Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2018-011 February 1, 2013

Risk Allocation for Temporal Risk Assessment

Andrew J. Wang

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Risk Allocation for Temporal Risk Assessment

by

Andrew J. Wang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2013

Author .....
Department of Electrical Engineering and Computer Science
January 31, 2013

Certified by .....
Brian C. Williams
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Prof. Dennis M. Freeman
Chairman, Masters of Engineering Thesis Committee

Risk Allocation for Temporal Risk Assessment

by

Andrew J. Wang

Submitted to the Department of Electrical Engineering and Computer Science
on January 31, 2013, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Temporal uncertainty arises when performing any activity in the natural world. When activities are composed into temporal plans, then, there is a risk of not meeting the plan requirements. Currently, we do not have quantitatively precise methods for assessing temporal risk of a plan. Existing methods that deal with temporal uncertainty either forgo probabilistic models or try to optimize a single objective, rather than satisfy multiple objectives. This thesis offers a method for evaluating whether a schedule exists that meets a set of temporal constraints, with acceptable risk of failure.

Our key insight is to assume a form of risk allocation to each source of temporal uncertainty in our plan, such that we may reformulate the probabilistic plan into an STNU parameterized on the risk allocation. We show that the problem becomes a deterministic one of finding a risk allocation which implies a schedulable STNU within acceptable risk. By leveraging the principles behind STNU analysis, we derive conditions which encode this problem as a convex feasibility program over risk allocations. Furthermore, these conditions may be learned incrementally as temporal conflicts. Thus, to boost computational efficiency, we employ a generate-and-test approach to determine whether a schedule may be found.

Thesis Supervisor: Brian C. Williams
Title: Professor of Aeronautics and Astronautics

Acknowledgments

I am grateful to those who cared for my thesis in the making. Their support meant much. They include:

- First and foremost, my advisor, Prof. Brian Williams. His scholarship and his dedication to his students show me concepts that I would rarely learn on my own.

- My fellow lab members in the Model-based Embedded & Robotics Systems group. Those who have collaborated most directly are Larry Bush, Cheng Fang, Peng Yu, and Pedro Santana. Others who I have or still interact with daily include David Wang, Eric Timmons, Steve Levine, Andreas Hoffman, Masahiro Ono, Wesley Graybill, Robert Effinger, Shannon Dong, James Paterson, Enrique Fernandez, and Ameya Shroff.

- Jerry Jaeger and Michael Boulet at MIT Lincoln Laboratory. They generously funded this research, and they have taken great interest in it as well.

- Anne Hunter, Vera Sayzew, and Linda Sullivan in the Course 6 Undergraduate Office, who work tirelessly on behalf of students.

- And my family, which I am very fortunate to be part of.

Contents

1 Introduction **9**

1.1 Motivation 9

1.2 Core Insight 11

1.3 Solution Method 12

1.4 Experimental Results 13

1.5 Thesis Outline 14

2 Problem Statement **15**

2.1 Motivating Scenario 15

2.2 pSTN Model 17

2.2.1 Primitives 18

2.2.2 Model definition 21

2.3 pSTP Scheduling Problem 24

2.3.1 Execution strategy 25

2.3.2 Problem definition 26

3 Reformulation into Convex Risk Allocation **31**

3.1 Approach Overview 32

3.1.1 Challenges and approach to solving the pSTP 32

3.1.2 Online execution strategy 36

3.1.3 Offline scheduling run-through example 38

3.2 Reformulation Procedure 42

3.2.1 Risk allocation into STNU form 44

3.2.2 Reduction to convex feasibility 45
3.2.3 Incremental temporal constraint discovery 48

4 Experimental Validation **53**
4.1 Hypothesis 53
4.2 Test-case Generation 53
4.3 Experiment Design 54
4.4 Results 55

5 Conclusion **59**
5.1 Contributions 59
5.2 Future Work 60
5.2.1 Algorithm and Analysis Extensions 60
5.2.2 Integration with Other Capabilities 61

Chapter 1

Introduction

1.1 Motivation

Temporal uncertainty is a natural aspect of performing real-world activities. For instance, the time it takes to drive down a block is subject to road traffic, pedestrians crossing, whether the road curves or goes straight, and plenty of other factors. These factors may not amount to much for one short block, but if one needs to drive across town for an appointment, then the accumulated temporal effects begin to matter.

Such temporal effects pose serious risks in time-critical missions where lives or high-value assets are at stake. Disaster relief responders must traverse dangerous and unknown terrain to reach survivors, who may only have hours to live. Autonomous Mars exploration rovers must complete their science objectives with enough daylight left to recharge their batteries. The assembly line in a manufacturing warehouse must tolerate the varying rates at which workers assemble parts in order to avoid costly bottlenecks.

All these scenarios involve constraints on the time difference between various activities. For example, we are very familiar with deadline constraints. The seriousness of such deadlines is apparent in the disaster relief and Mars rover scenarios. Sometimes, though, we may not want to work as fast as possible, but rather ensure a minimum passage of time. Such a situation might arise in the manufacturing scenario, where one would prefer to produce a part slower in order to match the rate of the assembly

line, instead of accumulating a huge stockpile of said parts. In general, different sets of activities need to finish within certain time bounds.

However, temporal uncertainty is involved in executing any activity. Whether driving through flooded roads, drilling a soil sample, or building parts by hand, there are always uncontrollable and natural effects which will tend to speed one up or slow one down. As these effects accumulate over activities in sequence, they may make it easier or harder to satisfy the temporal constraints, but it will always be impossible to guarantee satisfaction of all temporal constraints.

Therefore, we must deploy missions on the basis of weighing the risks of multiple objectives against their relative priorities. If our disaster relief mission is to reach two survivors in two different locations, it would be more urgent to reach the one in more critical condition first, and that would inform how we balance our temporal risks when scheduling the mission. Currently, the complex implications of temporal uncertainty limit deployed missions to those whose temporal risks can be analyzed by hand. However, computational methods could enable us to confidently assess and schedule more ambitious missions, hence raising performance in many applications.

The state-of-the-art in scheduling under temporal uncertainty is insufficient in several regards. Vidal [16] introduced the *simple temporal network with uncertainty (STNU)*, which assumes that uncontrollable temporal disturbances are limited to a fixed interval. This representation provides ease of analysis, and there exist several polynomial-time scheduling algorithms for STNUs. Unfortunately, it is sensitive to modeling errors; a perturbation to one of the intervals may require recomputing the schedule. More recently, Tsamardinos [14, 15] and Li [10, 9] have pursued probabilistic temporal constraint networks and developed scheduling strategies for those. However, they attempt to optimize the risk for the entire mission, rather than addressing different goals according to priority. Other techniques either optimize a single utility rather than satisfying multiple goals [1], are based on most likely inaccurate probabilistic models [13, 8], or perform expensive Monte Carlo simulations [5]. None of these approaches are sufficient for the needs laid out above.

This thesis addresses the problem of chance-constrained scheduling, given proba-

bilistic temporal uncertainty. We model the mission as a *probabilistic simple temporal network (pSTN)* and apply chance constraints over subsets of its temporal constraints. We then provide an offline scheduling algorithm that either finds a schedule that meets the chance constraints, or claims no such schedule exists.

1.2 Core Insight

The key intuition behind our scheduling algorithm is to allocate risk to each source of temporal uncertainty by assuming its outcome will lie in a finite range. This will occur with some probability for each source, which is the risk we allocate to it. By representing the bounds of these ranges as risk variables, we reformulate the pSTN model into an STNU parameterized on the risk variables. This is advantageous because efficient algorithms exist for scheduling and dispatching STNUs [11]. Hence, if we find a risk allocation such that it meets the chance constraints and the resulting STNU can be scheduled, then the STNU's schedule is a valid solution to our chance-constrained pSTN.

By assuming this risk allocation, we are reformulating the original stochastic problem into a deterministic one. It is deterministic because we are no longer concerned with individual outcomes of temporal uncertainty and their probability densities, but rather entire intervals and their probability masses. The original problem is defined over schedules and individual probabilistic outcomes, whereas the reformulated one is only over risk variables.

Our scheduling problem is an instance of stochastic constraint satisfaction, or a stochastic CSP, over continuous variables. This strategy of reformulating into a deterministic problem is a common thread in solving this class of problems. For instance, Blackmore [2] applied reformulation to obstacle path-planning with stochastic vehicle dynamics. By approximating the expected vehicle position with a collection of particles, the problem became deterministic in terms of the particle set. One downfall of this approach is that in order to achieve good accuracy, a large number of particles is needed, and resulting deterministic CSP becomes intractable again. Ono [12] followed

up on this path-planning problem, using risk allocation to reformulate the problem into the deterministic one of calculating safety margins around obstacles. This is much like our approach, except in the context of physical state rather than time.

Tsamardinos [15] and Li [10] also reformulated their probabilistic temporal problems into deterministic optimization. Tsamardinos offers an approach that is quite similar to ours in spirit. Although he does not use the term "risk allocation," he does frame the problem in terms of finding ranges for each source of uncertainty. Nevertheless, where his reformulation ends, our algorithm reformulates one step further for efficiency gains. Also, he is concerned with dynamic scheduling and optimizing the entire mission, whereas our algorithm focuses on static scheduling and satisfying individual chance constraints. On the other hand, Li's reformulation is based on summary statistics of each source's temporal distribution, and hence does not take full advantage of the information in each distribution. Li applies "risk allocation" only as a heuristic after optimization for curbing conservatism, and hence it is not an integral part of his reformulation technique.

1.3 Solution Method

The goal of our offline scheduling algorithm is twofold. First, we want to reformulate the original chance-constrained pSTN into a set of deterministic constraints over the risk variables. Then, we need to solve for a feasible risk allocation. The reformulation should address both the temporal and the probabilistic aspects of the original problem; our approach considers each separately. This produces two sets of constraints, which must be solved together in the last step.

The output of the temporal reformulation in the first step is to construct an *simple temporal network (STN)* out of the original pSTN, parameterized on the risk variables. During this step is when we apply the key insight to construct the parameterized STNU from the pSTN and the risk variables. We then apply a reformulation algorithm for the STNU to turn it into an equivalent STN. Algebraically, an STN is simply a set of inequality constraints.

The probabilistic reformulation rewrites each chance constraint, which is defined over temporal constraints, in terms of the risk variables. Note that only certain sources of temporal uncertainty could affect the satisfaction of chance constraints. For example, suppose one goes shopping at two supermarkets, first A then B. Finishing shopping at supermarket A by noon does not depend on how long it takes to drive home from B, but it does depend on how long it took to drive from home to A and how long one spends at A. Detecting which sources are relevant is achieved by applying a cycle detection algorithm on the temporal network. Then, each chance constraint may be rewritten in terms of the corresponding risk variables as an inequality constraint as well.

Finally, we solve these inequality constraints via an off-the-shelf solver. First, we note that the STN constraints are linear, and the chance constraints are proven to be convex. Thus, we can use a convex solver to take advantage of well known barrier and interior-point methods [3, 4]. Second, the STN constraints are actually in terms of scheduled event times as well as the risk variables. To avoid solving over the additional event time variables, we learn constraints equivalent to the STN over iterations of generating and testing candidate risk variable assignments. The generator solves the convex satisfaction problem over the inequality constraints, while the test then checks consistency of the risk allocation against the temporal constraints. If inconsistent, a temporal constraint on the risk variables is learned, and this is fed to the convex solver in order to prune the space of candidate risk allocations. This continues until the risk allocation is no longer temporally inconsistent, upon which a schedule may be derived, or until the convex solver receives an infeasible set of constraints.

1.4 Experimental Results

We aim to empirically validate the range of problems for which our algorithm tractably solves. In particular, we run the algorithm with and without the last step of generate-and-test with constraint learning, in order to evaluate the efficiency gains of that strategy. We generate our test corpus of pSTNs modeling a single agent performing a

sequence of activities, with temporal constraints between events along the way. The problem size grows as the number of activities and temporal constraints.

