Robust Controllability of Temporal Constraint Networks under Uncertainty

Hoong Chuin LAU
School of Information Systems
Singapore Management University
hclau@smu.edu.sg

Jia LI
School of Information Systems
Singapore Management University
jli@smu.edu.sg

Roland H.C. YAP
School of Comuting
National University of Singapore
ryap@comp.nus.edu.sg

Abstract

Temporal constraint networks are embedded in many planning and scheduling problems. In dynamic problems, a fundamental challenge is to decide whether such a network can be executed as uncertainty is revealed over time. Very little work in this domain has been done in the probabilistic context. In this paper, we propose a Temporal Constraint Network (TCN) model where durations of uncertain activities are represented by random variables. We wish to know whether such a network is robust controllable, i.e. can be executed dynamically within a given failure probability, and if so, how one might find a feasible schedule as the uncertainty variables are revealed dynamically. We present a computationally tractable and efficient approach to solve this problem. Experimentally, we study how the failure probability is affected by various network properties of the underlying TCN, and the relationship of failure rates between robust and weak controllability.

1 Introduction

Reasoning about *Temporal Constraint Networks* is a common task in AI. Typically, we want to verify if a plan or schedule is temporally consistent. *Temporal Networks with Uncertainty* are used to model temporal networks with two kinds of events or time-points. To make explicit the temporal nature, in this paper we will mostly talk about a time-point to refer to an event at a particular time. Time-points which are under the control of the planner are considered as "executable" and are the decision variables. The uncertainty arises from the remaining time-points which are the result of external factors not under the control of the planner, the "observables". Different problems arise depending on how observables affect the executable time-points which leads to the notions of strong, weak and dynamic controllability.

Most papers on temporal networks with uncertainty deal with execution under 100% confidence. This may be too conservative. In practice, this can lead to wasteful resource allocation or infeasible networks. In this paper, we study uncertainty in a probabilistic context for networks where each contingent (uncertain) edge is represented by a random variable. We call such networks *Robust Temporal Constraint Networks* because they allow one to ensure the feasibility of the network within a specified degree of confidence. Among the three forms of uncertainty (strong, dynamic and weak), it is dynamic controllability which is the most interesting from a practical point of view. However, the problem of dynamic controllability in a probabilistic setting is difficult because it would seem that with a continuous probability distribution, the number of scenarios would be infinite or exponential with discrete probabilities. We propose, under some assumptions on the relationship between controllable and uncontrollable (contingent) time-points, a computationally tractable approach for verifying and executing the dynamic controllability problem on robust temporal constraint networks. The strength of this approach is that we need very little information about the uncertainties, only the mean and variance of the uncontrollable time-points is needed. We then investigate the effect of network properties on the failure probability and compare dynamic against weak controllability. We remark that

our approach also works when the uncontrollable time-points do not have bounds[1] but some small changes with definitions will be needed and we would also need to know information about the probability distribution.

2 Related Work

Temporal constraint networks were introduced by [DMP91]. The widely popular Simple Temporal Constraint Network (STN) is formulated as a 4-tuple $\langle V, \mathcal{E}, l, u\rangle$, where V is the set of nodes representing time-points, $\mathcal{E}$ is the set of edges denoting binary constraints on the temporal duration between two time-points, l and u are functions defining the lower and upper bounds of the durations. A STN can be represented using a DAG. Each directed edge $(i, j) \in \mathcal{E}$ is labeled by an interval $[l_{ij}, u_{ij}]$ which represents the constraint $l_{ij} \leq V_j - V_i \leq u_{ij}$. A tuple $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a *solution* if the assignment $\{V_1 = x_1, \ldots, V_n = x_n\}$ satisfies all the constraints in $\mathcal{E}$. A STN is *consistent* if there exists at least one solution.

Simple Temporal Network with Uncertainty (STNU) [VG96] extends STNs to model the uncertain (or uncontrollable) duration between time-points. A STNU distinguishes a subset C of $\mathcal{E}$ which are the *contingent edges* representing uncertain durations – these can only be measured/observed after a certain time-point has been reached, i.e. $\forall c_{ij} \in C$, the duration of c_{ij} can take *any* value inside the interval $[l_{ij}, u_{ij}]$. Let $\mathcal{S}$ be the set of all possible scenarios ranging over all the possible values of C. A STNU is defined to be *strongly controllable* if there exists a solution such that it satisfies all the constraints in $\mathcal{E}$ irrespective of all the scenarios in $\mathcal{S}$. A STNU is *weakly controllable* if for each possible scenario in $\mathcal{S}$ there exists a solution. A STNU is *dynamically controllable* if a solution can always be built up incrementally based on the outcomes of contingent edges in the past, but not on those in the future.

Strong controllability is the strictest form of STNU. It is useful in applications where contingent events cannot be observed exactly. A strongly controllable network means that the plan can be executed without regard to the contingent events. Weak controllability is the weakest kind of STNU and is relevant in applications where contingent events are observable before execution and the corresponding solution can be computed beforehand. In other applications, contingent events are only observable after they happen, i.e. events caused by the external environment. Thus, an execution strategy using dynamic controllability is needed to produce an incremental solution based on the subsequent revelation of contingent events. Clearly, strong controllability implies dynamic controllability, which implies weak controllability.

