Distributing the Control of a Temporal Network among Multiple Agents

Luke Hunsberger
124 Raymond Ave.
Box 444, Vassar College
Poughkeepsie, NY 12604-0444, USA
hunsberg@cs.vassar.edu

ABSTRACT

Agents collaborating on a set of tasks subject to temporal constraints must coordinate their activities to ensure that all of the temporal constraints are ultimately satisfied. Simple Temporal Networks (STNs) can be used to concisely represent temporal constraints; however, most algorithms for manipulating such networks presume that a single agent controls the network. Although recent research considers the controllability of networks in which Nature independently controls some temporal intervals, it nonetheless presumes that a single agent controls the *rest* of the network.

This paper makes the following contributions. First, it argues for STNs augmented to accommodate the real-time execution of tasks. Although borrowing from existing approaches, it differs by sharply distinguishing between constraints in the network and the distribution of control over that network. Second, it introduces a more general conception of distributing control of a temporal network, one that is able to accommodate not only networks partially controlled by Nature, but also networks controlled by multiple agents. Third, it construes an existing algorithm for partitioning temporal networks into independent subnetworks as an algorithm for distributing control of a temporal network among multiple agents, each agent having sole control over one subnetwork. It then presents a more general algorithm that allows one of the subnetworks to remain dependent on the rest, thereby enabling the overall network to be less constrained. Restrictions on the control of the dependent subnetwork, specified in terms of necessary and sufficient bounds, guarantee an effective distribution of control over the network.

Categories and Subject Descriptors

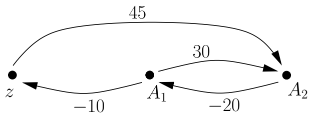
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Theory, algorithms

Keywords

Temporal reasoning



$\mathcal{D}$	z	A_1	A_2
z	0	25	45
A_1	-10	0	30
A_2	-30	-20	0

Figure 1: A graphical representation of an STN and its associated distance matrix

1. INTRODUCTION

This section uses a simple example of a task subject to temporal constraints to introduce Simple Temporal Networks [2] and basic, relevant definitions and results.

EXAMPLE 1. *Let A be a task subject to the following:*

- *A must begin at or after time 10;*
- *A's duration must be between 20 and 30; and*
- *A must end at or before time 45.*

1.1 Simple Temporal Networks

DEFINITION 1. *A* Simple Temporal Network *is a pair, $(\mathcal{T}, \mathcal{C})$, where $\mathcal{T}$ is a set $\{t_0, t_1, \ldots, t_n\}$ of time-point variables and $\mathcal{C}$ is a set of binary constraints on those variables, each constraint having the form $t_j - t_i \leq \delta$ for some real number δ. The "variable" t_0 represents an arbitrary, fixed reference point on the time-line. (In this paper, t_0 is fixed at 0 and is referred to as z, or the* zero *time-point variable.)*

Example 1 can be represented by the STN $(\{z, A_1, A_2\},\ \mathcal{C})$, where A_1 and A_2 represent the beginning and ending times for task A, and $\mathcal{C}$ contains the following constraints:

$$\begin{array}{rcrl} z - A_1 & \leq & -10 & \text{(i.e., } 10 \leq A_1) \\ A_2 - A_1 & \leq & 30 & \\ A_1 - A_2 & \leq & -20 & \text{(i.e., } 20 \leq A_2 - A_1 \leq 30) \\ A_2 - z & \leq & 45 & \text{(i.e., } A_2 \leq 45) \end{array}$$

This STN is depicted graphically at the left side of Figure 1, where each node in the graph corresponds to a time-point variable and each labelled, directed edge corresponds to a binary temporal constraint. The label on an edge E is also called the *length* of E, and is denoted by $|E|$.

Adding together explicit constraints in an STN yields implicit constraints, as illustrated below:

$$\begin{array}{rcccccl} & z & - & A_1 & \leq & -10 & \text{(i.e., } 10 \leq A_1) \\ & A_1 & - & A_2 & \leq & -20 & \text{(i.e., } A_1 + 20 \leq A_2) \\ \hline \Longrightarrow & z & - & A_2 & \leq & -30 & \text{(i.e., } 30 \leq A_2) \end{array}$$

The implicit constraint, $z - A_2 \leq -30$, is represented in the graph by a *path* from A_2 to z (via A_1) whose length is -30. For convenience, a path P may be denoted by the sequence of time-points it contains, and its length by $|P|$. Thus, the path from A_2 to A_1 to z is written $P = (A_2, A_1, z)$, and $|P| = -30$. In general, shorter paths correspond to stronger constraints. For example, the edge of length 30 from A_1 to A_2 corresponds to a stronger constraint ($A_2 - A_1 \leq 30$) than does the path (A_1, z, A_2) of length 35 (which corresponds to $A_2 - A_1 \leq 35$). For each pair of time-points (t_i, t_j), it is useful to keep track of the *strongest implicit constraint* (i.e., the shortest path) from t_i to t_j.