Runtime improvements become apparent for problem sizes involving hundreds of activities. Few improvements, if any, are seen for trivially small problems. This is due to several iterations of solving and discovering new constraints, whereas the full STN approach solves it all at once. However, the total number of constraints considered for generate-and-test grows slower than the problem does. Hence, this approach is tractable for a much larger range of problems.

1.5 Thesis Outline

The body of this thesis is comprised of two technical chapters plus a third on experimental validation. First, Chapter 2 defines the chance-constrained pSTN scheduling problem. Chapter 3 then presents our novel temporal risk allocation algorithm, which either finds a schedule that satisfies the chance constraints over a given pSTN, or determines that no such schedule exists. The scalability of the algorithm's generate-and-test approach in the last step is empirically investigated in Chapter 4. Finally, we summarize our contributions in Chapter 5. We discuss future extensions, analysis, and potential integration of our work with other planning and scheduling capabilities.

Chapter 2

Problem Statement

2.1 Motivating Scenario

As a guiding example in explaining our model and algorithm, we consider a simplified version of the disaster relief scenario introduced in Chapter 1. Although simple, this scenario expresses the three key features of our scheduling problem: 1) the desired temporal coordination between activities, 2) the temporal flexibility and stochastic uncertainty in executing each activity, and 3) the desire to robustly achieve (1) in the face of execution uncertainty from (2). In this section, we describe a scenario that exemplifies these features.

A major hurricane has devastated a coastal city, and in response, a disaster relief organization sets up a supply depot on the city's outskirts. This depot will supply food, water, and basic utilities until sufficient infrastructure is repaired. As a field worker for this organization, your daily job is to lead supply convoys into surrounding neighborhoods to deliver these supplies. A typical deployment consists of driving into a particular neighborhood, dropping off your supplies, and returning to base for your next convoy assignment.

At the high-level, this mission plan consists of three major activities in sequence: driving there, dropping off supplies, and driving back. Each segment requires time to complete, but you must finish them within certain timing constraints. Your first responsibility is to the people: to arrive there and give them what they need within

one hour. Since you will not know their needs exactly until you arrive and assess their condition, you make sure to carry a slight surplus. Once there, you radio back to the depot staff what you have decided to unload, so the staff can restock in preparation for your return. They require at least 80 minutes to prepare your next batch of supplies. However, the depot is on a tight schedule, so once the depot staff start processing your request, they need you back within two hours to pick up your supplies. Nevertheless, satisfying this depot logistics constraint is not as crucial as getting supplies to the people on time.

From your experience running such missions, you have an estimate of how long each activity will take. These estimates are based on how much control you have in guiding each activity towards completion. For example, in driving to the neighborhood, you may be comfortable driving at speeds which would get you there as fast as 20 minutes but no longer than 40 minutes. However, you do not know the road conditions ahead of time. Due to widespread flooding, the roads may be more flooded or more dry than you expect, which would tend to slow you down or speed you up. *Having a probabilistic belief about Nature's temporal effects is a key aspect of this scenario.*

Similar estimates could be made for the other two activities. For unloading supplies, your experience says you can do it as quickly as 15 minutes, but you don't mind spending an extra 20 minutes to further distribute the supplies you have unloaded. Also, depending on how organized the people are, they may help you shave five minutes off or require an additional ten minutes of your assistance, approximately. Then, by the time you start leaving for base, you expect dusk to have fallen, so you will naturally drive slower under less visibility. If you plan to reduce your speed by a factor of one-third, then you will make the trip between 30 and 60 minutes.

Your job, as convoy leader, is to pace these activities so that you deliver your supplies expediently, while returning to the depot at an appropriate time. Due to the stochastic uncertainty in your activity duration estimates, adhering to these goals might not be achievable under all situations. For instance, should you encounter heavily flooded roads, you might have to drive much slower than anticipated and therefore

miss your first one-hour deadline. If you believe there is a significant possibility of such flooding, you would want to negotiate for more time to complete your mission. Otherwise, barring no other conceivable dangers, you would be willing to proceed.

Everyone at your organization recognizes such operational risks, and therefore no one expects perfect guarantees of temporal success. However, your superiors coordinating the city-wide relief effort want to be notified if any convoy leader does not believe he has a 90% chance of meeting all deadlines. In addition, you would like to personally guarantee that the people will receive your supplies on time with 95% probability. Prior to embarking, you must demonstrate that you have a strategy for achieving these temporal goals within such safety margins.

This small example's scheduling requirements and temporal risks are intuitive to grasp, but assessing risk in the context of larger missions becomes quite complex. By encoding these features in formal models, we may apply computational methods to evaluate risks in scenarios that are beyond humans' inference abilities. In Section 2.2, we discuss constraints that model global coordination and local execution. Then, in Section 2.3, these are combined with chance constraints that model acceptable risk, which leads us to the formal problem statement.

2.2 pSTN Model

We model the constraints on global coordination and local execution through the **probabilistic simple temporal network (pSTN)**. First, we describe what modeling primitives are needed to represent our scenario. Then, we summarize these primitives as the definition of a pSTN. We apply some reasonable assumptions on how these primitives may be composed, so that the restricted class of pSTNs matches those scenarios we are concerned with.

2.2.1 Primitives

Modeling temporal coordination requirements

To model temporal coordination, we begin with the notion of an *event*, which represents a point in time. Each activity has two endpoints, a start event and an end event. These events are *controllable*, meaning that we determine when they should occur. This corresponds to the idea of *scheduling*. When we coordinate activities, we schedule their endpoints to happen close enough to each other (or far enough from each other). Formally, this is expressed as a set of **simple temporal constraints (STCs)**, each consisting of a lower and upper bound on the temporal difference between two events.

For example, Figure 2-1 models our scenario's global coordination requirements using four events and two STCs. Our scenario is composed of three activities in series: driving to the neighborhood, unloading supplies, and driving back. In sequence, they start and end on events e_1 and e_2, e_2 and e_3, and e_3 and e_4. Each activity is drawn as a thick arrow from start to end, while both temporal constraints are drawn as thin arrows with acceptable intervals written above. The first temporal constraint associated with dropping off supplies is an upper bound on the time between leaving the depot (e_1) and finishing unloading (e_3). This is represented by the interval $(-\infty, 60]$, where units are in minutes. Likewise, the other temporal constraint concerning depot logistics is an interval $[80, 120]$ on the time between arriving at the drop-off site (e_2) and returning to the depot (e_4).

STCs are equivalently called **requirement links** because they specify temporal requirements to be satisfied. The definition follows.

Definition 1 (Requirement Link)**.** *A* ***requirement link*** *constrains the time between two events e_a and e_b to a set of allowed durations D. That is, $t(e_b) - t(e_a) \in D$. When D is an interval with lower and upper bounds, $[l, u]$, the requirement link is in the form of a* simple temporal constraint*.*

The STC model of activity coordination is widely applicable. It readily handles our scenario where activities are laid in series. An event can simultaneously be the end of

one activity and the start of the next. Alternatively, we can insert an implicit temporal constraint of zero (i.e., an interval of $[0, 0]$) between two activities in sequence. In more complex scenarios, where activities occur simultaneously or in parallel, STCs are equally applicable. Finally, if we require coordination in the middle of an activity, then we can split that activity in two, creating a middle event as the point of coordination.

This model of temporal coordination draws directly from the *Simple Temporal Network (STN)*, which is simply a set of events and STCs. The term "network" refers to the graphical depiction of a set of constraints. The main limitation of this model is that constraints are restricted to the form of STCs, which constrain the allowed duration between two events to a single interval. However, in most situations, if two different durations are acceptable, then any duration in between is also acceptable; this is equivalent to having an interval of durations that are temporally feasible.

There are instances, though, where some environmental condition would determine whether one interval or another is applicable. That would require modeling conditional constraints, which is beyond the scope of this work.

Modeling activity execution

If we modeled each activity's duration as a requirement link, this would imply that we have full control over completing each activity within some time interval. However, activity execution in our scenario is more subtle. We assume that in addition to our nominal range of control, Nature will inject stochastic effects, such as the effects of flooding. Therefore, we cannot determine when exactly an activity will end, but throughout activity execution, we may be able to adjust our control efforts in response to Nature.

We represent this model of activity execution by an **contingent links** followed by a requirement link. The contingent link's spans the time between the activity's start event to an *uncontrollable event*, which is a "virtual" event in the "middle" of activity execution. The outcome of the contingent link is a random variable distributed over the set of reals. Its probability distribution summarizes the effect of Nature over the entire course of the activity's execution. Note that this duration can be negative, as

Nature may speed you up or slow you down. Then, the requirement link that follows represents the range of control you have. It begins on the uncontrollable event and ends back on the activity's end event, which is controllable. Hence, you do not know when the uncontrollable duration will end, but given an outcome for it, you maintain some control over when the entire activity ends.

Figure 2-2 illustrates this model applied to the first activity in our scenario, driving into the neighborhood. The uncontrollable event is represented by a square, as opposed to circles for the activity's controllable start and end events. The solid arrow from the uncontrollable duration to the end event is a regular STC with lower and upper bounds of 20 and 40 minutes, respectively. This encodes your model of how fast or slow you can drive under nominal conditions. The dotted arrow from the start event to the uncontrollable event is the uncontrollable duration. Its associated distribution is Gaussian, centered at 0 and with a standard deviation σ of 2.5 minutes. That means the durations falls within the 2σ range of ± 5 minutes with 95.5% likelihood, encoding your rough estimate that Nature would affect your driving time by at most 5 minutes.

Definition 2 (Contingent Link)**.** *A* ***contingent link*** *defines a temporal distribution for the time between two events e_a and e_b. That is, $t(e_b) - t(e_a) = \omega$, where ω is a real-valued random variable, and it is distributed according to probability density function $f(\omega)$ and corresponding cumulative density function $F(\omega)$. Because the link ends on event e_b, this event is said to be* uncontrollable*.*

This model of a contingent link followed by a requirement link reflects the intuition of aiming to stay on schedule while executing an activity. The schedule to be constructed would assign specific values to the activity's start and end events, hence implying a total duration. This duration would determine how much control effort you have to put in nominally. In the case of driving to the neighborhood, if it is 10 miles away, and your schedule says to reach there in 30 minutes, then you should drive at 20 mph. Along the way, if you have to slow down to traverse a flooded section of the road, then you would speed back up on dry land, so that you average 20 mph in

the long run. However, if the flooding slowed you to a crawl, you might not be able to recover all the lost time. Thus, a more realistic model for activity execution may be a series of tiny uncontrollable and controllable segments. Nevertheless, this could be equivalently summarized as two durations in sequence, the first a contingent link and the second a requirement link.

2.2.2 Model definition

We now define the pSTN, using the intuition developed above for modeling temporal coordination and activity execution. This definition actually encompasses a wider set of temporal situations than those we are currently interested in. Therefore, we impose some limitations on the structure of the pSTN, and we explain why these assumptions do not greatly restrict the class of useful scenarios. Those extra classes would be useful to consider in future work, using different modeling and solution strategies.

Definition 3 (pSTN). *A **probabilistic simple temporal network** is a tuple* $\mathcal{N}_p = \langle \mathcal{E}, \mathcal{L}_{\text{req}}, \mathcal{L}_{\text{ctg}} \rangle$*, representing:*

- *A set of* events $\mathcal{E}$*, partitioned into* controllable events $\mathcal{E}^c$ *and* uncontrollable events $\mathcal{E}^u$*, i.e.,* $\mathcal{E} = \mathcal{E}^c \bigcup \mathcal{E}^u$*.*
- *A set of* requirement links $\mathcal{L}_{\text{req}}$*, each specifying a simple temporal constraint linking event* e_i *to event* e_j*, where* $e_i, e_j \in \mathcal{E}$ *and* $e_i \neq e_j$*. The temporal bounds are denoted* $[l_{ij}, u_{ij}]$*.*
- *A set of* contingent links $\mathcal{L}_{\text{ctg}}$*, each linking an event* $e_i \in \mathcal{E}$ *to an uncontrollable event* $e_j^u \in \mathcal{E}_u$*. The duration between them is denoted by the random variable* ω_{ij}*, which has probability density function* $f_{ij}(t)$ *and corresponding cumulative density function* $F_{ij}(t)$*.*

A *schedule* $t : \mathcal{E}^c \rightarrow \mathbb{R}^{|\mathcal{E}^c|}$ assigns a timepoint to every controllable event. During execution, each contingent link's probabilistic outcome ω_{ij} is an *observation*. Since each $\omega_{ij} \in \mathbb{R}$, the space of complete observations for a pSTN's contingent links is

$\Omega = \mathbb{R}^{|\mathcal{L}_{\text{ctg}}|}$. Together, a schedule and a complete observation determine the timepoints of the uncontrollable events.

