Optimising Bounds in Simple Temporal Networks with Uncertainty under Dynamic Controllability Constraints

Jing Cui
ANU & NICTA
u5392410@anu.edu.au

Peng Yu, Cheng Fang
MIT
{yupeng,cfang}@mit.edu

Patrik Haslum
ANU & NICTA
patrik.haslum
@anu.edu.au

Brian C. Williams
MIT
williams@mit.edu

Abstract

Dynamically controllable simple temporal networks with uncertainty (STNU) are widely used to represent temporal plans or schedules with uncertainty and execution flexibility. While the problem of testing an STNU for dynamic controllability is well studied, many use cases – for example, problem relaxation or schedule robustness analysis – require optimising a function over STNU time bounds subject to the constraint that the network is dynamically controllable. We present a disjunctive linear constraint model of dynamic controllability, show how it can be used to formulate a range of applications, and compare a mixed-integer, a non-linear programming, and a conflict-directed search solver on the resulting optimisation problems. Our model also provides the first solution to the problem of optimisation over a probabilistic STN subject to dynamic controllability and chance constraints.

Introduction

The simple (non-disjunctive) temporal network with uncertainty, or STNU (Vidal and Fargier 1999), is a widely used model for representing schedules or temporal plans that have both uncertainty about the timing of some events (for example, the time needed to complete an activity) and flexibility for the executing agent to choose the timing of other events (for example, the time to start an activity). An STNU is a constraint satisfaction problem over real-valued time point variables, with constraints that are (upper and lower) bounds on the differences between pairs of variables. However, some time points are *uncontrollable*, meaning that in any execution of the schedule or plan, their values will be chosen non-deterministically (by the environment) within the given bounds, while the values of remaining time point variables are chosen by the executing agent, subject to the constraints.

If there is no uncertainty, it suffices to verify that the STN is consistent to know that executing agent will be able to meet the constraints. Consistency of an STN can be formulated as a set of linear constraints. An STNU, however, requires *controllability*, meaning, informally, that the executing agent has a valid (constraint-satisfying) response to any choice by the environment. If the agent can make this choice before making any observation, the network is strongly controllable. The more practically useful, but also more complex, property of dynamic controllability means the agent can choose a value for each controllable time point variable using only observations of uncontrollable events that have taken place earlier, such that all constraints will be respected under any outcome of future uncertainties.

It is known that deciding if a *given* STNU is dynamically controllable can be done in polynomial time (Morris, Muscettola, and Vidal 2001). The problem that we consider is optimising an objective function over the bounds on time point differences, subject to the constraint that the network is dynamically controllable. This has a broad range of applications. As illustrative examples, we consider minimally relaxing an over-constrained (non-controllable) STNU to make it dynamically controllable (Yu, Fang, and Williams 2014); finding the minimum schedule flexibility needed to maintain dynamic controllability (Wah and Xin 2004); maximising different measures of schedule robustness; and optimising a preference function over a probabilistic STN with chance constraints (Fang, Yu, and Williams 2014).

The problem of optimising time bounds under dynamic controllability was previously considered by Wah and Xin (2004), who formulated a non-linear constraint optimisation model. In fact, dynamic controllability is a disjunctive linear constraint, and using this insight we consider several alternative ways of dealing with it, including a conflict-driven search (Yu, Fang, and Williams 2014), a formulation as a mixed-integer linear program with 0/1 variables, and the non-linear encoding proposed by Wah and Xin.

Background

Formally, an STNU (Vidal and Fargier 1999) consists of a set of nodes $X = X_E \cup X_U$, representing executable (X_E) and uncontrollable (X_U) time points, and a set of links $E = R \cup C$, called *requirement* and *contingent* links. Each link e_{ij} has a lower bound L_{ij} and upper bound U_{ij}, representing the constraints $L_{ij} \leq t_j - t_i \leq U_{ij}$. Each uncontrollable time point has exactly one incoming contingent link, whose lower bound is non-negative. In other words, executable time points correspond to choices of the agent, contingent links represent uncontrollable durations, and the uncontrollable time points are when the agent finds out what duration the environment has chosen. Requirement links may connect any pair of time points.

A *schedule* is an assignment of values to all time points.

The schedule is consistent iff it satisfies the bounds of all requirement links. A *projection* of an STNU replaces each contingent link $e_{ij} \in [L_{ij}, U_{ij}]$ with a requirement link $e_{ij} \in [d, d]$ for some $L_{ij} \leq d \leq U_{ij}$. The resulting network represents a possible outcome of the uncontrollable choices, and has no remaining uncertainty; it is called an STN. In an STN, the tightest bounds on the difference between any two time points that are implied by the given links can be computed in polynomial time by solving the set of linear constraints (1). Thus, there is, implicitly, a requirement link between every pair of time points. If any link's bounds are inconsistent ($L_{ij} > U_{ij}$), the network has no satisfying schedule (Dechter, Meiri, and Pearl 1991).

An *execution strategy*, S, maps projections of the STNU to schedules. S is *valid* iff $S(\pi)$ is consistent for every projection π. Given a schedule, T, and a time point $x \in X$, $T_{<x}$ is the restriction of T to all time points scheduled before x. An execution strategy S is *dynamic* iff $S(\pi_1)_{<x} = S(\pi_2)_{<x}$ implies $S(\pi_1)(x) = S(\pi_2)(x)$ for all projections π_1, π_2 and executable time point x. This means that the time the strategy assigns to the executable point x can only depend on uncontrollable durations observed earlier. An STNU is *dynamically controllable* iff there exists a valid dynamic execution strategy for it.

Problem Formulation

The general form of the optimisation problem can be stated as follows: We are given the structure of an STNU, that is, the set of time points $X = X_E \cup X_U$ and links $E = R \cup C$, but not the upper and lower bounds on (all) links, and an objective function. The problem is then to set those bounds so as to optimise the objective function value:

$$\begin{array}{ll} \text{opt} & f_{obj}(l_{ij}, u_{ij} \mid e_{ij} \in E) \\ \text{s.t.} & L_{ij} \leq l_{ij} \leq u_{ij} \leq U_{ij} \\ & N(l_{ij}, u_{ij} \mid e_{ij} \in E) \text{ is dynamically controllable} \\ & \text{application-specific side constraints} \end{array}$$

The decision variables, l_{ij} and u_{ij}, represent the lower and upper bounds on link e_{ij}. Thus, a satisfying assignment defines an STNU, $N(l_{ij}, u_{ij} \mid e_{ij} \in E)$, and this STNU must be dynamically controllable. The main contribution of this paper is the formulation of dynamic controllability as set of disjunctive linear constraints. We then show how these constraints can be reformulated into a mixed-integer linear program (MIP) or a non-linear program (NLP), which can be given to different optimisation solvers.