DEFINITION 2. *Given an STN* $\mathcal{S} = (\{t_0, t_1, \ldots, t_n\}, \mathcal{C})$, *the* distance matrix *for* $\mathcal{S}$ *is the* $(n+1)$*-by-*$(n+1)$ *matrix* $\mathcal{D}$ *defined by:* $\mathcal{D}(t_i, t_j) = \text{ shortest path from } t_i \text{ to } t_j \text{ in } \mathcal{S}$.

The distance matrix for an STN can be computed in time $O(n^3)$ using Floyd-Warshall's all-pairs shortest-path algorithm [1, 2]. The distance matrix for the STN in Figure 1 is shown at the right side of the figure; the entries corresponding to explicit constraints have been highlighted.

DEFINITION 3. *A* solution *to an STN* $(\{z, t_1, \ldots, t_n\}, \mathcal{C})$ *is a set of variable assignments* $\{z = 0, t_1 = v_1, \ldots, t_n = v_n\}$ *satisfying all of the constraints in* $\mathcal{C}$. *An STN that has at least one solution is called* consistent.

Dechter et al. [2] showed that an STN is consistent if and only if there are no negative cycles (i.e., loops with negative path-length) in its associated graph or, equivalently, if and only if the diagonal entries in the distance matrix are all non-negative. Since this holds for the distance matrix in Figure 1, the corresponding STN is consistent. That STN has many solutions, including:

$$\{z = 0,\ A_1 = 13,\ A_2 = 37\}.$$

1.2 Adding Constraints to an STN

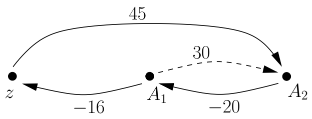
As described above, an STN is a static entity: a fixed set of time-point variables together with a fixed set of binary constraints on those variables. However, when using an STN to represent temporal constraints among tasks, it typically becomes necessary to modify that STN over time (e.g., by adding new constraints to it).

Recall that task A is constrained to start at or after time 10. To make room for some other task, it might be necessary to strengthen this constraint, for example, to require that A start at or after time 16. Of course, this should be done only if it would not threaten the consistency of the STN. As the following definition and theorem show, the distance matrix contains the relevant information.[1] Whereas $\mathcal{D}(t_i, t_j)$ gives the strongest implicit constraint from t_i to t_j, the negative-transpose entry, $-\mathcal{D}(t_j, t_i)$, specifies the degree to which the constraint from t_i to t_j may be strengthened without threatening the consistency of that STN.

DEFINITION 4. *The candidate constraint* $t_j - t_i \leq \delta$ *satisfies the* negative-transpose criterion *(NTC) with respect to the distance matrix* $\mathcal{D}$ *if* $\delta \geq -\mathcal{D}(t_j, t_i)$.

THEOREM 5. *If a constraint* c *satisfies the NTC with respect to the distance matrix* $\mathcal{D}$ *for a* consistent *STN* $(\mathcal{T}, \mathcal{C})$, *then the STN* $(\mathcal{T}, \mathcal{C} \cup \{c\})$ *is also consistent.*

Given that $-16 \geq -25 = -\mathcal{D}(z, A_1)$, the candidate constraint, $z - A_1 \leq -16$, satisfies the NTC. Thus, it may be safely added to the network, as shown in Figure 2. Note that the edge from A_1 to



$\mathcal{D}'$	z	A_1	A_2
z	0	25	45
A_1	-16	0	29
A_2	-36	-20	0

Figure 2: The STN and distance matrix resulting from adding a constraint to the STN in Figure 1

A_2 is shown as a dashed line because the constraint corresponding to the path (A_1, z, A_2) of length 29 is stronger than the constraint corresponding to the edge from A_1 to A_2 of length 30.

In general, when adding a constraint to an STN, the entries in the distance matrix either decrease or stay the same (i.e., the strongest implicit constraints get stronger or stay the same). As a result, the negative-transpose values in the NTC either *increase* or stay the same. Thus, as constraints are added to the network, the NTC imposes stronger restrictions (in the form of greater lower bounds).

2. SINGLE-AGENT CONTROL OF AN STN

In the original research on STNs, the question of who *controlled* an STN (i.e., had the authority to modify an STN) was not addressed. However, for a single agent in complete control of an STN, the original results and algorithms suffice. The following corollary to Theorem 5 shows that as long as an agent adds constraints that at each stage satisfy the NTC, then the network shall remain consistent.