Structurally, the pSTN is similar to the STNU [17], which also extends the STN to include temporal uncertainty. However, the STNU model restricts each uncontrollable duration to an interval. That is, in driving to the neighborhood, you would only be able to model Nature's temporal influence as somewhere in the interval $[-5, 5]$. On one hand, restricting a duration to a finite interval simplifies analysis. On the other, this is a less accurate model, and does not provide the rich notion of likelihood of success that this work addresses.

Assumptions

We make three assumptions on the form of the elements in $\mathcal{L}_{\text{req}}$ and $\mathcal{L}_{\text{ctg}}$. The first two assumptions govern where and what type of constraints may be drawn between which events, and the third limits the type of temporal distributions we will consider.

Assumption 1. *Every contingent link begins on a controllable event. That is, given $\ell \in \mathcal{L}_{\text{ctg}}$ linking e_i to e_j^u,*

$$\ell \implies e_i \in \mathcal{E}^c.$$

Assumption 2. *All requirement links must include at least one controllable event. That is, given $\ell \in \mathcal{L}_{\text{req}}$ linking e_i to e_j,*

$$\ell \implies (e_i \in \mathcal{E}^c) \bigvee (e_j \in \mathcal{E}^c).$$

Together, these two assumptions restrict the class of pSTNs to model each activity as a contingent followed by a requirement link, as discussed before. Furthermore, it allows two activities to share the same source of uncertainty. For example, when delivering supplies to the people of the neighborhood, you may have both food and medical equipment to distribute. The medical equipment may be more delicate to handle, so you cannot distribute it as fast. However, as you distribute both in parallel to the same crowd, the uncertainty in both activities is due to how efficiently you process the same lines. The corresponding temporal model for this situation is shown

in Figure 3-1. Event e_2 marks the beginning of distributing supplies. There are two requirement links from the uncontrollable event to events e_{3a} and e_{3b}, corresponding to distributing food and medical supplies, respectively. These last two events converge back at event e_3, waiting for whichever finishes first.

These restrictions form a valid model for many useful scenarios. Specifically, these are the situations it excludes.

- Assumption 1 does not allow a contingent link to follow another one immediately. Together, they would represent a single contingent link, and the only useful purpose for modeling like that is if one tries to coordinate by reaching into the middle of a contingent link. As mentioned before, this case would be better served by splitting the activity in two.

- Assumption 2 gives you at least one direct knob on the satisfiability of each requirement link. If both events are controllable, then you would have full control to satisfy the constraint via scheduling. If only one is controllable e_i and the other is uncontrollable e_j, then since Assumption 1 says e_j arrives some unspecified duration after a controllable event e'_j, you could coordinate e_i and e'_j to protect the constraint against one source of uncertainty. This corresponds to scheduling the start and end event of an activity, which we have discussed above.

  However, we would rarely encounter a requirement link with both e_i and e_j uncontrollable, for two reasons. First, this does not make sense in the interpretation of uncontrollable events as "virtual" events. Second, even if these were real events, you could only coordinate their direct predecessor events e'_i and e'_j. Now your schedule to e'_i and e'_j must protect against two sources of uncertainty, which again, does not have a clear physical interpretation during local execution of activities.

Both assumptions essentially restrict Nature to at most one degree of freedom. When executing an activity, the first assumption lets you aggregate all the sources of uncertainty into one contingent link, and the second means you have to control against

only the uncertainty in that link.

Assumption 3. *Each temporal distribution's probability density function f_u is semi-concave. The definition of a* semi-concave function *is that given f_u's mode $d^\star$, we must have*

$$\forall d < d^\star,\ f'_u(d) \geq 0$$
$$\forall d > d^\star,\ f'_u(d) \leq 0.$$

Equivalently, for any constant c, the set D for which $f_u(d) \geq c, \forall d \in D$, is an interval, which could possibly be empty.

Pictorially, this third assumption means that f_u is unimodal. Intuitively, semi-concavity of a probability density function reflects the idea that stochasticity is centered around a deterministic process. If a probability density function has two modes, then perhaps there is some underlying switching condition determines whether process A or process B happens, each with its own nominal duration and stochasticity. Modeling such conditions is beyond our current scope. Practically, semi-concave functions will allows us to take advantage of corresponding convexity in cumulative distribution masses.

2.3 pSTP Scheduling Problem

The pSTN encodes temporal coordination and activity durations by composing contingent and requirement links. However, this only models what we would like and what we can do. To assess temporal risk, we need to know the chance of achieving our desired temporal coordination, while constrained to the physical laws of activity execution. This is what a **chance constraint** represents. It mandates the maximum acceptable risk with respect to achieving a set of temporal goals.

Chance constraints are applicable to pSTNs because as discussed previously, there is no way to always guarantee satisfaction of the temporal coordination constraints, given the stochasticity in activity execution. However, given a strategy for carrying

out the mission, it would imply a stochastic model for every event's actual execution time. From this model, we could infer each coordination constraint's likelihood of being satisfied.

Therefore, our temporal risk assessment problem is to find an **execution strategy** that would satisfy a set of chance constraints over a pSTN model. In this section, we first explain what an execution strategy is, so that we may then define a chance constraint in terms of one. Finally, we define the **probabilistic simple temporal problem (pSTP)** in terms of chance constraints and the pSTN.

2.3.1 Execution strategy

An execution strategy defines the control actions we use to execute our mission. These actions, combined with Nature's stochastic effects, result in an actual execution time for each event. For instance, suppose we want to arrive the neighborhood in 30 minutes. If we simply assume nominal conditions and drive without adjusting to Nature's effects, then according to our activity model, this is equivalent to arriving at event e_2^u by sampling a temporal outcome ω_{12} from the contingent link, and then the activity ends at event e_2 30 minutes later. The actual time spent getting there would be:

$$t(e_2) - t(e_1) = \omega_{12} + 30.$$

Now, suppose we adopt the more flexible strategy of continually observing Nature's effects during execution, and we try to adjust to meet the 30-minute specification. This is equivalent to observing *when* we arrive at e_2^u, and at that moment, deciding the duration until e_2. Since the range of control we have is between 20 and 40 minutes, the range of uncertainty we can tolerate is ± 10 minutes in order to arrive in 30 minutes. If the disturbance extends the driving duration more than 10 minutes, then the best we can do is to drive as fast as possible at the speed corresponding to 20 minutes, and vice versa if the disturbance is less than -10 minutes. Hence, the actual duration of

this driving activity would be:

$$t(e_2) - t(e_1) = \begin{cases} \omega_{12} + 40 & : \quad \omega_{12} < -10 \\ 30 & : \quad -10 \leq \omega_{12} \leq 10 \\ \omega_{12} + 20 & : \quad \omega_{12} > 10 \end{cases} \tag{2.1}$$

Hence, this execution strategy yields a different schedule than the previous one. Here is the formal definition for an execution strategy.

Definition 4 (Execution Strategy)**.** *An* ***execution strategy*** *for a pSTN $\mathcal{N}^p$ is a function $t_x : \mathcal{E}^c \times \Omega \rightarrow \mathbb{R}^{|\mathcal{E}^c|}$ that determines the actual schedule followed during execution, conditioned on complete observation $\omega \in \Omega$. For convenience, we will denote a single event's execution time as $t_x(e_i)$.*

Note that in our definition, the complete observation is provided but not necessarily used. In our example, we did not use it at all the first time, and the second time, we observed only what was happening at the current activity to plan for the activity's end timepoint. In real life, we could observe the entire history up to what has happened so far, but not beyond into the future. We may choose not to use the entire history, though, for instance, if it is too expensive computationally. Whether we discard the observation, use only the history, or use the complete observation our execution strategy corresponds to the notions of strong, dynamic, and weak controllability in STNU execution, respectively.

2.3.2 Problem definition

The actual schedule that transpires from following an execution strategy determines whether the temporal coordination constraints hold. Since that schedule depends on a complete observation, and is therefore nondeterministic, we use chance constraints to express how important it is for the schedule to meet certain coordination constraints. In our scenario, the chance constraints would correspond to the 90% requirement on the entire mission and the 95% requirement on delivering supplies on time. The former requires both requirement links in the pSTN to be satisfied, whereas the latter

concerns only the $[0, 60]$ link. In principle, given an execution strategy, we could derive a joint probability distribution for when each event would occur. This would then determine the likelihood of satisfying each requirement link or a set of requirement links at once. This concept is mathematically formulated below.

Definition 5 (Chance Constraint)**.** *Given a pSTN $\mathcal{N}^p$ and an execution strategy t_x, a **chance constraint** requires a minimum probability Δ of satisfying all requirement links in a subset of $\mathcal{L}_{\text{req}}$. That is,*

$$\Pr\left[\bigwedge_{\substack{\ell \in \mathcal{L}' \\ \subseteq \mathcal{L}_{\text{req}}}} t_x(e_j) - t_x(e_i) \in [l_{ij}, u_{ij}]\right] \geq \Delta.$$

We are specifically interested in chance constraints over requirement links that encode temporal coordination. Given our activity execution model, this means these requirement links only link controllable events in the pSTN. It would not mean much to consider those requirement links involving an uncontrollable event, for two reasons. First, our uncontrollable events are "virtual" events that only summarize the net effect of Nature's stochasticity over an entire activity, and there is no physical way to verify that a constraint involving a virtual event has been satisfied. Second, in our model, those requirement links reflect the range of feasible control we have over an activity's duration. In the real world, physically, they cannot be violated, no matter what the execution strategy.

The ability to express multiple chance constraints is key. If we allow the scenario to become arbitrarily large, then we are adding more sources of uncertainty, so the chance that everything will go perfectly plummets. What we actually care about is that individual parts of the mission plan will succeed. Some goals may be more important than others, so it may be easier to focus on satisfying a few really well than to satisfy them all equally badly. This is what the two chance constraints in our scenario demonstrate. To us, delivering supplies on time is the most important part of the mission; if a constraint had to be bent, the logistics constraint $[80, 120]$ be first to be modified. During execution, then, to satisfy the delivery-specific 95%

requirement and the mission-wide 90% requirement, we would be extra-mindful of the time when driving out and unloading supplies, and willing to take more temporal risk on the way back. Thus, satisfying multiple chance constraints over a pSTN is our core problem, and that is what the pSTP expresses.