Consistency algorithms and tractable classes for the standard STNU were proposed in [VF99], among others. Execution algorithms for STNU trading between flexibility and efficiency were presented in [MM00]. Dynamic controllability was previously believed to be hard, until a recent work by Morris and Muscettola [MM05] who proposed a pseudo-polynomial algorithm which is based on constraint satisfaction. It utilizes a ternary structure to tighten the intervals on each edge until either inconsistency rises or there are 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.

Another approach for dealing with uncertainties in temporal duration is with Fuzzy Simple Temporal Constraint Network [NCBS97]. This extends STNs with fuzzy temporal constraints which is interpreted using the techniques from possibilistic distributions. Closely related to fuzzy networks are STNUs with preferences, [RVY04] studies Simple Temporal Problems with Preferences and Uncertainty.

Although there is a body of work on various uncertain forms of temporal networks, there has been little work on the most obvious kind of uncertainty, namely a probabilistic one. The closest related work is Probabilistic Simple Temporal Networks (PSTN) [T02] which extends STNUs with conditional probability density functions for each uncontrollable event. One difference between the probabilistic and our robust networks is that the probability distributions are with respect to a prior controllable time-point. That work dealt mainly with strong controllability in PSTN. Subsequently [TPR03] looked the problem of determining the probability of executing a dynamically controllable strategy for a PSTN. Rather than a tractable polynomial time algorithm, they presented heuristic techniques for approximating the upper and lower bounds on the execution probability. Under normal distribution assumptions for contingent edges, they proposed a method based on [MV01] to find a lower bound probability of success.

The difference with our work in this paper is that we use a robust optimization approach which, under some reasonable statistical conditions, provides a computationally tractable algorithm for verifying dynamic

[1] [T02] argues that the bounds in STNUs are somewhat artificial and hence a probability distribution of the time-points may be closer to real-life.

controllability on a probabilistic temporal network. We only require partial information about the uncertainty, only the mean, variance and bounds of the uncontrollable events, rather than any complete information on the probability distributions. Unlike PSTN, our work does not assume having no constraints between uncontrollable events.

Robust optimization models have been proposed recently for immunizing uncertain mathematical optimization against infeasibility while preserving the tractability of the model, e.g. [BN00] and [BS04]. More recently, [CSS04] proposed a robust optimization technique for solving approximate solutions for stochastic optimization models with chance constraints. Their solution approach is based on second-order cone programming which is a generalization of semidefinite programming. The model permits more sophisticated uncertainty sets that are mapped from random variables. This technique provides a convenient and computationally tractable solution strategy for verifying and solving consistency of networks whose contingent edges are defined on random variables. [LOS05] first applied ideas from robust optimization specifically to solve weak controllability. In this paper, we generalize their work to handle dynamic controllability. Essentially, the underlying network model needs to be expanded to pay special attention to distinguishing between executable vs observable time-points, and controllable vs contingent edges. Algorithmically, we are confronted with additional dynamic issues such as dependency and onsite executions. We also improve upon the efficiency of the robust approach through a more refined execution policy and a heuristic to allocate failure probabilities among constraints.

3 Problem Formulation

Notationally, a random variable is denoted by $\tilde{x}$. Bold face lower case letters such as $\boldsymbol{x}$ represent vectors and the corresponding upper case letters such as $\boldsymbol{A}$ denote matrices.

Definition 1 *A Temporal Constraint Network (TCN) is represented by a 4-tuple $\mathcal{N} = \langle \mathcal{V}_\mathcal{X}, \mathcal{V}_\mathcal{O}, \mathcal{E}_\mathcal{R}, \mathcal{E}_\mathcal{T} \rangle$ where*
$\mathcal{V}_\mathcal{X}$: Set of nodes representing executable time-points
$\mathcal{V}_\mathcal{O}$: Set of nodes representing observable time-points
$\mathcal{E}_\mathcal{R}$: Set of controllable edges where each edge represents the following temporal constraint:

$$l_{ij} \leq V_j - V_i \leq u_{ij}, \forall (i,j) \in \mathcal{E}_\mathcal{R}, V_i \in \mathcal{V}_\mathcal{X} \cup \mathcal{V}_\mathcal{O}, V_j \in \mathcal{V}_\mathcal{X} \cup \mathcal{V}_\mathcal{O} \quad (1)$$

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

$$V_j - V_i = \tilde{d}_{ij}, \quad \forall (i,j) \in \mathcal{E}_\mathcal{T}, V_i \in \mathcal{V}_\mathcal{X} \cup \mathcal{V}_\mathcal{O}, V_j \in V_O \quad (2)$$

In this paper, we will make use of bounded intervals for the contingent edges. This frees us from the requirement for a probability distribution, though it is possible to also make further use of one. However, it is straightforward to relax this condition and not require the durations of contingent edges to be within an interval.

