; 0.00e+00	4.25e-02; 2.00e-03
	$\epsilon = 0.1$	5.02e-02; 0.00e+00	7.59e-02; 0.00e+00	8.38e-02; 0.00e+00	8.85e-02; 0.00e+00	7.35e-02; 0.00e+00	8.15e-02; 0.00e+00	8.77e-02; 0.00e+00
70065-70067	$\epsilon = 0.01$	9.72e-03; 3.98e-01	6.82e-03; 1.40e-03	7.54e-03; 1.54e-02	8.31e-03; 6.78e-02	6.57e-03; 8.00e-04	7.30e-03; 6.20e-03	8.07e-03; 4.06e-02
	$\epsilon = 0.05$	4.79e-02; 1.50e-01	4.20e-02; 4.00e-04	4.37e-02; 9.00e-03	4.53e-02; 5.58e-02	4.14e-02; 0.00e+00	4.32e-02; 4.80e-03	4.49e-02; 3.86e-02
	$\epsilon = 0.1$	9.61e-02; 7.54e-02	8.79e-02; 2.20e-03	9.04e-02; 1.72e-02	9.25e-02; 2.34e-02	8.70e-02; 4.00e-04	8.96e-02; 8.00e-03	9.21e-02; 2.34e-02
70067-70069	$\epsilon = 0.01$	9.74e-03; 4.42e-01	6.81e-03; 1.80e-03	7.53e-03; 1.48e-02	8.31e-03; 5.14e-02	6.56e-03; 1.60e-03	7.29e-03; 7.40e-03	8.06e-03; 4.02e-02
	$\epsilon = 0.05$	4.16e-02; 0.00e+00	4.11e-02; 0.00e+00	4.28e-02; 0.00e+00	4.43e-02; 7.00e-03	4.05e-02; 0.00e+00	4.23e-02; 0.00e+00	4.40e-02; 4.60e-03
	$\epsilon = 0.1$	6.97e-02; 0.00e+00	8.25e-02; 0.00e+00	8.31e-02; 0.00e+00	8.53e-02; 0.00e+00	8.24e-02; 0.00e+00	8.28e-02; 0.00e+00	8.47e-02; 0.00e+00

Table 6.1: Table of empirical results for different traversal datasets. For each entry, the first number is the average number empirical proportion of traversals larger than the bound which should be smaller than the specified ϵ, and the second number is the empirical proportion of incorrect bounds which should be smaller than the specified α. The datasets cover the segments: 70063-70065 from Davis to Porter; 70065-70067 from Porter to Harvard; and 70067-70069 from Harvard to Central.

		KDE	Distribution			Finite execution		
			$\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.05$
70069-70071	$\epsilon = 0.01$	9.92e-03; 4.52e-01	6.86e-03; 1.40e-03	7.58e-03; 1.58e-02	8.35e-03; 7.72e-02	6.61e-03; 1.00e-03	7.33e-03; 9.00e-03	8.09e-03; 4.88e-02
	$\epsilon = 0.05$	4.53e-02; 1.14e-02	4.23e-02; 2.00e-03	4.40e-02; 7.00e-03	4.56e-02; 1.36e-02	4.17e-02; 1.00e-03	4.35e-02; 7.00e-03	4.52e-02; 1.14e-02
	$\epsilon = 0.1$	6.92e-02; 0.00e+00	8.71e-02; 0.00e+00	8.96e-02; 0.00e+00	9.16e-02; 4.00e-04	8.63e-02; 0.00e+00	8.89e-02; 0.00e+00	9.11e-02; 0.00e+00
70071-70073	$\epsilon = 0.01$	9.82e-03; 4.42e-01	6.81e-03; 2.20e-03	7.56e-03; 1.56e-02	8.31e-03; 7.48e-02	6.56e-03; 8.00e-04	7.31e-03; 9.00e-03	8.06e-03; 4.84e-02
	$\epsilon = 0.05$	4.93e-02; 4.15e-01	4.24e-02; 1.20e-03	4.42e-02; 9.80e-03	4.58e-02; 6.50e-02	4.18e-02; 6.00e-04	4.37e-02; 5.40e-03	4.54e-02; 4.78e-02
	$\epsilon = 0.1$	9.89e-02; 3.38e-01	8.91e-02; 2.20e-03	9.17e-02; 7.60e-03	9.39e-02; 5.22e-02	8.82e-02; 8.00e-04	9.09e-02; 3.40e-03	9.34e-02; 3.60e-02

Table 6.2: Table of empirical results for traversal datasets. MBTA Redline segments: 70069-70071 from Central to Kendall; and 70071-70073 from Kendall to MGH.