Problem 1 (pSTP)**.** *The* ***probabilistic simple temporal problem*** *is a tuple* $\mathcal{P}^p = \langle \mathcal{N}^p, \mathcal{C} \rangle$*, where:*

- $\mathcal{N}^p$ *is a pSTN with components* $\langle \mathcal{E}, \mathcal{L}_{\text{req}}, \mathcal{L}_{\text{ctg}} \rangle$*.*
- $\mathcal{C}$ *is a set of chance constraints, each constraint* c *specifying a minimum probability* Δ_c *over a subset of the requirements links,* $\mathcal{L}_c \subseteq \mathcal{L}_{\text{req}}$*.*

A solution to $\mathcal{P}^p$ *is an execution strategy* t_{x} *that satisfies all the chance constraints in* $\mathcal{C}$*.*

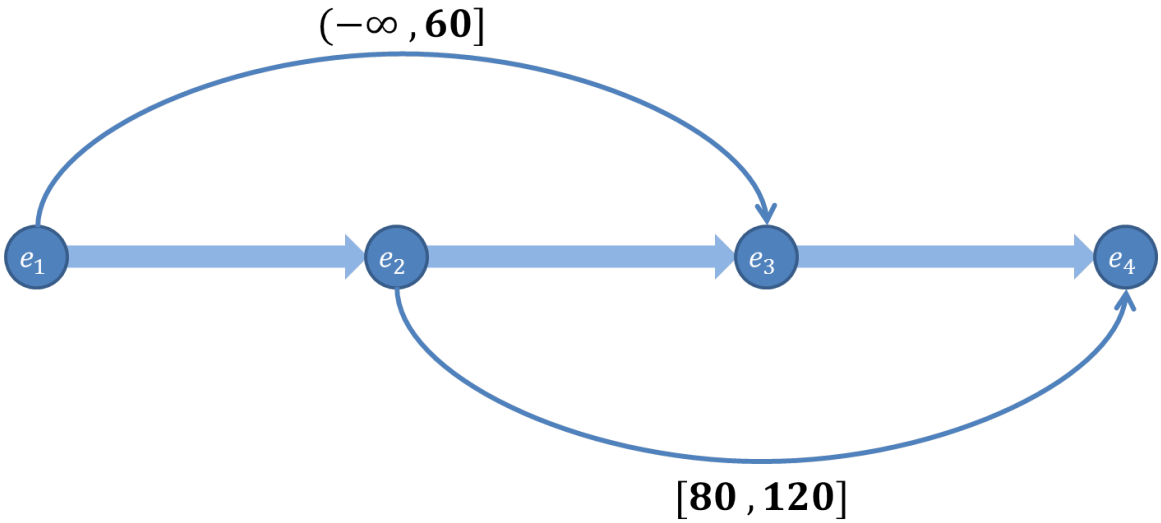

Figure 2-1: Temporal schematic of activities with temporal coordination constraints linking their start and end events.

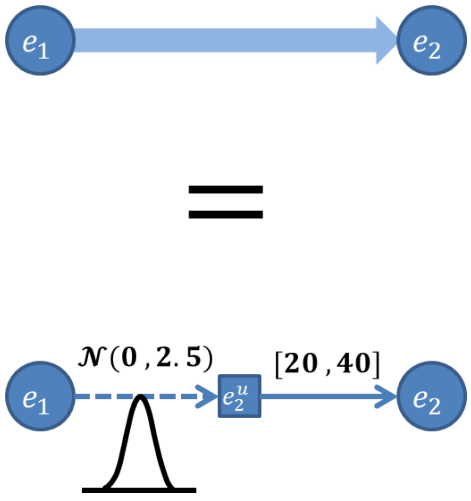

Figure 2-2: Modeling an activity as a probabilistic contingent link plus a requirement link.

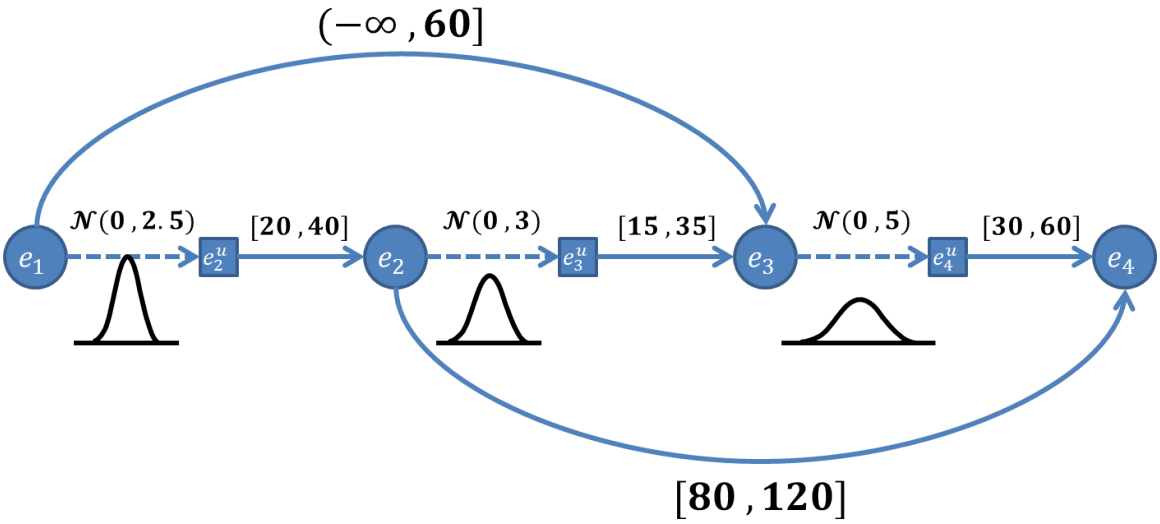

Figure 2-3: Example pSTN modeling the disaster relief scenario.

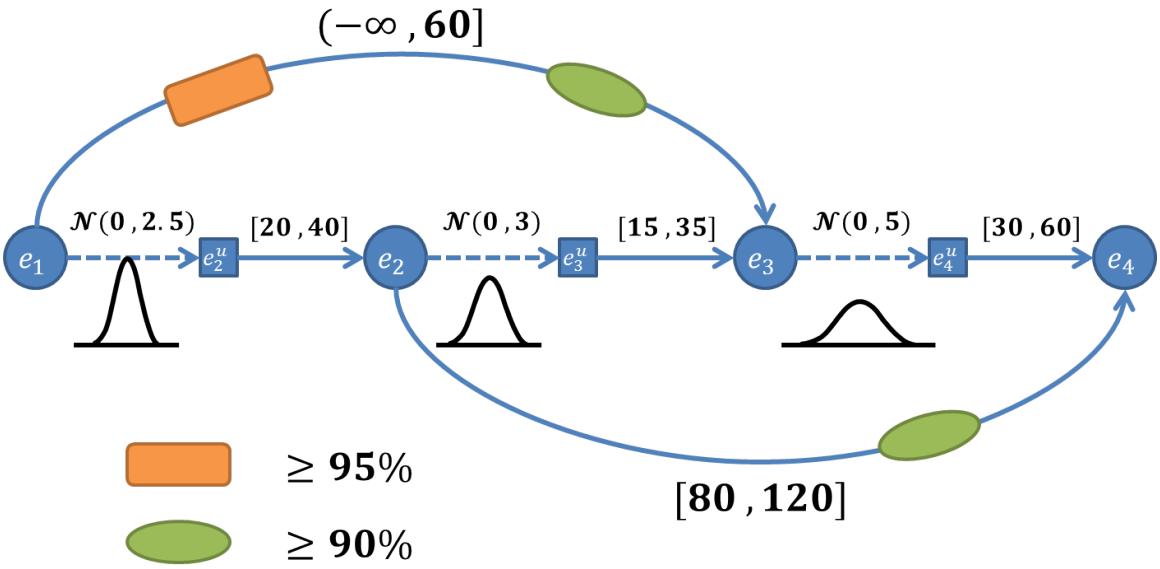

Figure 2-4: Example pSTP modeling the disaster relief scheduling problem.

Chapter 3

Reformulation into Convex Risk Allocation

This chapter presents our solution to the pSTP, which involves offline scheduling plus online execution that tries to meet the schedule. Executing a schedule online is relatively simple. However, designing a schedule to satisfy the chance constraints is considerably more complex and requires knowledge of the online execution strategy. This is the main component of our algorithm.

Our approach to offline scheduling is to reformulate the problem's structure so that the probabilistic aspects are more tractable to reason about. This reformulation is a novel combination of concepts from risk allocation, STNU controllability, convex optimization, and conflict detection. The goal is to construct a schedule that can be verified against the pSTP's chance constraints. We do this by allocating risk to each source of temporal uncertainty, making sure the total risk still satisfies the chance constraints, and deriving a schedule that can avoid such risk while respecting the temporal constraints. Along the way, our reformulation borrows the form of the STNU to frame our problem in terms of convex feasibility. Then we can use an off-the-shelf solver to find an acceptable schedule, if one exists. Our usage of the convex solver is further optimized by incrementally detecting temporal conflicts and inserting them in place of the original set of temporal constraints.

Section 3.1 presents an overview of the methods used in our approach. It includes

a formal statement of the online execution strategy and a run-through example of the offline scheduling algorithm on the disaster relief scenario. Then, we describe each step of the reformulation in detail in Section 3.2.

3.1 Approach Overview

In this section, we give a high-level motivation and overview of our algorithm. First, we outline the computational challenges posed by the pSTP, and we highlight the methods used to address these challenges. These methods are demonstrated on the disaster relief scenario. However, they rely on assessing the risk of our strategy for executing a pSTN according to a schedule. Thus, we define the behavior of the online portion first, and then run through the offline portion applied to the scenario.

3.1.1 Challenges and approach to solving the pSTP

Challenges

To solve the problem, we need to compute an execution strategy and verify it against each chance constraint. This entails temporal and probabilistic reasoning over the space of complete observations Ω. According to Definition 5, an execution strategy t_x satisfies a chance constraint c if and only if there exists a subset $\Omega' \subseteq \Omega$ with these two properties:

(a) Given any observation in Ω', the execution strategy would find execution times that satisfy all the requirement links in c. In other words, the execution strategy is robust against Ω'.

(b) The probability that the actual observation lies in Ω' is at least Δ_c.

Hence, to verify a solution, we need to be given a candidate subset Ω' and be able to verify conditions (a) and (b). Additionally, to solve the problem, we need to construct such a subset or prove one does not exist. Each of these three capabilities is computationally challenging for the following reasons:

1. It is hard to verify whether an execution strategy is robust against a given Ω'. This is because requirement links are specified in terms of events' execution times, whereas Ω' is a set of observations. The mapping from observations to event times, given by the execution strategy, may be difficult to analyze. For example, consider the hour-long requirement link on driving to the neighborhood and delivering supplies. First, our driving behavior and reaction to temporal uncertainty along the way determines when we arrive. Then, given the amount of time left until 60 minutes, we may start distributing supplies faster or slower, but dealing with a disorganized crowd could still push us over schedule.

   Thus, an activity's duration could depend on both its own stochasticity as well as how we respond to the execution of previous activities. The resulting execution strategy could be arbitrarily nonlinear and intractable to analyze. Another complication is determining the implied set of observations that are relevant to this requirement link. In this case, it is obvious that only the durations for driving and unloading matter, but determining this for more complicated activity networks is nontrivial.

2. It is hard to calculate the probability mass of an arbitrary Ω'. This requires us to integrate a joint probability distribution over Ω', which involves dependent limits of integration among different variables. For instance, suppose if the driving disturbance is at most $+5$ minutes, then we can tolerate a disturbance of at most $+8$ minutes when unloading supplies. Now if the driving disturbance increases to $+10$ minutes, then even if we work faster to unload, we might only tolerate up to a $+6$ minute disturbance for this latter activity. In addition to dependent limits of integration, the individual distributions for each contingent link's outcome are most likely nonlinear. Together, these two complexities would make such a multivariate integral hard to evaluate.

3. It is hard to construct an Ω' that will meet these conditions, due to the same complications that make these conditions hard to verify. Namely, the complexity of the execution strategy and the nonlinearity of the joint probability

distribution make it difficult to decide how far we may push out the boundaries of Ω'. In particular, these two aspects have competing effects. On one hand, the execution strategy can only be robust to a certain extent of temporal uncertainty, but on the other, the probability mass of Ω' needs to be large enough. The relative sensitivities of these conditions at every point along the boundary of Ω' determines whether we want to pull it in (less uncertainty) or push it out (more probability mass).

Approach

To address these three challenges, our approach involves two major methods. First, we restrict our execution strategy to creating a full schedule prior to the mission and following that schedule as best we can during execution. Rather than assess the risk of meeting the required temporal coordination, we assess only the risk of meeting the pre-planned schedule. Second, we represent the risk of such a schedule by allocating risk to each contingent link. The variables representing the risk allocation are then used to reformulate the original set of chance constraints into a convex feasibility problem. This allows us to solve for a risk allocation, from which we can extract a schedule that meets both the temporal and probabilistic requirements of the chance constraints. Below, we discuss why we choose these two strategies and how they resolve the three challenges.

