View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by ScholarBank@NUS

ROBUST CONTROLLABILITY OF TEMPORAL CONSTRAINT NETWORK WITH UNCERTAINTY

LI JIA

NATIONAL UNIVERSITY OF SINGAPORE

· 2006 ·

ROBUST CONTROLLABILITY OF TEMPORAL CONSTRAINT NETWORK WITH UNCERTAINTY

LI JIA

(B.Sc.(Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF COMPUTER

SCIENCE DEPARTMENT OF SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

· 2006 ·

Acknowledgements

I would like to express my gratitude to all those who helped me and supported me to complete this one-year journey. I want to thank Dr Lau Hoong Chuin and Dr Roland Yap for guiding me towards the correct direction, and giving me inspirational ideas along the way. I would also thank Dr Melvyn Sim and Thomas Ou for their valuable comments. Lastly but not less importantly, I would like to give heartfelt thanks to my dear friends who gave me every bit of joy and encouragement for the past one whole year.

Contents

Summary

Temporal constraint networks with uncertainty are embedded in many scheduling problems. The fundamental problem is to decide whether such network can be executed under different uncertainty scenarios. Few works in the literature raise the question of probabilistic dynamic execution. In this thesis, we propose the Robust Temporal Constraint Network (RTCN) model where durations of uncertain activities are represented by random variables. We wish to know the Robust Controllability problem whether such a network can be executed dynamically with failure probability less than or equals to a given $0 \leq \epsilon \leq 1$. If so, how one might find a feasible schedule on the fly as the uncertainty variables are revealed dynamically. We present a computationally tractable and efficient approach to solve the RTCN controllability problem. Experimentally, we will examine how the failure probability ϵ is affected by several properties of RTCN, and how the failure probability of robust controllability differs from that of a weaker form of controllability. We will also propose some enhancements to improve the result.

Keywords:

Uncertainty, Temporal Constraint Network, Robust Optimization

Implementation Software and Hardware:

Pentium 4 3.0GHz, 1GB DDR2 RAM, Windows XP, MatLab 7, MOSEK 3

List of Figures

Chapter 1

Introduction

Reasoning about temporal constraint is a very important topic in Artificial Intelligence. Several formalisms and corresponding algorithms have been created to model temporal constraints in a planning system. Their applications could be found in areas like project scheduling [1], diagnosis and temporal database [2], multimedia authoring systems [3] and more. *Simple Temporal Network* (STN) is one of such formalism where relations between events are expressed in numerical constraints with the form $l \leq T_x - T_y \leq u$. T_x and T_y represent start event and end event of an activity. l and u are positive real numbers. This constraint specifies that the time the activity takes is no less than l units and no more than u units. Every activity of STN is specified with such an interval $[l, u]$. Figure.1.1 is an example of STN where there are 17 edges, each of which represents an activity and is associated with an interval, and 10 nodes, each of which represents the start or end event of an

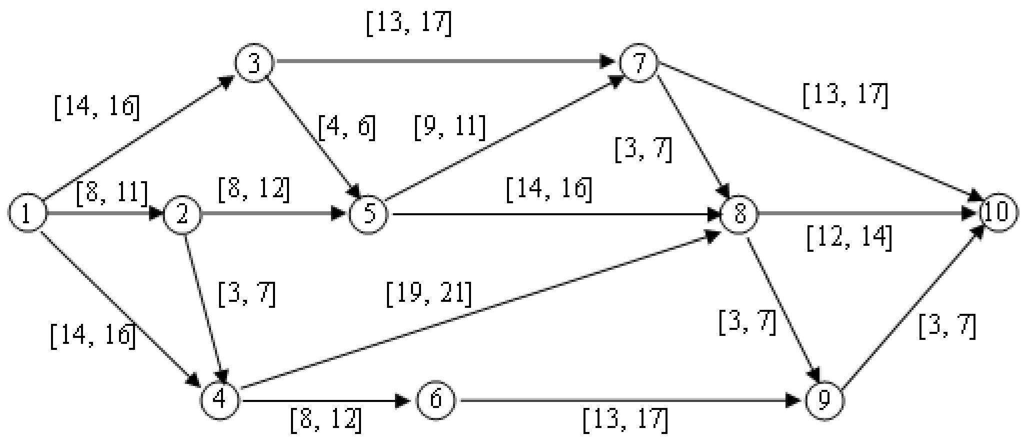

Figure 1.1: An example of Simple Temporal Network

activity. Formalism like STN is able to answer the plan consistency problem. A plan is consistent if there exists an assignment to all the timepoints of the STN such that no constraint is violated. Thus STN helps temporal system designers to construct valid plans.

Although studies on STN have been fruitful, it is still limited in representing some realistic event types. The STN formalism is assuming all events are controllable by the system designer and thus he could assign any desired value to the events to execute the plan. However, in real-world domain, some events are uncontrollable by the system designer. Instead, their time is decided outside the planning system such as by Nature. For example, suppose we have a shipment activity. The end event of this activity is the time the shipment arrives. The system designer is unable to decide the exact time as it is subject to status of the Nature like weather condition. *Simple Temporal Network with Uncertainty* (STNU) is coined to overcome this drawback. This new formalism is able to model two kinds of activities. The first kind is controllable activities as in STN, and the second kind is contingent activities whose durations are uncertain.

STNU plays a much more significant role than STN. In many applications the durations of contingent activities are only known after their occurrence but decision must be made when to execute the set of executable timepoints. This uncertainty in the timing implicitly creates many possible scenarios and plan of this kind will be executed without knowledge of the exact scenario in advance. For this family of applications, system designers have to execute the plan dynamically. The STNU is said to be *dynamically controllable* if it can be executed incrementally as the duration of contingent activities are revealed gradually. The formal definition will be introduced in later part, but essentially a dynamically controllable network must survive all possible scenarios. This requirement is too conservative and could be hard to meet in practice. *This concern in conservativeness motivates us to define a new level of controllability which could both inherit the property of dynamic controllability to describe the dynamic execution and meanwhile measures the level of conservativeness.*

The new controllability we propose in this thesis is called *robust controllability*. It is associated with a given value ϵ which we call *failure probability*. A network with uncertainty is robust controllable if it can be executed dynamically with probability at least $1 - \epsilon$. We will model the contingent activities with some probability distribution. The resulted network model is given the new name *Robust Temporal Constraint Network* (RTCN) upon which robust controllability will be formally defined. The resulted problem is a stochastic problem which traditional method is inefficient to solve. We will apply a recent and promising technique named *robust optimization approach* to solve the robust Controllability problem. Furthermore, the technique offers the flexibility to adjust the degree of conservativeness in guaranteeing controllability.

This report is organized as follows: Beginning of Chapter 2 gives background knowledge on types of temporal network models where we emphasize the model STNU, in Addition, Chapter 2 introduces three levels of controllabilities defined on STNU as well as their respective complexity. Chapter 3 explains Robust Optimization technique which is related to solving the problem of this thesis. Chapter 4 presents results on solving Dynamic and Weak controllabilities with Affine Linear Policy. Chapter 5 is the main part of our thesis where we define the Robust Controllability problem, present a solution approach, and give some enhancements. Chapter 6 gives experimental results, and we conclude this report with Chapter 7.

Chapter 2

Background on Temporal Networks

This chapter will give necessary background knowledge on Temporal Networks. The first section will introduce two categories of Temporal Network models namely *Qualitative Temporal Network* and *Quantitative Temporal Network*. The network models discussed in this thesis (STN, STNU and RTCN coming later) all belong to the latter category. The second section and third section will define STNU and various controllabilities respectively. STNU provides the context to define three levels of controllabilities, one of which is the basis to propose our robust controllability problem.

2.1 Models of Temporal Network

Temporal Networks have been widely used to solve problems in many fields [1, 2, 3]. These problems commonly involve a set of activities and certain temporal relations in between. Temporal Network models are created to systematically study temporal problems. Categorized by the representation, we have two kinds of network models: *qualitative* temporal network model and *quantitative* temporal network model.

2.1.1 Qualitative Temporal Networks

For problems where relations between activities are described by chronological ordering not in numerical form, qualitative temporal network is sufficient for the representation. There are two kinds of qualitative temporal networks represented in Interval Algebra (IA) [5] and Point Algebra (PA) [6] respectively. In IA based model, the basic elements are *intervals* and constraints are the relations of paired intervals. This model allows disjunctive relations which is chosen from the set {*before, meets, overlaps, starts, during, finishes, equal*}. In PA based model, basic elements are *timepoints* and constraints are the ordering of paired timepoints. Each constraint is one element from the set $\{=, >, <, \leq, \geq \neq\}$. Both models can be described as a constraint graph, and thus classic constraint propagation algorithms such as path consistency can be applied to check graph consistency. However, qualitative networks are limited in its power to deal with metric information. Quantitative networks overcome this deficiency and provide a convenient formalism to deal with numerical information like the durations of events. Next section will introduce quantitative network models.

2.1.2 Quantitative Temporal Networks

Different from qualitative counterpart, quantitative temporal network, as its name suggests, models problems where relations between activities are quantified in numbers. The concept of quantitative temporal network is first proposed by Rina [4] and has the name *Temporal Constraint Satisfaction Problem* (TCSP) . In this framework, variables represent timepoints and temporal information are represented by a set of unary and binary constraints.

Definition 1. *A Temporal Constraint Satisfaction Problem (TCSP) involves a set of variables, $X_1, \ldots, X_n$, having continuous domains; each variable represents a timepoint. Each Constraint is represented by a set of intervals $\{I_1, \ldots, I_k\} = \{[a_1, b_1], \ldots, [a_k, b_k]\}$.*
A unary constraint T_i restricts the domain of variable x_i to the given set of intervals; that is, it represents the disjunction $(a_1 \leq x_1 \leq b_1) \vee \cdots \vee (a_k \leq x_k \leq b_k)$.
A binary constraint T_{ij} constraints the permissible values for the distance $x_j - x_i$; it represents the disjunction $(a_1 \leq x_j - x_i \leq b_1) \vee \cdots \vee (a_k \leq x_j - x_i \leq b_k)$.

TCSP is very expressive not only because it enables metric information but also allows disjunctive relations among time points. However, the advantage in expressiveness is at the cost of high computational cost. For example checking consistency for TCSP is NP-hard. A restricted yet reasonably expressive and computationally tractable version of TCSP is Simple Temporal Network(STN) [1] presented in the same paper [4].

STN is a subclass of TCSP where disjunctive relations do not exist. In such a network, each edge is labelled by an interval $[a_{ij}, b_{ij}]$, which represents the constraint $a_{ij} \leq X_j - X_i \leq b_{ij}$. Thus solving consistency problem of STN amounts to solving a set of linear inequalities possibly with well-known mathematical methods. Alternatively, the structure of STN problem allows more efficient polynomial algorithm which is based on the property that STN is consistent if and only if no negative cycle is contained in the distance graph which can be derived from the original.

2.2 Simple Temporal Network with Uncertainty

Although STN is capable of modelling certain temporal problems, Vidal and Fragier [12] found the deficiency of STN in representing some realistic applications. A STN is consistent as long as there is a set of valid assignment to

[1]It is called Simple Temporal Problem in the original paper, and Simple Temporal Network in some other papers. We adopt the latter name to emphasize it is a network formalism and to keep name consistency with names of other formalisms in the report

all the timepoints. This requires system designer to have full-control of all the events. However, in many real applications, the duration of some event is out of control of the system designers. For example, a system requires when a signal indicates emergency, one must deal with the dangerous situation within 20 minutes. One can control the process of treating emergency but he himself cannot control the happening of emergency. Different from the set of controllable edges, duration of certain kind of edges is decided by "external world". We call this set of edges as *contingent* edges whose durations can only be measured after execution. To distinguish from contingent edges, we give the term *controllable* edges to the rest edges whose durations are controlled by system designer. Simple Temporal Network with Uncertainty (STNU) [12] is invented to deal with contingent activities.

Definition 2. *The Simple Temporal Network with Uncertainty (STNU) is a connected and acyclic graph* Γ*. It is labelled by a four-tuple* $\Gamma = \langle \mathcal{V}_{\mathcal{X}}, \mathcal{V}_{\mathcal{O}}, \mathcal{E}_{\mathcal{R}}, \mathcal{E}_{\mathcal{T}}, \rangle$ *where*

$\mathcal{V}_{\mathcal{X}}$*: Set of nodes representing executable timepoints*

$\mathcal{V}_{\mathcal{O}}$*: Set of nodes representing observable timepoints*

$\mathcal{E}_{\mathcal{R}}$*: Set of controllable edges. The duration of each edge must fall between a given interval.*

$\mathcal{E}_{\mathcal{T}}$*: Set of contingent edges. The duration of each edge is a uncertain value, and can be any number in a given interval.*

Although both controllable and contingent edge are associated with a given interval, they differ in the way that system designer could choose the assignment of a controllable edge while he cannot choose the assignment of a contingent edge. The nodes are related to the edges in the way that each contingent edge is ended with an observable timepoint. Since the duration of contingent edge is controlled outside the planning system, the time of observable timepoint is uncertain but we assume it could be observed and the planning system knows the valid range of the duration of the contingent edge. Without loss of generality, we require no two contingent edges in the graph to share the same observable timepoint (or else the network is apparently inconsistent in too many situations) so contingent edge and observable timepoint are in one-to-one correspondence.

2.3 Controllabilities and Their Complexities

This section is going to bring up the definition and detailed illustration of *controllabilities*. Although controllability is originally defined with respect to the model STNU [12], it applies to any temporal network with uncertainty models as long as their definitions comply with that of STNU.

Refer to Definition 2., as the duration of contingent edges can take any value from a continuous interval and we assume the durations of contingent edges vary independently, there are infinitely many combinations of durations to consider. If we call the network assigned with each such combination an instance, there will be infinitely many instances to consider in terms of network consistency. Thus consistency is redefined as *Controlla-*

bility in temporal networks with uncertainty. A network is controllable if each instance is consistent and the property of the consistent solution for each instance determines the level of controllability. There are three levels of controllability, namely *Strong*, *Weak*, and *Dynamic* controllability. Each level of controllability plays a role in its respective application area. Before giving the definition of respective controllability, several terms are explained first.

A *scenario* ω is an assignment of valid durations to all the contingent edges.

A *current scenario* $\omega_{\prec t}$ is an assignment of observed durations to all the contingent edges occurred up to current time t.

A *future scenario* $\omega_{\succ t}$ is an assignment of possible durations to all the contingent edges not yet observed at time t.

An *instance* γ_ω of a STNU is a temporal network constructed from the original STNU by assigning each contingent edge with the corresponding duration in ω.