L_{ij} and U_{ij} are constants, which constrain the range of the bounds variables. In principle, these constants are not needed. They can be set to $-\infty$ and ∞, respectively, leaving the bounds variables unrestricted. Many of the application problems we consider (cf. section after next), however, specify narrower ranges, and we use this to simplify the constraint formulation.

Constraint Model of Dynamic Controllability

Checking dynamic controllability of an STNU was first shown to be tractable by Morris, Muscettola and Vidal (2001). Their algorithm repeatedly applies a set of reductions, tightening the bounds on requirement links, until no more reductions apply (in which case the network is controllable) or the network becomes inconsistent (implying it is not controllable). Our constraint model uses essentially the same reduction rules, but in the form of constraints between the decision variables that represent link bounds. Therefore, we briefly review the algorithm before presenting the model. For a detailed explanation, we refer to their paper.

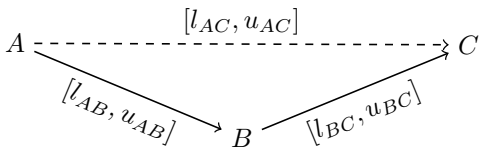

Figure 1: An STNU triangle. The A–C link is contingent.

The DC Checking Algorithm

Morris, Muscettola and Vidal's (2001) algorithm repeatedly examines each triangle of time points in the STNU, considering at most one contingent link each time. Figure 1 shows a triangle, with time points A, B and C. The link between A and C is contingent, the other two are requirement links. (If e_{AB} is also contingent, it is considered in a separate triangle.) First, the algorithm applies the implied (shortest path) bounds (e.g., $L_{BC} \leftarrow \max(L_{BC}, L_{AC} - U_{AB})$ and $U_{BC} \leftarrow \min(U_{BC}, U_{AC} - L_{AB})$).

Recall that there is an implicit requirement link between each pair of time points, including parallel to each contingent link. If the implied requirements on a pair of variables connected by a contingent link are tighter than the contingent links' bounds, the contingent link is said to be "squeezed". In this case, the network cannot be controllable.

The next step depends on the relation between B and C:

- If $U_{BC} < 0$, B must be scheduled after C (hence, after C has been observed), so no further adjustments are needed. This is called the "follow" case.
- If $L_{BC} \geq 0$, B must be scheduled before or simultaneously with C (i.e., before C has been observed). This is called the "precede" case, and the bounds on the e_{AB} link are updated to $L_{AB} \leftarrow \max(L_{AB}, U_{AC} - U_{BC})$ and $U_{AB} \leftarrow \min(U_{AB}, L_{AC} - L_{BC})$.
- If $L_{BC} < 0$ and $U_{BC} \geq 0$, B may be scheduled before or after C. This case, called the "unordered" case, introduces a conditional bound, called a "wait", $\langle C, U_{AC} - U_{BC} \rangle$ on e_{AB}, with the meaning that execution of B must wait for either C to occur or at least $U_{AC} - U_{BC}$ after A. If $U_{AC} - U_{BC} \leq L_{AC}$, C cannot occur before the wait expires, so the wait is replaced by the unconditional bound $L_{AB} \leftarrow \max(L_{AB}, U_{AC} - U_{BC})$. Otherwise, $L_{AB} \leftarrow \max(L_{AB}, L_{AC})$, since the wait for C will delay B to at least L_{AC} after A.

Tighter bounds on a requirement link propagate to any other triangle that the link is part of. In addition, the algorithm performs "wait regression", in which conditional bounds are propagated to other links. If $\langle C, w \rangle$ is a wait on e_{AB}, where $w \leq U_{AC}$, then (i) if there is any link e_{DB} with upper bound U_{DB}, a wait $\langle C, w - U_{DB} \rangle$ is added to e_{AD}; and (ii) if $w \geq 0$

and there is a contingent link e_{DB}, where $B \neq C$, with lower bound L_{DB}, a wait $\langle C, w - L_{DB}\rangle$ is added to e_{AD}.

Disjunctive Linear Model

Constraints are formulated over each triangle of nodes in the STNU, considering at most one contingent link each time.

Shortest path constraints

$$\begin{array}{rcl} l_{AC} & \leq & u_{AB} + l_{BC} \leq u_{AC} \\ l_{AC} & \leq & l_{AB} + u_{BC} \leq u_{AC} \\ u_{AC} & \leq & u_{AB} + u_{BC} \\ & & l_{AB} + l_{BC} \leq l_{AC} \end{array} \quad (1)$$

The shortest path constraints can propagate in any direction (i.e., from the contingent link e_{AC} to the requirement links and vice versa.) This may seem contradictory, since a contingent link may not be squeezed by requirements. However, l_{AC} and u_{AC} here are variables, whose values will be the bounds on the contingent link. In some applications (e.g., the problem of minimising flexibility which motivated Wah and Xin) these variables are fixed to given constant values but other applications (e.g., problem relaxation) allow the bounds of contingent links to vary.

If no link in the triangle is contingent, these are the only constraints. Assuming, w.l.o.g., that the e_{AC} link is contingent, what constraints are needed is determined by the outer bounds on the e_{BC} link, following the cases in Morris, Muscettola and Vidal's (2001) algorithm. If $U_{BC} < 0$, then $u_{BC} < 0$ and the triangle must always be in the follow case. Thus, no additional constraints are needed.