	Average Headway (seconds)	Average # of Failures	# of High Failure Schedules
KDE	611.465	980.530	343
Distribution	641.331	867.666	5
Finite execution	654.171	821.804	0

Table 6.3: Performance of Example 5 schedules optimised with upper bounds on traversal derived with different methods. High failure schedules are those which empirically fail more than 10% of the time, and thus do not meet the required risk bound.

STNU description of the problem. Finding a static schedule is an instance of a strong controllability problem for STNUs. Taking inspiration from [16], we encoded the linear constraints ensuring strong controllability, and optimised the linear function for headway using the standard linear solver in [26]. Each resulting schedule was then tested against 10000 combinations of outcomes for the traversals. This procedure was repeated 1000 times, and we gathered statistic on the resulting headway and the empirical rate of schedule failure. This information is summarised in Table 6.3.

In all cases, the average number of failure for each schedule was lower than that allowed. However, the number of high failure schedules, schedule which had higher than 10% probability of failure, was 34.3% for schedules optimised using bounds from KDE. This contrasts with the negligible number of high failure schedules optimised with the proposed bounds. We may also note that, in this case, the cost of having probabilistic guarantees on the chance-constraint is very small in terms of the differences in headway - under a minute when compared to KDE.

6.5 Summary

Chance-constrained scheduling relies on accurate descriptions of uncertainty in providing probabilistic guarantees. When the underlying distributions are not known and only data sets of past samples are available, simply fitting distributions to the data sets do not provide accurate descriptions. As scheduling under uncertainty is typically a model-based approach, the correctness of the probabilistic guarantees depend on the correctness of the model.

In this chapter, we focused on constructing models from data for use in scheduling under uncertainty. We have motivated an alternative representation of uncertainty, using bounded sets which cover a required probability mass, allowing us to leverage STNU algorithms while guaranteeing probability of success. We have also shown how to construct such sets, either as bounds for the underlying distribution, or as bounds for a finite number of future outcomes, with quantitative confidence on the correctness of the bounds in each case.

The correctness of these theoretical results were empirically validated with extensive testing. We have shown the inadequacies of distribution fitting with standard KDEs, and the correctness of the two proposed methods. We have further demonstrated, in a motivating scenario, that the use of the proposed bounded sets with correctness guarantees do not lead to schedules with drastically lower utility.

There are several avenues for future work. The first is to extend this work into a risk-allocation scheme, in which we allow different risks to be prescribed to different random variables within the same problem. The approaches proposed in this chapter also only provide bounds on marginal distributions. Providing analogous bounds for joint distributions is thus an open problem. Lastly, while this chapter has demonstrated the correctness of the uncertainty sets in a strong controllability setting, the resulting uncertainty sets are equally applicable for STNU dynamic controllability. Further work will concentrate on leveraging recent advances in dynamic controllability for correct chance-constrained dynamic execution.

Chapter 7

Conclusion

As robotics and artificial intelligence advances, we must address difficulties in learning to trust autonomous decision making. One approach is to provide plans which are robust to uncertainty, with correct quantifiable estimates of the risk involved.

The chance-constrained mixed logical-linear program is a problem formulation which allows us to describe many real world problems, including cyber security and autonomous exploration. However, the problem is difficult to solve, due to the combinatorics introduced by the logical constraints, and because the continuous portion of the problem is nonlinear.

To address the difficulties in solving CC-MLLPs, I have proposed a decomposition framework in Chapter 3, combining a discrete search for assigning logical variables, and a dedicated solver for the chance-constrained linear programs which feature as the continuous subproblems.