A sequence $S = \{s_i : i \in \mathcal{V}_\mathcal{X}\}$ where s_i is an assignment to executable timepoint i is a *solution* of instance γ_ω if it satisfies all the temporal constraints of γ_ω.

A *partial assignment* $S_{\prec t} = \{s_i : i \in \mathcal{V}_\mathcal{X} \cap i \prec t\}$ at time t is an assignment to all the executable timepoints occurred up to time t.

Definition 3. *The three levels of controllabilities are defined as:*

A STNU is ***strongly controllable*** *iff* $\exists$ *a single sequence* S *such that* $\forall$

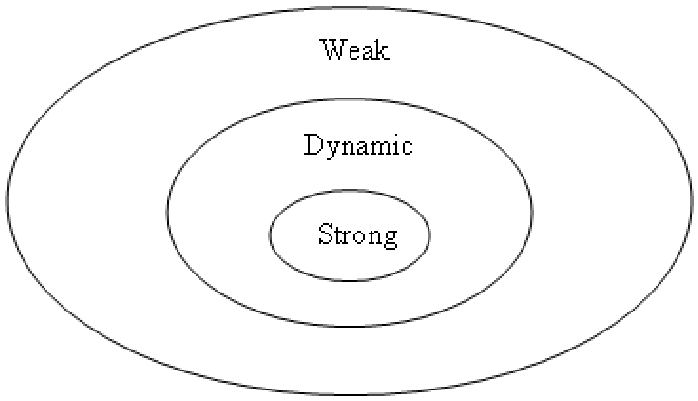

Figure 2.1: Relation among Strong, Dynamic and Weak Controllabilities

scenario ω, S is a solution of instance γ_ω.

A STNU is ***weakly controllable*** *iff $\forall$ scenario ω, $\exists$ a sequence $S(\omega)$ such that $S(\omega)$ is a solution of instance γ_ω.*

A STNU is ***dynamically controllable*** *iff at any arbitrary time t, $\forall$ future scenario $\omega_{\succ t}$, the current partial assignment $S_{\prec t}$ can be extended to a full sequence S such that S is a solution for instance $\gamma_{(\omega_{\prec t} \cup \omega_{\succ t})}$.*

The relation between three levels of controllability is that strong controllability implies dynamic controllability and dynamic controllability implies weak controllability. Thus the set of strongly controllable networks is included in the set of the dynamically controllable networks which in turn is included in the set of weakly controllable networks described by a Venn Diagram 2.1, Strong controllability is the most rigorous, it is relevant in applications where contingent events can not be observed exactly or a full solution must be fixed beforehand. Weak controllability is least demanding, it is relevant in applications where contingent events are observable before execution and there is time to calculate a corresponding solution beforehand. In many realistic applications however, contingent events are only observable after it happens because such events are controlled by external world as in the definition. If the network is strongly controllable with a solution S, the agent can use S to execute the plan no matter how long the contingent events take. But for more generous purpose, these applications need an execution policy to produce a growing solution according to the subsequent revelation of contingent events. In a word, dynamic controllability is a desirable property to describe whether a STNU could be executed incrementally as the uncertainty is revealed along time. This property is most useful and is the one this thesis focus on.

Vidal and Fargier have discussed the complexity of checking strong controllability and weak controllability [12]. Following result is quoted from the paper stated.

Property 1. *Checking strong controllability is polynomial.*

Sketch of proof: The problem of deciding strong controllability of a STNU can be represented by means of a classical STN such that the STNU is strongly controllable iff the STN is consistent in the classical sense. The idea is to consider the relation between tasks in the worst case, assuming that a contingent duration $d_i \in [l_i, u_i]$ is equal to u_i in any constraint of the form $x - y > d_i$ and equal to l_i in any constraint of the form $x - y < d_i$. Hence, since determining whether a STN is consistent is a polynomial problem, so is it for deciding strong controllability.

Conjecture 1. *Checking weak controllability is Co-Np-complete*

Proof: The Co-problem of checking weak controllability is: is there a scenario ω such that γ_ω is an inconsistent instance? Since checking that a STN is inconsistent is a polynomial problem, this co-problem belongs to NP. Hence, weak controllability belongs to Co-Np. The difficulty of the problem (Co-Np-complete) remains to be proven.

Besides above conjecture, they propose the following property:

Property 2. *a STNU is weakly controllable iff for any* $\omega^{bnd} \in \{l_1, u_1\} \times \ldots \times \{l_G, u_G\}, \gamma_{\omega^{bnd}}$ *is a consistent temporal network.*

This property suggests an enumeration algorithm in which all the temporal networks constructed by combination of boundary values are checked. The complexity is exponential to the number of contingent edges. Alternatively, a tree searching algorithm with search space pruning simplifies the average complexity. A new branch is created as we choose a contingent edge to assign lower bound value and upper bound value. At each tree node, some procedures will check consistency of the current partial tree to decide whether stop searching. Because if one subtree fails the set of corresponding network instances fail. The detailed algorithm is described in [12].

Dynamic controllability is previously believed to be complicated, however a recent work by Morris and Muscettola [9] proposed a polynomial method which is a constraint satisfaction algorithm. It utilizes a ternary structure to tighten the intervals on each edge until either inconsistency rises or no more intervals to be tightened which indicates dynamic controllability. The set of intervals in the latter case is a minimum form of network equivalent to the original network. The execution of a dynamically controllable network will be based on the reduced network produced by their constraint satisfaction algorithm.

2.4 Probabilistic Controllability

Tsamardinos [7] first proposes the problem of probabilistic execution of temporal plans with uncertainty. This first work deals with the probabilistic counterpart of strong controllability. It is regarding the probability a single solution survives all possible execution. A more interesting work is regarding probabilistic counterpart of dynamic controllability [8]. In that paper, the authors deal with two problems.

The first problem is to access the probability of a plan being legally executed under an execution policy by bounding the probability from above and below. The upper bound can be used to reject a plan which falls below a given threshold, while a lower bound can be used to accept a plan if it is satisfiable and computation of lower bound also gives a guidance on how to execute the plan although it is not dynamically controllable. The authors propose two methods namely binary search for loosest bounds and iterative tightening to solve this problem. But both of them are heuristic methods, and the authors have list flaws associated with the two methods in their paper.

The second problem they deal with is to maximize the probability of satisfying all the execution constraints given that the contingent durations follow uniform distribution. Their method is based on Morris' algorithm [9] but they impose restrictions on the original algorithm under a particular condition, thus in a strict sense they are maximizing a lower bound probability. Despite of the deficiency of the solutions, the authors provide insights in both reasoning and methodology regarding probabilistic treatment of temporal networks.

Chapter 3

Robust Optimization In Stochastic Programming

3.1 Robust Optimization Technique

Probably the first work to bring Robust Optimization to light is work by Soyster [16] in 1973, however, it is not until the 90s have people brought up interest to this research area again. The classical way to deal with optimization problems usually assumes deterministic data. For example, in a shortest path problem the weight of each edge is a deterministic value. However, in real life, the data of many applications is not known exactly when a decision needs to be made. Robust Optimization is essentially used to hedge decision against data uncertainty. Most influential works in this area are from the following researchers: Bel-Tal & Nemirovski [17, 18, 19] , Kouvelis & Yu [20], El-Ghaoui [24, 25] and Bertsimas & Sim [21, 22, 23]. The common goal of these people's works is to find a reliable solution that remains safe when there are data perturbations. They differ in the way to model data uncertainty and solve the problem. For example, data presented in Kouvelis & Yu's work [20] is modelled as discrete scenarios, while most of the other works consider uncertainty data set in the form of ellipsoidal or intersection of ellipsoidal sets.

Two important issues in Robust Optimization are *conservativeness* and *computational tractability*. On one hand, a robust solution is acceptable for many possible data realizations even for the worst-case instance; on the other hand, some realizations might be so improbable that dwelling on them could result in unnecessarily pessimistic decisions. In other words, a too conservative solution could often deteriorate the objective value. This is why the degree of conservativeness is an important issue. Works by Bel-Tal & Nemirovski [17, 18, 19] and Bertsimas & Sim [21, 22, 23] address the control of conservativeness by providing an adjustable probability guarantee of the feasibility of the solution subject to the size of the uncertainty data set $\mathcal{U}$. The bigger the size of $\mathcal{U}$, the more scenarios are taken care of, thus the safer is the solution. Meanwhile, due to the large-scale nature of optimization problems, the time complexity of the robust counterpart is generally worse than that of the nominal problem. It is possible that under certain data assumptions, a polynomially solvable problem (such as Minimum Spanning Tree) has a NP-hard robust counterpart (see Kouvelis & Yu's book [20]).

Under Bel-Tal & Nemirovski's model [19], the robust counterpart of a Linear Program with ellipsoidal uncertain set becomes a continuous Conic Quadratic Problem which is polynomially solvable. One of Bertsimas & Sim's work [21, 22] could even preserve the computational tractability of the nominal problem.

3.2 Applying Robust Optimization To Solve Stochastic Problem

We first consider an uncertain linear constraint,

$$\tilde{\boldsymbol{a}}'\boldsymbol{x} \geq \tilde{\boldsymbol{b}} \tag{3.1}$$

in which the vectors $\tilde{\boldsymbol{a}}$ and $\tilde{\boldsymbol{b}}'$ are subjected to random perturbation as follows,

$$(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}}) = (a^0, b^0) + \sum_{j=1}^{N} (\Delta a^j, \Delta b^j)\tilde{z}_j$$

where (a^0, b^0) are the nominal value of the data, $(\Delta a^j, \Delta b^j), j \in \{1 : N\}$ is the direction of data perturbation. $\tilde{z}_j$ is the primitive uncertainty which has mean zero and support in $[-\underline{z}_j, \overline{z}_j], \underline{z}_j, \overline{z}_j > 0$.

We are interested in finding feasible solution $X(\epsilon)$ such that $x \in X(\epsilon)$ is feasible to the linear constraint 3.1 with probability at least $1 - \epsilon$,

$$X(\epsilon) = \{x : P(\tilde{\boldsymbol{a}}'\boldsymbol{x} \geq \tilde{\boldsymbol{b}}) \geq 1 - \epsilon\}, \tag{3.2}$$

The parameter ϵ varies the conservatism of the solution. The traditional way to solve this kind of linear stochastic problem often requires the use of non-linear optimization which can be computationally challenging. [14] proposes to solve a tractable "robust counterpart" problem to obtain the robust feasible solution as follows,

$$X_r(\Omega) = \{x : \tilde{\boldsymbol{a}}'\boldsymbol{x} \geq \tilde{\boldsymbol{b}}\ \forall(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}}) \in \mathcal{U}_\Omega\} \tag{3.3}$$

where $\mathcal{U}_\Omega$ is a compact uncertainty set and the parameter Ω, referred to as the budget of uncertainty, varies the size of the uncertainty set radially from the central point $\mathcal{U}_{\Omega=0} = (a^0, b^0)$ to $\mathcal{U}_{\Omega_w}$ which is the smallest uncertainty set satisfying $P((\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}}) \in \mathcal{U}_{\Omega_w}) = 1$. If we consider the worst case budget Ω_w, it is equivalent to consider full range of uncertain data $(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}})$ which leads to a very conservative approximation of $X(\epsilon)$. To derive a less conservative approximation, one needs an appropriate choice of the budget of uncertainty Ω.

The design of uncertainty set $\mathcal{U}_\Omega$ depends on knowledge of probability distribution. [14] introduces forward deviation and backward deviation which takes into account asymmetric data distribution to model the primitive uncertainty variables. The uncertainty set is designed as,

$$\begin{aligned}\mathcal{U}_\Omega = \{(a,b) : \exists v, w \in \mathcal{R}^N, (a,b) = (a^0, b^0) + \textstyle\sum_{j=1}^{N} (\Delta a^j, \Delta b^j)(v_j - w_j),\\ \|\mathcal{P}^{-1}v + \mathcal{Q}^{-1}w\| \le \Omega, -\underline{z} \le v - w \le \overline{z}, w, v \ge 0\}\end{aligned} \tag{3.4}$$

where $\mathcal{P} = diag(p_1, \ldots, p_N), \mathcal{Q} = diag(q_1, \ldots, q_N)$ with $p_j, q_j > 0, j \in 1 : N$. Here p_j is the forward deviation for each primitive random variable z_j, and q_j is the backward deviation for each primitive random variable z_j. The definition of forward and backward deviation will be explained later. With this uncertainty data settting, the "robust counterpart" of 3.3 is proved to be equivalent to,

$$\left\{\begin{array}{l} \exists \boldsymbol{u}, \boldsymbol{r}, \boldsymbol{s} \in \Re^N, h \in \Re \\ \ a^{0'}x + \Omega h + \boldsymbol{r}'\overline{\boldsymbol{z}} + \boldsymbol{s}'\underline{\boldsymbol{z}} \le b^0 \\ \ \|\boldsymbol{u}\| \le h \\ \ u_j \ge p_j(\Delta a^{j'}x - \Delta b^j - r_j + s_j) \ \ \forall j \in \{1, \ldots, N\}, \\ \ u_j \ge -q_j(\Delta a^{j'}x - \Delta b^j - r_j + s_j) \ \ \forall j \in \{1, \ldots, N\}, \\ \ \boldsymbol{u}, \boldsymbol{r}, \boldsymbol{s} \ge \boldsymbol{0}. \end{array}\right\} \tag{3.5}$$

which is a Conic Quadratic Program (CQP) solvable in polynomial time. Furthermore, if there exists a solution, then it satisfies the condition

$$P(\tilde{a}'x \le \tilde{b}) \ge 1 - \epsilon$$

We remains to explain the definition of forward and backward deviation.

Forward and Backward Deviation, p. q. Let $\tilde{z}$ be a random variable with zero mean and $M_{\tilde{z}}(s) = \mathrm{E}(\exp(s\tilde{z}))$ be its moment generating function. They define the set of values associated with forward deviations of $\tilde{z}$ as follows,

$$\mathcal{P}(\tilde{z}) = \left\{\alpha \ : \ \alpha \geq 0, M_{\tilde{z}}\left(\frac{\phi}{\alpha}\right) \leq \exp\left(\frac{\phi^2}{2}\right), \forall\phi \geq 0\right\}. \tag{3.6}$$

Likewise, for backward deviations, they define the following set,

$$\mathcal{Q}(\tilde{z}) = \left\{\alpha \ : \ \alpha \geq 0, M_{\tilde{z}}\left(\frac{-\phi}{\alpha}\right) \leq \exp\left(\frac{\phi^2}{2}\right), \forall\phi \geq 0\right\}. \tag{3.7}$$

Furthermore $\mathcal{P}(c) = \mathcal{Q}(c) = \Re_+$ for any constant c.