Precede constraints If $L_{BC} \geq 0$, then $l_{BC} \geq 0$ and the triangle will be in the precede case. The following constraints must hold:

$$\begin{array}{rcl} u_{AB} & \leq & l_{AC} - l_{BC} \\ l_{AB} & \geq & u_{AC} - u_{BC} \end{array} \quad (2)$$

This together with (1) is equivalent to

$$\begin{array}{rcl} u_{AB} & = & l_{AC} - l_{BC} \\ l_{AB} & = & u_{AC} - u_{BC} \end{array} \quad (2')$$

since $l_{AB} \leq u_{AB}$ is always required. If $L_{BC} < 0$ and $U_{BC} \geq 0$, the triangle can be in any case, depending on the values given to l_{BC} and u_{BC}. The precede constraint then becomes disjunctive:

$$(l_{BC} < 0) \ \vee \ \left(\begin{array}{rcl} u_{AB} & \leq & l_{AC} - l_{BC} \\ l_{AB} & \geq & u_{AC} - u_{BC} \end{array} \right) \quad (3)$$

Triangular wait constraints If it is possible that the triangle may be in the unordered case ($L_{BC} < 0$ and $U_{BC} \geq 0$), a variable representing the conditional wait bound is added:

$$w_{ABC} \ \geq \ u_{AC} - u_{BC} \quad (4)$$

Regression of waits (described below) may introduce wait variables w_{ABX}, where X is any uncontrollable time point (not necessarily in the same triangle as A and B). For each requirement link e_{AB} and wait variable w_{ABX}, the following disjunctive constraint must hold:

$$l_{AB} \ \geq \ \min(l_{AX}, w_{ABX}) \quad (5)$$

If $U_{AB} \leq L_{AX}$, this simplifies to $w_{ABX} = l_{AB}$. The constraint $w_{ABX} \leq u_{AB}$ must also hold.

Wait regression Each wait bound $\langle X, t\rangle$ on a link e_{AB} is represented by a variable w_{ABX}. If there is a wait w_{ABX} and a contingent link e_{DB}, then wait regression implies the constraint

$$(w_{ABX} < 0) \ \vee \ (\ w_{ADX} \ \geq \ w_{ABX} - l_{DB} \ ) \quad (6)$$

It is disjunctive because the regression only applies when $w_{ABX} \geq 0$. The weaker constraint

$$w_{ADX} \ \geq \ w_{ABX} - u_{DB} \quad (7)$$

holds in all cases (i.e., also when e_{DB} is a requirement link, or the wait is negative).

Correctness

The correctness of our constraint formulation can be shown to follow from that of Morris, Muscettola and Vidal's (2001) DC checking algorithm. Their algorithm applies a set of reduction rules, which tighten the bounds on links, until quiescence; if the network at that point is consistent, the original network is dynamically controllable.

Let $N(l_{ij}, u_{ij} \mid e_{ij} \in E)$ be an STNU that is defined by a solution to our model, i.e., constraints (1)–(7). We can then show that applying all the reduction rules from the DC checking algorithm to this STNU will not result in a tightening of any bound. That is, the STNU is already at quiescence. Since it must satisfy the shortest path constraints (1), it is also consistent. Thus, the DC checking algorithm applied to N will report that it is dynamically controllable.

Reducing the Size of the Model

The model formulated above has up to $O(n^3)$ constraints and $O(|X_U|n^2)$ variables, where $n = |X|$ is the number of time points. Wah and Xin (2004; 2007) proposed several rules for eliminating redundant constraints and variables from the model. Although there is, in principle, a requirement link for every pair of time points, not all of these must be represented with decision variables. For example, if, in Figure 1, $U_{BC} < 0$ so that B must follow C, only the shortest path constraints apply to l_{AB} and u_{AB}, in this triangle. If there are no other constraints on e_{AB} these can always be satisfied, in which case it is not necessary to include them. Any link with bounds constrained in the input STNU must be represented (unless fixed to a constant), as must any link whose bounds can potentially be tightened by precede or wait constraints. Among the remaining implicit links, which are subject only to shortest path constraints, a sufficient set is found by a triangulation of the network.

Formulation as a MIP

The disjunctions in constraints (3) and (5) mean the model is not a linear program. Wah and Xin (2004) used non-linear constraints to encode the disjunctions (as explained in the next section) and tackled it with the non-linear programming solver SNOPT. As an alternative, we formulate a mixed-integer linear programming (MIP) formulation, where disjunctions are encoded using binary (0/1) variables. Although

MIP is an NP-hard problem, MIP solvers such as CPLEX or Gurobi are often very efficient in practice, and, in particular, typically more efficient than non-linear solvers. Experiment results across all application problems confirm this.

The wait bound constraint (5) can be formulated as

$$\text{if } \alpha > 0 \text{ then } \beta \geq 0 \text{ else } \gamma \geq 0$$

where $\alpha = w_{ABX} - l_{AX}$, $\beta = l_{AB} - l_{AX}$ and $\gamma = l_{AB} - w_{ABX}$ are all linear expressions. The disjunction can be replaced by the following linear constraints

$$\alpha - xU_\alpha \leq 0 \quad (8a)$$
$$\alpha - (1-x)(L_\alpha - 1) > 0 \quad (8b)$$
$$\beta - (1-x)L_\beta \geq 0 \quad (8c)$$
$$\gamma - xL_\gamma \geq 0 \quad (8d)$$

where $x \in \{0, 1\}$ is a binary variable, L_α, L_β and L_γ are constant lower bounds on α, β and γ, respectively, and U_α is a constant upper bound on α. This forces $\alpha > 0$ and $\beta \geq 0$ when $x = 1$, and $\alpha \leq 0$ and $\gamma \geq 0$ when $x = 0$. In the wait bound constraint, where $\alpha = w_{ABX} - l_{AX}$, we can choose $U_\alpha = U_{AB} - L_{AX}$, because U_{AB} is an upper bound on w_{ABX} ($w_{ABX} \leq u_{AB} \leq U_{AB}$) and L_{AX} is a lower bound on l_{AX}. The wait w_{ABX} is lower-bounded by the maximum of all wait constraints – triangular and regressed – on e_{AB}. The triangular wait lower bound is $w^t_{ABX} = u_{AX} - u_{BX}$, and from the shortest path constraint $l_{AB} + u_{BX} \leq u_{AX}$ we have $w^t_{ABX} \geq l_{AB}$. Thus, we can choose $L_\alpha = L_{AB} - U_{AX}$. For the lower bounds on $\beta = l_{AB} - l_{AX}$ and $\gamma = l_{AB} - w_{ABX}$ we have $L_\beta = L_{AB} - U_{AX}$ and $L_\gamma = L_{AB} - U_{AB}$, respectively.