COROLLARY 6. *Let* $(\mathcal{T}, \mathcal{C}_0)$, $(\mathcal{T}, \mathcal{C}_1)$, $\ldots$, $(\mathcal{T}, \mathcal{C}_k)$ *be a sequence of STNs such that* $(\mathcal{T}, \mathcal{C}_0)$ *is consistent and, for each* $i \in \{1, \ldots, k\}$, $\mathcal{C}_i = \mathcal{C}_{i-1} \cup \{c_i\}$, *where* c_i *satisfies the NTC for* $(\mathcal{T}, \mathcal{C}_{i-1})$. *Then each STN in the sequence is consistent.*

If an agent is using an STN to represent some scenario in the distant future, the above theory of STNs suffices. The question of *when* constraints are added to the network is irrelevant—as long as all modifications to the STN are completed (i.e., all time-point variables are constrained to fixed values) prior to executing any of the tasks. However, if the STN represents a set of tasks such that modification of the STN is interleaved with the execution of tasks, then the given theory is incomplete: it does not explain how to accommodate the ever-changing present moment or the real-time execution of tasks, both of which involve modifying the STN in real-time. The next section addresses these issues.

3. HANDLING THE PASSAGE OF TIME

Several researchers have used STNs to model task-execution scenarios in which the effects of the passage of time on the network must be accommodated [8, 10, 9, 7, 5, 6]. Because these approaches have used STNs that do not explicitly represent the ever-changing present moment, reasoning about the effects of the passage of time must be done outside the representation of STNs. This section argues for STNs augmented to include an explicit representation of the ever-changing present moment. One advantage of this approach is that the *execution* of a time-point (i.e., the real-time assignment of a time-point variable to the present moment) can be defined as a formal operation on Augmented STNs.[2] In addition, Augmented STNs provide a framework for presenting the concept

[1] Definition 4 and Theorem 5 draw directly from Dechter et al. [2], but the NTC terminology is new.

[2] Other researchers have discussed executing time-points in STNs [8, 6], but have not defined time-point execution as a formal operator on STNs; nor have they explicitly represented the impact of time-point execution in terms of constraints on the ever-changing present-moment.

of distributing the control of a temporal network among a group of agents.

3.1 Augmented STNs (ASTNs)

ASTNs differ from existing approaches, as follows.

- Each ASTN includes a time-point variable, `now`, that explicitly represents the ever-changing present moment.
- ASTNs do not require the use of a particular scheme for distributing control of a temporal network among a group of agents and/or Nature.
- Executing time-points and adding new constraints are explicitly defined as operators on ASTNs.

ASTNs enable a sharp distinction to be drawn between constraints in a temporal network and restrictions on who is authorized to modify various portions of that network. In addition, ASTNs can accommodate alternative schemes for distributing the control of a temporal network among a group of agents. For example, ASTNs can accommodate scenarios in which Nature controls certain task durations, as well as scenarios in which multiple agents seek to coordinate their execution of a set of temporally related tasks, each agent controlling a portion of the temporal network.

Although ASTNs will be used throughout this paper, this section restricts attention to the case of a single agent whose control over a network is complete—except that it does not control the passage of time. As will be seen, the passage of time can, by itself, threaten the consistency of a network.

In addition to the zero time-point, z, and the present-moment time-point, `now`, each ASTN includes two sets of time-points: $\mathcal{T}_x$ and $\mathcal{T}_{nx}$. Each time-point in $\mathcal{T}_x$ has already been executed and, thus, is fixed on the time-line. Each time-point in $\mathcal{T}_{nx}$ has not yet been executed and, thus, is explicitly constrained to occur at or after `now`.

To represent the asymmetric flow of time into the future, each ASTN includes a constraint of the form $z - \texttt{now} \leq -d$ (i.e., $\texttt{now} \geq d$), where d is a variable quantity. Replacing d by a numeric constant transforms an ASTN into an ordinary STN; thus, an ASTN is a parameterized family of STNs.