Having a pre-planned schedule offers two computational advantages. Assuming the schedule is actually followed during execution, then it becomes easy to verify whether the temporal coordination constraints will be met, and it also mandates an exact duration for each activity. The first property decouples the coordination requirements from the effects of temporal uncertainty. Given a schedule that already achieves temporal coordination, all we need is to find an Ω' for which our execution strategy can always meet the schedule. This resolves the first challenge. The second property decouples temporal uncertainty between activities. If every activity is assigned a duration, independent of any other activities' durations, then the temporal uncertainty each activity can tolerate is independent of other activities' temporal

uncertainties. Therefore, Ω' may be composed from individual activities' ranges of uncertainty. Each activity's risk is easily calculated over the univariate distribution of that activity's contingent link. Thus, $\Pr(\Omega')$ is easily computed as the product of their risks, hence resolving the second challenge.

Of course, during execution, there may be outcomes where the schedule cannot be met; some will still satisfy the temporal coordination requirements, and some will not. If the schedule already satisfies all temporal coordination constraints, then the probability mass of this schedule-related Ω' is a conservative estimate of the true risk of not satisfying temporal coordination. However, it is precisely this schedule-related Ω' that makes it easy to verify both the temporal and the probabilistic conditions of a chance constraint. Therefore, we choose to accept the conservatism and only assess the risk of meeting a schedule.

Since a schedule's risk is represented by a decomposable Ω', then conversely, an Ω' in decomposable form represents a schedule or a family of schedules with equivalent risk. Our risk allocation approach defines an Ω' in decomposable form, parameterized on each activity's range of acceptable uncertainty. Since Ω' corresponds to both a schedule and a probability mass, then in terms of its parameters, we can analytically express whether Ω' has the properties to satisfy each chance constraint. Furthermore, these expressions are proven to be convex, so we can leverage efficient numerical methods from convex optimization. Hence, just the reformulation to convex optimization needs to be efficient, and this solves the third challenge of finding a suitable Ω'.

To further boost the efficiency of running the convex solver, we do not solve for all temporal constraints. Instead, we discover only those that are violated, and we include them one-by-one into our convex formulation. That way, starting from an initial candidate solution, we only consider options that push the candidate towards a feasible region, hence arriving at a solution faster if one exists. If there is no solution, we also expect to determine that faster, which will happen by the time we collect the minimum inconsistent set of temporal constraints.

3.1.2 Online execution strategy

Missions are executed by carrying out the planned activities, which means that there are two types of temporal decisions to be made. First, we must decide when to start each activity, and second, once an activity has started, we must decide when to end it. Our job, given a schedule, is to make these two decisions so that we conform to the schedule as much as possible.

To know when to start activities, we must first consider the physical flow of events. Any mission plan, scheduled or not, is fundamentally composed of strings of activities. For example, our scenario is one continuous string of three activities. Figure 2-1 had depicted this by showing three activities following each other in sequence. We could have multiple strings if we had additional vehicles performing other things, such as other convoys driving to other neighborhoods or aerial scouts assessing road conditions. Such a representation might be considered by our superiors running the disaster relief effort. Note that activities are distinguished from temporal coordination constraints in that the former represent physical interactions with the world, while the latter are only abstract desires. Activities must have positive duration, whereas temporal constraints may specify positive or negative relationships in time.

In addition, strings may split or merge. Splitting means that we branch into doing more than one activity simultaneously. Conversely, merging corresponds to activities waiting for each other to finish. Note that waiting is also an activity, as we are physically there and spending positive duration. A wait activity's contingent duration simply degenerates to a deterministic value of 0. Figure 3-1 shows a simple example of splitting and merging on the activity of delivering supplies to the crowd. Here, we have split our supplies into two types, food and medicine. Upon arrival, our convoy team begins to unload these two supplies in parallel. Because the uncertainty model is based on the crowd, the contingent link is shared, but we have distinct requirements links indicating how fast we can unload these two commodities. Whichever branch finishes first waits for the other before we pack up and start the return trip. Here, the merge applies to the same split, but that does not have to be the case. Indeed, if we

had two convoys, but a mobile team was dispatched from one and later rendezvoused with the other, the split and merge would be unrelated.

To execute a mission, then, means to execute strings of activities, which represent physical progress. For activities at the beginning of a string, we can start right when scheduled. However, any other activity must follow another on a string, and is therefore dependent on the completion of its predecessor(s). For example, we cannot unload before arriving at the neighborhood, and we cannot drive back before finishing unloading. Furthermore, our mission plan says to start unloading immediately upon arrival and to drive back immediately after we unload everything.

This means every activity has a precise arrival time. We arrive at those without a preceding activity when the schedule says those activities should start, because there is no physical dependence on other activities' completion. For all other activities, we arrive at them once their start event is activated by the preceding activity or activities ending. During mission execution, then, we keep checking for activities that have arrived, and we execute them once they do.

Once we begin executing an activity, though, we have to decide when to end it. This is the second type of decision to be made. A complete schedule implies a precise duration for each activity, which is the difference between the activity's start and end times. Because an activity's duration is directly related to its temporal risk, it follows that in order to preserve the acceptable risk levels that the schedule is designed for, we should execute each activity according to duration. In our scenario, suppose the mission start e_1 is scheduled for time 0, arrival at the neighborhood e_2 is at time 30, and unloading concludes at time $e_3 = 55$. Under normal circumstances, we would finish driving in 30 minutes and have 25 minutes to unload. However, even if driving ends up taking 35 minutes, we still aim for 25 minutes of unloading, so e_3 would happen at time 60.

To execute an activity according to duration, we must observe the result of its contingent link and adjust our control effort within the bounds of the following requirement link. This was demonstrated in Equation 2.1, where we discussed the flexible strategy of observing and reacting to Nature's effects during an activity's

execution. Here, we formalize that strategy. Suppose we are aiming to execute an activity for a duration d. Let the activity's contingent link have duration ω, and let its requirement link have lower and upper bounds l and u, respectively. Our job is to choose a control effort $x \in [l, u]$ such that $\omega + x$ approximates d as much as possible. This is given by the function below.

$$x(\omega) = \begin{cases} u & : \quad \omega < d - u \\ d - \omega & : \quad d - u \leq \omega \leq d - l \\ l & : \quad \omega > d - l \end{cases} \tag{3.1}$$

This function is depicted in Figure 3-2. Intuitively, it means the amount of temporal disturbance we can tolerate is precisely the margin between the desired duration and the lower and upper bounds of control. Beyond that margin, we will not be able to meet the desired duration d; the best we can do is to stay at the limit of our control effort that best counters the disturbance, either l or u.

Thus, this online execution strategy simply follows strings of activities laid out in the mission plan, executing them according to the durations implied by the schedule. It is only a proxy for carrying out the schedule. Any intent to meet the temporal and chance constraints is therefore the responsibility of offline scheduling.

3.1.3 Offline scheduling run-through example

The main intuitions behind our offline scheduling algorithm are concretely demonstrated below. We will continue our running example of the disaster relief scenario, where we last framed the problem as a pSTP at the end of Section 2.3. Our algorithm operates in three major steps.

Step 1. Initialize risk allocation variables.

We saw previously that we could execute an activity for a specific duration, assuming a certain range of contingent link outcomes. Specifically, that occurs when $\omega \in [d - u, d - l]$. In our scenario, suppose we decide to spend $d_{12} = 30$ minutes driving,

$d_{23} = 25$ minutes unloading, and $d_{34} = 45$ minutes returning. Based on Equation 3.1, which describes our online execution strategy, we can meet these durations if and only if the contingent links' durations lie in the ranges $\omega_{12} \in [-10, 10]$, $\omega_{23} \in [-10, 10]$, and $\omega_{34} \in [-15, 15]$. This is depicted in Figure 3-3.

According to our pSTN model, this will not happen all the time, but rather with a certain probability. For any particular ω_{ij}, it lies in its associated range $\left[\underline{\omega}_{ij}, \overline{\omega}_{ij}\right]$ with probability $F_{ij}\left(\overline{\omega}_{ij}\right) - F_{ij}\left(\underline{\omega}_{ij}\right)$, where F_{ij} is the contingent link's cumulative density function. This is easy to calculate in our scenario because the probability distributions are univariate Gaussians. In general, any univariate cumulative distribution is easily approximated by a table of values.

Therefore, by deciding a duration for an activity, we are "allocating" to that activity the risk of whether we can meet that duration during execution. That means we have reformulated the problem into assigning durations such that a) we can extend it to a schedule that meets the temporal coordination constraints, and b) the activities' individual risks compose to satisfy the chance constraints.

Step 2. Formulate as convex feasibility.

The next step is then to mathematically express these two conditions over our durations. With regards to the schedule, it needs to obey both the durations and the temporal coordination constraints. We introduce the variables $t_i = t(e_i)$ to represent the scheduled time of event e_i. The first set of constraints encodes the durations of our three activities.

$$t_2 - t_1 = d_{12} \tag{3.2}$$

$$t_3 - t_2 = d_{23} \tag{3.3}$$

$$t_4 - t_3 = d_{34} \tag{3.4}$$

The second set reflects the coordination constraints.

$$t_3 - t_1 \leq 60 \tag{3.5}$$

$$80 \leq t_4 - t_2 \leq 120 \tag{3.6}$$

To express the chance constraints, we only have to encode the composed risk, because temporal coordination is already addressed by the scheduling constraints. However, we have to know what activities' risks are relevant to compose for each chance constraint. The 95% chance constraint only concerns the one-long temporal constraint between e_1 and e_3, whereas the 90% chance constraint is over both temporal constraints. Whether the first temporal constraint is satisfied depends on the durations d_{12} and d_{23}, so those are the relevant durations for the first chance constraint. Meanwhile, the second temporal constraint depends on durations d_{23} and d_{34}, so the second chance constraint depends on all three durations. Hence, the chance constraints may be expressed as

$$R_{12}R_{23} \geq 0.95 \tag{3.7}$$

$$R_{12}R_{23}R_{34} \geq 0.90, \tag{3.8}$$

where $R_{ij} = F_{ij}\left(d_{ij} - u_{ij}\right) - F_{ij}\left(d_{ij} - l_{ij}\right)$ expresses the probability of meeting activity ij's duration.

Together, Equations 3.3 through 3.8 encode the reformulated problem. Note that the scheduling equations are all linear. In addition, the chance constraints may be proven to be convex, due to Assumption 3, which restricts us to semi-concave probability density functions. Thus, given an initial set of proposed durations, such as $d_{12} = 30$, $d_{23} = 25$, and $d_{34} = 45$, we can use off-the-shelf convex optimization software to find a feasible set of durations and a schedule.

Table 3.1: Iterations of constraint discovery.

Added Constraint	d_{12}	d_{23}	d_{34}	$R_{12}R_{23}$	$R_{12}R_{23}R_{34}$
–	30	25	45	0.9991	0.9964
$60 + d_{34} \geq 80 + d_{12}$	28	25	48	0.9985	0.9901
$d_{23} + d_{34} \geq 80$	28	30	50	0.9516	0.9299

Step 3. Discover temporal conflicts incrementally.

Solving five linear plus two nonlinear constraints is easy, but there would be many more constraints for large missions. In particular, when planning larger missions, the number of activities and temporal coordination constraints would tend to increase more rapidly than the number of chance constraints. Thus, we further reformulate the linear temporal constraints so that they only need to be discovered and considered one-by-one. Below we illustrate.