For known distributions, we define the forward deviation of $\tilde{z}$ as $p^*_{\tilde{z}} = \inf \mathcal{P}(\tilde{z})$ and the backward deviation as $q^*_{\tilde{z}} = \inf \mathcal{Q}(\tilde{z})$. [14] show that these values are no less than the standard deviation. Interestingly, under normal distribution, these values coincide with the standard deviation.

Although for most distributions we may not be able to obtain close form solutions of p^* and q^*, we can still approve their values numerically through simulation method provided by the authors. For instance, if $\tilde{z}$ is uniformly distributed over $[-1, 1]$, we can determine numerically that $p = q = 0.58$ (This thesis will give a proof in later part that p^* and q^* equals to standard deviation in case of uniform distribution).

Chapter 4

Affine Linear Policy for STNU

It is already mentioned in Section 2.3 that there is an existing algorithm to check dynamic controllability using constraint satisfaction algorithm, and checking weak controllability remains a difficult problem due to the conjecture on its complexity. Here we are going to propose an *Affine Linear* policy to check dynamic and weak controllability uniformly and using mathematical programming technique provides a basis to study robust controllability later.

To ease the explanation, this chapter will be based on a concrete example of STNU. In Fig.4.1, there are five edges and five timepoints in this network. Among them, two are contingent edges represented by dotted lines. Timepoint $1, 3, 5$ are executable timepoints whose values are to be determined. Suppose the starting time (the value of timepoints 1) is always 0. Let t_3, t_5 be decision variables representing the value of executable timepoints $3, 5$ respectively. Let z_1, z_2 be random variables representing the duration of the

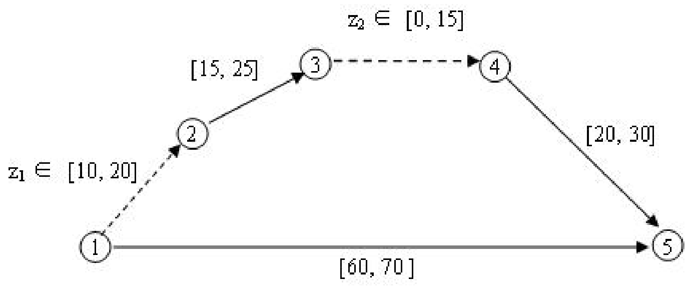

Figure 4.1: An example of Temporal Network with Uncertainty

two contingent edges ended with timepoint $2, 4$ respectively. Then we have the expressions for the two observable timepoints: $t_2 = z_1$ and $t_4 = t_3 + z_2$. To obtain an controllable network, we want to ensure for every network instance the constraints on the three controllable edges $2 \rightarrow 3, 4 \rightarrow 5, 1 \rightarrow 5$ are satisfied, which could be described by an inequality system 4.1. Before we illustrate the *Affine Linear* policy on dynamic and weak controllability with this example, we show this network is not strongly controllable.

$$\begin{cases} 10 \leq t_3 - t_2 \leq 20 \\ 20 \leq t_5 - t_4 \leq 30 \\ 55 \leq t_5 \leq 65 \end{cases} \tag{4.1}$$

Strong controllability requires a fixed assignment to the set of executable timepoints to fit all possible network instances created by combining different values of contingent edges. Inequalities (4.2) is formulated directly according to the definition, inequalities (4.3) is obtained by replacing $t_2 = z_1$ and

$t_4 = t_3 + z_2$. Inequalities (4.4) and (4.5) are subsequent reductions of (4.3).

$$\exists t_3, t_5 \quad \forall z_1, z_2 \in [10,20] \times [0,15] \quad \left\{ \begin{array}{lcll} 10 & \leq & t_3 - t_2 & \leq & 20 \\ 20 & \leq & t_5 - t_4 & \leq & 30 \\ 55 & \leq & t_5 & \leq & 65 \end{array} \right. \tag{4.2}$$

$\Rightarrow$

$$\exists t_3, t_5 \quad \forall z_1, z_2 \in [10,20] \times [0,15] \quad \left\{ \begin{array}{lclll} 15 & \leq & t_3 - z_1 & \leq & 25 \\ 20 & \leq & t_5 - t_3 - z_2 & \leq & 30 \\ 60 & \leq & t_5 & \leq & 70 \end{array} \right. \tag{4.3}$$

$\Rightarrow$

$$\exists t_3, t_4 \quad \left\{ \begin{array}{lcl} min\{t_3 - z_1\} & \geq & 15 \\ max\{t_3 - z_1\} & \leq & 25 \\ min\{t_5 - t_3 - z_2\} & \geq & 20 \\ max\{t_5 - t_3 - z_2\} & \leq & 30 \\ t_5 & \geq & 60 \\ t_5 & \leq & 70 \end{array} \right. \tag{4.4}$$

$\Rightarrow$

$$\exists t_3, t_4 \quad \begin{cases} t_3 - 20 & \geq \; 15 \\ t_3 - 10 & \leq \; 25 \\ t_5 - t_3 - 15 & \geq \; 20 \\ t_5 - t_3 & \leq \; 30 \\ t_5 & \geq \; 60 \\ t_5 & \leq \; 70 \end{cases} \tag{4.5}$$

The above inequality system has an empty solution space. Thus we conclude that the network is not strongly controllable.

4.1 Affine Linear Policy To Solve Dynamic Controllability

Dynamic controllability says at any time, a current partial solution can always be extended to a full solution no matter what is the uncertain situation yet to come. What's more, the partial solution depends only on the values of previous contingent edges but not on those after. This implies for each executable timepoint v_i, there is actually a set of information the decision when to execute v_i should depend on. We call this set a dependency set D_i for v_i, which consists of the set of random variables related to previous contingent edges with respect to v_i. For example, $D_3 = \{z_1\}, D_5 = \{z_1, z_2\}$.

Having a dependency set for each executable timepoint is not enough to execute it. System designers need guidance to determine when to execute each point v_i. We will call this guidance the *policy* throughout this report. A policy can be expressed in terms of a function whose input are random variables indicating durations of previous contingent edges with respect to v_i and whose output is the time of v_i. More formally, we have

$$f_i : D_i \rightarrow t_i \tag{4.6}$$

For example, we could regard t_3 and t_5 as functions $f_3(z_1)$ and $f_5(z_1, z_2)$ respectively. The formulation taking into consideration the constraints in 4.1 is as following:

$$\forall z_1, z_2 \quad \begin{cases} 15 \leq f_3(z_1) - z_1 \leq 25 \\ 20 \leq f_5(z_1, z_2) - f_3(z_1) - z_2 \leq 30 \\ 60 \leq f_5(z_1, z_2) \leq 70 \end{cases} \tag{4.7}$$

The following part will illustrate how to solve dynamic controllability of our example step by step. But before that, there is a need to discuss how to obtain $D(v_i)$ and very importantly, to discuss a reasonable representation of policy used in this thesis.

4.1.1 Determine the Dependency set

The dependency set in the example is explicitly obtained by hand. For more complicated networks, we need a systematic way to find the dependency set for each executable timepoint. This essentially requires us to know which are the observable timepoints trigged before a given executable timepoint as the information of those timepoints trigged after is not available yet to decide the assignment of the executable timepoint. This requirement needs the knowledge of the relation between every pair of observable and executable timepoints. However, since not every observable timepoint and executable timepoint are linked in the original network, the relation between them is absent. It is desirable to derive these relations from the already knowns. Furthermore, we want the lower and upper bound to be as tight as possible in order to obtain a more accurate relation. There is one algorithm to fit our purpose well, which is the one used by Dechter [4] to determine the minimum network. Although their algorithm applies to STN and is used for different purpose, the result of the algorithm is to obtain the tightest relation for every pair of timepoints based on the original interval of each edge and is what we need.

Recall that a STN is a graph where each edge (i, j) is labelled by an interval $[l_{ij}, u_{ij}]$. In Dechter's work [4], they associate an STN $G = (V, E)$ with a directed edge-weighted graph $G_d = (V, E_d)$, called distance graph. It has the same node set as G, and each edge, (i, j) is labelled by a weight a_{ij}, representing the inequality $t_j - t_i \leq a_{ij}$. For each edge (i, j) with $[l_{ij}, u_{ij}]$ in G, the corresponding edges in G_d is edge (i, j) labelled with u_{ij} and edge (j, i) labelled with $-l_{ij}$ (The weight of an edge in distance graph could thus be negative). Because $l_{ij} \leq t_j - t_i \leq u_{ij}$ is equivalent to $t_j - t_i \leq u_{ij}$ and $t_i - t_j \leq -l_{ij}$, G and G_d are equivalent fundamentally. Now suppose we run an all-pair shortest-path algorithm on the distance graph G_d and let d_{ij} denote the shortest distance for each edge (i, j). Then an important corollary given by Dechter [4] is: given a consistent STN, G, the equivalent STN, M, each of whose edge (i, j) is associated with $[-d_{ji}, d_{ij}]$ is the minimal network representation of G. A minimum network is the most economical representation of original STN whose intervals can not be tightened any more.

After running this algorithm, we associate each pair of observable timepoint v_i and executable timepoint v_j with an interval $[u, w]$ where $u = -d_{ji}$ and $w = d_{ij}$. The value of u and w could be of any sign because the weight of an edge in the distance graph could be either positive or negative. Thus if we examine the relation $u \leq v_j - v_i \leq w$, we have following cases,

(1)If $u \geq 0$, we have $v_j - v_i \geq 0$. It means observable timepoint v_i never happens after executable timepoint v_j. Thus $z_i \in D_j$.

(2)If $w \leq 0$, we have $v_j - v_i \leq 0$. It means executable timepoint v_j never happens after observable timepoint v_i. Thus $z_i \notin D_j$.

(3)If $u < 0 \cap w > 0$, the chronological ordering of v_i and v_j is not clear. We assume $z_i \notin D_j$ for this case.

4.1.2 Determine the Execution Policy

We have formulated the dynamic controllability problem of our example, but we have not determined the function representing the policy described as 4.6. Unfortunately, finding such policy is generally difficult. The size of the policy space is potentially infinite as the underlying distribution of its parameter (input random variable of the function) is continuous. Hence, to make computation efficient, one needs to place further restriction on the class of policies. Here we restrict the policy $f_i(D_i)$ to be affine dependent on the elements of set D_i as shown in equation 4.8.

$$t_i = f_i(D_i) = x_0 + \sum_i x_i \tilde{z}_i \quad \tilde{z}_i \in D_i \tag{4.8}$$

In above expression, x_i denotes the decision variables in our model. Note that although the assumption of affine linear function is somewhat restrictive, finding a feasible policy within this space is still a non-trivial problem.

4.1.3 Deterministic Dynamic Execution

According to our Affine Linear policy, we have the expression $t_3 = x_1 + x_2 \cdot z_1$ and $t_5 = x_3 + x_4 \cdot z_1 + x_5 \cdot z_2$, replace them into the formula 4.7, we have:

$$\forall z_1, z_2 \quad \begin{cases} 15 \leq x_1 + (x_2 - 1) \cdot z_1 & \leq 25 \\ 20 \leq x_3 - x_1 + (x_4 - x_2) \cdot z_1 + (x_5 - 1) \cdot z_2 & \leq 30 \\ 60 \leq x_3 + x_4 \cdot z_1 + x_5 \cdot z_2 & \leq 70 \end{cases} \tag{4.9}$$

The above inequality system requires each constraint to be consistent for any z_1, z_2 value. Actually, as long as each constraint is consistent for all combinations of upper and lower bound values of the random variables, the problem 4.9 is consistent too. However, trying all combinations could potentially suffer from exponential complexity. Next we will apply a series of transformation to solve above inequalities efficiently.

Let $\bar{z}_i$ denotes the median of each z_i, and Δz_i denotes the difference between boundary value and mean of each z_i. So $\bar{z}_i = (\check{z}_i + \hat{z}_i)/2$ and $\Delta z_i = \hat{z}_i - \bar{z}_i$. The transformation of previous inequality system is:

$$\forall z_1, z_2$$

$$\begin{cases} 15 \leq x_1 + (x_2 - 1) \cdot \bar{z_1} + (x_2 - 1) \cdot (z_1 - \bar{z_1}) \\ x_1 + (x_2 - 1) \cdot \bar{z_1} + (x_2 - 1) \cdot (z_1 - \bar{z_1}) \leq 25 \\ 20 \leq x_3 - x_1 + (x_4 - x_2) \cdot \bar{z_1} + (x_5 - 1) \cdot \bar{z_2} + (x_4 - x_2) \cdot (z_1 - \bar{z_1}) + (x_5 - 1) \cdot (z_2 - \bar{z_2}) \\ x_3 - x_1 + (x_4 - x_2) \cdot \bar{z_1} + (x_5 - 1) \cdot \bar{z_2} + (x_4 - x_2) \cdot (z_1 - \bar{z_1}) + (x_5 - 1) \cdot (z_2 - \bar{z_2}) \leq 30 \\ 60 \leq x_3 + x_4 \cdot \bar{z_1} + x_5 \cdot \bar{z_2} + x_4 \cdot (z_1 - \bar{z_1}) + x_5 \cdot (z_2 - \bar{z_2}) \\ x_3 + x_4 \cdot \bar{z_1} + x_5 \cdot \bar{z_2} + x_4 \cdot (z_1 - \bar{z_1}) + x_5 \cdot (z_2 - \bar{z_2}) \leq 70 \end{cases} \tag{4.10}$$

$\Rightarrow$

$\forall z_1 z_2$

$$\begin{cases} min\{x_1 + (x_2-1)\cdot\bar{z_1} + (x_2-1)\cdot(z_1-\bar{z_1})\} \geq 15 \\ max\{x_1 + (x_2-1)\cdot\bar{z_1} + (x_2-1)\cdot(z_1-\bar{z_1})\} \leq 25 \\ min\{x_3 - x_1 + (x_4-x_2)\cdot\bar{z_1} + (x_5-1)\cdot z_2 + (x_4-x_2)\cdot(z_1-\bar{z_1}) + (x_5-1)\cdot(z_2-\bar{z_2})\} \geq 20 \\ max\{x_3 - x_1 + (x_4-x_2)\cdot\bar{z_1} + (x_5-1)\cdot z_2 + (x_4-x_2)\cdot(z_1-\bar{z_1}) + (x_5-1)\cdot(z_2-\bar{z_2})\} \leq 30 \\ min\{x_3 + x_4\cdot\bar{z_1} + x_5\cdot\bar{z_2} + x_4\cdot(z_1-\bar{z_1}) + x_5\cdot(z_2-\bar{z_2})\} \geq 60 \\ max\{x_3 + x_4\cdot\bar{z_1} + x_5\cdot\bar{z_2} + x_4\cdot(z_1-\bar{z_1}) + x_5\cdot(z_2-\bar{z_2})\} \leq 70 \end{cases} \tag{4.11}$$