It is important to note that this proposed model is different from [LOS05] in that it distinguishes the set of executable points from observable points as well as controllable edges from contingent edges. Note that each contingent edge is incident on an observable time-point, and hence the end nodes of contingent edges are contained in the set of observable time-points. Note also that each observable time-point is associated with a unique contingent edge, since two contingent edges ending with the same observable time-point will in general cause the network to be inconsistent. We model the duration of each contingent edge (i, j) with a random variable $\tilde{d}_{ij}$ (or simply $\tilde{d}_j$ since it is unique to j), where we assume the mean and variance are known but not necessarily the underlying probability distributions. The remaining time-points are executable time-points whose values can be determined by the planner.

A *schedule* refers to an assignment of the executable time-points $\{V_1 = t_1, \ldots, V_n = t_n\}$ which are in $\mathcal{V}_\mathcal{X}$. 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 $V_i \in \mathcal{V}_\mathcal{X}$ given information on known contingent edges. We denote the decision for each such V_i as $V_i(\tilde{\boldsymbol{d}})$.

Definition 2 *A TCN $\mathcal{R}$ is said to be robust controllable (or simply controllable henceforth) iff for a given $0 \leq \epsilon < 1$, the probability that an execution strategy produces a schedule satisfying all the constraints in $\mathcal{E}_\mathcal{R}$ is at least $1 - \epsilon$. The problem of deciding, given a TCN and ϵ, whether the network is robust controllable is known as the RTCN problem.*

In this paper, we will focus mainly on the problem of robust dynamic controllability. In the experiments, we also present how to solve robust weak controllability using our approach.

Note that for dynamic controllability, the partial solution should depend only on the values of previous contingent edges at any time-point. Hence, for each executable point V_i, we define a dependency set $D(V_i)$ to be the set of duration random variables related to previous contingent edges with respect to V_i. Intuitively, this is determined by examining the constraint $[l_{ij}, u_{ij}]$ between V_i and each observable time-point V_j, where V_j is related if $u_{ij} < 0$ which means V_j will always

happen before V_i. Furthermore, to improve computational efficiency subsequently, we want these lower and upper bounds to be as tight as possible. To obtain $[l_{ij}, u_{ij}]$ for every pair of V_i and V_j, we adapt the algorithm by Dechter et al. [DMP91] for finding minimum network in an STN. Although this algorithm was used for a different purpose, the result of the algorithm was to obtain the tightest relation for every pair of time-points based on an interval for each edge. Our modified algorithm is described as follows.

Recall that a STN is a graph where each edge (i, j) is labeled by an interval $[l_{ij}, u_{ij}]$. In [DMP91], an STN $G = (V, E)$ is associated with a directed edge-weighted graph $G_d = (V, E_d)$ called distance graph. Each edge (i, j) is labeled 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) labeled with u_{ij} and edge (j, i) labeled with $-l_{ij}$ (The weight of an edge in distance graph could thus be negative). Graphs G and G_d are basically equivalent since $l_{ij} \leq t_j - t_i \leq u_{ij}$ is equivalent to $t_j - t_i \leq u_{ij}$ and $t_i - t_j \leq -l_{ij}$. 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 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 cannot be tightened any more.

We run the above algorithm and associate each pair of observable time-point V_j and executable time-point V_i with an interval $[u, w]$ where $u = -d_{ij}$ and $w = d_{ji}$. 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 $u \leq V_i - V_j \leq w$, we have following cases,

1. If $u \geq 0$, we have $V_i - V_j \geq 0$. It means the observable time-point V_j never happens after executable time-point V_i. Thus $\tilde{d}_j \in D(V_i)$.

2. If $w \leq 0$, we have $V_i - V_j \leq 0$. It means executable time-point V_i never happens after observable time-point V_j. Thus $\tilde{d}_j \notin D(V_i)$.

3. If $u < 0 \cap w > 0$, the chronological ordering of V_j and V_i is not clear. We assume $\tilde{d}_j \notin D(V_i)$ for this case.

Hence, the policy $V_i(\tilde{\boldsymbol{d}})$, which is used to decide when to execute each V_i, can be expressed in terms of a function on $D(V_i)$. Finding such a policy is generally difficult. The size of the policy space is potentially infinite if the underlying distribution is continuous. Even

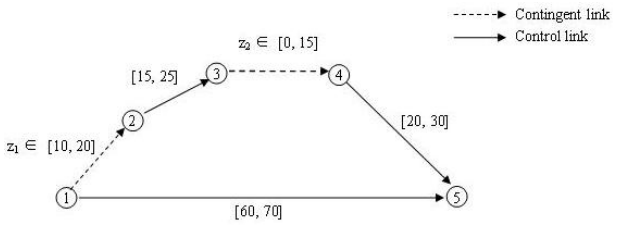

Figure 1. Example of a TCN

if the distributions are discrete, the size of the policy space is exponential. Hence, to ensure tractable computation, one needs to place a further restriction on the class of policies. Here, we restrict the policy $V_i(\tilde{\boldsymbol{d}})$ to be an affine function of the set $D(V_i)$ with decision variables V_{ij}:

$$V_i(\tilde{\boldsymbol{d}}) = V_{i0} + \sum V_{ik}\tilde{d}_k \quad \tilde{d}_k \in D(V_i) \tag{3}$$