The precede constraint (3) and regressed wait bound (6) are similar, except they have conditions only in one of the two cases (either "then" or "else"). Where one side of a disjunction consists of (a conjunction of) several linear constraints, as in (3), it is only necessary to add a constraint like (8c) for each conjunct, all using the same binary variable.

The wait regression constraint (6) can be strengthened to

$$(w_{ABX} \leq l_{AX}) \vee (w_{ADX} \geq w_{ABX} - l_{DB}) \quad (6')$$

The advantage of this is that one disjunct, $w_{ABX} \leq l_{AX}$, is the same as the branching condition in (5), so both constraints can be captured with one binary variable.

Constraint (6') is valid because regressing a wait w_{ABX} through a contingent link e_{DB}, via (6), is redundant if w_{ABX} is not greater than the lower bound of the contingent link e_{AX} that causes the wait. If $w_{ABX} \leq l_{AX}$, the wait bound constraint (5) implies $l_{AB} \geq w_{ABX} \geq 0$. Hence, the triangle DAB is in the precede case and $l_{AD} = -u_{DA} = -(l_{DB} - l_{AB}) = w_{ABX} - l_{DB}$, which implies the wait regression constraint (6).

Formulation as an NLP

The non-linear model formulated by Wah and Xin (2004) uses quadratic constraints and terms in the objective function to encode disjunctions. The precede constraint (3) is formulated as follows:

$$l_{BC}(l_{AB} - u_{AC} + u_{BC}) \geq 0 \quad (9a)$$
$$l_{BC}(u_{AB} - l_{AC} + l_{BC}) \leq 0 \quad (9b)$$

For each wait bound constraint (5), they introduce an auxiliary variable β and the following constraints:

$$(l_{AX} - w_{ABX})(l_{AB} - w_{ABX}) \geq 0 \quad (10a)$$
$$\beta \geq 0 \quad (10b)$$
$$\beta \geq w_{ABX} - l_{AX} \quad (10c)$$
$$\beta(l_{AB} - l_{AX}) \geq 0 \quad (10d)$$

Furthermore, a quadratic term $\beta(\beta - (w_{ABX} - l_{AX}))$ is added to the objective function. (This term must be minimised; if the problem is one of maximisation, its negative is used.) Its purpose is to ensure that β is 0 when $w_{ABX} \leq l_{AX}$ and otherwise equal to the difference $w_{ABX} - l_{AX}$. For this to work, the penalty incurred by a non-zero value of this must term outweigh the actual objective function. Note also that there is a situation in which this formulation fails to impose the lower bound on l_{AB}, since if $l_{AX} = w_{ABX}$, constraint (10a) is satisfied even if $l_{AB} \not\geq w_{ABX}$, and (10b)–(10d) are satisfied by setting $\beta = 0$. A similar encoding is used for the wait regression constraint (6').

Conflict-Directed Relaxation with Uncertainty

Finally, we consider the Conflict-Directed Relaxation with Uncertainty (CDRU) algorithm (Yu, Fang, and Williams 2014). CDRU takes an STNU that is *not* dynamically controllable and finds a least-cost relaxation of the temporal constraints that makes it controllable. Relaxing an STNU means tightening contingent links (reducing uncertainty) and/or loosening requirement links. The cost of a relaxation is a function of the change to each link, and some links may be excluded from change.

Relaxing over-constrained STNUs is one application of optimising bounds subject to dynamic controllability (cf. next section). In other applications, the initial STNU may already be dynamically controllable and the aim is to improve the value of an objective function by increasing uncertainty or tightening requirements, while maintaining controllability. We adapt CDRU to such problems by taking a dual approach: initial link bounds are chosen to maximise the objective, disregarding controllability, and the resulting (typically uncontrollable) STNU is then gradually relaxed to a feasible solution.

CDRU draws on conflict-directed A* (Williams and Ragno 2002), extending conflict learning and resolution to continuous variables (Yu and Williams 2013) and contingent constraints. The algorithm explores candidate relaxations in best-first order by interleaving two steps:

- First, it runs a dynamic controllability checking algorithm on the current best candidate STNU. If it is not dynamically controllable, the check returns an unsatisfied disjunction of linear constraints over subsets of link bounds. This is called a conflict: at least one constraint in the set must be satisfied to make the STNU controllable. The conflict is recorded as a new constraint that any solution must satisfy.
- Given a set of conflicts, the algorithm computes least-cost relaxations that eliminate them. A relaxation may loosen the bounds of requirement links, or tighten the bounds of contingent links. Since conflicts can be disjunctive, a conflict may generate several new candidates.

To apply CDRU to optimisation problems other than linear-cost relaxations, we have to replace the objective function and redefine the initial candidate accordingly. For example, in the maximum delay problem (see next section), the objective is to maximise the minimum width of contingent link intervals. Thus, we need a max-min function: $\max(\min_{e_{ij} \in C}(u_{ij} - l_{ij}))$. Finding the least-cost resolver of a single conflict disjunct (linear constraint) under this cost function can still be formulated as a linear program, and solved by a fast linear optimiser. The revised version of CDRU is shown as Algorithm 1. The input is an STNU in which the upper bounds of all contingent links are set to a very large number[1]. A candidate is a pair $\langle UB, C_r \rangle$, where UB is a set of tightenings to the upper bounds of contingent links and C_r the set of conflicts resolved by this candidate. For the initial candidate both sets are empty.