In Chapter 4, I provided a method of solving CC-LPs which quickly determines feasibility, and can be combined with off-the-shelf solvers to quickly return optimal solutions. The method relies on successive linear approximations of the chance constraint using cutting planes.

The linear approximations are further exploited in Chapter 5, in which I expand on the conflict extraction necessary to the decomposition algorithm. By identifying linear constraints preventing feasible solutions or higher quality solutions, we are able to trace back and find the logical conditions which lead to the conflicting linear constraints. These are explicitly avoided in the subsequent search, to avoid unpromising portions of the search space.

In Chapter 6, I consider the case when the distributions for uncertainty are unknown, and we must make chance constrained decisions using only data. Focusing on the case of scheduling, I derive uncertainty bounds which cover the required probability mass with quantifiable amounts of confidence. In addition to estimating the appropriate bounds for the true distribution, I consider the real world application in which each uncertain variable is observed a finite number of times. The resulting bounds meet empirical service quality guarantees over a finite number of executions.

7.1 Future Work

There are three main potential avenues of future work. The most immediate is a closer unification of the data-driven uncertainty representation with the CC-MLLP framework. Separately, while the treatment of chance constraints in this thesis has concentrated on a risk allocation approach, methods should be developed for CC-MLLPs with exact probabilities of success given continuous random variables. Further, while the present work concentrates on continuous random variables only, incorporating methods from stochastic constraint programming will allow the introduction of stochasticity in the logical variables.

In this thesis, I have concentrated separately on solution methods for CC-MLLPs given known distributions, and extracting uncertainty representation from data. However, these two thrusts could be more closely interwoven. The uncertainty representation derived can be thought of as random variables with discrete domains. If the representations are directly deployed in CC-MLLPs, the cutting planes techniques developed for CC-LPs will not be applicable. This is because the cutting planes require the evaluation of the probability density functions. This is an important and technically interesting challenge.

This thesis has also concentrated on solving CC-MLLPs with the risk allocation approximation. However, in the chance-constrained optimisation community, recent efforts have considered exact evaluations of the probability of success using the method of moments [25, 59]. Just as this thesis builds off prior work in risk allocation for CC-LPs, analogous approaches using exact risk for CC-LPs may also be deployed. This has two advantages. The first is that the risk allocation approach is conservative, and better solutions may be found with the method of moments approach. The second is that the method of moments allows chance constraints to be specified over nonlinear constraints as well. A method of moments approach may then allow a generalisation to chance-constrained mixed logical nonlinear programs. The key technical challenge would be to identify the corresponding conflict extraction techniques, given the different subsolver.

Finally, there has been significant efforts in the constraint programming community dedicated to stochastic constraint programming [58, 53]. As previously noted, this has resulted in insights into handling discrete stochasticity. This would be useful to a CC-MLLP framework in which the agent must deal with unknown operation scenarios. The expertise in dealing with discrete random variables may also help in unifying the data-driven uncertainty representation with the current conflict extraction framework. Lasting, while the current thesis has dealt with CC-MLLPs in an open-loop manner, the stochastic constraint programming work have dealt with policies responding to observations of random variables. Incorporating these insights would allow use to develop a reactive approach to CC-MLLPs.

Bibliography

[1] Sylvain Arlot, Alain Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[2] Jacques F Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962.

[3] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

[4] Dimitris Bertsimas, Vishal Gupta, and Nathan Kallus. Data-driven robust optimization. *arXiv preprint arXiv:1401.0212*, 2013.

[5] Daniel Bienstock, Michael Chertkov, and Sean Harnett. Chance-constrained optimal power flow: Risk-aware network control under uncertainty. *Siam Review*, 56(3):461–495, 2014.

[6] Lars Blackmore, Hui Li, and Brian Williams. A probabilistic approach to optimal robust path planning with obstacles. In *American Control Conference, 2006*, pages 7–pp. IEEE, 2006.

[7] Lars Blackmore and Masahiro Ono. Convex chance constrained predictive control without sampling. In *AIAA Guidance, Navigation, and Control Conference*, page 5876, 2009.