DEFINITION 7. *A 5-tuple* $(\mathcal{T}_{nx}, \mathcal{T}_x, \mathcal{C}, \mathcal{C}_\nu, b)$*, where:*

- $\{z\}, \{\texttt{now}\}, \mathcal{T}_{nx}$ *and* $\mathcal{T}_x$ *are disjoint sets*
- $\mathcal{C}$ *is a set of binary temporal constraints over time-points in* $\{z\} \cup \mathcal{T}_{nx} \cup \mathcal{T}_x$
- $\mathcal{C}_\nu = \{(\texttt{now} - t \leq 0) \mid t \in \mathcal{T}_{nx}\} \cup \{(z - \texttt{now} \leq -d)\}$
- b *is a real number*

is called an ASTN. *This 5-tuple determines a parameterized family of STNs, as follows. For each* $d \in [b, \infty)$*, the STN,*

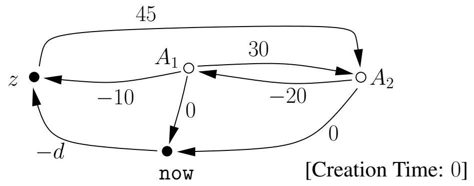
$$\mathcal{A}(d) = (\{z, \texttt{now}\} \cup \mathcal{T}_{nx} \cup \mathcal{T}_x,\ \mathcal{C} \cup \mathcal{C}_\nu)$$

is called the associate STN *at d. The number b is called the* creation time *of* $\mathcal{A}$*.*

A sample ASTN based on Example 1 is depicted in Figure 3, where the open circles are used to indicate that A_1 and A_2 are as yet unexecuted (i.e., $A_1, A_2 \in \mathcal{T}_{nx}$). The creation time has been arbitrarily set to 0.

For $d \leq 25$, the associate STN at d is consistent. For example, if $d = 12$, it has the following solution:

$$\{\texttt{now} = 14, z = 0, A_1 = 16, A_2 = 38\}.$$



$\mathcal{D}_d$	now	z	A_1	A_2
now	0	$-d$	$25-d$	$45-d$
z	25	0	25	45
A_1	0	$\min\{-10, -d\}$	0	$\min\{30, 45-d\}$
A_2	-20	$\min\{-30, -20-d\}$	-20	0

Figure 3: An Augmented STN (based on Example 1) together with its distance matrix

For $d > 25$ (which implies $\texttt{now} > 25$), the associate STN at d is inconsistent, as evidenced by the loop $(\texttt{now}, z, A_2, A_1, \texttt{now})$ having negative path-length. Thus, the mere passage of time can cause a consistent network to become inconsistent.

DEFINITION 8. *Let* $\mathcal{A}$ *be an ASTN created at time b.* $\mathcal{A}$ *is called* consistent at its creation time *if the associate STN* $\mathcal{A}(b)$ *is consistent. The* interval of consistency *for an ASTN that is consistent at its creation time is the unique interval* $[b, e]$ *such that* $\mathcal{A}(d)$ *is consistent if and only if* $d \in [b, e]$.

Given that the edge from `now` to z in an ASTN has the variable length, $-d$, computing the distance matrix for an ASTN (i.e., a parameterized family of STNs) is a little different from the case of an ordinary STN. However, since only one edge has variable length, the problem is easily solved.

DEFINITION 9. *Given an ASTN* $\mathcal{A} = (\mathcal{T}_{nx}, \mathcal{T}_x, \mathcal{C}, \mathcal{C}_\nu, b)$*, let*

$$\mathcal{S}^- = (\{z, \texttt{now}\} \cup \mathcal{T}_{nx} \cup \mathcal{T}_x,\ \mathcal{C} \cup \mathcal{C}_\nu \setminus \{(z - \texttt{now} \leq -d)\})$$

be the STN resulting from removing the edge, $z - \texttt{now} \leq -d$*, from* $\mathcal{A}$*. Let* $\mathcal{D}^-$ *denote the distance matrix for* $\mathcal{S}^-$*.*

Since a shortest path between two time-points in an ASTN $\mathcal{A}$ either includes the edge, $z - \texttt{now} \leq -d$, or does not, the distance matrix for $\mathcal{A}(d)$ may be computed as follows:

$$\mathcal{D}_d(t_i, t_j) = \min\{\ \mathcal{D}^-(t_i, t_j),\ \mathcal{D}^-(t_i, \texttt{now}) - d + \mathcal{D}^-(z, t_j)\ \}$$

The distance matrix for the ASTN in Figure 3 is shown at the bottom of the figure. Since $\mathcal{A}(d)$ is inconsistent for $d > 25$, the values shown are for $d \leq 25$.

Since the edge, $z - \texttt{now} \leq -d$, satisfies the NTC if and only if $-d \geq -\mathcal{D}^-(z, \texttt{now})$ (i.e., $d \leq \mathcal{D}^-(z, \texttt{now})$), the interval of consistency for a ASTN that is consistent at its creation time is the interval, $[b, \mathcal{D}^-(z, \texttt{now})]$. For example, the interval of consistency for the ASTN in Figure 3 is $[0, 25]$.