$\Rightarrow$

$$\begin{cases} x_1 + (x_2-1)\cdot\bar{z_1} - |x_2-1|\cdot\Delta z_1 & \geq & 15 \\ x_1 + (x_2-1)\cdot\bar{z_1} + |x_2-1|\cdot\Delta z_1 & \leq & 25 \\ (x_3-x_1) + (x_4-x_2)\cdot\bar{z_1} + (x_5-1)\cdot\bar{z_2} - |x_4-x_2|\cdot\Delta z_1 - |x_5-1|\cdot\Delta z_2 & \geq & 20 \\ (x_3-x_1) + (x_4-x_2)\cdot\bar{z_1} + (x_5-1)\cdot\bar{z_2} + |x_4-x_2|\cdot\Delta z_1 + |x_5-1|\cdot\Delta z_2 & \leq & 30 \\ x_3 + x_4\cdot\bar{z_1} + x_5\cdot\bar{z_2} - |x_4|\cdot\Delta z_1 - |x_5|\cdot\Delta z_2 & \geq & 60 \\ x_3 + x_4\cdot\bar{z_1} + x_5\cdot\bar{z_2} + |x_4|\cdot\Delta z_1 + |x_5|\cdot\Delta z_2 & \leq & 70 \end{cases} \tag{4.12}$$

Replacing $\bar{z}_i, \Delta z_i$ with numerical value, we get:

$$\begin{cases} x_1 + 15 \cdot (x_2 - 1) - 5 \cdot |x_2 - 1| & \geq \ 15 \\ x_1 + 15 \cdot (x_2 - 1) + 5 \cdot |x_2 - 1| & \leq \ 25 \\ (x_3 - x_1) + 15 \cdot (x_4 - x_2) + 7.5 \cdot (x_5 - 1) - 5 \cdot |x_4 - x_2| - 7.5 \cdot |x_5 - 1| & \geq \ 20 \\ (x_3 - x_1) + 15 \cdot (x_4 - x_2) + 7.5 \cdot (x_5 - 1) + 5 \cdot |x_4 - x_2| + 7.5 \cdot |x_5 - 1| & \leq \ 30 \\ x_3 + 15 \cdot x_4 + 7.5 \cdot x_5 - 5 \cdot |x_4| - 7.5 \cdot |x_5| & \geq \ 60 \\ x_3 + 15 \cdot x_4 + 7.5 \cdot x_5 + 5 \cdot |x_4| + 7.5 \cdot |x_5| & \leq \ 70 \end{cases} \tag{4.13}$$

The above is a linear inequality system with absolute terms. It can be observed that the absolute terms could all be moved to left-hand side of $\leq$, this enables a transformation to pure linear inequalities. Take $|A|+|B| \leq c$ as an example, we could add two more variables x, y to form the new inequalities

$$\begin{cases} x + y \leq c \\ -x \leq A \leq x \\ -y \leq B \leq y \end{cases} \tag{4.14}$$

Applying this technique, we can transform (4.13) to following:

$$\begin{cases} x_1 + 15 \cdot (x_2 - 1) - 5 \cdot y_1 & \geq \quad 15 \\ x_1 + 15 \cdot (x_2 - 1) + 5 \cdot y_1 & \leq \quad 25 \\ (x_3 - x_1) + 15 \cdot (x_4 - x_2) + 7.5 \cdot (x_5 - 1) - 5 \cdot y_2 - 7.5 \cdot y_3 & \geq \quad 20 \\ (x_3 - x_1) + 15 \cdot (x_4 - x_2) + 7.5 \cdot (x_5 - 1) + 5 \cdot y_2 + 7.5 \cdot y_3 & \leq \quad 30 \\ x_3 + 15 \cdot x_4 + 7.5 \cdot x_5 - 5 \cdot y_4 - 7.5 \cdot y_5 & \geq \quad 60 \\ x_3 + 15 \cdot x_4 + 7.5 \cdot x_5 + 5 \cdot y_4 + 7.5 \cdot y_5 & \leq \quad 70 \\ -y_1 \leq x_2 - 1 \leq y_1 & \\ -y_2 \leq x_4 - x_2 \leq y_2 & \\ -y_3 \leq x_5 - 1 \leq y_3 & \\ -y_4 \leq x_4 \leq y_4 & \\ -y_5 \leq x_5 \leq y_5 & \end{cases} \tag{4.15}$$

The above system can now be solved by any LP solver. It turns out to be consistent which means the network is dynamically controllable. Recall that in previous section, we concluded the network is not strongly controllable. This example approves strong controllability is a stricter criterion than dynamic controllability. One possible solution is as following,

$$\begin{cases} x_1 = 25 \\ x_2 = \frac{1}{2} \\ x_3 = 57\frac{1}{2} \\ x_4 = \frac{1}{4} \\ x_5 = \frac{1}{2} \end{cases} \tag{4.16}$$

For this method, we have essentially replaced absolute terms with new variables and add two inequalities per absolute term to the inequality system. The number of variables in original formula and that of the transformed formula are both in order of $O(|\mathcal{V}_{\mathcal{X}}| \cdot |\mathcal{V}_{\mathcal{O}}|)$, while the number of constraints are in order of $O(|\mathcal{E}_R|)$ and $O(|\mathcal{V}_{\mathcal{O}}| \cdot |\mathcal{E}_R|)$ respectively. With the fact that the number of observable timepoints are practically small, our transformation does not degenerate the computation complexity largely.

4.2 Affine Linear Policy To Solve Weak Controllability

Weak controllability says for each possible instance of temporal network there exists a consistent assignment. The formulation of weak controllability is very similar to that of strong controllability, except the position of universal and existential quantifiers. We already know $t_2 = z_1$ and $t_4 = t_3 + z_2$, the formulation is:

$$\forall z_1, z_2 \in [10, 20] \times [0, 15] \quad \exists t_3, t_5 \quad \begin{cases} 15 \ \leq \ t_3 - z_1 \ \leq \ 25 \\ 20 \ \leq \ t_5 - t_3 - z_2 \ \leq \ 30 \\ 60 \ \leq \ t_5 \ \leq \ 70 \end{cases} \tag{4.17}$$

Property 2 mentioned in Section 2.3 suggests inequality system (4.17) is equivalent to the following:

$$\forall z_1, z_2 \in \{10, 20\} \times \{0, 15\} \quad \exists t_3, t_5 \quad \begin{cases} 15 \ \leq \ t_3 - z_1 \ \leq \ 25 \\ 20 \ \leq \ t_5 - t_3 - z_2 \ \leq \ 30 \\ 60 \ \leq \ t_5 \ \leq \ 70 \end{cases} \tag{4.18}$$

This can be done by solving $2^2 = 4$ independent instances of inequality systems. Each system may have its own solution to t_3, t_5, but unless all the inequality systems have nonempty solutions, the network is not weakly controllable. Notice that solving weak controllability by above inequality system has exponential complexity. We could simplify the problem if we assume some dependency between time of executable timepoints and the duration of contingent edges. Similar to the affine linear dependency discussed in previous section on dynamic controllability, we could assume $t_3 = x_1 + x_2 * z_1 + x_3 * z_2$ and $t_5 = x_4 + x_5 * z_1 + x_6 * z_2$, then the formulation 4.18 becomes 4.19. Note that each timepoint depends on all contingent edges instead of only those in its dependency set for dynamic controllability case.

$$\forall z_1, z_2 \quad \begin{cases} 15 & \leq & x_1 + (x_2 - 1) \cdot z_1 + x_3 \cdot z_2 & \leq & 25 \\ 20 & \leq & x_4 - x_1 + (x_5 - x_2) \cdot z_1 + (x_6 - x_3 - 1) \cdot z_2 & \leq & 30 \\ 60 & \leq & x_4 + x_5 \cdot z_1 + x_6 \cdot z_2 & \leq & 70 \end{cases} \tag{4.19}$$

To solve this inequality system, we would apply a series of transformation to convert it to a linear program. The transformation is very similar to the one we applied in previous section to solve dynamic controllability, thus it is omitted. One possible solution is shown below. The example is weakly controllable as predicted because it is already known to be dynamically controllable.

$$\begin{cases} x_1 = 25 \\ x_2 = \frac{1}{2} \\ x_3 = 0 \\ x_4 = 55 \\ x_5 = \frac{1}{2} \\ x_6 = \frac{1}{3} \end{cases} \tag{4.20}$$

Besides the successful usage in this chapter, affine linear policy will provide a suitable formulation to obtain solutions for robust controllability to be presented in the next chapter. We will further compare the behavior of weak controllability and dynamic controllability with Affine Linear policy in probabilistic context.

Chapter 5

Robust Controllability

Although dynamic controllability is a desirable property to designers of temporal system, it is too conservative and it provides no further information when a network is not dynamically controllable. In practice, system designers may rather accept a high quality network which is not dynamically controllable but survives most of the possible scenarios, or in our terminology, is consistent in most instances. Here the quality of the network can be measured with a probability $\mathrm{P}(\xi|\mathrm{t}(\xi) \in \mathrm{sol}(\xi), \xi \in \Xi)$ which is the probability of successfully executed instances. Here Ξ is the set of all possible instances; $t(\xi)$ is the solution produced by execution policy t under instance ξ; $sol(\xi)$ is the set of all valid solutions under instance ξ; $t(\xi) \in sol(\xi)$ says the policy executes successfully for the given instance.

In this thesis, the duration of each contingent edge will be modelled by a random variable $\tilde{z}_i$ with some probability distribution and bounded by $[l_i, u_i]$.

We make the assumption that $\tilde{z}_i$ is independent from each other.

5.1 Robust Temporal Constraint Network

The following part will formally define our problem started by describing the network model Robust Controllability is defined on.

5.1.1 RTCN Model

Definition 4. *A Robust Temporal Constraint Network is represented by a 5-tuple* $\mathcal{R} = \langle \mathcal{V}_{\mathcal{X}}, \mathcal{V}_{\mathcal{O}}, \mathcal{E}_{\mathcal{R}}, \mathcal{E}_{\mathcal{T}}, \epsilon \rangle$

$\mathcal{V}_{\mathcal{X}}$*: Set of nodes representing executable timepoints*

$\mathcal{V}_{\mathcal{O}}$*: Set of nodes representing observable timepoints*

$\mathcal{E}_{\mathcal{R}}$*: Set of controllable edges where each edge represents the following temporal constraint:*

$$l_{ij} \leq V_j - V_i \leq u_{ij}, \quad \forall (i,j) \in \mathcal{E}_{\mathcal{R}} \tag{5.1}$$

$\mathcal{E}_{\mathcal{T}}$*: Set of contingent edges whose durations are represented by the following function:*

$$V_j - V_i = \tilde{d}_{ij}, \quad \forall (i,j) \in \mathcal{E}_{\mathcal{T}}, V_j \in \mathcal{V}_{\mathcal{O}} \tag{5.2}$$

ϵ*: failure probability of temporal constraint violation*

Our proposed model is a kind of temporal network with uncertainty as defined in beginning of Chapter 3. Each contingent edge is ended with an observable timepoint, whose value is not decided by the system designer.

The set of end nodes of each contingent edge constitutes the set of observable timepoints. While the set of all other nodes forms the executable timepoints and their times must be determined by the system designer. Note that each observable timepoint is associated with a unique contingent edge, since two contingent edges ending with the same observable timepoint will always cause network inconsistency. We model the duration of each contingent edge (i, j) with a random variable $\tilde{d}_{ij}$. It is assumed that we know the mean and variance values of these random variables, but not the actual probability distributions. Without loss of generality, we specify $\tilde{d}_{ij} = t_{ij}(1 + \tilde{z}_{ij})$ where t_{ij} is a constant and $\tilde{z}_{ij}$ is a primitive random variable with zero mean and bounded by $[-1, 1]$.

5.1.2 Robust Controllability

This section will define Robust Controllability with respect to RTCN model just described. As mentioned above, each contingent edge is associated with a random variable which takes value from a bounded interval, therefore there are infinitely many scenarios due to the variety on contingent edges. As Mentioned in Section 2.3, an *instance* refers to a particular STN where every random variable is instantiated to a fixed value. A *schedule* refers to an assignment of the executable timepoints $\{V_1 = t_1, \ldots, V_n = t_n\}$. As mentioned in Section 4.1, an *execution policy* specifies how a schedule can be incrementally generated over time as the durations of contingent edges become known. More specifically, it outputs the decision of the time to execute each $V_i \in \mathcal{V}_{\mathcal{X}}$

according to information on known contingent edges. We denote the decision for each V_i as $V_i(\tilde{\boldsymbol{z}})$.

Definition 5. *A RTCN $\mathcal{R}$ is said to be robust controllable iff at any time, the probability an execution policy produces a schedule satisfying all the constraints in $\mathcal{E}_{\mathcal{R}}$ is at least $1 - \epsilon$.*

The RTCN Feasibility problem is defined as: given an RTCN, determine whether it is robust controllable. This problem is equivalently stated as the problem of finding a *robust policy* such that a feasible schedule can be derived from the policy with probability $1 - \epsilon$ over the set of all network instances. The RTCN Optimization problem is to find a policy that optimizes a given objective function while preserving robust controllability. In this paper, we focus on the feasibility problem.

5.1.3 Measurement of ϵ

Note that the RTCN problem is a core problem underlying many real-world planning and scheduling problems. For example, in robust scheduling, one is concerned with minimizing the total makespan of the machines where job arrivals or processing times are uncertain. The value of ϵ defines the robustness of the RTCN in the sense that the smaller the value of ϵ the more robust the network will be against uncertain temporal constraint violation. However, certain RTCN may not be robust controllable when its constraints are too tight. In such cases, system designers may decide to take more risk of temporal constraint violation by increasing ϵ to obtain a robust controllable RTCN.

5.2 Solving Robust Controllability Problem with Affine Linear Policy

Previous section has defined the Robust Controllability problem, the math formulation will be given in this chapter as well as how to solve the problem with an efficient technique from a published paper [14]. This section will begin with a framework to solve Robust Controllability problem followed by detailed explanations including how the original formulation is transformed to an efficient solvable counterpart and resolving the parameters needed in the solution process.