Thus, $V_i(\tilde{\boldsymbol{d}})$ expresses the relationship between an executable time-point and observable time-points under dynamic controllability. Although an affine function is somewhat restrictive, finding a feasible policy is still a non-trivial problem. The variables V_{i0} and V_{ik} are the auxiliary decision variables in our model. In our resulting model, these constraint on V_i are required for all paths in the graph G to V_i with non-empty $D(V_i)$. Note that for an observable timepoint V_j with contingent edge (i, j), we have $V_j = V_i + \tilde{d}_{ij}$.

3.1 Example

We illustrate with an example as shown in Fig 1. Here, $\mathcal{V}_{\mathcal{X}} = \{V_1, V_3, V_5\}$ and $\mathcal{V}_{\mathcal{O}} = \{V_2, V_4\}$. The controllable edges $(2, 3), (4, 5), (1, 5)$ are defined by intervals $[15, 25]$, $[20, 30]$, $[60, 70]$ respectively. The contingent edges $(1, 2)$ and $(3, 4)$ are defined by random variables $\tilde{d}_1$ and $\tilde{d}_2$, having support in $[10, 20]$ and $[0, 15]$ respectively. Given $D(V_3) = \{\tilde{d}_1\}$ and $D(V_5) = \{\tilde{d}_1, \tilde{d}_2\}$, the execution policy is:

$$\begin{pmatrix} V_3(\tilde{d}) = V_{30} + V_{31}\tilde{d}_1 \\ V_5(\tilde{d}) = V_{50} + V_{51}\tilde{d}_1 + V_{52}\tilde{d}_2 \end{pmatrix} \tag{4}$$

We can view ϵ as the upper bound on the probability that at least one temporal constraint in (1) is violated. In other words, it defines the desired level of conservativeness we allow for temporal constraint violations. Note that when ϵ becomes 0, the set of feasible schedules are the ones which satisfy (1) under the worst case scenario. As we increases ϵ, we are in fact increasing the size of the set of feasible schedules.

Thus when a TCN is not robust controllable because its constraints are too tight, the planner can decide to increase the risk of temporal constraint violation by increasing the value of ϵ.

4 Solution Approach

In this paper, we make use of a new robust optimization result proposed in [CSS04], described briefly as follows. Consider an uncertain linear constraint, $\tilde{\boldsymbol{a}}'\boldsymbol{x} \geq b$ in which the coefficient vector $\tilde{\boldsymbol{a}}$ is subjected to random perturbation. We want to find feasible solutions such that, $\mathrm{P}(\tilde{\boldsymbol{a}}'\boldsymbol{x} \geq b) \geq 1-\epsilon$. In robust optimization, this is transformed to what is known as the "robust counterpart" given by

$$\boldsymbol{a}'\boldsymbol{x} \geq b \quad \forall \boldsymbol{a} \in \mathcal{G}_\Omega, \tag{5}$$

where $\mathcal{G}_\Omega$ is a compact uncertainty set parameterized by Ω, which directly controls the robustness of the solution. Intuitively, as seen later, Ω is inversely proportional to ϵ, which determines the size of the uncertainty set $\mathcal{G}_\Omega$. The larger the uncertainty set, the more scenarios are taken care of, and thus the safer is the resulting solution.

The form of uncertainty set depends on our knowledge of the probability distributions. [CSS04] introduced new deviation measures that help in designing uncertainty sets from knowledge of the probability distributions. We will adopt the same approach in mapping probability distributions of the *primitive uncertainties* to the uncertainty sets used in our robust optimization framework.

4.1 Uncertainty Model

As described in equation 3, we represent each contingent edge with a random variable $\tilde{d}_k$ specified with a mean, variance and bounded within an interval. Without loss of generality, we associate $\tilde{d}_k$ with a primitive random variable $\tilde{z}_k$ which has mean zero and support in $[-\underline{z}_k, \bar{z}_k]$, $\underline{z}_k, \bar{z}_k > 0$. We call $\{\tilde{z}_k\}_{k=1:N}$, the primitive uncertainties. Under the affine policies, we can express a model with uncertainties solely by using the primitive uncertainties, $\tilde{\boldsymbol{z}}$.

Operating on random variables with known distributions is generally cumbersome, e.g. deciding whether the α-quantile of a linear sum of random variables exceeds certain value is usually computationally intractable. The key idea in robust optimization is to approximate probabilistic constraints with tractable formulations by incorporating mild distribution information such as the support and deviation parameters. [CSS04] introduces two new parameters: forward and backward deviation parameters, which we use to obtain nontrivial probability bounds against temporal constraint violation.