Input: An STNU $N = \langle X, E = R \cup C \rangle$.
Output: A set of tightened contingent link upper bounds $\{u_{ij} | e_{ij} \in C\}$ making N dynamically controllable.
Initialization:
$Cand \leftarrow \langle UB, C_r \rangle$; the first candidate;
$Q \leftarrow \{Cand\}$; a priority queue of candidates;
$Cft \leftarrow \emptyset$; the set of known conflicts;
Algorithm:
while $Q \neq \emptyset$ **do**
  $Cand \leftarrow$Dequeue(Q);
  $currCft \leftarrow$UnresolvedConflicts$(Cand, Cft)$;
  if $currCft == null$ **then**
    $newCft \leftarrow$ DynamicControllable?$(Cand)$;
    if $newCft == null$ **then**
      return $Cand$;
    else
      $Cft \leftarrow Cft \cup \{newCft\}$;
      $Q \leftarrow Q \cup \{Cand\}$;
    endif
  else
    $Q \leftarrow Q \cup$ExpandOnConflict$\{Cand, currCft\}$;
  endif
end
return $null$;

Algorithm 1: The CDRU algorithm adapted for the maximum delay problem.

As an example, consider the STNU shown in Figure 2(a), with three contingent and three requirement links. The upper bounds of the contingent links are initialised to 10^7. This STNU is not dynamically controllable, since the upper bound of link BF is less than the sum of the upper bounds of CD and EF. This generates a conflict involving five links: $u_{BF} - l_{DE} - u_{CD} - u_{EF} - l_{BC} \geq 0$. The least-cost resolver of this conflict tightens u_{CD} and u_{EF}, and generates the new best candidate shown in Figure 2(b). However, this candidate is still not controllable. Checking returns the conflict $u_{BF} - u_{EF} - u_{CD} - l_{BC} - u_{AB} + l_{AB} - l_{DE} \geq 0 \vee u_{BF} - u_{EF} - u_{CD} - l_{DE} \geq 0$. This conflict is a disjunction of two linear inequalities, meaning it can be resolved in two ways. This is the property of dynamic controllability: we can either apply relaxations to make the "negative cycles" in the graph positive, or apply relaxations to shift the order of some constraints such that the cycle itself is eliminated. Only one of the linear inequalities needs to be satisfied to resolve this conflict. CDRU finds two alternative sets of relaxations, leading to the two new candidates shown in Figure 2(c) and (d). It first considers candidate (c), since it allows a width of 4 for all contingent links, while candidate (d) allows only 3. Since candidate (c) is also controllable, it is returned as the preferred solution to the input STNU.

[1]Due to numerical issues, we do not use $+\infty$

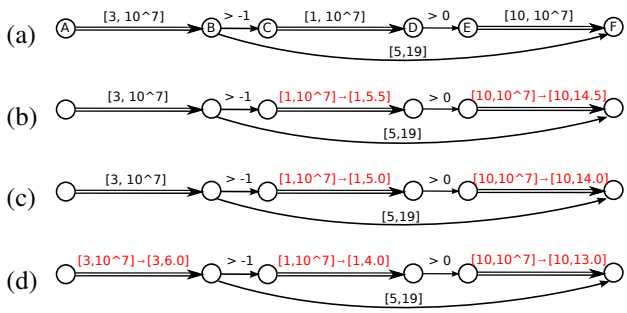

Figure 2: (a) Example of an uncontrollable STNU (upper bounds on contingent links are too large); (b–d) first, second and third relaxation candidate.

Applications

Next, we review different problems from the literature that can be formulated as optimisation over an STNU with dynamic controllability constraints, and compare the effectiveness of different methods of solving them. All experiments were run on 3.1GHz AMD cores with 64Gb memory.

Relaxing Over-Constrained Problems

An STNU that is not dynamically controllable often arises in planning and scheduling because users' goals (requirements) are too ambitious or the desired robustness to uncertainty (width of contingent links) is too great. In this situation, dynamic controllability can be restored by relaxing the problem: widening requirement links and/or tightening contingent links (Yu, Fang, and Williams 2014).

Let $\hat{l}_{ij}$ and $\hat{u}_{ij}$ denote the original (desired) bounds on link e_{ij}. The relaxation problem can be formulated as

$$\begin{aligned} \min \quad & \textstyle\sum_{e_{ij} \in E} f_{ij}(\delta^l_{ij}, \delta^u_{ij}) \\ \text{s.t.} \quad & l_{ij} = \hat{l_{ij}} - \delta^l_{ij} + \tau^l_{ij} \leq u_{ij} = \hat{u_{ij}} + \delta^u_{ij} - \tau^u_{ij} && e_{ij} \in R \\ & l_{ij} = \hat{l_{ij}} + \delta^l_{ij} \leq u_{ij} = \hat{u_{ij}} - \delta^u_{ij} && e_{ij} \in C \\ & \delta^l_{ij}, \delta^u_{ij}, \tau^l_{ij}, \tau^u_{ij} \geq 0 \\ & \text{dynamic controllability (1)–(7)} \end{aligned}$$

where $f_{ij}(\delta^l, \delta^u)$ encodes the relative cost relaxing the lower and upper bounds on link e_{ij} by δ^l and δ^u, respectively. The dynamic controllability constraints enforce not only that the network is dynamically controllable, but also that bounds are minimal (i.e., the tightest implied) on each requirement link. Because of this, we must allow also for requirement links to

be tightened, without affecting the objective function. This is why deviations from the target value are split into two non-negative variables, e.g., δ^l_{ij} and τ^l_{ij} for the lower bound, and only the relaxation part appears in the objective. (Note that minimal bounds are computed also as part of the reductions made by the DC checking algorithm, but a network does not have to be minimal to be dynamically controllable.)

For contingent links we can set the constants $L_{ij} = \hat{l_{ij}}$ and $U_{ij} = \hat{u_{ij}}$, since their bounds can only shrink. For requirement links there are no given limits in this problem.

Comparison of Solvers We compare the conflict-directed relaxation procedure (CDRU), the MIP model solved with Gurobi[2] and the non-linear model solved with SNOPT[3] on 2400 relaxation test cases used by Yu, Fang and Williams (2014). The STNUs have between 14 and 2000 nodes, and a number of contingent and requirement links approximately linear in the number of nodes. The objective function is a linear function of the amount of relaxation; it is symmetric w.r.t. relaxation of lower and upper bounds (i.e., $f_{ij}(\delta^l_{ij}, \delta^u_{ij}) = c_{ij}(\delta^l_{ij} + \delta^u_{ij})$).