5.2.1 Framework of Algorithm

A framework of algorithm is given below:

Step 1 Decide the tightest bound $[l_i, u_i]$ of every edge in the network. This is determined by propagating bounds of adjacent edges. Algorithm to apply is a modified all-pair shortest-path algorithm.

Step 2 Decide the dependency set for each executable timepoint t_i. Formulate the affine linear expression for t_i based on the dependency set.

Step 3 Set up the inequality system based on the constraint of each controllable edge.

Step 4 Decide separate chance constraint ϵ_i, forward and backward deviation for each random variable.

Step 5 Apply robust optimization approach to transform the above inequality system to its counterpart and solve it. If there exists a solution the plan is satisfiable, otherwise we drop the plan.

Step 1 is used for two purposes. On one hand, a tightened interval for each edge is obtained. All values outside the interval will fail the plan. On another hand, it provides a basis to find the set of contingent edges each executable timepoint depends on in step 2. Step 2 is already explained in Section 4.1.1. Step 3 is straightforward. Step 4 is to set up additional parameters used to solve the problem from already known parameters. Step 5 is the most important step. Here we adopt work by Chen [14] to solve the problem formulated by step 3. As explained in Section 3.2, their method transforms original problem to a robust counterpart which is efficient to solve. The next section will first give the formulation of the original problem as well as its counterpart.

5.2.2 Applying Robust Optimization Approach to Solve Our Problem

In the RTCN Feasibility problem, we are interested to know whether the probability that a temporal constraint network being controllable is at least $1-\epsilon$ over all instances. Recall we have an Affine Linear policy to determine the time of each executable timepoint i which we denote as $V_i(\tilde{z})$. The network is consistent if the constraint on every controllable edge is satisfied:

$$l_{ij} \leq V_j(\tilde{\boldsymbol{z}}) - V_i(\tilde{\boldsymbol{z}}) \leq u_{ij} \quad \forall (i,j) \in \mathcal{E} \tag{5.3}$$

The robust feasibility problem is equivalent to determine whether there exists a policy $V(\tilde{\boldsymbol{z}})$ such that all constraints are satisfied:

$$\mathrm{P}\left(\left\{\begin{array}{c} V_j(\tilde{\boldsymbol{z}}) - V_i(\tilde{\boldsymbol{z}}) \geq l_{ij} \\ \forall (i,j) \in \mathcal{E} \\ V_j(\tilde{\boldsymbol{z}}) - V_i(\tilde{\boldsymbol{z}}) \leq u_{ij} \\ \forall (i,j) \in \mathcal{E} \end{array}\right\}\right) \geq 1-\epsilon. \tag{5.4}$$

We can view ϵ as the upper bound probability that at least one of the temporal constraints specified in 5.3 is violated. In other words, it defines the desired level of conservativeness we allow for temporal constraint violations. Note that when $\epsilon = 0$, the set of feasible schedules are the ones which satisfy 5.3 under the worst case scenario. As we increases ϵ, we are in fact increasing the size of the set of feasible robust schedules.

Notice that equation 5.4 is a *probabilistic constraint problem*. Particularly it is in joint chance constraint form. But solving a joint chance constraint program in above form by traditional stochastic programming method is rather complicated which involves non-linear optimization. Instead we recourse to the technique proposed by Chen et. al [14] which fits our model well. Throughout this report, we associate the symbol $\acute{}$ with any variables related to lower bound, and $\grave{}$ with any variables related to upper bound

Proposition 1. *For any $\acute{\epsilon_{ij}}$ and $\grave{\epsilon_{ij}}$, $\forall(i,j) \in \mathcal{E}_R$ such that*

$$\sum_{(i,j)\in\mathcal{E}_R} \acute{\epsilon_{ij}} + \sum_{(i,j)\in\mathcal{E}_R} \grave{\epsilon_{ij}} \leq \epsilon \tag{5.5}$$

if there exists measurable functions $V_i(\tilde{z})$ for every node i satisfying

$$\mathrm{P}(\mathrm{V_j}(\tilde{\boldsymbol{z}}) - \mathrm{V_i}(\tilde{\boldsymbol{z}}) \geq \mathrm{l_{ij}}) \geq 1 - \acute{\epsilon_{ij}} \quad \forall(\mathrm{i,j}) \in \mathcal{E}_\mathrm{R}$$
$$\mathrm{P}(\mathrm{V_j}(\tilde{\boldsymbol{z}}) - \mathrm{V_i}(\tilde{\boldsymbol{z}}) \leq \mathrm{u_{ij}}) \geq 1 - \grave{\epsilon_{ij}} \quad \forall(\mathrm{i,j}) \in \mathcal{E}_\mathrm{R}$$

the probability the RTCN is dynamically controllable is at least $1 - \epsilon$.

Proof: For any realization z of $\tilde{z}$, the policy produces a consistent solution if and only if the following is satisfied:

$$V_j(\tilde{\boldsymbol{z}}) - V_i(\tilde{\boldsymbol{z}}) \geq l_{ij} \quad \forall(i,j) \in \mathcal{E}_R$$
$$V_j(\tilde{\boldsymbol{z}}) - V_i(\tilde{\boldsymbol{z}}) \leq u_{ij} \quad \forall(i,j) \in \mathcal{E}_R$$

we have

P (RTCN is dynamically controllable)

$= \mathrm{P}\ (\bigcap_{\forall(\mathrm{i,j})\in\mathcal{E}_\mathrm{R}} \ (\mathrm{V_j}(\tilde{\boldsymbol{z}}) - \mathrm{V_i}(\tilde{\boldsymbol{z}}) \geq \mathrm{l_{ij}} \ \cap \ \mathrm{V_j}(\tilde{\boldsymbol{z}}) - \mathrm{V_i}(\tilde{\boldsymbol{z}}) \leq \mathrm{u_{ij}})\ )$

$$= 1 - \mathrm{P}\left(\bigcup_{\forall(\mathrm{i,j})\in\mathcal{E}_\mathrm{R}} \quad (\mathrm{V_j}(\tilde{\boldsymbol{z}}) - \mathrm{V_i}(\tilde{\boldsymbol{z}}) < \mathrm{l_{ij}} \quad \cup \quad \mathrm{V_j}(\tilde{\boldsymbol{z}}) - \mathrm{V_i}(\tilde{\boldsymbol{z}}) > \mathrm{u_{ij}}) \right)$$
$$\geq 1 - \left(\sum_{\forall(i,j)\in\mathcal{E}_R} \mathrm{P}\left(\mathrm{V_j}(\tilde{\boldsymbol{z}}) - \mathrm{V_i}(\tilde{\boldsymbol{z}}) < \mathrm{l_{ij}}\right) + \sum_{\forall(i,j)\in\mathcal{E}_R} \mathrm{P}\left(\mathrm{V_j}(\tilde{\boldsymbol{z}}) - \mathrm{V_i}(\tilde{\boldsymbol{z}}) > \mathrm{u_{ij}}\right)\right)$$
$$\geq 1 - \left(\sum_{\forall(i,j)\in\mathcal{E}_R} \acute{\epsilon_{ij}} + \sum_{\forall(i,j)\in\mathcal{E}_R} \grave{\epsilon_{ij}}\right)$$
$$\geq 1 - \epsilon$$

The first $\geq$ follows from *Bonferroni's Inequality.* By assigning each constraint a probability bound, we are actually solving a separate constraint program to approximate joint chance constraint program.

Expression 3.5 has presented the robust counterpart of a linear constraint. For a given RTCN with $|\mathcal{E}_\mathcal{R}|$ number of controllable edges, there will be $2|\mathcal{E}_\mathcal{R}|$ linear constraints, each of which is converted to a set of inequalities as specified in 3.5. For a given edge (i, j), we choose $\acute{\Omega}_{ij} = \sqrt{-2\ln\acute{\epsilon}_{ij}}$ as the uncertainty budget for the linear constraint associated with its lower bound, and $\grave{\Omega}_{ij} = \sqrt{-2\ln\grave{\epsilon}_{ij}}$ as the uncertainty budget for its upper bound, then a sufficient condition for RTCN feasibility is to find whether there exists a linear decision rule, parameterized by

$$\boldsymbol{V}^0, \ldots, \boldsymbol{V}^N,$$

that satisfy the following

$$\left\{\begin{array}{l}
\boldsymbol{V}^0, \ldots, \boldsymbol{V}^N \ \ : \\
\exists\ \acute{h}, \grave{h} \in \Re^{|\mathcal{E}_R|}, \acute{v}^{kl}, \grave{v}^{ij}, \acute{r}^{ij}, \grave{r}^{ij}, \acute{s}^{ij}, \grave{s}^{ij} \in \Re^{|\mathcal{E}_\mathcal{T}|} \ \ \forall (i,j) \in \mathcal{E}_\mathcal{R} \\
\ V_j^0 - V_i^0 + \acute{\Omega}_{ij} * \acute{h}_{ij} + \boldsymbol{\acute{r}}^{ij\prime} \bar{\boldsymbol{z}} + \boldsymbol{\acute{s}}^{ij\prime} \underline{\boldsymbol{z}} \leq -l_{ij} \\
\qquad \forall (i,j) \in \mathcal{E}_\mathcal{R} \\
\ \acute{v}_{kl}^{ij} \geq p_{kl}(V_j^{kl} - V_i^{kl} - \acute{r}_{kl}^{ij} + \acute{s}_{kl}^{ij}) \\
\qquad \forall (k,l) \in \mathcal{E}_\mathcal{T}, (i,j) \in \mathcal{E}_\mathcal{R} \\
\ \acute{v}_{kl}^{ij} \geq -q_{kl}(V_j^{kl} - V_i^{kl} - \acute{r}_{kl}^{ij} + \acute{s}_{kl}^{ij}) \\
\qquad \forall (k,l) \in \mathcal{E}_\mathcal{T}, (i,j) \in \mathcal{E}_\mathcal{R} \\
\ \|\boldsymbol{\acute{v}}^{ij}\|_2 \leq \acute{h}_{ij} \\
\qquad \forall (i,j) \in \mathcal{E}_\mathcal{R} \\
\ V_j^0 - V_i^0 + \grave{\Omega}_{ij} * \grave{h}_{ij} + \boldsymbol{\grave{r}}^{ij\prime} \bar{\boldsymbol{z}} + \boldsymbol{\grave{s}}^{ij\prime} \underline{\boldsymbol{z}} \leq u_{ij} \\
\qquad \forall (i,j) \in \mathcal{E}_\mathcal{R} \\
\ \grave{v}_{kl}^{ij} \geq p_{kl}(V_j^{kl} - V_i^{kl} - \grave{r}_{kl}^{ij} + \grave{s}_{kl}^{ij}) \\
\qquad \forall (k,l) \in \mathcal{E}_\mathcal{T}, (i,j) \in \mathcal{E}_\mathcal{R} \\
\ \grave{v}_{kl}^{ij} \geq -q_{kl}(V_j^{kl} - V_i^{kl} - \grave{r}_{kl}^{ij} + \grave{s}_{kl}^{ij}) \\
\qquad \forall (k,l) \in \mathcal{E}_\mathcal{T}, (i,j) \in \mathcal{E}_\mathcal{R} \\
\ \|\boldsymbol{\grave{v}}^{ij}\|_2 \leq \grave{h}_{ij} \\
\qquad \forall (i,j) \in \mathcal{E}_\mathcal{R} \\
\ \boldsymbol{\acute{r}}^{ij}, \boldsymbol{\grave{r}}^{ij}, \boldsymbol{\acute{s}}^{ij}, \boldsymbol{\grave{s}}^{ij} \geq \boldsymbol{0} \quad \forall (i,j) \in \mathcal{E}_\mathcal{R}
\end{array}\right\} \tag{5.6}$$

is feasible. In above inequality system, $\boldsymbol{V}^0, \ldots, \boldsymbol{V}^N$, are the decision variables which determine the policy. $\acute{h}, \grave{h}, \acute{v}^{kl}, \grave{v}^{ij}, \acute{r}^{ij}, \grave{r}^{ij}, \acute{s}^{ij}, \grave{s}^{ij}$ are auxiliary variables from Expression 3.5.

Deciding $\acute{\epsilon}_{ij}$, $\grave{\epsilon}_{ij}$, p and q

$\acute{\epsilon}_{ij}$, $\grave{\epsilon}_{ij}$, p and q are the parameters used in 5.6, this section will explain how to get the value of these parameters.

We have presented the conditions for $\acute{\epsilon}_{ij}$ and $\grave{\epsilon}_{ij}$ to satisfy in expression 5.5. On another hand, The value of $\acute{\epsilon}_{ij}$ and $\grave{\epsilon}_{ij}$ take direct effect on the uncertainty budget $\acute{\Omega}_{ij}$ and $\grave{\Omega}_{ij}$. One way to determine each $\acute{\epsilon}_{ij}$ and $\grave{\epsilon}_{ij}$ is through an optimization problem that minimize total budget of uncertainties as presented in [14], because the less is the budget, the smaller is the size of the uncertainty set, and thus the easier for each constraint to be satisfied.

$$\begin{array}{ll} min & \sum_{(i,j)\in\mathcal{E}} \acute{\Omega}_{ij} + \sum_{(i,j)\in\mathcal{E}} \grave{\Omega}_{ij} \\ s.t. & \sum_{(i,j)\in\mathcal{E}} \acute{\epsilon}_{ij} + \sum_{(i,j)\in\mathcal{E}} \grave{\epsilon}_{ij} \le \epsilon \end{array} \tag{5.7}$$

where $\acute{\Omega}_{ij} = \sqrt{-2ln\,\acute{\epsilon}_{ij}}$ and $\grave{\Omega}_{ij} = \sqrt{-2ln\,\grave{\epsilon}_{ij}}$ Solving above optimization problem get $\acute{\epsilon}_{ij} = \grave{\epsilon}_{ij} = \frac{\epsilon}{2|\mathcal{E}_R|}$.

Previous section has presented definition of forward deviation set, backward deviation set (see expression 3.6 and 3.7). Forward deviation is defined as $p^* = \inf \mathcal{P}(\tilde{z})$ and backward deviation is defined as $q^* = \inf \mathcal{Q}(\tilde{z})$. For some distributions like normal distribution, the authors derived the value of $p*$ and $q*$ analytically. Their values coincide with standard deviation, i.e., $p* = q* = \sigma$. When the random variables are bounded by an interval, the distribution of the random variable becomes a truncated version, and the

$p*$ and $q*$ are hard to derive analytically. However, the authors provide a method to determine an approximated deviation denoted as p and q numerically by simulation. In this way, the distribution allowed is in wide range as long as we are able to obtain a sufficient sample for simulation. In this thesis, we assume normal distribution for all the random variables. In the experiment part we apply the provided simulation method to obtain the p and q value for truncated normal distribution. The procedures are similar to apply for other distributions commonly used to model temporal parameters such as uniform distribution and poisson distribution. For uniform distribution we discovered the $p*$ and $q*$ value also coincide with the standard deviation. As the authors only derived p and q by simulation for uniform distribution, we show the steps of mathematical deduction below.