Let $\tilde{z}$ be a random variable with zero mean and bounded support. Then, there exists a pair of associated deviation parameters $p_{\tilde{z}}$ and $q_{\tilde{z}}$ such that for any value $\phi \geq 0$, the probability that $\tilde{z}$ deviates from the range of values $[-\phi q_{\tilde{z}}, \phi p_{\tilde{z}}]$ is exponentially small in ϕ. Intuitively, this is a much stronger form of Chebyshev's inequality. [CSS04] shows that indeed such parameters are well-defined and non-empty. More precisely, they show that if $\tilde{z}$ is symmetric in $[-1, 1]$, then one may define $p_{\tilde{z}} = q_{\tilde{z}} = 1$ and if $\tilde{z}$ is bounded in $[-1, 1]$ but not necessarily symmetric, then one may define $p_{\tilde{z}} = q_{\tilde{z}} = 1.07$. While there is still the difficulty of obtaining closed form solutions for $p_{\tilde{z}}$ and $q_{\tilde{z}}$, one can determine their values numerically by simulation. For instance, if $\tilde{z}$ is uniformly distributed over $[-1, 1]$, simulation shows that $p_{\tilde{z}} = q_{\tilde{z}} = 0.58$.

Suppose the contingent edges can be represented by their primitive bounded random variables $\{\tilde{z}_k\}_{k=1:N}$ with deviation parameters p_k and q_k, $\forall k = \{1, \ldots, N\}$. To determine whether the network is controllable with probability $1-\epsilon$, we define a uncertainty budget Ω associated with ϵ, and a corresponding uncertainty set parameterized by Ω as follows,

$$\begin{aligned} \mathcal{G}_\Omega = \big\{ \boldsymbol{z} \ : \ & \exists \boldsymbol{v}, \boldsymbol{w} \in \Re^N_+, \boldsymbol{z} = \boldsymbol{v} - \boldsymbol{w}, \\ & \|\boldsymbol{P}^{-1}\boldsymbol{v} + \boldsymbol{Q}^{-1}\boldsymbol{w}\|_2 \leq \Omega, -\underline{\boldsymbol{z}} \leq \boldsymbol{v} - \boldsymbol{w} \leq \bar{\boldsymbol{z}} \big\} \end{aligned} \tag{6}$$

where $\boldsymbol{P} = \mathrm{diag}(p_1, \ldots, p_N)$ and $\boldsymbol{Q} = \mathrm{diag}(q_1, \ldots, q_N)$.

Note that the size of $\mathcal{G}_\Omega$ is inversely proportional to the value of ϵ, meaning that the more conservative (i.e. the smaller the value of ϵ), the larger the size of the uncertainty set we contend with. The relationship between ϵ and Ω will become clear later.

The following theorem shows that checking the feasibility of a system of inequalities $y(\boldsymbol{z}) \geq 0$ over different instantiations of $\boldsymbol{z}$ taken from a given uncertainty set $\mathcal{G}_\Omega$ can be formulated as a second-order cone program:

Theorem 1 *[CSS04] Let*

$$y(\boldsymbol{z}) = y_0 + \sum_{k=1}^{N} y_k z_j.$$

The robust counterpart of

$$y(\boldsymbol{z}) \geq 0 \quad \forall \boldsymbol{z} \in \mathcal{G}_\Omega, \tag{7}$$

is equivalent to

$$\left\{\begin{array}{l} y_0,\ldots,y_N \ : \\ \exists v_0 \in \Re, \boldsymbol{v}, \boldsymbol{r}, \boldsymbol{s} \in \Re^N \\ \ y_0 \geq \Omega v_0 + \boldsymbol{r}'\bar{\boldsymbol{z}} + \boldsymbol{s}'\underline{\boldsymbol{z}} \\ \ v_k \geq q_k(y_k - s_k + r_k) \ \ \forall k \in \{1,\ldots,N\}, \\ \ v_k \geq -p_k(y_j - s_k + r_k) \ \ \forall k \in \{1,\ldots,N\}, \\ \ \|\boldsymbol{v}\|_2 \leq v_0 \\ \ \boldsymbol{r}, \boldsymbol{s} \geq \boldsymbol{0}. \end{array}\right\} \tag{8}$$

where $y_i \in \Re$ are decision variables, $\boldsymbol{v}, \boldsymbol{r}, \boldsymbol{s}$ are auxiliary variables and p_k, q_k, Ω are input parameters. In addition, if $y_0,\ldots,y_N$ satisfies the robust counterpart (8), then

$$\mathrm{P}(y(\boldsymbol{z}) \geq 0) \geq 1 - \exp(-\Omega^2/2).$$

In the above formulation, $\|\boldsymbol{v}\|_2$ refers to the L2 norm and hence can be solved in polynomial time using second order cone programming [AG03].