3.2 Operations on Augmented STNs

This section defines two operations on ASTNs. The *Add-Constraint* (AC) operation modifies an ASTN by adding a constraint to it; the *Execute-Time-Point* (EX) operation modifies an ASTN by executing one of the as-yet-unexecuted time-points. In either case, an ASTN modified at time T yields a new ASTN whose creation time is T. For the AC operation, the constraint being added is not allowed to involve the `now` time-point. (Such constraints are typically either redundant or impossible to satisfy.) The EX operation assigns a previously unexecuted time-point t to a fixed value T by adding binary constraints on t and z, while *removing* the constraint

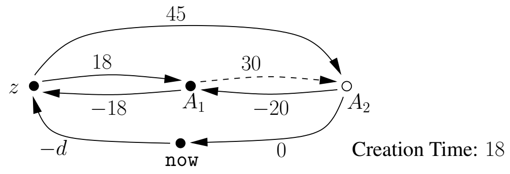


Figure 4: The ASTN after executing A_1 at time 18.

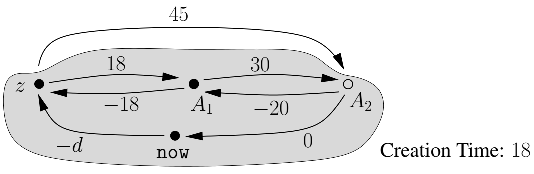


Figure 5: The subnetwork controlled by Nature

that t occur at or after `now` (since, otherwise, the network would ever after be inconsistent). For the ASTN from Figure 3, executing the time-point A_1 at time 18 results in the ASTN shown in Figure 4. In this ASTN, A_1 is effectively constrained to be fixed at 18; in addition, the edge from A_1 to `now` has been removed. The interval of consistency for this ASTN is $[18, 45]$. In other words, by executing a time-point, the network has gained some flexibility with respect to the passage of time.

DEFINITION 10. *Let $\mathcal{A} = (\mathcal{T}_{nx}, \mathcal{T}_x, \mathcal{C}, \mathcal{C}_\nu, b)$ be an ASTN, T a real number such that $T \geq b$, c an arbitrary constraint on time-points in $\mathcal{T}_x \cup \mathcal{T}_{nx} \cup \{z\}$, and t an arbitrary time-point in $\mathcal{T}_{nx}$. Then, AC and EX are defined as follows:*

$$AC(\mathcal{A}, c, T) = (\mathcal{T}_{nx}, \mathcal{T}_x, \mathcal{C} \cup \{c\}, \mathcal{C}_\nu, T) \text{ and}$$
$$EX(\mathcal{A}, t, T) = (\mathcal{T}_{nx} \backslash \{t\}, \mathcal{T}_x \cup \{t\}, \mathcal{C} \cup \{c_1, c_2\}, \mathcal{C}_\nu \backslash \{c_f\}, T)$$

where: c_1 is the constraint: $t - z \leq T$ (i.e., $t \leq T$);
c_2 is the constraint: $z - t \leq -T$ (i.e., $t \geq T$); and
c_f is the constraint: $\mathtt{now} - t \leq 0$ (i.e., $t \geq \mathtt{now}$).

3.3 Prior Work on Using STNs in Real-Time

Tsamardinos et al. [8] investigated the use of STNs in cases where tasks are being executed in real time. They observed that a consistent STN may always be successfully executed in real-time by following a simple strategy. However, in practice, during the time that an agent is computing updates to the distance matrix in response to some change to the STN, the passage of time might itself cause the network to become inconsistent. Thus, their paper focuses on carrying out fast incremental updates of the distance matrix in real time as time-points are being executed. By transforming the graph of the STN into an equivalent STN, they show that the distance matrix may be incrementally computed using only local (i.e., one-step) propagation of constraints rather than full-fledged propagation through the entire network. They do not explicitly represent the ever-changing present moment in their STNs; nor do they formally define operations on STNs.

3.4 Single-Agent Control of an ASTN

If an agent is authorized to apply instances of the Add-Constraint and Execute-Time-Point operations to an ASTN in any manner whatsoever, then that agent is said to have complete control over that ASTN. However, even in that case, the agent does not control the passage of time. However, given an ASTN that is consistent at its creation, the following strategy for executing time-points ensures that the ASTN will remain consistent over time—despite the passage of time (which cannot be directly controlled).

Let $\mathcal{A}_0$ be an ASTN that is consistent at its creation time, b. Its interval of consistency is $[b, e]$, for some $e \geq b$. Since any $d > e$ would introduce an inconsistency (i.e., a loop of negative path-length), there must be some loop L of length 0 in the associate STN $\mathcal{A}(e)$ that includes the edge from `now` to z. Since the only edges terminating at `now` are those of length 0 from as-yet-unexecuted time-points, one of those edges, say, $\mathtt{now} - t_i \leq 0$, must be part of the path L. Execute the time-point t_i at time e, generating a new ASTN $\mathcal{A}_1$ that is consistent at its creation time. Apply the preceding strategy to this new ASTN, and so on, until all time-points have been executed. Since the last ASTN is consistent at its creation time, and there are no more edges terminating at `now`, that ASTN will be forever consistent.