As the uniform distribution is a symmetric distribution, its backward deviation $q*$ is the same as its forward deviation $p*$. Let's assume we have a random variable $\tilde{z}$ with uniform distribution between [-1,1], the following will illustrate $p* = \sigma = \sqrt{(\frac{1}{3})}$. The moment generating function for $\tilde{z}$ is

$$M(t) = \begin{cases} \frac{e^{tb}-e^{ta}}{t(b-a)} & t \neq 0 \\ 1 & t = 0 \end{cases} \tag{5.8}$$

To find $P(\tilde{z})$, one needs to find the set of α such that $M_{\tilde{z}-E(\tilde{z})}\left(\frac{\phi}{\alpha}\right) \leq$

$exp(\frac{\phi^2}{2})$. We have the following sequence:

$$\frac{e^{\frac{\phi}{\alpha}} - e^{-\frac{\phi}{\alpha}}}{\frac{2\phi}{\alpha}} \leq e^{\frac{\phi^2}{\alpha}} \tag{5.9}$$

$\Rightarrow$

$$\alpha(e^{\frac{\phi}{\alpha}} - e^{-\frac{\phi}{\alpha}}) \leq 2\phi e^{\frac{\phi^2}{\alpha}} \tag{5.10}$$

Apply Taylor's expansion $\Rightarrow$

$$\alpha\left(\left(1 + \frac{\phi}{\alpha} + \frac{(\frac{\phi}{\alpha})^2}{2!} + \frac{(\frac{\phi}{\alpha})^3}{3!} + M_1\right) - \left(1 - \frac{\phi}{\alpha} + \frac{(\frac{\phi}{\alpha})^2}{2!} - \frac{(\frac{\phi}{\alpha})^3}{3!} + M_2\right)\right) \leq 2\phi e^{\frac{\phi^2}{\alpha}} \tag{5.11}$$

M_1, M_2 are terms with high order exponent, and their difference is approximately 0. $\Rightarrow$

$$\alpha\left(2\frac{\phi}{\alpha} + 2\frac{(\frac{\phi}{\alpha})^3}{3!}\right) \leq 2\phi e^{\frac{\phi^2}{\alpha}} \tag{5.12}$$

$\Rightarrow$

$$\alpha \geq \frac{\phi}{\sqrt{6\left(e^{\frac{\phi^2}{2}} - 1\right)}} \tag{5.13}$$

The above inequality must be valid for all ϕ. In order to find $p* = inf(\alpha)$, one needs to find the maximum of right hand side first. Let $f(x) = \frac{x}{\sqrt{6\left(e^{\frac{x^2}{2}} - 1\right)}}$ where $x > 0$, the maximum is achieved at $f'(x) = 0$

$$f'(x) = \frac{1}{\sqrt{6}}\left(e^{\frac{\phi^2}{2}} - 1\right)^{-\frac{3}{2}}\left[\left(e^{\frac{\phi^2}{2}} - 1\right) - \frac{1}{2}x^2 e^{\frac{x^2}{2}}\right] = 0 \tag{5.14}$$

$\Rightarrow$

$$\left(e^{\frac{\phi^2}{2}}-1\right)-\frac{1}{2}x^2e^{\frac{x^2}{2}}=0 \tag{5.15}$$

$\Rightarrow$

$$1-\frac{x^2}{2}=e^{-\frac{x^2}{2}} \tag{5.16}$$

By plotting the graph of both sides, we observe there is only one intersection at $x = 0$ Thus $\max f(x) = lim_{x\to 0}f(x)$. Because $lim_{x\to 0}f^2(x) = lim_{x\to 0}\frac{x^2}{6\left(e^{\frac{x^2}{2}}-1\right)} = \frac{1}{3}$, thus $p* = \max f(x) = \sqrt{\frac{1}{3}}$ which is the same as its standard deviation.

5.3 Further Enhancement

This section is going to present two refinements to improve the results. The first one is regarding the network, a more refined execution policy will be proposed to deal with a special case. The second is regarding the solution process, a heuristic to allocate failure probability among constraints will decrease the total failure probability.

5.3.1 Enhancement in Execution Policy

In section 4.1.1, we have illustrated how to find dependency set for each executable timepoint. This section will refine the previous by dealing with a special case specifically. Consider a pair of observable timepoint C and executable timepoint B depicted in Fig.5.3.1 where (A, C) is the contingent

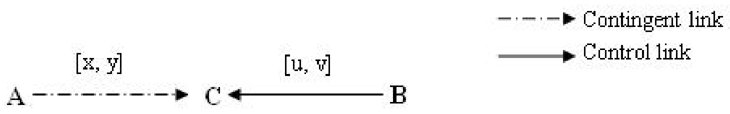

Figure 5.1: Determining the Dependency of An Executable Timepoint

edge associated with C:

Suppose $[u, v]$ is the constraint between B and C. We have described three possibilities in Section 4.1.1 and will mention them briefly again here. When $u \geq 0$, it means B precedes C, C is not included in the dependency set. When $v \leq 0$, it means B follows C, C is added to the dependency set. However for some of the derived edges (those not existed in original network) it is possible that $u < 0$ and $v > 0$. Let $\tilde{z}$ be the random variable representing the duration of contingent edge $A \rightarrow C$. Previously we would exclude $\tilde{z}$ from the dependency set of B and the expressions we have while formulating the problem is

$$\begin{aligned} &u \leq t_C - t_B \leq v \\ &\tilde{z} \notin D_B \end{aligned} \tag{5.17}$$

Morris and Vidal [9] has analyzed this special case, and discovered the rule if activity on $A \rightarrow C$ has not happened within $y - v$ time units, B must also wait till $y - v$ units expires with respect to timepoint A. Otherwise suppose B is executed before $y - v$ units with respect to A, when the duration of the activity on (A, C) takes the upper bound y, $T_C - T_B$ will be larger than v and violates the constraint on $B \rightarrow C$.

Here $y - v$ serves as a breakpoint. We have the following refined policy to the special case when $u < 0$ and $v > 0$:

$$\begin{aligned} &when \;\; \tilde{z} < y - v && u \leq t_C - t_B \leq 0 \\ & && \tilde{z} \in D_B \\ &when \;\; \tilde{z} \geq y - v && t_B - t_A \geq y - v \\ & && u \leq t_C - t_B \leq v \\ & && \tilde{z} \notin D_B \end{aligned} \tag{5.18}$$

The refined policy says the system allows an observation time within $y-v$ units. On one hand if activity on $A \rightarrow C$ happens during the observation period, $\tilde{z}$ is added to dependency set D_B to allow more execution flexibility and the upper bound on $B \rightarrow C$ is modified to 0 because system waits for event C to happen before deciding value of B. On another hand, if activity on $A \rightarrow C$ happens after the observation period, we are unable to know the relation between B and C. Thus we simply exclude $\tilde{z}$ from dependency set D_B and reserve the constraint on $B \rightarrow C$.

Having this new policy would result in larger feasible set because it utilize a rule without which, the constraint in equation 5.17 may fail to be satisfied. An concrete example will help to explain. Here we consider dynamic controllability instead of robust controllability. For one reason, it is more explicit to verify the effects of refined policy. For another, if the policy enables larger feasibility set for non-probabilistic case, it intuitively does the same to the probabilistic case. To simply the problem, we consider only a part of some network as in Fig. 5.3.1 and assume A is the origin of time. Suppose the duration of contingent activity on $A \rightarrow C$ is within $[3, 8]$, and the derived constraint on $B \rightarrow C$ is $[-1, 2]$. According to equation 5.17, we have

$$-1 \leq \tilde{z} - x_B \leq 2 \;\; \forall \tilde{z} \in [3, 8] \tag{5.19}$$

Apparently, above inequality system has no solution. Applying the refined policy in 5.18, we have:

$$\begin{array}{lll} when & \tilde{z} < 8 - 2 & -1 \leq \tilde{z} - (x_1 + x_2\tilde{z}) \leq 0 \\ when & \tilde{z} \geq 8 - 2 & x_B - 0 \geq 8 - 2 \\ & & -1 \leq \tilde{z} - x_B \leq 2 \end{array} \tag{5.20}$$

Note timepoint B has different expressions subject to value of $\tilde{z}$. It is $x_1 + x_2\tilde{z}$ is when $\tilde{z} < y - v$ and x_B otherwise. A solution for above system is: $x_1 = 2, x_2 = 2/3, x_B = 6$. It means If activity on $A \rightarrow C$ happens before 6 time units, we use the expression $2 + 2/3 \cdot \tilde{z}$ to decide time of B. Otherwise, we simply execute B at time 6.

In this example the refined policy works better than the original policy which ignores the special case where the derived[1] constraint between a contingent timepoint and executable timepoint has negative lower bound and positive upper bound. The executable timepoint in such cases will have two expressions, each of which is associated with different intervals of the contin-

[1]Note that the constraint is derived as any original lower and upper bound representing the duration of activities must be positive.

gent edge. While formulating the robust controllability problem, we could replace each original constraint involving the executable timepoint with two sets of constraints as listed in 5.18. Then we remove the condition "when" as in 5.18 by two random variables each representing one of the divided intervals of the original random variable.

5.3.2 Enhancement In ϵ_{ij} Allocation

With our solution approach, a RTCN with given failure probability ϵ is robust infeasible if it fails to satisfy the expression 5.21. However, as our solution approach is not exact, the network may be actually robust feasible with the given ϵ failure probability especially when the result by replacing given ϵ of the network with a slightly larger value is feasible. There is a possibility to refine the procedure of allocating ϵ_{ij} to gain an improved result. Notice that we are using ϵ_{ij} to represent both $\acute{\epsilon}_{ij}$ and $\grave{\epsilon}_{ij}$ as they are treated equally.

According to section 5.3.2, the failure probability ϵ is divided equally among each constraint in order to gain a minimum total budget (Larger budget means larger coverage or more stringent attention). However, intuitively if the inequality system is not feasible, it is normally caused by violation on a subset of the constraints. If we could unload excessive budget from those easily violated constraint and transfer it to those less easily violated ones, the system may become consistent. This section is to improve the solution approach by assigning unequal ϵ_i to different constraints by some *heuristics*. Observe the constraint on each of the controllable edge:

$$\mathrm{P}\left(\{l_{ij} \leq V_j(\tilde{\boldsymbol{z}}) - V_i(\tilde{\boldsymbol{z}}) \leq u_{ij}\}\right) \geq 1 - \epsilon_{ij} \quad \forall (i,j) \in \mathcal{E}_R. \tag{5.21}$$

It seems reasonable to assume that constraints with wider interval tend to have bigger probability P in above expression, or in another word, the constraints with larger $u_{ij} - l_{ij}$ value are more possible to be satisfied. Thus we develop following heuristic:

Suppose a network is not robust feasible with failure probability ϵ but feasible with a slightly larger failure probability ϵ' given that we have allocated equal ϵ'_{ij} to each controllable edge (i, j). Now we sort the lengths of intervals of controllable edges $len_{ij} = u_{ij} - l_{ij}$ by decreasing order. The higher is the rank len_{ij} in the sorted list (nearer to the front of the list), the more possible the constraint on the corresponding edge is satisfiable, and the less likely the infeasibility is caused by this constraint. Thus we choose to increase the Ω'_{ij} (the larger Ω'_{ij} , the smaller ϵ'_{ij}) value by a suitable predefined amount (for example 0.2) until it causes violation again. We repeat this process by trying the edges on the sorted list sequentially. If at any stage, the accumulated failure probability decreases to the original ϵ, we are done. Notice that edges with higher rank not necessarily sustain larger budget than edges with lower rank. The reason behind sorting the interval list is to reach given ϵ more quickly because intuitively larger len_{ij} value implies a support of higher budget.

Chapter 6

Experimental Result

Although temporal network has been studied for years, we are surprised to find there is little experiment result in the published papers. We have to generate and design our own experiments rather than compare with any benchmarks. For this thesis, we will perform three groups of experiments. The first two groups are intended to test how failure probabilities are affected by the property of the network and by the controllability requirement, the last is the proposed improvements. Before discussion of each individual experiment, there is a need to describe the workflow of the experiment commonly shared by each experiment of this report.

6.1 Experimental Setup

The workflow could be depicted as Fig 6.1 including three main modules:

(1) RTCN generation,

(2) Problem construction,

(3) Problem solution.

The figure shows each module has its own functions which are shown in the graph too together with their inputs and outputs. For each experiment, the three modules will be run sequentially. First we generate RTCN according to the required parameters. The second module will subsequently work on the generated RTCN to construct formulation of the Robust Controllability problem . The solver in the last module will then resolve the problem based on the output of an interface which translates the problem to an accepted form by the solver. The following part will explain each module with an emphasis on RTCN generation module rather than the implementation issues on constructing and solving the problem.