[8] Kyle EC Booth, Goldie Nejat, and J Christopher Beck. A constraint programming approach to multi-robot task allocation and scheduling in retirement homes. In *International conference on principles and practice of constraint programming*, pages 539–555. Springer, 2016.

[9] Giuseppe Carlo Calafiore and Laurent El Ghaoui. On distributionally robust chance-constrained linear programs. *Journal of Optimization Theory and Applications*, 130(1):1–22, 2006.

[10] Elsa Carvalho, Jorge Cruz, and Pedro Barahona. Probabilistic continuous constraint satisfaction problems. In *2008 20th IEEE International Conference on Tools with Artificial Intelligence*, volume 2, pages 155–162. IEEE, 2008.

[11] Abraham Charnes and William W Cooper. Chance-constrained programming. *Management science*, 6(1):73–79, 1959.

[12] Abraham Charnes and William W Cooper. Deterministic equivalents for optimizing and satisficing under chance constraints. *Operations research*, 11(1):18–39, 1963.

[13] Wenqing Chen, Melvyn Sim, Jie Sun, and Chung-Piaw Teo. From cvar to uncertainty set: Implications in joint chance-constrained optimization. *Operations research*, 58(2):470–485, 2010.

[14] IBM ILOG CPLEX. V12. 1: UserâĂŹs manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

[15] Jack Elzinga and Thomas G Moore. A central cutting plane algorithm for the convex programming problem. *Mathematical Programming*, 8(1):134–145, 1975.

[16] Cheng Fang, Peng Yu, and Brian C Williams. Chance-constrained probabilistic simple temporal problems. 2014.

[17] Pablo Fleurquin, José J Ramasco, and Victor M Eguiluz. Systemic delay propagation in the us airport network. *Scientific reports*, 3, 2013.

[18] Arthur M Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260, 1972.

[19] Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

[20] JM Grosso, Carlos Ocampo-Martínez, Vicenç Puig, and B Joseph. Chance-constrained model predictive control for drinking water networks. *Journal of process control*, 24(5):504–516, 2014.

[21] Gurobi. Inc.,âĂIJgurobi optimizer reference manual,âĂİ 2015. *URL: http://www. gurobi. com*, 2014.

[22] John N Hooker and Maria A Osorio. Mixed logical-linear programming. *Discrete Applied Mathematics*, 96:395–442, 1999.

[23] John N Hooker and Greger Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.

[24] Raj Jagannathan. Chance-constrained programming with joint constraints. *Operations Research*, 22(2):358–372, 1974.

[25] AM Jasour, NS Aybat, and CM Lagoa. Semidefinite programming for chance constrained optimization over semialgebraic sets. *SIAM Journal on Optimization*, 25(3):1411–1440, 2015.

[26] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2017-03-06].

[27] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[28] Erez Karpas, Steven James Levine, Peng Yu, and Brian Charles Williams. Robust execution of plans for human-robot teams. In *ICAPS*, pages 342–346, 2015.

[29] George Katsirelos and Fahiem Bacchus. Generalized nogoods in csps. In *AAAI*, volume 5, pages 390–396, 2005.

[30] James E Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[31] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[32] Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[33] Hui Li and Brian Williams. Generalized conflict learning for hybrid discrete/linear optimizationâŃĘ. In *Principles and Practice of Constraint Programming-CP 2005: 11th International Conference, CP 2005, Sitges Spain, October 1-5, 2005*, volume 3709, page 415. Springer, 2005.

[34] Xiao Liu, Simge Küçükyavuz, and James Luedtke. Decomposition algorithms for two-stage chance-constrained programs. *Mathematical Programming*, 157(1):219–243, 2016.

[35] James Luedtke. A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support. *Mathematical Programming*, 146(1-2):219–244, 2014.

[36] MBTA. Online trip planning tools. `http://old.mbta.com/rider_tools/developers/default.asp?id=21895`, 2017. Accessed: 2017-11-20.