4. SHARING CONTROL WITH NATURE

Several researchers have investigated the use of STNs in cases where a single agent shares control of a temporal network with Nature, in particular, where Nature independently controls the durations of certain tasks [10, 9, 7, 5, 6]. For example, I might control when I turn on my television, but Nature controls how long it takes for the picture tube to warm up. Morris et al. [6] provide a polynomial-time algorithm for determining whether such a network is *dynamically controllable* (i.e., whether there exists a real-time strategy for executing time-points that guarantees that all temporal constraints will ultimately be satisfied—regardless of the choices made by Nature). Importantly, such a strategy can, at any given time, only depend on what has already happened, not on what will happen. They show that if a network is dynamically controllable, then a strategy for controlling it can be expressed in terms of (self-imposed) conditional restrictions on the agent's authority to execute the time-points under its direct control. For example, an agent might wait to execute a time-point B until Nature executes some other time-point C or until at least ΔT units have elapsed since the execution of yet another time-point A.

Their use of STNs does not include an explicit representation of the ever-changing present moment. Also, they do not explicitly define the execution of a time-point variable as a formal operation on a temporal network.

Using ASTNs, the work of Morris et al. may be characterized as follows. First, they restrict attention to the Execute-Time-Point operation on ASTNs. Second, Nature is authorized to execute certain time-points in $\mathcal{T}_{nx}$ under certain restrictions; the agent is authorized to execute all other time-points in $\mathcal{T}_{nx}$. For example, the agent might be authorized to execute A_1 in the ASTN in Figure 3, whereas Nature might be authorized to execute A_2. However, Nature's use of the Execute-Time-Point operation is restricted to $\Delta T \in [20, 30]$ time-units after the execution of A_1. If the agent executes A_1 at time 18 (as shown in Figure 4), then Nature's subsequent control over the execution of A_2 is *equivalent* to Nature having complete control over the highlighted portion of the ASTN in Figure 5, subject only to the constraint that that *subnetwork* remain consistent.

When multiple task durations are subject to Nature's control, they are assumed to be independently controlled. Thus, each such duration is, in effect, controlled by a separate instance of a simple *Nature-agent*. Each Nature-agent's control over its corresponding subnetwork is activated by the execution of a particular time-point variable. In this scenario, control over the entire ASTN is distributed among the agent and multiple Nature-agents. Importantly, each Nature-agent controls its subnetwork without considering the impact of its Execute-Time-Point operation on the global network.

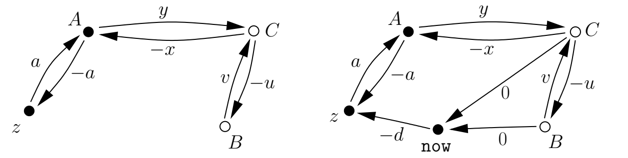


Figure 6: A canonical case covered by Morris et al.

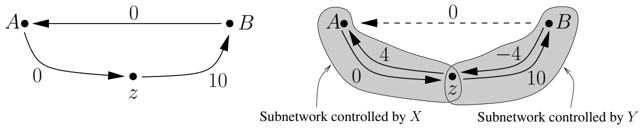


Figure 7: The TD Algorithm on a simple example

Thus, ensuring the consistency of the entire network, over time, is the sole responsibility of the agent.

Some of the canonical cases covered by Morris et al. have a natural interpretation in terms of ASTNs. For example, suppose an agent directly controls the execution of time-points A and B, but Nature controls C, subject to the constraints shown in the left-hand side of Figure 6. Morris et al. derive the following rule: if Nature has not yet executed C, then the agent must execute B no later than $(x - u)$ time-units after A is executed. However, if $u < 0$, then B must be executed after C and, hence, this rule does not apply.

The ASTN on the right-hand side of Figure 6 represents the same situation, except that it explicitly includes the `now` time-point variable. The reason for the above rule is apparent from this ASTN. In particular, the path $(C, B, \texttt{now}, z, A)$ has length $-u + 0 - d + a$, which, for values of d greater than $a + (x - u)$, is less than $-x$ and, thus, represents a stronger constraint than that represented by the *edge* from C to A. Since Nature's control is governed solely by the *edges* linking C and A, Nature might very well violate any tighter constraint imposed by the agent. By executing B before time $a + (x - u)$, the edge from B to `now` is removed, thereby deleting the threat represented by the path $(C, B, \texttt{now}, z, A)$, which no longer exists. This ASTN also illustrates why the rule does not apply when $u < 0$: in that case, the path $(C, B, \texttt{now}, z, A)$ is longer (i.e., weaker) than the path $(C, \texttt{now}, z, A)$.