Not surprisingly, the CDRU is the fastest on this problem and scales much better than both the MIP and non-linear solver, as shown in Figure 3(a). The non-linear solver does not guarantee solution optimality: of the solutions it finds on the relaxation problem set, 84.2% are within 1% of the optimal objective value (provided by the other solvers).

Robustness with Non-Probabilistic Uncertainty

Providing flexibility in schedules and temporal plans is viewed as a means to increase their robustness against unforeseen disturbances, such as delays. Several metrics for the flexibility of a schedule have been proposed (e.g., Cesta, Oddi, and Smith 1998; Aloulou and Portmann 2003; Wilson et al. 2014), as well as algorithms for finding high-flexibility schedules (e.g., Aloulou and Portmann 2003; Policella et al. 2009; Banerjee and Haslum 2011).

However, flexibility does not necessarily imply robustness: this depends on how "robustness" itself is defined. In abstract terms we may define robustness as the greatest level of disturbance (deviation from expected outcomes) at which the schedule is still successfully executed. (If we assume a probability distribution over deviations is given, the level of disturbance at which the schedule breaks equates to the probability of it breaking during execution. We consider this case in a later section.) To operationalise this definition, we have to specify what kind of disturbances are considered, and how the schedule executive can use flexibility to cope with them. Here, we exemplify by assuming (1) that the possible disturbances are deviations in the time taken to execute an activity from its normal duration, and (2) a partial-order schedule with a dynamic execution strategy.

A partial-order schedule (POS) consists of a set of time constraints between activities such that any realisation that meets these constraints is also resource feasible. In the deterministic case, where the duration of each activity i is a constant d_i, the POS can be represented as an STN with time points t_{s_i} and t_{e_i} for the start and end, respectively, of each activity. Assuming the duration of each activity can vary within some bounds, $[l_{s_i,e_i}, u_{s_i,e_i}]$, the schedule can be modelled as an STNU where the link $e_{s_ie_i}$ from each activity's start to its end is contingent, while remaining time constraints are requirement links. Thus, given a POS we can ask, what is the maximum deviation (i.e., width of the contingent bound) on any activity at which the STNU is dynamically controllable? This defines our measure of robustness. To compute it, we solve the following problem:

$$
\begin{array}{lll}
\max & \Delta & \\
\text{s.t.} & l_{s_i,e_i} = d_i - \delta_i \geq 0 & \forall i \\
& u_{s_i,e_i} = d_i + \delta_i & \forall i \\
& 0 \leq \Delta \leq \delta_i & \forall i \\
& \text{POS constraints (requirement links)} & \\
& \text{dynamic controllability (1)–(7)} &
\end{array}
$$

As explained above, requirement link bounds must be allowed to shrink, but their outer limits can be set to the given POS constraints. Since contingent links represent durations, a hard lower bound $L_{s_ie_i} = 0$ applies.

We can also define a one-sided variant of this robustness metric, accounting for delays only, by fixing $l_{s_i,e_i} = d_i$ (i.e., adding deviations only to the upper bound).

Comparison of Solvers We compared solvers on the one-sided (maximum delay) variant of the problem. As test cases, we use 3400 partial-order schedules for RCPSP/max problems (e.g., Kolisch and Padman 2001) with 10–18 jobs[4]. The schedules are generated by a scheduler that optimises a measure of POS flexibility (Banerjee and Haslum 2011). The STNU representation of a schedule has a time point for the start and end of each activity, as described above. Hence, the number of nodes and contingent links is determined by the number of jobs, but the number of (given) requirement links varies from 50 to 300.

The adapted CDRU algorithm is very effective for this problem, and the relative runtimes of the MIP and non-linear solvers, as shown in Figure 3(b), are similar to the previous case study. The non-linear solver frequently fails to find optimal solutions. To remedy this issue we run SNOPT repeatedly, using the the last solution as the starting point for the next iteration and using its objective value as a lower bound. This configuration (labelled "NLP_M" in Figure 3) is more time-consuming, but improves final solution quality: 93% of solutions found by NLP_M are optimal, compared to only around 70% for a single run of the solver.

Strong vs. Dynamic Controllability We can evaluate the robustness of a schedule under strong controllability as well. This requires only solving a linear program. For 84.3% of the test cases, the value of the maximum delay metric is the same. However, 6.38% of cases allow zero delay (i.e., have no robustness) under strong controllability but admit a non-zero value with dynamic exection.

[2]http://www.gurobi.com/

[3]http://ccom.ucsd.edu/~optimizers/; cf. also Gill, Murray and Saunders (2002)

[4]Set J10 from PSPLIB (http://www.om-db.wi.tum.de/psplib/), plus additional problems generated to have more max time lag constraints and allow more variation in resolving resource conflicts.

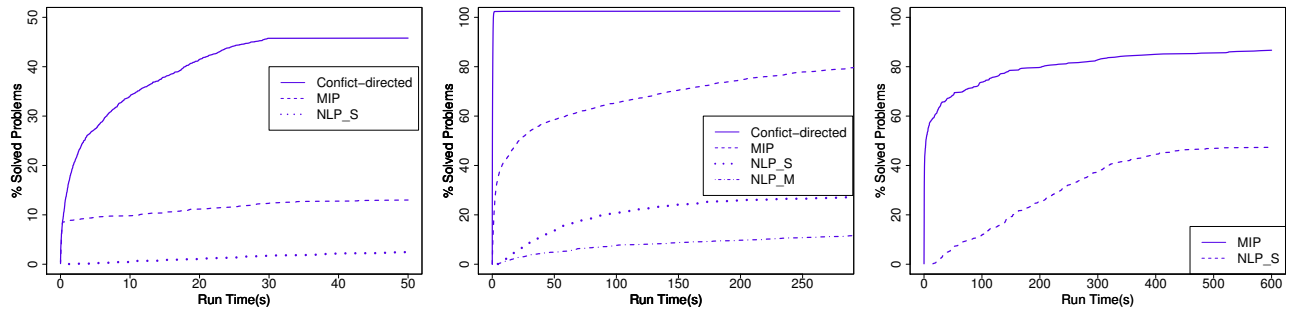

Figure 3: Runtime distributions for three different solvers (conflict-directed relaxation (CDRU), the MIP model solved with GuRoBi and the non-linear model solved with SNOPT) on (a) minimum relaxation problems, (b) schedule robustness (maximum delay) problems and (c) minimum flexibility problems.