4.2 Solving RTCN

We are now ready to solve the RTCN problem. Recall that we want to know whether the probability that a temporal constraint network being controllable is at least $1-\epsilon$ over all uncertainty scenarios. For each scenario represented by $\boldsymbol{z}$, we check if an execution strategy exists. Define the set

$$\chi(\boldsymbol{z}) = \{\boldsymbol{V} : l_{ij} \leq V_j(\boldsymbol{z}) - V_i(\boldsymbol{z}) \leq u_{ij} \ \ \forall (i,j) \in \mathcal{E}\}.$$

The goal is to determine whether,

$$\mathrm{P}(\{\chi(\tilde{\boldsymbol{z}}) \neq \emptyset\}) \geq 1 - \epsilon. \tag{9}$$

Equation 3 expresses each executable time-point with dependencies as an affine function of random variables associated with the relevant contingent links. Thus, equation 9 is equivalent to determining whether there exists a policy $V(\tilde{\boldsymbol{z}})$:

$$\mathrm{P}\left(\left\{\begin{array}{l} V_j(\tilde{\boldsymbol{z}}) - V_i(\tilde{\boldsymbol{z}}) - l_{ij} \geq 0 \ \ \forall (i,j) \in \mathcal{E}_{\mathcal{R}} \\ V_i(\tilde{\boldsymbol{z}}) - V_j(\tilde{\boldsymbol{z}}) + u_{ij} \geq 0 \ \ \forall (i,j) \in \mathcal{E}_{\mathcal{R}} \end{array}\right\}\right) \geq 1-\epsilon \tag{10}$$

Note that we have $2|\mathcal{E}_{\mathcal{R}}|$ number of inequalities from equation 10 (having the same form as equation 7). By choosing $\Omega = \sqrt{-2\ln(\epsilon/2|\mathcal{E}_{\mathcal{R}}|)}$ and setting the values of p, q by simulation, we can transform the inequalities to their robust counterparts using Theorem 1 and solve the resulting second order cone program in polynomial time.

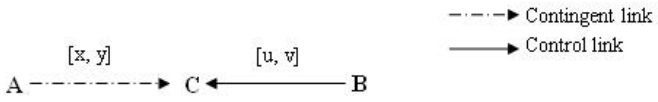

Figure 2. Dependency Enhancement

5 Further Enhancements

We now present two refinements to improve the quality of results. The first one is a more refined execution policy. The second is a heuristic to allocate the failure probability among constraints which is intended to decrease the total failure probability.

Execution Policy Enhancement Consider a pair of observable time-point C and executable time-point B depicted in Fig. 2 where (A, C) is the contingent edge associated with C. Suppose $[u, v]$ is the constraint between B and C. When $u \geq 0$, it means B precedes C, C is not included in the dependency set of B. When $v \leq 0$, it means B follows C, C is added to the dependency set. However for some of the derived edges (i.e. those not existing in the 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 which gives,

$$\begin{array}{l} u \leq t_C - t_B \leq v \\ \tilde{z} \notin D_B \end{array} \tag{11}$$

Morris and Vidal [MV01] have analyzed this special case, and formulated the rule that if the 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 time-point A. Here $y - v$ serves as a breakpoint. We have the following policy refinement for the special case when $u < 0$ and $v > 0$:

$$\begin{array}{lll} \textit{when} \ \ \tilde{z} < y - v & u \leq t_C - t_B \leq 0 & \\ & \tilde{z} \in D_B & \\ \textit{when} \ \ \tilde{z} \geq y - v & t_B - t_A \geq y - v & \\ & u \leq t_C - t_B \leq v & \\ & \tilde{z} \notin D_B & \end{array} \tag{12}$$

The refined policy says the system allows an observation time within $y - v$ units. On the 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 the 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 the dependency set D_B and reserve the constraint on $B \rightarrow C$.

Having this new policy would result in larger feasible set because it uses a rule without which, the constraint in equation 11 may fail to be satisfied.

Enhancing the ϵ_{ij} Allocation Given a failure probability ϵ, a TCN is not controllable if it fails to satisfy the constraint 10. However, as our solution admits the possibility of failure, the network may actually be controllable with the given ϵ failure probability especially when replacing the given ϵ with a slightly larger value makes it controllable. In [CSS04], the failure probability ϵ is divided equally among each constraint in order to minimize the total uncertainty budget. Let ϵ_{ij} be the failure probability associated with each constraint. In the following, we discuss how to refine the procedure of allocating ϵ_{ij} to gain an improved result.

Intuitively if the constraint system is not feasible, we would expect it to be due to violations of only a subset of the constraints. If we can unload excessive budget from the more easily violated constraints and transfer it to the less easily violated ones, then the TCN may become controllable. Our idea is to try to improve the solution by assigning unequal ϵ_i to different constraints by some *heuristics*. Now the constraint on each controllable edge becomes:

$$\mathrm{P}\left(\{l_{ij} \leq V_j(\tilde{\boldsymbol{z}}) - V_i(\tilde{\boldsymbol{z}}) \leq u_{ij}\}\right) \geq 1 - \epsilon_{ij} \quad \forall (i,j) \in \mathcal{E}_R \tag{13}$$

It seems reasonable to assume that constraints with a wider interval tend to have a larger probability P in the above expression. Thus, we expect that constraints with a larger $u_{ij} - l_{ij}$ value are also more likely to be satisfied. This leads to the following heuristic.