Suppose we initialize the durations as we have described in Steps 1 and 2. If we just insert the two nonlinear constraints into the solver, it will verify that these durations satisfy each chance constraint's acceptable risk level. This is shown in the first line of Table 3.1. However, if we examine the solution, we will notice an inconsistency on the duration between e_1 and e_4. We know e_3 is supposed to happen no later than 60 minutes after e_1, and the duration d_{34} is set to 45 minutes. On the other hand, the duration d_{12} is set to 30 minutes, and e_4 should not occur earlier than 80 minutes after e_2. Hence, we have two conflicting requirements, one saying the duration between e_1 and e_4 is no more than 105 minutes, and the other saying no less than 110 minutes.

To resolve this conflict, we derive the linear temporal constraint saying that the upper bound on this duration must be at least the lower bound. This is reflected in the second line of Table 3.1. When we add this constraint to the original two nonlinear constraints, the solver returns a new solution, where d_{12} has been reduced by 2 minutes and d_{34} increased by 3 minutes, thus resolving the conflict.

However, we then discover another conflict, which is that d_{23} and d_{34} do not add up to at least 80 minutes. When we add that to our constraint set, we now arrive

at the durations listed in the third line. Meanwhile, the risks associated with the chance constraints have been affected, too, but not to the point of violation. The final solution is to schedule $e_1 = 0$, $e_2 = 28$, $e_3 = 58$, and $e_4 = 108$. This satisfies all the temporal coordination constraints, and the associated risks with the two chance constraints are 95.12% and 92.99%, respectively.

By incrementally discovering inconsistent temporal constraints, we ended up only adding two linear constraints to our system. Also, we did not have to consider the schedule variables t_1 through t_4 to extract these constraints. These savings, although modest, become very valuable for large problems.

3.2 Reformulation Procedure

In this section, we provide full details on the offline scheduling algorithm, which is this thesis's core contribution. Our discussion formalizes the three steps of the run-through example. Here, we present the top-level algorithm, which breaks into three subroutines. These subroutines are then discussed individually.

In this more formal discussion, the details of our presentation differ slightly in two ways from that of the run-through example. First, our risk allocation variables will represent windows on the contingent links' outcomes, rather than representing activity durations. This is because in general, and as discussed before, different activities may share the same contingent link. In our scenario, there is only one activity series, without any splits. Hence, it was equivalent and more intuitive to talk about activity durations. Nevertheless, risk windows on contingent links are a more direct representation of risk allocation, so that is what we use here.

Second, note that the schedule variables t_1 through t_4 were ultimately unnecessary. Therefore, we will not introduce them here. They were presented to contrast the size of the resulting convex problem, had we not utilized the incremental temporal constraint discovery approach. By the end of our algorithm, either we will have produced a consistent STN, from which we can extract a schedule, or the problem will have been deemed infeasible.

Algorithm 1: Chance-constrained offline scheduling.

Input: a pSTP $\mathcal{P}^p = \langle \mathcal{N}^p, \mathcal{C} \rangle$
Output: a valid schedule t_s, or infeasible

$\langle \mathcal{R}_{\text{ctg}}, \overline{\mathcal{N}}, r^0 \rangle \leftarrow$ `InitializeRiskAllocation`$(\mathcal{N}^p)$.
$\langle \mathcal{C}' \rangle \leftarrow$ `ReformulateChanceConstraints`$(\mathcal{C}, \mathcal{N}^p, \mathcal{R}_{\text{ctg}})$.
$\langle \mathcal{N}, r^f \rangle \leftarrow$ `SolveIncrementally`$(\mathcal{C}', \overline{\mathcal{N}}, r^0)$.
if `SolveIncrementally` *found no solution* **then**
    return infeasible
else
    Extract and **return** schedule t_s from $\mathcal{N}$.

Algorithm 1 is the top-level scheduling algorithm. We begin by creating and initializing the risk allocation variables on the pSTN $\mathcal{N}^p$. As mentioned, these variables represent windows on each contingent link, and therefore, each contingent link has an associated risk that can be expressed in terms of its risk allocation variable. The set $\mathcal{R}_{\text{ctg}}$ contains all these individual risk expressions for the contingent links in $\mathcal{N}^p$. At the same time, we reformulate the pSTN into an STN parameterized in terms of the risk allocation. The parameterized STN is written as $\overline{\mathcal{N}}$. Finally, the initialized values of all the risk allocation variables are listed in the vector r^0.

Next, we reformulate the pSTP's chance constraints in terms of the risk allocation variables. To do so, we need to match relevant contingent links in the pSTN $\mathcal{N}^p$ to each chance constraint in $\mathcal{C}$. Once the relevant links are found, we can compose their individual risk expressions from $\mathcal{R}_{\text{ctg}}$ into reformulated risk expressions for each chance constraint. These reformulated chance constraints are all collected into the set $\mathcal{C}'$.

With the probabilistic constraints of the pSTP encoded in $\mathcal{C}'$, we now incrementally encode the temporal constraints and solve for a risk allocation. We begin by giving the convex solver all the probabilistic constraints and the initial risk allocation r^0. The parameterized STN $\overline{\mathcal{N}}$ is provided to extract the next temporal conflict at any stage. In the end, if the convex solver reaches an iteration where there is no solution, then the problem is infeasible. Otherwise, it will return a final risk allocation r^f and the consistent STN it induces $\mathcal{N} = \overline{\mathcal{N}}(r^f)$. Therefore, we can use standard

STN decomposition algorithms [7] to extract a complete offline schedule t_{s}.

This offline scheduling algorithm, combined with the online execution strategy, is sound with respect to solving the original pSTP. However, there are a few points of conservatism, and hence it is not complete. We discuss this at the end of this chapter.

3.2.1 Risk allocation into STNU form

The motivation behind temporal risk allocation is that we cannot guarantee each activity will finish at a specific time or even within some range of times. However, if we assume the outcome of temporal uncertainty will fall in some window, then we have a concrete set of uncertainty that we can control against. We just have to accept the risk that comes with making this assumption. The goal of `InitializeRiskAllocation` is to introduce risk allocation variables, which simultaneously reformulate probabilistic uncertainty into set-bounded uncertainty and imply risks taken at each source of uncertainty. With separate expressions for both, we will have decoupled the temporal and the probabilistic aspects of the pSTN.

Our risk allocation variables define windows $\left[\underline{\omega}_{ij}, \overline{\omega}_{ij}\right]$ on each contingent link $\ell_{ij} \in \mathcal{L}_{\mathrm{ctg}}$. If we assume the actual outcome ω_{ij} falls in this window, then we are taking a risk R_{ij} of falling outside the window

$$R_{ij} = 1 - F_{ij}(\overline{\omega}_{ij}) + F_{ij}(\underline{\omega}_{ij}).$$

This is what line 2 in Algorithm 2 expresses. We represent all contingent links' windows by a risk allocation vector r, which collects all the $\underline{\omega}_{ij}$ and $\overline{\omega}_{ij}$ variables.

If we accept the risk and focus on scheduling, then we have reformulated the pSTN $\mathcal{N}^p$ into an STNU $\overline{\mathcal{N}}^u$ whose contingent links are parameterized in terms of r, as shown in line 6. Whether we can find a schedule for an STNU is a problem of STNU strong controllability. The standard method for determining strong controllability is to reformulate the STNU into an equivalent STN via triangular reductions [11]. Any schedule that satisfies the STN's constraints is also a strongly controllable schedule for the STNU.

An example triangular reduction is shown in Figure 3-4. It simply replaces a contingent link followed by a requirement link with a single requirement link. The replacement link indicates the range of durations between e_1 and e_2 guaranteed to be achievable regardless of the contingent link's outcome. If we apply triangular reductions to our STNU $\overline{\mathcal{N}}^u$ with parameterized contingent links, then we get a parameterized STN $\overline{\mathcal{N}}$, as shown in line 7. Hence, in order to find a schedule, we must solve for r that will make $\overline{\mathcal{N}}$ consistent. This is what `SolveIncrementally` will do.

Lastly, we provide an initialization r^0 to the risk allocation variables r, which will be used as the initial values for the convex solver in `SolveIncrementally`. We initialize each risk window's bounds to both be the mode of the probability density function. When solving, these bounds will move apart to the left and right of the mode, so that R_{ij} is convex in terms of these bounds. Lemma 1 discusses why this is, and it will help us construct convex risk expressions for each chance constraint in `ReformulateChanceConstraints`.

Lemma 1. *The risk of a contingent link, R_{ij}, is convex in terms of $\underline{\omega}_{ij}$ and $\overline{\omega}_{ij}$ over the domain $\underline{\omega}_{ij} \leq \mathsf{mode}$ and $\overline{\omega}_{ij} \geq \mathsf{mode}$.*

Proof. Consider the lower bound $\underline{\omega}_{ij}$ first. The concavity at any point is given by

$$\frac{\partial^2 R_{ij}}{\partial \underline{\omega}_{ij}{}^2} = F''_{ij}(\underline{\omega}_{ij}) = f'_{ij}(\underline{\omega}_{ij}).$$

By Assumption 3 that f_{ij} is semi-concave, because $\underline{\omega}_{ij}$ is less than the mode of f_{ij}, the concavity at $\underline{\omega}_{ij}$ is non-negative. Hence, R_{ij} is convex in terms of $\underline{\omega}_{ij}$.

An analogous argument holds for the upper bound $\overline{\omega}_{ij}$. □

3.2.2 Reduction to convex feasibility

Our ultimate goal is to encode the pSTP's chance constraints in terms of the risk allocation variables. `InitializeRiskAllocation` reformulated the pSTN model structure to where the risk and temporal relationships, $\mathcal{R}_{\text{ctg}}$ and $\overline{\mathcal{N}}$, were explicitly laid

Algorithm 2: `InitializeRiskAllocation`

Input: a pSTN $\mathcal{N}^p$
Output: a set of contingent links' risk expressions $\mathcal{R}_{\text{ctg}}$
Output: a parameterized STN $\overline{\mathcal{N}}$
Output: initial values to the risk allocation variables r^0

1 **foreach** *contingent link* $\ell_{ij} \in \mathcal{L}_{\text{ctg}}$*, linking event* e_i *to event* e_j^u **do**
2 $\quad R_{ij} \leftarrow \left\langle 1 - F_{ij}(\overline{\omega}_{ij}) + F_{ij}(\underline{\omega}_{ij}) \right\rangle$
3 $\quad$ Insert R_{ij} into $\mathcal{R}_{\text{ctg}}$.
4 $\quad \mathsf{mode} \leftarrow$ Find the mode of ω_{ij}'s probability density function f_{ij}.
5 $\quad$ Append $\left\langle \underline{\omega}_{ij}^0 \leftarrow \mathsf{mode} \right\rangle$ and $\left\langle \overline{\omega}_{ij}^0 \leftarrow \mathsf{mode} \right\rangle$ onto r^0.
6 $\overline{\mathcal{N}}^u \leftarrow$ Replace each probabilistic contingent link in $\mathcal{N}^p$ with a set-bounded contingent link $\omega_{ij} \in [\underline{\omega}_{ij}, \overline{\omega}_{ij}]$.
7 $\overline{\mathcal{N}} \leftarrow$ `TriangularReduce(`$\overline{\mathcal{N}}^u$`)`.

out in terms of r. Now, we can express the chance constraints in terms of $\mathcal{R}_{\text{ctg}}$ and $\overline{\mathcal{N}}$, and by extension, r. First, `ReformulateChanceConstraints` rewrites the probabilistic condition of each chance constraint. Then, `SolveIncrementally` inserts temporal constraints as needed into the system of equations to solve.

The main hurdle for `ReformulateChanceConstraints` is to determine which contingent links are relevant to each chance constraint. If a contingent link does not affect satisfaction of a chance constraint, but we factor in its risk anyway, then we are considering irrelevant risk. In the real world, only activities that execute prior to and lead up to either event of a requirement link could affect whether that requirement is satisfied. Hence, our approach identifies the contingent links belonging to such activities it finds.

Algorithm 3 reformulates each chance constraint in terms of the risk allocation variables. For each chance constraint c, first in lines 2 to 5, we find out which contingent links ℓ_{mn} are relevant to each requirement link the chance constraint is defined over. Note that multiple requirement links may share the same contingent link; we do not include the same expression twice. Having identified all the contingent links relevant to c in $\mathcal{L}_c$, we collect their total risk in line 9, requiring that all of them fall within their assumed risk windows. Then we specify in line 10 that this total probability must meet the chance constraint's threshold Δ_c.