Minimising Flexibility

Although flexibility may improve robustness, there can also be a cost associated with it. For example, to permit a job to start at any point in an interval it may be necessary to reserve resources for the entire time from its earliest starting time to its latest ending time, an interval that may be much larger than the duration of the job. Thus, for a given, fixed level of temporal uncertainty we may seek the smallest flexibility – meaning the tightest requirement bounds – that is sufficient to maintain dynamic controllability. This is the application that motivated Wah and Xin (2004). Their formulation is:

$$\begin{array}{lll} \min & \sum_{e_{ij}\in R}(u_{ij}-l_{ij}) & \\ \text{s.t.} & L_{ij}=l_{ij}\le u_{ij}=U_{ij} & e_{ij}\in C \\ & L_{ij}\le l_{ij}\le u_{ij}\le U_{ij} & e_{ij}\in R \\ & \text{dynamic controllability (1)–(7)} & \end{array}$$

where L_{ij} and U_{ij} are the bounds of the input STNU. That is, contingent link bounds are fixed, and requirement link bounds may only be tightened.

Comparison of Solvers We use a set of 600 random STNUs of the "GRID" family, generated by the same method used by Wah and Xin (2004). The number of nodes ranges from 41 to 201, contingent links from 6 to 63, and requirement links from 60 to 360. The runtime distribution is shown in Figure 3(c). Since there is no CDRU implementation for this problem, we compare only the MIP and non-linear solvers.

Robustness with Probabilistic Uncertainty

It is a natural extension to the STNU model to associate probabilities with the outcome (timing) of uncontrollable events (Tsamardinos 2002; Fang, Yu, and Williams 2014). As mentioned earlier, this allows us to define robustness as the probability of successful plan or schedule execution.

In the probabilistic STN (pSTN) model proposed by Fang, Yu and Williams (2014) the duration of each contingent link e_{ij} (i.e., the difference $t_j - t_i$) is a random variable D_{ij}. The model makes no assumption about independence or the distribution of these variables.

It is straightforward to transfer the STNU representation of a partial-order schedule, described above, to a pSTN. Since the random variable $D_{s_ie_i}$ represents the duration of an activity we assume it is non-negative. The probability of a successful dynamic schedule execution is then at least

$$\begin{array}{lll} \max & \mathbf{P}\left(\bigwedge_i l_{s_ie_i}\le D_{s_ie_i}\le u_{s_ie_i}\right) & \\ \text{s.t.} & 0\le l_{s_i,e_i}\le u_{s_i,e_i} & \forall i \\ & \text{POS constraints (requirement links)} & \\ & \text{dynamic controllability (1)–(7)} & \end{array}$$

The objective value is a conservative (lower) bound on the success probability, because it is the probability that all uncontrollable events fall inside the chosen bounds. The schedule may still execute successfully even if some durations fall outside these bounds, as the outcomes of other events may fortuitously compensate so that no constraints are violated.

The objective function form depends on the probability distributions over activity durations. For example, if each $D_{s_ie_i}$ is uniform over an interval, we can conservatively approximate the success probability with a linear function. Other distributions give rise to non-linear objectives.

Flexibility vs. Robustness As we are now able to compute different measures of schedule robustness, we can put to test the hypothesis that different measures of schedule flexibility correlate with robustness. We consider two flexibility metrics: *fluidity*, defined by Aloulou and Portmann (2003), and *improved flex*, defined by Wilson et al. (2014). Both metrics are averages of some form of temporal slack in the schedule.

We use the same set of RCPSP/max problems as before. For each of the two metrics, we take the schedules with the least and greatest fluidity/flex score for each instance. (Of course, we can only use instances for which the scheduler found at least two schedules with different fluidity/flex values.) For both schedules in each such pair, we compute the non-probabilistic (maximum delay) and probabilistic measures of robustness (where this is possible within reasonable time). For probabilistic robustness, we use uniformly distributed durations over an interval between $\pm 30\%$ of the nominal activity duration. We then count how often the schedule that has the higher score according to each flexibility metric has a higher, lower or equal robustness score,

	Maximum Delay			Prob. of Success		
	$>$	$=$	$<$	$>$	$=$	$<$
fluidity	291	332	125	185	441	77
improved flex	476	513	192	334	698	141

Figure 4: Correlation between schedule flexibility, as measured by *fluidity* (Aloulou and Portmann 2003) and *improved flex* (Wilson et al. 2014), and robustness. Each entry is the number of instances in which the schedule with a higher fluidity/flex score has a higher ($>$), equal ($=$) or lower ($<$) max delay and probability of success, respectively, compared to that of the schedule with lower fluidity/flex.

according to each of the two robustness measures. The results are summarised in Table 4. If there was no correlation between flexibility and robustness, we should find roughly equal proportions of schedules with higher and lower robustness scores. A simple binomial test shows that the observed result is extremely unlikely under this hypothesis. Thus, we conclude that there is a (positive) correlation between fluidity/flex and robustness, although it is quite weak.

Dynamic Controllability with Chance Constraints

Fang, Yu and Williams (2014) argue that in many situations, minimising risk (probability of failure) is too conservative, and may compromise other objectives (such as cost) too much. Instead, users may prefer to give an absolute bound on risk and optimise other metrics subject to this constraint. They propose a pSTN optimisation algorithm subject to strong controllability and the chance constraint

$$\sum_{e_{ij} \in C} (1 - \mathbf{P}(l_{ij} \leq D_{ij} \leq u_{ij})) \;\leq\; \rho. \tag{11}$$

This makes use of the union bound (Boole's inequality), so it is a conservative estimate of risk: That is, it ensures the probability of violating a requirement is ρ *or less*.

Combining the dynamic controllability model (1)–(7) with (11) enables us to find dynamic execution strategies under chance constraints. Whether the constraint is linear or non-linear, and hence which solvers can be applied, depends on the probability distributions, as noted above.