The work by Morris et al. represents an important contribution to the theory of STNs. However, they make several assumptions that are not appropriate in many multi-agent scenarios. For example:

- Each subnetwork controlled by Nature corresponds, in effect, to a single temporal interval.
- The time-points defining the interval controlled by Nature are required to have a specified order.
- Nature's authorization is limited to a single Execute-Time-Point operation for each subnetwork it controls.
- Nature does not negotiate. If a network is not dynamically controllable, all hope is lost.

The next section treats scenarios involving multiple agents (but no Nature-agents) seeking to coordinate their execution of tasks subject to various temporal constraints. That work does not consider real-time, dynamic controllability issues but does involve more complex schemes for distributing the control of a temporal network among a group of agents.

5. DISTRIBUTING CONTROL OF A NETWORK AMONG MULTIPLE AGENTS

An earlier paper [3] presents a Temporal Decoupling (TD) algorithm that computes a set of constraints to add to a given STN sufficient to partition that STN into a set of independent subnetworks. The characteristic property of a decoupled STN, called the *Mergeable Solutions Property*, is that arbitrary solutions to the decoupled subnetworks may be combined to yield a solution to the entire network. Thus, if each subnetwork corresponds to a set of tasks being done by a single agent, each agent may operate independently on its own subnetwork without threatening the consistency of the overall network. Thus, in a decoupled network, control over the network can be effectively distributed to the participating agents such that each agent independently controls its own subnetwork.

Consider the following example. Two time-points, A and B, are subject to the following constraint: $0 \leq A \leq B \leq 10$. The corresponding STN is shown in the left-hand side of Figure 7. The goal is to distribute control over the network to two agents, X and Y, such that X controls the execution of A, and Y controls the execution of B. As things currently stand, X and Y cannot be given independent and complete control over their respective subnetworks: were they to execute their time-points independently, they might violate the constraint that A be executed before B.

The TD algorithm computes a set of constraints to add to the network sufficient to temporally decouple specified subnetworks. The right-hand side of Figure 7 shows a possible result of running the TD algorithm on the STN at the left-hand side of the figure. Notice that the edge from A to B (which represents an *inter-subnetwork* constraint) has been made redundant by the new path (B, z, A). As a result, no matter when X executes A (while maintaining the consistency of its local subnetwork) and no matter when Y executes B (while maintaining the consistency of its local subnetwork), the consistency of the global network is ensured. Although this example is quite simple, it represents the canonical trick employed by the TD algorithm to decouple an arbitrarily complex network. The main result of that paper is that any consistent STN can be decoupled into independent subnetworks according to any pre-defined partition of the non-z time-points. In addition, the network can be fully decoupled by processing only the inter-subnetwork *edges* of the sort shown in Figure 7 and ignoring all other inter-subnetwork *paths.* A more recent work [4] extended that result by showing that the resultant decoupling can be required to be *minimal* in the sense that any weakening of its constraints would foil the Mergeable Solutions Property.

Although the TD algorithm does not address real-time issues such as dynamic controllability, it enables a group of agents working on a set of temporally related tasks to distribute their control of that network among themselves such that if each agent maintains the consistency of its local subnetwork, the consistency of the global network will be ensured. This type of control distribution is more general than that considered by Morris et al. in that the subnetworks controlled by each agent can be arbitrarily complex.

One shortcoming of a temporally decoupled network is that all of the subnetworks are required to be fully decoupled from one another. As a result, temporally decoupled networks can sometimes be tightly constrained. The next section introduces a relaxed form of temporal decoupling that results in less-constrained networks.

6. A MORE GENERAL METHOD FOR DISTRIBUTING CONTROL OF AN STN

This section introduces *relative temporal decouplings*, which are the same as ordinary temporal decouplings, except that one of the subnetworks (comprising the set of time-points $\mathcal{T}_W$ in the formal

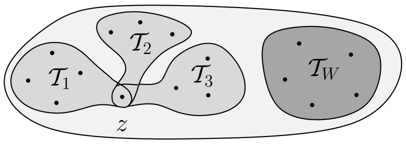


Figure 8: A sample relative z-partition

treatment that follows) is not required to be fully decoupled from the others. There are several important reasons that such a slight change can have a non-trivial impact for agents using a temporal network to coordinate their activities. First, the Relative Temporal Decoupling (RTD) algorithm (presented in this section) generates networks that are less constrained than those generated by the TD algorithm. Second, the RTD algorithm can be recursively applied to subnetworks, generating a hierarchically decoupled network in which each level of the hierarchy is less constrained than its fully decoupled counterpart. Third, the subnetworks at any level of such a hierarchy (including the top level) are controllable according to a simple scheme described in this section.