To find which contingent links are relevant to the satisfaction of a given requirement link ℓ_{ij}, we consider when the two events e_i and e_j will be executed by on the online strategy t_{x}. If an event e_n has no preceding activities, then it will always execute according to the schedule t_{s}. There is no temporal uncertainty associated with that. However, if e_n ends some activity, then its execution time $t_{\mathrm{x}}(e_n)$ depends on when that activity starts $t_{\mathrm{x}}(e_m)$ and the outcome of its contingent link ω_{mn}. Likewise, if e_m ends some other activity, we can keep tracing back until the beginning of the activity string. Hence, in line 3 of our algorithm, we trace back a tree of activity strings leading up to the event. We include all the contingent links associated with the tree's edges in line 5.

Lastly, having obtained an expression for each chance constraint's risk R_c, we ensure it is convex before sending it to the solver in `SolveIncrementally`. Line 8 does this by ensuring the mode of each contingent link's probability density function is contained within its risk window. By Lemma 1, each R_{mn} that contributes to R_c is convex in terms of its risk window's bounds, and by Lemma 2, R_c must therefore be convex.

Constraining the risk allocation windows to contain the mode places additional restrictions on the types of temporal distributions we can handle. Notably, we may be too conservative with respect to distributions that have modes on their extremities, such as those shown in Figure 3-5. A feasible solution might exist, but only if the risk allocation window is somewhere near the center of the distribution, which our constraints in line 8 would preclude. However, if the chance constraint requirements Δ_c are reasonably high, that means each individual R_{ij} would have to be quite low, and therefore the window on ω_{ij} would have to be considerably wide relative to f_{ij}. In particular, the median is always contained if every $\Delta_c \geq 0.5$. Therefore, if the mode is reasonably close to the median, then it has a good chance of being included in the window if a solution exists. In the real world, it is reasonable to model most temporal distributions with sufficiently long tails on either side, so this restriction should not be too conservative.

Lemma 2. *If each R_{mn} is convex over a domain of possible windows on ω_{mn}, then*

R_c is convex over the joint domain of all contingent links' risk allocation windows.

Proof. The initialization of R_c in line 6 with the iterative inclusion of R_{mn} in line 9 means the final expression for R_c is

$$R_c = 1 - \prod_{\ell_{mn} \in \mathcal{L}^c} (1 - R_{mn}).$$

For convenience, let P_{mn} denote $1 - R_{mn}$, and let P_c denote the entire product expression, which is equivalent to $1 - R_c$. Thus, $P_c = \prod_{\ell_{mn} \in \mathcal{L}^c} P_{mn}$, and we need to show that P_c is concave.

□

Algorithm 3: `ReformulateChanceConstraints`

Input: a set of pSTP chance constraints $\mathcal{C}$
Input: a pSTN $\mathcal{N}^p$
Input: a set of contingent links' risk expressions $\mathcal{R}_{\text{ctg}}$
Output: a set of reformulated chance constraints $\mathcal{C}'$

1 **foreach** *chance constraint* $c \in \mathcal{C}$*, over requirement links* $\mathcal{L}'$ **do**
2 **foreach** *requirement link* $\ell_{ij} \in \mathcal{L}'$*, linking event* e_i *to event* e_j **do**
3 Perform two depth-first traversals, one each from e_i and e_j. Include only parameterized links leading up to either.
4 **foreach** *link traversed, parameterized in terms of* $\underline{\omega}_{mn}$ *and* $\overline{\omega}_{mn}$ **do**
5 Insert the contingent link ℓ_{mn} into $\mathcal{L}_c$. Avoid duplicates.
6 $R_c \leftarrow 0$.
7 **foreach** *contingent link* $\ell_{mn} \in \mathcal{L}_c$ **do**
8 Insert $\langle \underline{\omega}_{mn} \leq \mathsf{mode} \rangle$ and $\langle \overline{\omega}_{mn} \geq \mathsf{mode} \rangle$ into $\mathcal{C}'$.
9 $R_c \leftarrow 1 - (1 - R_c)(1 - R_{mn})$.
10 Insert $\langle 1 - R_c \geq \Delta_c \rangle$ into $\mathcal{C}'$.

3.2.3 Incremental temporal constraint discovery

In the last subsection, `ReformulateChanceConstraints` wrote the probabilistic aspect of each chance constraint in terms of the risk windows on each contingent link. Now, corresponding to the temporal aspect, `SolveIncrementally` must ensure those

windows result in a consistent STN. STN consistency is equivalent to there being no negative cycles in the STN's distance graph [6]. Therefore, if we could extract all cycles from the parameterized STN $\overline{\mathcal{N}}$, expressed in terms of the risk allocation r, then we could enforce them to all be positive, combine these conditions with the reformulated chance constraints, and submit them to the convex solver. A cycle constraint is linear in terms of the risk allocation variables, so it is already convex.

However, it is usually not necessary to consider every cycle. There might be a much smaller subset of cycles such that if we satisfy their constraints, then all other cycle constraints will be satisfied, too. Such a subset would be dependent on the initial values of the risk allocation. To construct such a subset, we could repeatedly solve our constraint system, discovering a negative cycle after each iteration and adding its constraint before the next, until there are no more negative cycles to be discovered. The inclusion of each new cycle constraint would prune out the possibility of violating many other cycles. Hence, we expect there to be few iterations, with a reasonable problem size during each.

This approach is implemented by `SolveIncrementally` in Algorithm 4. We start with the initial r^0 calculated in `InitializeRiskAllocation` and the chance constraints $\mathcal{C}'$ from `ReformulateChanceConstraints`. Passing these through a convex optimizer `SolveConvexSystem` will give us a new risk allocation r that satisfies the chance constraints. Plugging these values into the parameterized STN $\overline{\mathcal{N}}$ gives us a concrete STN $\mathcal{N}$, on which we check for negative cycles. Negative cycles are detected using a single-source shortest path algorithm like Bellman-Ford [6]. Given a negative cycle, we activate the cycle's constraint by deriving its parameterized weight in $\overline{\mathcal{N}}$.

When there are no more negative cycles to be found, that means the current risk allocation creates a consistent STN, and we can return that as our final risk allocation r^f. Otherwise, there are two ways the algorithm could fail. If the convex solver returns no solution, then up to the precision of the solver, we have arrived at a set of mutually inconsistent constraints. Since this is a subset of the entire set of constraints, there is no solution to the original pSTP. The other failure case is if we find an un-parameterized negative cycle, which must be composed only of requirement

links from the original pSTN. This would mean the temporal coordination constraints are inconsistent with themselves, and there is no amount of risk allocation that could resolve that.

Algorithm 4: `SolveIncrementally`

Input: a set of reformulated chance constraints $\mathcal{C}'$
Input: a parameterized STN $\overline{\mathcal{N}}$
Input: an initial risk allocation r^0
Output: a grounded STN $\mathcal{N}$ and a final risk allocation r^f, or no solution

$r \leftarrow$ Initialize risk allocation to r^0.
system $\leftarrow$ Initialize system of equations to $\mathcal{C}'$.
repeat
    $r \leftarrow$ `SolveConvexSystem(`system, r`)`.
    $\mathcal{N} \leftarrow \overline{\mathcal{N}}(r)$.
    cycle $\leftarrow$ Extract negative cycle from `SSSP(`$\mathcal{N}$`)`.
    Insert $\langle$cycle's weight in $\overline{\mathcal{N}} \geq 0\rangle$ into system.
until *no more negative cycles, or* `SolveConvexSystem` *returns* no solution
$r^f \leftarrow r$.

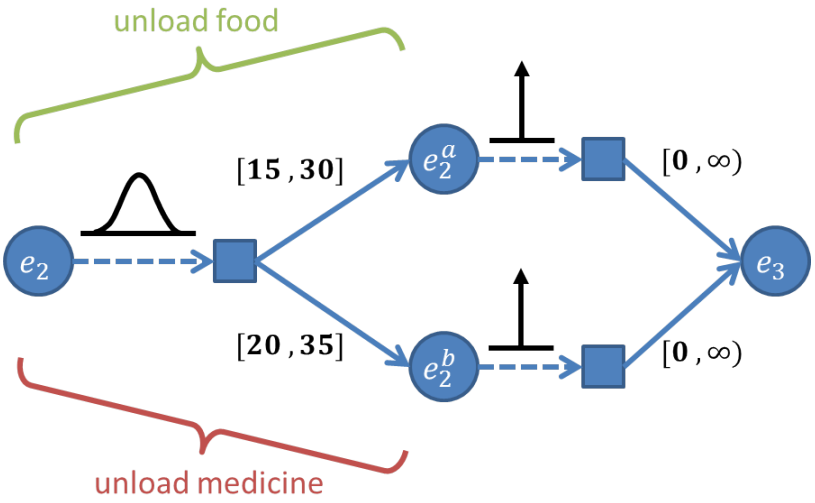

Figure 3-1: Splitting the unloading activity into two sub-activities, then merging them through waiting.

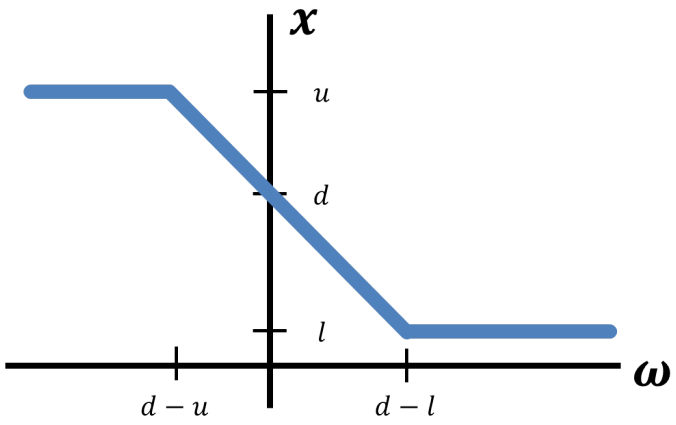

Figure 3-2: How we adapt our control effort in response to observed temporal stochasticity.

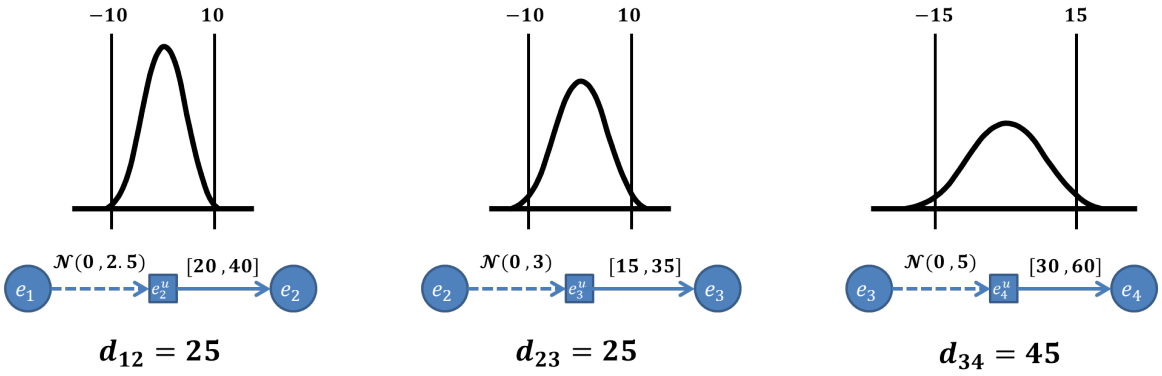

Figure 3-3: Initial risk allocation for the disaster relief scenario.