Suppose a TCN is not controllable with failure probability ϵ but controllable with a slightly larger failure probability ϵ' given that we have allocated equal ϵ'_{ij} to each controllable edge (i,j). We rank controllable edges by sorting on the interval length $len_{ij} = u_{ij} - l_{ij}$ in decreasing length. The assumption is that a higher rank means the the constraint is less likely to be infeasible. Thus we choose to increase Ω_{ij} (consequently decrease ϵ_{ij} since the larger Ω'_{ij} is, the smaller ϵ'_{ij} becomes) by a suitable predefined amount until it causes violation again. We repeat this process by trying the edges sequentially in their rank ordering. If at any stage, the accumulated failure probability decreases to the original ϵ, we are done. Note that edges with higher rank do not necessarily sustain a larger budget than edges with a lower rank. The reason behind sorting the interval list is to reach a given ϵ more quickly due to our assumptions that larger len_{ij} value support a larger budget.

6 Experimental Results

We have conducted some preliminary experiments to study the behavior of the worst-case (i.e. minimum) failure probability ϵ such that a given TCN is still controllable. All our algorithms are implemented in Matlab v7.0 and use the second-order cone solver in MOSEK version 4 (`http://www.mosek.com`). The hardware platform is a Pentium 4-2.4GHz processor with 256MB RAM under Windows XP.

6.1 Impact of Edge Density and Contingent Ratio

We first study how ϵ is affected by the structure of the network, in terms of edge density and contingent ratio. The edge density is defined as the ratio of the number of edges to the maximum number of possible edges for a given network. The contingent ratio C is defined as the ratio of the number of observable time-points to the total number of time-points, thus $C \leq 1$. We generate random instances of TCNs based on the TimeNetManager [COS99] network generation algorithms. This generates sets of temporal networks controlled by a set of macro-parameters that characterize the topology and the temporal flexibility of the networks. We extend their algorithm to generate TCNs by adding additional parameters to set contingent edges and distribution of temporal durations.

We fix the node size to be 20, and vary edge density (between 0.2 and 0.5) and contingent ratio $C = 0.1$ to 0.4. For larger values of edge density and contingent ratio, the random instances generated are almost always non-feasible, and hence we exclude them in our experiments. In our experiment, we determine the optimal value of ϵ, i.e. the smallest value such that the network is still controllable, through binary search. This makes use of the fact that if the network is controllable for a given ϵ, then it is also controllable for any large ϵ'. For each setting, we ran 8 random instances and plot the average optimal value of ϵ.

The CPU time required is between 12 to 25 seconds and the results are shown in Fig. 3. This shows our approach is efficient, since conventional stochastic programming approaches can take considerably longer. From Fig. 3, we observe that:

1. As the edge density increases, ϵ also increases in a linear fashion; and

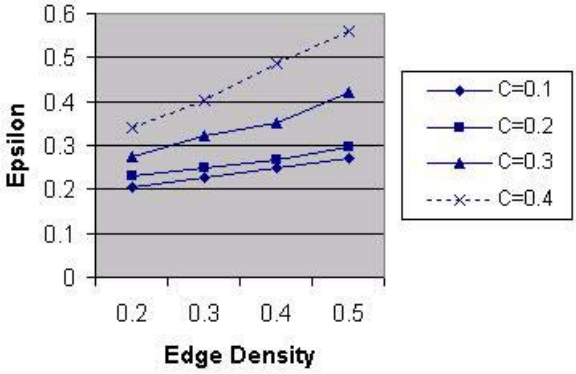

Figure 3. Impact of edge density & contingent ratio

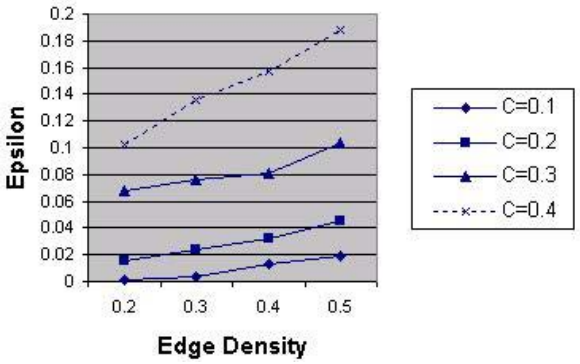

Figure 4. Effect of Enhancing the ϵ allocation

2. As C increases, the rate of increase of ϵ also increases (as the slope is steeper).

Next, we repeat the same set of experiments, this time using the ϵ allocation strategy proposed in Section 5. The results in Fig. 4 shows that the optimal value of ϵ is significantly reduced to a much smaller value compared to those shown in Fig. 3. This demonstrates the effectiveness of our proposed allocation strategy. The CPU time required is however much longer, between 80 to 170 seconds.