[37] Catharine LR McGhan, Richard M Murray, Romain Serra, Michel D Ingham, Masahiro Ono, Tara Estlin, and Brian C Williams. A risk-aware architecture for resilient spacecraft operations. In *Aerospace Conference, 2015 IEEE*, pages 1–15. IEEE, 2015.

[38] Bruce L Miller and Harvey M Wagner. Chance constrained programming with joint constraints. *Operations Research*, 13(6):930–945, 1965.

[39] Paul Morris. Dynamic controllability and dispatchability relationships. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 464–479. Springer, 2014.

[40] Paul H Morris and Nicola Muscettola. Temporal dynamic controllability revisited. In *AAAI*, pages 1193–1198, 2005.

[41] Arkadi Nemirovski and Alexander Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2006.

[42] Yu Nesterov. Complexity estimates of some cutting plane methods based on the analytic barrier. *Mathematical Programming*, 69(1-3):149–176, 1995.

[43] Mikael Nilsson, Jonas Kvarnström, and Patrick Doherty. Incremental dynamic controllability in cubic worst-case time. In *Temporal Representation and Reasoning (TIME), 2014 21st International Symposium on*, pages 17–26. IEEE, 2014.

[44] Masahiro Ono, Lars Blackmore, and Brian C Williams. Chance constrained finite horizon optimal control with nonconvex constraints. In *American Control Conference (ACC), 2010*, pages 1145–1152. IEEE, 2010.

[45] Masahiro Ono and Brian C Williams. An efficient motion planning algorithm for stochastic dynamic systems with constraints on probability of failure. In *Proceedings of the 23rd national conference on Artificial intelligence-Volume 3*, pages 1376–1382. AAAI Press, 2008.

[46] Masahiro Ono, Brian C Williams, and Lars Blackmore. Probabilistic planning for continuous dynamic systems under bounded risk. *Journal of Artificial Intelligence Research*, 46:511–577, 2013.

[47] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.

[48] Francesca Rossi, Peter Van Beek, and Toby Walsh. Constraint programming. *Foundations of Artificial Intelligence*, 3:181–211, 2008.

[49] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.

[50] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming*, pages 417–431. Springer, 1998.

[51] Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[52] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1.

[53] S Armagan Tarim, Suresh Manandhar, and Toby Walsh. Stochastic constraint programming: A scenario-based approach. *Constraints*, 11(1):53–80, 2006.

[54] Eric Timmons and Brian C Williams. Enumerating preferred solutions to conditional simple temporal networks quickly using bounding conflicts. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[55] Thierry Vidal. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence*, 11(1):23–45, 1999.

[56] Michael P Vitus and Claire J Tomlin. On feedback design and risk allocation in chance constrained control. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 734–739. IEEE, 2011.

[57] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[58] Toby Walsh. Stochastic constraint programming. In *ECAI*, volume 2, pages 111–115, 2002.

[59] Allen Wang, Ashkan Jasour, and Brian C Williams. Non-gaussian chance-constrained trajectory planning for autonomous vehicles under agent uncertainty. *IEEE Robotics and Automation Letters*, 5(4):6041–6048, 2020.

[60] Andrew J Wang and Brian C Williams. Chance-constrained scheduling via conflict-directed risk allocation. 2015.

[61] Qianfan Wang, Yongpei Guan, and Jianhui Wang. A chance-constrained two-stage stochastic program for unit commitment with uncertain wind power output. *Power Systems, IEEE Transactions on*, 27(1):206–215, 2012.

[62] Brian C Williams and Robert J Ragno. Conflict-directed a* and its role in model-based embedded systems. *Discrete Applied Mathematics*, 155(12):1562–1595, 2007.

[63] Peng Yu, Cheng Fang, and Brian Williams. Resolving uncontrollable conditional temporal problems using continuous relaxations. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*, 2014.

[64] Peng Yu, Cheng Fang, and Brian Williams. Resolving over-constrained probabilistic temporal problems through chance constraint relaxation. 2015.

[65] Steve Zymler, Daniel Kuhn, and Berç Rustem. Distributionally robust joint chance constraints with second-order moment information. *Mathematical Programming*, 137(1-2):167–198, 2013.