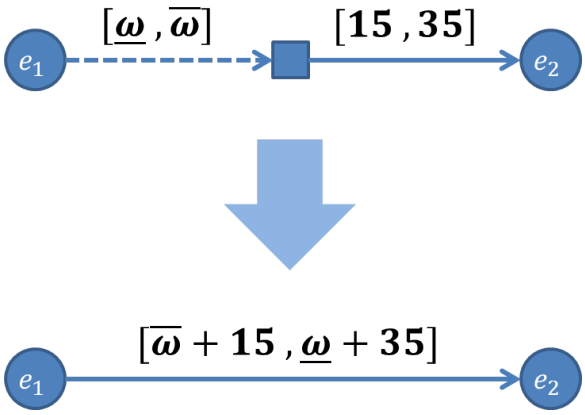

Figure 3-4: Fundamental operation of the triangular reduction. Constraining the duration between e_1 and e_2 to $[u + 15, l + 35]$ guarantees satisfaction of the $[15, 35]$ requirement link prior to reduction.

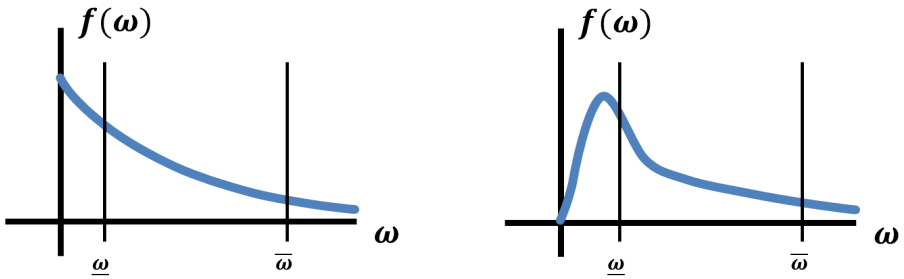

Figure 3-5: Examples of distributions with modes on the extreme left side, which might not lie in the solution's risk allocation window.

Chapter 4

Experimental Validation

4.1 Hypothesis

Our objective is to demonstrate that our algorithm scales tractably as the problem size grows. In particular, we wish to highlight the effect of our incremental generate-and-test approach versus the inefficiencies of solving the full parameterized STN. First, we describe how we generate test cases. Our strategy intends to capture the competing temporal interaction among activities, temporal constraints, and chance constraints. Then, we explain how we vary the parameters of our generated test cases in order to compare the two algorithmic approaches.

4.2 Test-case Generation

A complete test case consists of activities, temporal constraints, and chance constraints. We will consider the case of modeling a single agent performing a sequence of consecutive activities.

An activity has a contingent constraint plus a requirement constraint. We randomly generate an interval inside $[0, 100]$ for the requirement constraint. The contingent constraint we model as a Normal distribution centered at 0. We randomly choose a standard deviation between 0 and 0.1 times the width of the activity's requirement constraint.

To form a temporal constraint, we first randomly pick two events. We want the temporal constraint to actually be constraining on the activity durations between its two events. Hence, over all the activities in between, we first sum all the lower and all the upper bounds of their requirement constraints. The analogy here is assuming no temporal disturbance from the contingent constraints, this gives us a range of the minimum and maximum time needed to physically complete these activities. To simulate the intent that temporal constraints will restrict the durations of activities, we choose the temporal constraint bounds to cover a random one-third. of this feasible execution range.

Lastly, we randomly pick one, two, or three temporal constraints over which to define each chance constraint. The more constraints an operator defines a chance constraint over, the more likely that chance constraint represents as lower priority requirement. In the disaster relief scenario, we were more concerned about getting supplies to the people on time (95%) than with the entire mission's success (90%). For these three cases of one, two, or three temporal constraints, we assign chance constraint requirements of 95%, and 90%, and 80%.

4.3 Experiment Design

To compare how these two approaches scale, we increase the problem size on a variety of factors and measure the average runtime and number of temporal constraints considered. For the iterative approach, total runtime means the time spent across all iterations, whereas for the full STN approach, there is only one iteration. Hence, we expect the iterative approach to have worse total runtime on small examples, because it will require several iterations, whereas the single iteration on the entire STN finished very quickly, too. However, the number of temporal constraints that the iterative approach discovers should always be lower than those encoded by the STN.

There are several parameters on which we may adjust the problem size. Namely, there are the number of activities, the number of temporal constraints, and the number of chance constraints. In addition, we may widen the temporal distribu-

tions, which increases temporal uncertainty and hence makes it harder to satisfy the chance constraints. Changing the number of activities roughly decides the mission length. Given the number of activities, the number of temporal constraints affects how "crowded" the temporal network is. Finally, given the number of temporal constraints, the number of chance constraints shows how interdependent our objectives may be.

If we are to study the effects of varying each parameter, we must establish relative baselines between the numbers of each. These baselines should reflect a typical single-vehicle mission. Nominally, we will create one-fifth the number of temporal constraints as activities, and one-half the number of chance constraints as temporal constraints. When varying the number of activities, we keep these ratios constant. Then, when varying the number of temporal constraints, we keep at constant 300 activities. Likewise, we keep a constant 60 temporal constraints when varying the number of chance constraints. Finally, we retain those above numbers and 30 chance constraints when adjusting the distribution widths.

4.4 Results

Chapter 5

Conclusion

5.1 Contributions

This thesis developed an approach to scheduling under temporal uncertainty in order to meet temporal constraints under acceptable risk. First, we laid out the chance-constrained pSTN scheduling problem. This formulation models temporal uncertainty as probabilistic temporal distribution in order to more accurate represent how uncertain outcomes occur in the real world. It also defines chance constraints over subsets of temporal constraints, which allow mission planners and operators greater freedom to express their priorities.

Then, we offered an algorithm for offline scheduling of a pSTN to satisfy a set of chance constraints. The fundamental insight was to allocate risk in a way that reformulates the pSTN into a parameterized STNU. From that point, we were able to write the reformulation as a set of convex inequalities and linear temporal inequalities. Furthermore, the latter set of inequalities could be equivalently learned through temporal conflict detection. Our benchmarks demonstrated this algorithm is tractable for plans involving upwards of hundreds of activities, largely due to the generate-and-test approach.

Temporal uncertainty is pervasive in all our activities, and it can be difficult to reason about over even in moderately sized temporal networks. However, the problem formulation and solution method presented in this thesis offer a quantitatively

rigorous method to guarantee low-risk satisfaction of temporal constraints through chance-constrained scheduling. Our modeling of temporal plans is applicable to many real-world problems, and hence, so are our scheduling insights.

5.2 Future Work

The ideas developed in this work have potential to be extended as well as integrated with other algorithmic capabilities. We discuss these two aspects here.

5.2.1 Algorithm and Analysis Extensions

First, we would like to extend our algorithm to perform dynamic scheduling online. Our static schedules have no leeway during execution. If we could schedule an activity's end point based on current status and observed history, we would have a higher chance of success. For example, when distributing supplies to the people, we do not know how much help they need until we get there. How much time we decide to spend helping them affects how fast we drive back to the base and therefore how much risk we incur of arriving back late or early. The works by Tsamardinos [15] and Li [10] study dynamic scheduling of probabilistic temporal networks, and they leverage principles from analysis of STNU dynamic controllability. We could incorporate these ideas similarly into our chance-constrained framework.

With regard to algorithm analysis, we are interested in the sensitivity of the schedule's risk to modeling error. One of the main motivations for developing a probabilistic model was to be more robust to temporal uncertainty through more accurate modeling. However, no model is purely captures the physical world, and it becomes increasingly expensive to develop higher fidelity models. Nevertheless, we would expect probabilistic models to be more robust than the interval-bounded representation employed by the STNU. Further experiments could be carried out to determine how far our model's temporal distributions may diverge from the true distribution in order to maintain given sensitivity tolerance.

5.2.2 Integration with Other Capabilities

Currently, our algorithm simply outputs a "yes" or "no" response to whether there exists a schedule. We are all set if "yes," because we receive a corresponding schedule, but a "no" need not be taken as hopeless news. All it means is the problem, as specified, is over-constrained, but it doesn't say by how much. If we couldn't meet a 95% chance constraint, then perhaps 94% would be achievable, or perhaps only 80% is. Clearly, these are two different situations, and we would like to distinguish between them.

This relates to the idea of failing gracefully. If the problem is over-constrained, rather than dropping all objectives, we could drop them in order of priority so that we still achieve as much as possible. Fortunately, our problem formulation and scheduling insights support such a strategy. Our chance constraints already specify priorities, and our constraints are parameterized in terms of temporal and risk bounds. If our algorithm exits with an infeasible solution, then by studying constraints in the vicinity of that solution, we could determine how far to relax the constraints' bounds in order for the solution to be considered feasible. Yu and Williams [18] have pursued such ideas for negotiating with the human operator intent which temporal goals to relax and how much. We could extend this to include relaxations of chance constraints as well.

Lastly, our risk assessment capability has been developed in the context of meeting temporal constraints, but other types of goals and their associated risk exist in the real world. The ultimately capability would be an integrated assessment of richly-expressed mission goals over all sources of risk. Blackmore [2] and Ono [12] have considered the risk of violating physical state constraints, such as colliding with obstacles or missing a goal region. Like temporal risk assessment, vehicle path-planning involves continuous variables. There may also be discrete sources of risk. For example, when driving to the neighborhood in our disaster relief scenario, we may be forced to take a detour if a road is too heavily flooded. Our backup route's temporal distribution will be fundamentally different, so we must evaluate that separately in our

risk assessment. Integrating stochastic sources which affect time, state, and choice poses challenges through mixing tightly coupled continuous and discrete variables. However, the capability would lead to truly deployable decision-making systems that perform rigorous risk assessment .

Bibliography

[1] Hossein Arsham. Managing project activity-duration uncertainties. *Omega*, 21(1):111–122, 1993.

[2] Lars J. C. Blackmore. *Robust Execution for Stochastic Hybrid Systems*. PhD thesis, MIT, September 2007.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex Optiization*. Cambridge University Press, 2004.

[4] Jon Dattorro. *Convex Optimization and Euclidean Distance Geometry*. Meboo Publishing, 2012.

[5] Nashwan Dawood. Estimating project and activity duration: a risk management approach using network analysis. *Construction Management & Economics*, 16(1):41–48, 1998.

[6] Rina Dechter. Constraint processing. Morgan Kaufmann, 2003.

[7] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1):61–95, 1991.

[8] Dimitri Golenko-Ginzburg. A two-level decision-making model for controlling stochastic projects. *International journal of production economics*, 32(1):117–127, 1993.

[9] Hoong Chuin Lau, Jia Li, and Roland H. C. Yap. Robust controllability of temporal constraint networks under uncertainty. In *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*. IEEE, 2006.

[10] Jia Li. Robust controllability of temporal constraint network with uncertainty. Master's thesis, National University of Singapore, 2006.

[11] Paul Morris, Nicola Muscettola, Thierry Vidal, et al. Dynamic control of plans with temporal uncertainty. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 494–502. LAWRENCE ERLBAUM ASSOCIATES LTD, 2001.

[12] Masahiro Ono. *Robust, Goal-directed Plan Execution with Bounded Risk*. PhD thesis, MIT, September 2011.

[13] Matthew J Sobel, Joseph G Szmerekovsky, and Vera Tilson. Scheduling projects with stochastic activity duration to maximize expected net present value. *European Journal of Operational Research*, 198(3):697–705, 2009.

[14] Ioannis Tsamardinos. A probabilistic approach to robust execution of temporal plans with uncertainty. pages 97–108. SETN, 2002.

[15] Ioannis Tsamardinos, Martha E. Pollack, and Sailesh Ramakrishnan. Assessing the probability of legal execution of plans with temporal uncertainty. In *Workshop on Planning under Uncertainty and Incomplete Information*.

[16] Thierry Vidal. Controllability characterization and checking in contingent temporal constraint networks. In *Proceedings of Seventh International Conference on Principles of Knowledge Representation and Reasoning*, 2000.

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

[18] Peng Yu and Brian C. Williams. Diagnosis of temporal planning problems. In *Prepared for 22nd International Conference on Automated Planning and Scheduling*. ICAPS, 2012.