RTCN Generation

The first module includes three main functions: $TimeNet_Manager$, $Assign\_Contingency$ and $SIMUL$. $TimeNet_Manager$ is the main function and is based on the algorithm of a software tool named $TimeNetManager$ described in the work [15]. $TimeNetManager$ is able to build temporal networks by using a set of parameters $\{n, \alpha, \beta, D, S_i, S_a, U\}$ which controls the network structure. The output $\{V, \mathcal{E}, l, u\}$ of $TimeNetManager$ is a connected and acyclic graph where each edge of the graph is bounded by a randomly generated interval. Since $TimeNetManager$ does not specify con-

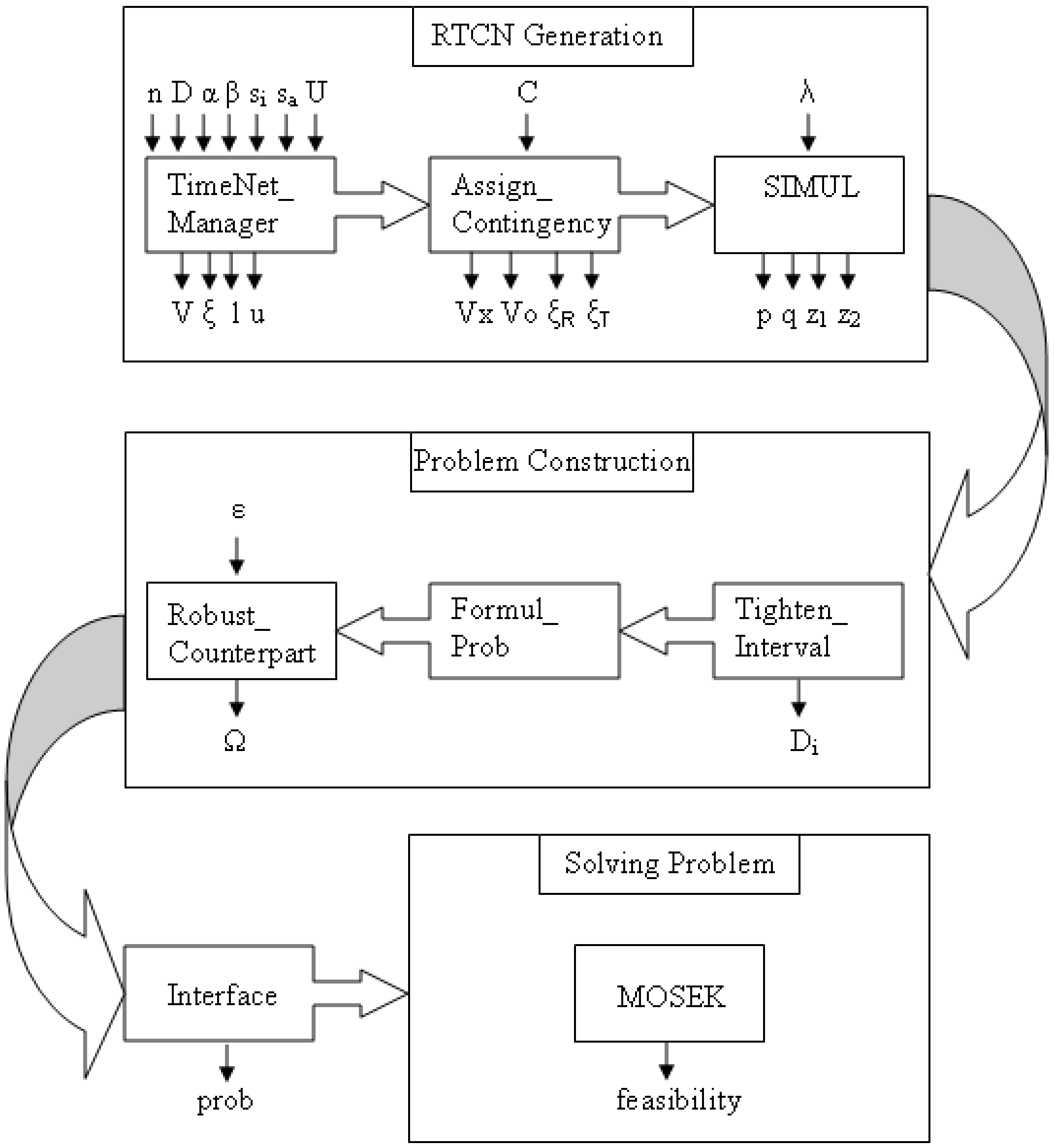

Figure 6.1: The Workflow of Experiment

tingent edges as it was originally used to generate random STN, we need the second function $Assign\_Contingency$ to specify the set of contingent edges controlled by some parameter. The parameter we use is called *contingency density* and denoted as C. It equals to the ratio between the number of contingent edges and the number of edges of the network. After running this function, the set of contingent edges and observable timepoints are decided. The last function $SIMUL$ is to generate vectors p, q, z_1, z_2 for the set of contingent edges. As said earlier in the report, the forward and backward deviation could be approximated numerically by a simulation function described by the authors of [14]. The input of the simulation function is a statistic sample of the values between an interval. As we assume normal distribution in this thesis, we could generate such samples easily using embedded functions of MatLab. The Mean of the normal distribution is the average of the lower bound and upper bound, the variance is decided by dividing the length of the interval by a parameter λ. There is a need to briefly explain the first function as it is quite complicated. More details can be found in the original paper [15].

The algorithm of $TimeNetManager$ controls four main graph's characteristics:

(1) the number of nodes n,

(2) the edges density D,

(3) the network topology and

(4) the temporal flexibility.

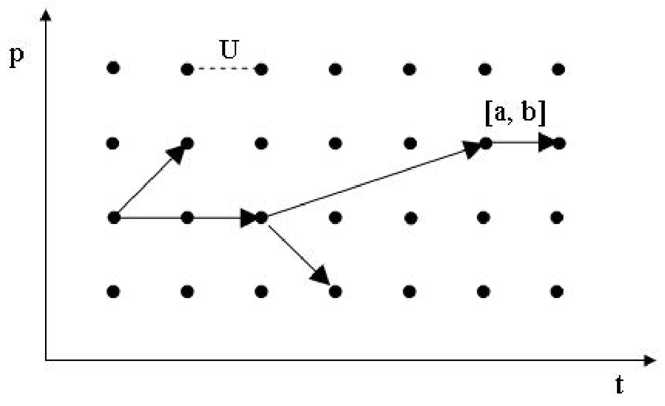

Figure 6.2: The Grid

The first two characteristics determine the number of nodes and the number of edges of the graph. The last two characteristics determine the arrangement of the nodes and edges. In order to control the last two characteristics, *TimeNetManager* use a reference structure called grid (see Fig.6.2). It is a matrix of points with discrete dimensions $T \times P$, where the horizontal dimension represents the time and the vertical one represents a quantity called degree of parallelism which represents the maximum number of contemporaneous timepoints in the same time period. The basic idea is to randomly map the set of timepoints on the set of grid points, in this way the timepoints are positioned and the distance between girds along horizontal dimension determines the average time between timepoints.

The algorithm takes as input the parameters $n, \alpha, \beta, D, S_i, S_a, U$. These parameters all together determine the required characteristics of a temporal network.

$\alpha = P/T$ is called the *grid ratio*.

$\beta = P \cdot T/n$ is the ratio between the number of the grid points and the number of nodes, and is called the *grid density*.

S_i and S_a are two real and nonnegative slack parameters controlling the amplitude of the intervals.

U is the number of temporal units per grid interval.

The generation process can be divided in three main phases: *initialization*, *tree generation* and *graph completion*. During initialization phase the random mapping is realized between the set of nodes and the set of points in the grid. In order to guarantee the connectivity of the output graph, a tree is created by tree generation phase. Meanwhile, the intervals are also generated based on the distance between two nodes in the grid and the two parameters S_i, S_a. In the last phase namely graph completion, other edges together with the associated intervals are added to the graph until edge density D is reached.

Problem Construction

The previous module has generated the RTCN we needed. The Construct Problem module will formulate the Robust Controllability problem based on the data of the generated RTCN. The first function $Tighten\_Interval$ is to tighten the interval and generate the dependency set for each executable timepoint. The algorithm to apply is all-pair shortest-path algorithm explained in Section 4.1.1. The next function $Formul\_Prob$ will formulate the problem by defining the executable timepoints with affine linear policy and expressing the constraints on the controllable edges as inequalities. The last function *Robust_Counterpart* will then construct the robust counterpart of the inequalities.

Problem Solution

The robust counterpart of our formulation is a *Second-Order Cone Program* (SOCP) [26] which is polynomially solvable by any SOCP solver. The solver we used in the last module is a powerful tool called MOSEK [27]. It provides interface to MatLab and is quite efficient. However, the input of MOSEK is limited to matrix form. An interface is needed between *Problem construction* and *Problem solution* module to transform our robust counterpart program to MOSEK-format.

Before moving on to the discussion of each experiment, several issues shared by the experiments need to be raised in advance. Firstly the number of nodes used in all the experiments are $n = 10$, and edge density $D = 0.5$, which means roughly 22 edges and 44 constraints. The generated network is of moderate size. Secondly and importantly, the problem previously proposed is a robust feasibility problem, which means given a parameter ϵ, one is to decide whether the network is dynamically controllable with probability $1 - \epsilon$. Thus if a network is robust controllable with $\epsilon = 5\%$, it is also robust controllable with $\epsilon = 10\%$ but not vice versa. In the experiment, we will use binary search to obtain the smallest ϵ for a given network for experimental purpose. We denote this smallest ϵ for robust controllability as ϵ_{DC} as it is regarding dynamic controllability. We will call the smallest failure probability ϵ a network is weakly controllable as ϵ_{WC}. The algorithm used to obtain ϵ_{DC} is stated in *procedure 1* which utilizes binary search to speed up the searching process.

Procedure 1 Calculate ϵ_{DC} by binary search

```
1: flag = true
2: $\epsilon = \epsilon_0$
3: while flag do
4:    while the network is robust controllable with $\epsilon$ do
5:       $\epsilon = \epsilon/2$
6:    end while
7:    $\epsilon_1 = \epsilon$;
8:    while the network is not robust controllable with $\epsilon$ do
9:       $\epsilon = \epsilon_1 + \epsilon/2$
10:   end while
11:   if $(2 * \epsilon_1 - \epsilon < threshold)$ then
12:      flag=false
13:   end if
14: end while
```

The basic idea is to halve previous ϵ if the network is feasible and increase the last unfeasible ϵ_1 by half of previous ϵ if the network is infeasible until it is feasible again and the obtained ϵ is satisfiable which means the difference between the current feasible ϵ and the last unfeasible ϵ_1 is less than a threshold.

6.2 Experiment One

Apparently, many factors could affect ϵ_{DC} such as n the number of nodes, m the number of edges, C the contingent density, the network topology, the amplitude of the interval associated with each edge and N the distribution of the contingent edge duration. Each of the above factors is able to affect ϵ_{DC} individually or through combinations by various degree. We are not going to test all the effects but instead we are more interested to see the effect of C the contingent density and N the distribution of the contingent edge duration as these two are unique to temporal network with uncertainty. Testing C is straightforward while testing the distribution of random variables have several possibilities. For example, we could test normal distribution versus uniform distribution, we could also test how varying λ changes the ϵ_{DC}. We choose to test the latter as λ is directly linked to the forward and backward deviation in our solution approach. Recall λ is used to produce standard deviation $\sigma = (u - \mu)/\lambda$ where u is the upper bound and μ is the mean. σ is subsequently used to generate forward and backward deviation. To have a reasonably large worst case budget Ω_ω, we choose three values for λ: $10, 12, 14$. The testing procedure is stated in *procedure 2: Testing procedure for Experiment One*

In this experiment, we are using the same network sample generated by $TimeNetManager$ (line 1 of *procedure 2*) which means we fix the number of nodes, edges, and the lower and upper bound of each edge but select

Procedure 2 Testing procedure for Experiment One

```
1: Generate a random network with $n = 10, |ED| = 0.5$
2: for $C = 0.1, 0.2, 0.3, 0.4$ do
3:     for $i = 1 : 10$ do
4:         Determine the set for contingent edges $\mathcal{E}_t$
5:         for $\lambda = \lambda_1, \lambda_2, \lambda_3$ do
6:             Generate the set of parameters for $\mathcal{E}_t$
7:             Calculate $\epsilon_{DC}$ for current network
8:             $\epsilon(i)^C_\lambda = \epsilon_{DC}$
9:         end for
10:    end for
11:    $\epsilon^C_\lambda = average(\epsilon(1:10)^C_\lambda)$
12: end for
```

certain edges to be contingent edges for each experiment instance. To be more detailed, we choose different numbers of contingent edges determined by the C values, and for each C value we vary the selection of contingent edges. Line 5 to line 9 is assigning different deviation parameters to the generated network instance. Line 11 then calculates the average ϵ_{DC} of 10 network instances for each contingent density C.

The result is plotted by Fig.6.3. There are two observations. Firstly, by fixing C, the larger the λ value, the less the failure probability ϵ_{DC}. This is because larger λ value means smaller standard deviation or smaller p, q values, and it in turn means random data is more concentrated to the middle of the interval. For a fixed network instance [1], the feasible interval for each contingent edge is fixed. The more the data is concentrated, the larger budget we allow. Thus larger λ results in a larger probability the

[1]Here fixed network means the number of nodes, edges, lower and upper bound for both controllable and contingent edges are fixed

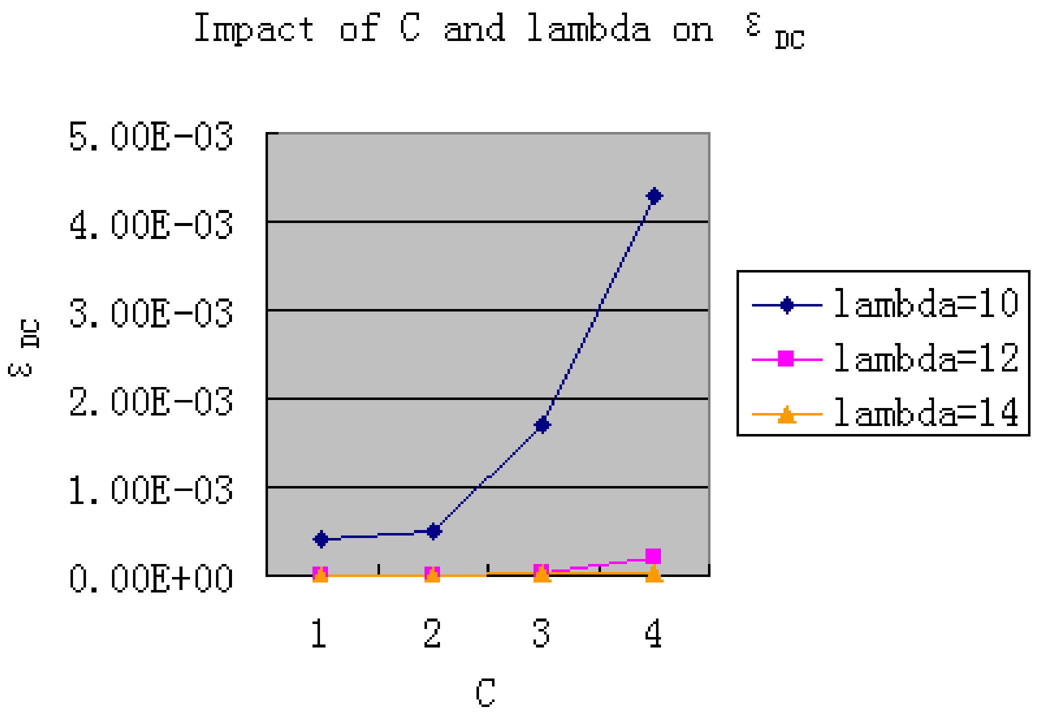

Figure 6.3: Plot of Experiment 1

network is dynamically controllable. Secondly, by fixed λ value, ϵ_{DC} increases as contingent density C increase, this says the more number of contingent edges, the smaller the probability a network is dynamically controllable. It is quite straightforward because more contingent edges directly means more network dynamic is required.