Dynamic vs. Strong Controllability Since a strongly controllable network is also dynamically controllable, the optimal value of any objective function can only improve as we consider dynamic instead of strong execution strategies. We can now evaluate how much it improves, by comparing the quality of solutions obtained under dynamic and strong controllability constraints on the test cases used by Fang, Yu and Williams (2014). The objective is to minimise makespan under chance constraints. Since the problems feature normally distributed uncertain durations, only the non-linear solver is able to tackle them.

Figure 5 shows the distribution of the improvement, measured by the reduction in makespan achieved under dynamic controllability from that of the optimal strongly controllable solution, expressed as a fraction of the latter. (Strongly controllable solutions to chance-constrained problems are generated with Fang, Yu and Williams' solver.)

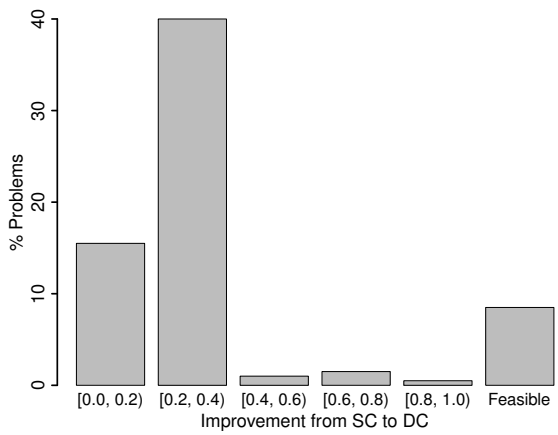

Figure 5: Reduction in makespan achieved with dynamic as opposed to strong controllability. Instances in the last column are infeasible under strong controllability, but have a valid dynamic execution strategy.

This clearly shows the value of using a dynamic execution strategy. (In fact, the possible improvement may be even greater since solutions returned by the non-linear solver are often not optimal.) 8.5% of problems are infeasible under strong controllability constraints – that is, no strong (unconditional) execution strategy exists – but are feasible under dynamic controllability constraints. On the other hand, optimisation with dynamic controllability is harder: the NLP solver failed to find a solution for 33% of the instances.

Conclusions

In many cases where an STNU is used to represent a plan or schedule, the ability to optimise a function of STNU time bounds subject to the constraint that the network is dynamically controllable make it far more useful than being able only to test if dynamic controllability holds. For example, we showed how it enables optimisation of a chance-constrained pSTN under dynamic controllability, leading to better solutions than possible if strong controllability is imposed. It also enabled us to measure schedule robustness in new ways, and thereby test the hypothesis that greater flexibility leads to more robust schedules.

Dynamic controllability is a disjunctive linear constraint. Comparing solution approaches, we found that dealing with disjunctions explicitly, as in the CDRU algorithm (Yu, Fang, and Williams 2014), is the most efficient. However, the MIP and NLP formulations are very flexible, allowing controllability to be combined with other constraints. Our constraint model follows closely the algorithm by Morris, Muscettola and Vidal (2001). It is an open question if a better model can be derived from Morris' (2006) structural characterisation of dynamic controllability.

Acknowledgements This work was supported in part by Boeing Corporation, under grant number MIT-BA-GTA-1,

and ARC project DP140104219, "Robust AI Planning for Hybrid Systems". NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

References

Aloulou, M. A., and Portmann, M.-C. 2003. An efficient proactive reactive scheduling approach to hedge against shop floor disturbances. In *Proc. 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA)*, 337–362.

Banerjee, D., and Haslum, P. 2011. Partial-order support-link scheduling. In *Proc. 21st International Conference on Automated Planning and Scheduling (ICAPS)*, 307–310.

Cesta, A.; Oddi, A.; and Smith, S. F. 1998. Profile-based algorithms to solve multiple capacitated metric scheduling problems. In *Proc. 4th International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, 214–223.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.

Fang, C.; Yu, P.; and Williams, B. C. 2014. Chance-constrained probabilistic simple temporal problems. In *Proc. 28th AAAI Conference on Artificial Intelligence*, 2264–2270.

Gill, P. E.; Murray, W.; and Saunders, M. A. 2002. SNOPT: an SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization* 12(4):979–1006.

Kolisch, R., and Padman, R. 2001. An integrated survey of project scheduling. *OMEGA International Journal of Management Science* 29(3):249–272.

Morris, P.; Muscettola, N.; and Vidal, T. 2001. Dynamic control of plans with temporal uncertainty. In *Proc. 17th International Conference on Artificial Intelligence (IJCAI)*, 494–499.

Morris, P. 2006. A structural characterization of temporal dynamic controllability. In *Proc. 12th International Conference on Principles and Practice of Constraint Programming (CP)*, 375–389.

Policella, N.; Cesta, A.; Oddi, A.; and Smith, S. 2009. Solve-and-robustify. *Journal of Scheduling* 12:299–314.

Tsamardinos, I. 2002. A probabilistic approach to robust execution of temporal plans with uncertainty. In *Methods and Applications of Artificial Intelligence (Proc. 2nd Hellenic Conference on AI)*, 97–108.

Vidal, T., and Fargier, H. 1999. Handling contingency in temporal constraint networks: From consistency to controllabilities. *Journal of Experimental and Theoretical AI* 11(1):23–45.

Wah, B. W., and Xin, D. 2004. Optimization of bounds in temporal flexible planning with dynamic controllability. In *Proc. 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 40–48.

Wah, B. W., and Xin, D. 2007. Optimization of bounds in temporal flexible planning with dynamic controllability. *International Journal on Artificial Intelligence Tools* 16(1):17–44.

Williams, B. C., and Ragno, R. J. 2002. Conflict-directed A* adn its role in model-based embedded systems. *Journal of Discrete Applied Mathematics* 155(12):1562–1595.

Wilson, M.; Klos, T.; Witteveen, C.; and Huisman, B. 2014. Flexibility and decoupling in simple temporal networks. *Artificial Intelligence* 214:26–44.

Yu, P., and Williams, B. 2013. Continuously relaxing over-constrained conditional temporal problems through generalized conflict learning and resolution. In *Proc. 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2429–2436.

Yu, P.; Fang, C.; and Williams, B. C. 2014. Resolving uncontrollable conditional temporal problems using continuous relaxations. In *Proc. 24th International Conference on Automated Planning and Scheduling (ICAPS)*, 341–349.