6.2 Robust Dynamic versus Weak Controllability

In the deterministic context, dynamic controllability implies weak controllability. It is not obvious what how this relationship generalizes in the probabilistic context. More precisely, we wish to investigate the gap in the value of ϵ between dynamic and weak controllability on the same network. Weak controllability is a simple modification of our approach for dynamic controllability since the difference lies in the dependence

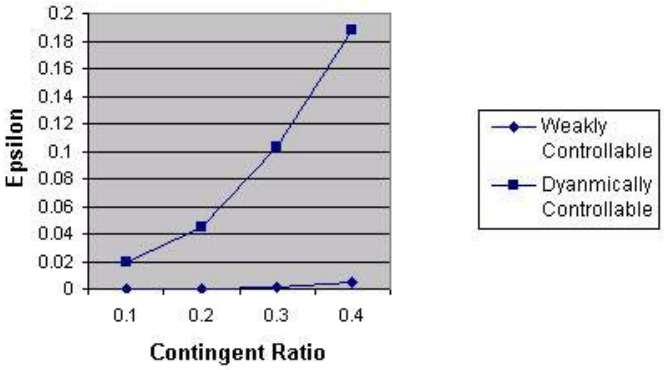

Figure 5. Failure prob. of dynamic vs weak controllability

set. We can check weak controllability by letting each executable time-point be dependent on all the random variables and using the same solution approach. The experimental results with node size 20 and edge density 0.5 are given in Fig. 5.

We observe from the results that: (i) ϵ increases with C for both dynamic and weak controllability; and (ii) the rate of increase of ϵ grows faster in dynamic than weak controllability.

7 Conclusion

We presented what we believe to be the first computationally tractable and efficient approach to handle dynamic controllability of probabilistic temporal constraint networks. This approach uses the techniques from robust optimization which have not been used in uncertain temporal networks. Experimentally, we demonstrate the efficacy of our approach on a range of networks. The results obtained are aligned to intuition and a couple of surprising observations have been made. We believe that a theoretical analysis to explain the experimental results is possible, though we have been unable to achieve it at the time of writing of this paper.

References

[AG03] F Alizadeh, D Goldfarb (2003): Second-order cone programming, *Mathematical Programming* 95(1), 3-51.

[BN00] Ben-Tal, A. and Nemirovski, A. (2000): Robust solutions of Linear Programming problems

contaminated with uncertain data, *Mathematical Programming*, 88, 411-424.

[BS04] Bertsimas, D. and Sim, M. (2004): Price of Robustness, *Operations Research*, 52(1), 35-53.

[COS99] Cesta, A., Oddi A., and Sussi A. (1999): TimeNetManager - A Software Tool for Generating Random Temporal Networks, in *Proc. AI*IA*, 143-154.

[CSS04] Chen X., Sim M., and Sun P. (2004): A Robust Optimization Perspective to Stochastic Programming.`https://netfiles.uiuc.edu/xinchen/www/papers/`. Submitted to *Operations Research*

[DMP91] Dechter R., Meiri I., and Pearl J. (1991): Temporal Constraint Networks, *Artificial Intelligence*, 49, 91-95.

[LOS05] Lau H., Ou T., and Sim M. (2005): Robust Temporal Constraint Network, in *Proc. ICTAI 2005*.

[NCBS97] Marin R. , Cardenas M. A., Balsa M., and Sanchez J.L. (1997): Obtaining solutions in fuzzy constraint networks, *International Journal of Approximate Reasoning*, 16, 261-288.

[MM00] Morris, P., and Muscettola, N. (2000): Execution of temporal plans with uncertainty, in *Proc AAAI 2000*, 491-496.

[MV01] Morris, P. and Vidal, T. (2001): Dynamic Control Of Plans With Temporal Uncertainty, in *Proc. IJCAI 2001*, 494-502.

[MM05] Morris P. and Muscettola, N. (2005): Temporal Dynamic Controllability Revisited, in *Proc. AAAI 2005*, 1193-1198.

[RVY04] F. Rossi, K. B. Venable, N. Yorke-Smith,: Controllability of Soft Temporal Constraint Problems, in *Proc. CP 2004*, 588-603.

[VG96] Vidal, T. and Ghallab, M (1996): Dealing with uncertain durations in temporal constraint networks dedicated to planning, in *Proc. ECAI 1996*, 48-52.

[VF99] Vidal, T. and H. Fragier (1999): Handling consistency in temporal constraint networks: from consistency to controllabilities, *Journal of Experimental & Theoretical AI*, 11, 23-45.

[T02] Tsamardinos I. (2002): A Probabilistic Approach to Robust Execution of Temporal Plans with Uncertainty. In Vlahavas and Spyrpoulos (Eds), in *SETN 2002, LNAI (2308)*, 97-108.

[TPR03] Tsamardinos I., Pollack M. E., Ramakrishnan S.(2003): Assessing the Probability of Legal Execution of Plans with Temporal Uncertainty, *Proc. ICAPS-03 Workshop on Planning under Uncertainty and Incomplete Information*.