6.3 Experiment Two

This second experiment will study the relation between probabilistic dynamic controllability (robust controllability) and probabilistic weak controllability. It is already proved that dynamic controllability is more strict than weak controllability but the relation between the two in probabilistic context is not clear yet up to this point. To be more specific, this experiment is going to examine the relation between ϵ_{WC} and ϵ_{DC} given a temporal network.

Intuitively the relation between the two controllabilities still holds in probabilistic context. In another word, the probability a network is dynamically controllable is less than the probability the network is weakly controllable. Indeed it can be proved. As stated in beginning of Chapter 4, the probability a temporal network is dynamically controllable is $\mathrm{P_{DC}} = \mathrm{P}(\xi|\mathrm{t}(\xi) \in \mathrm{sol}(\xi), \xi \in \Xi)$. Here Ξ is the set of possible network instances and ξ is a single instance. $t(\xi)$ is the output of execution policy. The probability it is weakly controllable is $\mathrm{P_{WC}} = \mathrm{P}(\xi|\mathrm{sol}(\xi) \neq \phi, \xi \in \Xi)$. Suppose $t(\xi)$ is a valid solution for network instance ξ $(t(\xi) \in sol(\xi))$, thus

$sol(\xi) \neq \phi$. The set $\{\xi | t(\xi) \in sol(\xi), \xi \in \Xi\} \subset \{\xi | sol(\xi) \neq \phi, \xi \in \Xi\}$, which means $\mathrm{P_{DC}} \leq \mathrm{P_{WC}}$ or $\epsilon_{DC} \geq \epsilon_{WC}$.

The proof above is about the relation between the optimal failure probabilities of two controllabilities. However, as the solution approach applied here produces an approximation to the optimal failure probability, the ϵ_{DC} and ϵ_{WC} could deviate from the optimal value by an unknown amount. If the ϵ value obtained by the solution approach is not tight enough, the property may not be shown at all. We will do experiment to test $\epsilon_{DC} - \epsilon_{WC}$ on the same temporal network. We generate four groups of RTCN whose contingent densities are $0.1, 0.2, 0.3, 0.4$ respectively. For each group of RTCN, we will generate k samples and compare the average ϵ_{DC} and ϵ_{WC} values. Details are described in *Procedure 3: Testing procedure for Experiment Two*

Procedure 3 Testing procedure for Experiment Two

```
1: for $C = 0.1, 0.2, 0.3, 0.4$ do
2:     for $i = 1 : 10$ do
3:         Generate a random network with $n = 10, |ED| = 0.5$
4:         Determine parameters for contingent edges $\mathcal{E}_t$
5:         Calculate $\epsilon_{DC}$ and $\epsilon_{WC}$ of current network
6:     end for
7:     Calculate average of $\epsilon_{DC}$ and $\epsilon_{WC}$ of 10 rounds
8: end for
```

The result is plotted by the Fig.6.4. From the experiment plot, we can see clearly $\epsilon_{DC} > \epsilon_{WC}$ which complies with the property we have proved, and shows that the ϵ produced by our solution approach is reasonable. Another discovery is as the contingent density D increases, the difference $\epsilon_{DC} - \epsilon_{WC}$ is enlarged. This shows ϵ_{DC} changes more rapidly in terms of the amplitude

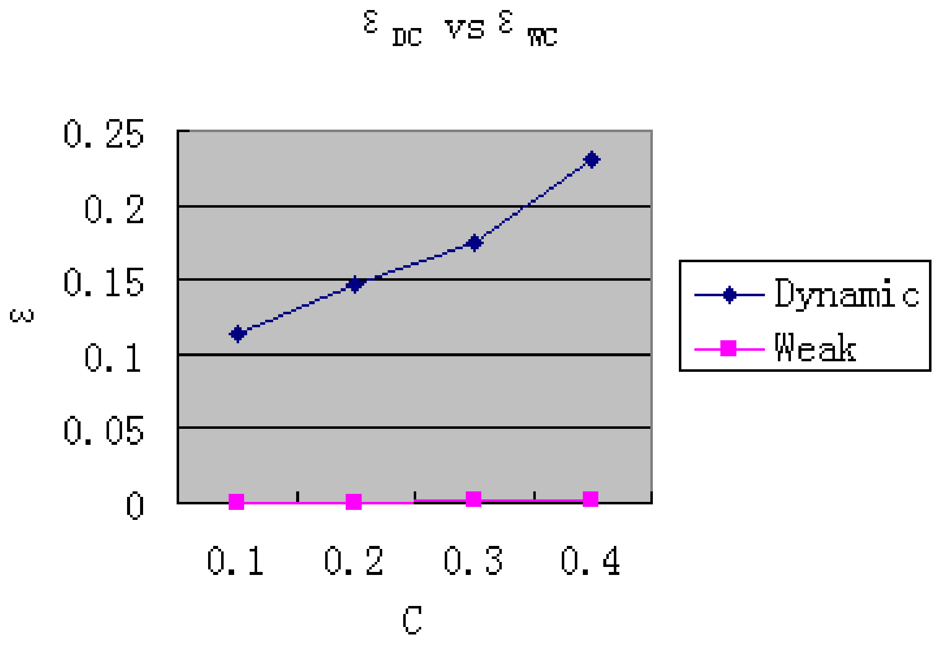

Figure 6.4: Plot of Experiment 2

of its value. This result complies with intuition too. When there is only one contingent edge, the probabilities of being dynamically controllable and being weakly controllable are both small. When the contingent edges increase to a certain number, the network may become hardly dynamically controllable while still weakly controllable in many scenarios as weak controllability is less strict than dynamically controllable. Thus the jump on ϵ_{DC} is larger than ϵ_{WC}.

6.4 Experiment Three

For this experiment, we examine the improvement effect of allocating unequal ϵ_{ij} by the heuristic introduced in section 5.3.2. Again instead of robust feasibility problem, we examine how ϵ_{DC} could be decreased by using our heuristics. We will first randomly generate a network instance, and calculate ϵ_{DC} by binary search. For an exhausted search, we could try to decrease ϵ_{ij} for every edge which does not need the sorting procedure. Further more, since the constraints are independent from each other. We could test the feasibility of the robust counterpart of each particular constraint alone instead of checking the feasibility of the whole problem each time we decrease the budget of a constraint. However, for large networks and when the runtime is an important issue, reaching a durable ϵ quickly is desirable. Thus we apply the sorted list heuristic here because the edges on top of the lists have biggest potential to stand smaller failure probability. To control time, we use a parameter η to limit the number of runs. The steps are described in *Procedure 4: Testing procedure for Experiment Three.*

Procedure 4 Testing procedure for Experiment Three

```
1: Generate a random network with $n = 10, |ED| = 0.5, C = 0.2$
2: Calculate $\epsilon_{DC}$
3: Sort $len_{ij} = u_{ij} - l_{ij}$ into a list $lst$ by decreasing order
4: $turn = 1, edge = lst.head()$
5: while $turn \leq \eta$ do
6:    while $edge$ is not last member of $lst$ do
7:       while $\epsilon_{ij} > 0$ do
8:          $\epsilon_{ij} = \epsilon_{ij} - unit$
9:          Check feasibility with $\epsilon_{ij}$
10:         if infeasible then
11:            $\epsilon_{ij} = \epsilon_{ij} + unit$
12:            break
13:         end if
14:      end while
15:      $edge = lst.next()$
16:   end while
17: end while
18: Calculate $\epsilon'_{DC} = \sum_{(i,j)\in\mathcal{E}_R} \epsilon_{ij}$
```

The number of η limits the number of runs, and in turns affect the amount of failure probability we are able to decrease. Meanwhile we could also increase *unit* value to allow faster decrement of failure probability. In this experiment η is set to 10 and η is set to 0.2. The outcome of the experiment is like following: ϵ_{DC} value before with unequal ϵ_{ij} is 0.2111, the obtained decreased ϵ'_{DC} is 0.1897. This preliminary result suggests our heuristic seems to be effective.

Chapter 7

Conclusion

Temporal network with uncertainty is a useful formalism to model important and practical problem in real-world scheduling and planning system. As it is often the case that the durations of some activities are not known exactly until they take place, it is not clear how to execute the scheduling/planning system in advance. Instead, the decisions are built incrementally as the durations of uncertain activities are gradually revealed. Previous literature has used the notation of dynamic controllability to describe whether a temporal network with uncertainty could be built in this way successfully.

In this thesis, we introduce the notation of probabilistic dynamic execution. We presented a Robust Temporal Constraint Network RTCN where the durations of uncertain activities are modelled by random variables with some probability distribution. Given a RTCN with an acceptable failure probability ϵ, we check if the probability an execution policy produces a schedule satisfying all the temporal constraints of RTCN is at least $1 - \epsilon$. To obtain a computational tractable problem, we assume an *Affine Linear* policy in which the time of every executable event is linearly dependent on the durations of some relevant uncertain activities. To solve the problem, we apply a recent promising technique called *Robust Optimization* which is able to solve certain Stochastic Programs efficiently. As the size of uncertainty budget controls the value of failure probability, this technique also enable the adjustment of degree of conservativeness in guaranteeing controllability. Lastly, we propose some refinements on both the policy and the solution approach to improve the result.

Experimentally, we first show that as the number of contingent activities increases, or as the distribution of the durations of contingent activities becomes more deviated, the chance the scheduling/planning can be executed dynamically decreases, or as presented in Fig 6.3 the failure probability increases. We also show that for a given temporal network, the probability it is dynamically controllable is less than the probability it is weakly controllable, or presented in Fig 6.4 the failure probability of being dynamically controllable (ϵ_{DC}) is more than that of being weakly controllable (ϵ_{WC}). Furthermore, as the number of contingent edges increases, the ratio between the two, $\epsilon_{DC}/\epsilon_{WC}$, seems to increase in exponential order as the former increases much faster than the latter in Fig 6.4.

For future work, we could explore more execution policy options and further improve the solution approach by proposing more heuristics.

Bibliography

[1] A Cesta, A Oddi, SF Smith, A constraint-based method for project scheduling with time windows, *Journal of Heuristics, vol 8(1)*: 109-136 (2002)

[2] Brusoni, V., Console, L., Pernici, B., and Terenziani, P., LaTeR: a General Purpose Manager of Temporal Information, *LNCS, vol 869*: 255-264 (1994)

[3] Jourdan, M., Layaïda, N., Sbry-Ismail, L., Time Representation and Management in MADEUS: an Authoring Environment for Multimedia Documents, *Multimedia Computing and Networking*:68-79 (1997)

[4] Dechter Rina, Itay Meiri, Judea Pearl), Temporal Constraint Networks, *Artificial Intelligence 49*: 61-95 (1991

[5] J.F. Allen (1983), Towards a General Theory of Action and Time, *Commun. ACM 26(11)*:832-843.

[6] M. Vilain and H. Kautz, Constraint Propagation Algorithms for Temporal Reasoning, *AAAI-1986*: 377-382 (1986)

[7] Ioannis Tsamardinos, A Probabilistic Approach to Robust Execution of Temporal Plans with Uncertainty, *SETN 2002*: 97-108 (2002)

[8] Ioannis Tsamardinos, Martha E. Pollack, Sailesh Ramakrishnan, Assessing the Probability of Legal Execution of Plans with Temporal Uncertainty, *ICAPS-03 Workshop on Planning under Uncertainty and Incomplete Information* (2002)

[9] Paul Morris, Thierry Vidal, Dynamic Control Of Plans With Temporal Uncertainty, *IJCAI 2001*: 494-502 (2001)

[10] Paul Morris, Nicola Muscettola, Temporal Dynamic Controllability Revisited, *AAAI-05*: 1193-1198 (2005)

[11] Paul Morris, Nicola Muscettola, Execution of Temporal Plans with Uncertainty, *AAAI/IAAI, 2000*: 491 - 496 (2000)

[12] Thierry Vidal, Helene Fragier, Controllability Characterization and Checking in Contingent Temporal Constraint Networks. *KR2000*:559-570 (2000)

[13] Thierry Vidal, Helene Fragier, Handling Contingency in Temporal Constraint Networks: from Consistency to Controllabilities, *J. Exp. Theor. Artif. Intell. 11(1)*: 23-45 (1999)

[14] X Chen, M Sim, P Sun (2005), A Robust Optimization Perspective to Stochastic Models. *Submitted to Operations Research, June 2005*

[15] Amedeo Cesta, Angelo Oddi, Angelo Susi, TimeNetManager - A Software Tool for Generating Random Temporal Networks. *Lecture Notes In Computer Science Vol. 1792*: 143 - 154 (1999)

[16] Soyster, A.L., Convex Programming With Set-inclusive Constraints and Applications to Inexact Linear Programming, *Oper. Res., 21*:1154-1157 (1073)

[17] Ben-Tal, A., Nemirovski, A., Robust Convex Optimization. *Math. Oper. Res.,23*: 769-805 (1998)

[18] Ben-Tal, A., Nemirovski, A., Robust Solutions to Uncertain Programs. *oper. Res. Let., 25*: 1-13 (1999)

[19] Ben-Tal, A., Nemirovski, A., Robust Solutions of Linear Programming Problems Contaminated with Uncertain Data. *Math. Progr., 88*: 411-424 (2000)

[20] P. Kouvelis, G. Yu, *Robust Discrete Optimization and its Applications*, Kluwer Academic Publishers, 1997.

[21] Bertsimas, D., Sim, M., Robust Discrete Optimization and Network Flows. *Math. Progr., 98*: 49-71 (2003)

[22] Bertsimas, D., Sim, M., Price of Robustness. *Oper. Res., 52(1)*: 35-53 (2004)

[23] Bertsimas, D., Sim, M., Robust Linear Optimization Under General Norms. *oper. Res. Let., 32*: 510-516 (2004)

[24] El-Ghaoui, Lebret, H., Robust Solutions to Least-square Problems to uncertain Data Matrices. *SIAM J. Matrix Anal. Appl., 18*: 1035-1064 (1997)

[25] El-Ghaoui, L. Oustry, F., Lebret, H., Robust Solutions to Uncertain Semidefinite Programs. *SIAM J. Optim., 9*: 33-52. (1998)

[26] F Alizadeh, D Goldfarb , Second-order Cone Programming. *Mathematical Programming 95(1)*: 3-51

[27] *http://www.mosek.com/*