The RTD algorithm enables agents to distribute independent control to those agents that require greater flexibility. Furthermore, even those agents whose control is partially restricted can, in the long run, end up having greater flexibility than they would have had in a fully decoupled network.

The rest of this section defines a relative temporal decoupling, presents an algorithm for generating such decouplings, and presents a scheme for distributing the control of a decoupled network among a group of agents based on a relative temporal decoupling. The control scheme distinguishes two types of subnetworks: one that is independently controlled and one whose control is restricted in that an agent may only add constraints that satisfy a set of necessary and sufficient bounds, called Lambda Bounds. Space limitations preclude giving detailed proofs of the theorems (which can be found in a longer work [4]); however, sketches are provided.

6.1 The RTD Problem

For an ordinary temporal decoupling, the time-points must first be partitioned into disjoint subsets—except that each subset contains the fixed time-point, z. For a relative temporal decoupling, the subset, $\mathcal{T}_W$, does not contain z.

DEFINITION 11 (RELATIVE Z-PARTITION). *Let $\mathcal{T}$, $\mathcal{T}_W$, and $\mathcal{T}_1, \ldots, \mathcal{T}_m$ be sets of time-points. A* z-partition of $\mathcal{T}$ relative to $\mathcal{T}_W$ *is a sequence $(\mathcal{T}_1, \ldots, \mathcal{T}_m; \mathcal{T}_W)$ such that:*

- *$\mathcal{T}_r \cap \mathcal{T}_s = \{z\}$, for all $r \neq s$, $1 \leq r, s \leq m$; and*
- *$\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_m = \mathcal{T} \setminus \mathcal{T}_W$.*

A relative z-partition (with $m = 3$) is depicted in Figure 8.

In an ordinary temporal decoupling, the merger of arbitrary solutions to the decoupled subnetworks is guaranteed to be a solution for the global network. In a relative temporal decoupling, the merger of arbitrary solutions to the subnetworks corresponding to $\mathcal{T}_1, \ldots, \mathcal{T}_m$ (i.e., all time-points except those in $\mathcal{T}_W$) must be *extendible* to a solution for the global network.

DEFINITION 12 (EXTENDIBLE PARTIAL SOLUTION). *Let $\mathcal{S}$ be an STN over time-points in $\mathcal{T}$. Let $\mathcal{T}' \subseteq \mathcal{T}$ be a set of time-points. An* extendible partial solution *for $\mathcal{S}$ over $\mathcal{T}'$ is a set of assignments for the time-points in $\mathcal{T}'$ that is extendible to a solution for $\mathcal{S}$.*

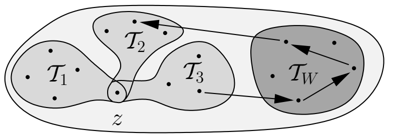


Figure 9: A sample tight, proper, mixed path

DEFINITION 13 (RELATIVE TEMPORAL DECOUPLING). *$\mathcal{S}_1 = (\mathcal{T}_1, \mathcal{C}_1)$, ..., $\mathcal{S}_m = (\mathcal{T}_m, \mathcal{C}_m)$ are said to temporally decouple $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ relative to $\mathcal{T}_W$ if:*

- *$(\mathcal{T}_1, \ldots, \mathcal{T}_m; \mathcal{T}_W)$ is a z-partition of $\mathcal{T}$ relative to $\mathcal{T}_W$;*
- *$\mathcal{S}_1, \ldots, \mathcal{S}_m$ are consistent STNs; and*
- *(Relative Mergeable Solutions Property) Any solutions $\mathcal{X}_1, \ldots, \mathcal{X}_m$ to $\mathcal{S}_1, \ldots, \mathcal{S}_m$ may be merged to form an extendible partial solution for $\mathcal{S}$ over $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_m$.*

DEFINITION 14 (RTD PROBLEM). *Given an STN $\mathcal{S}$ whose time-points $\mathcal{T}$ are z-partitioned by $\mathcal{T}_1, \ldots, \mathcal{T}_m$ relative to $\mathcal{T}_W$, find sets of constraints $\mathcal{C}_1, \ldots, \mathcal{C}_m$ such that the STNs $(\mathcal{T}_1, \mathcal{C}_1)$, ..., $(\mathcal{T}_m, \mathcal{C}_m)$ temporally decouple $\mathcal{S}$ relative to $\mathcal{T}_W$.*

6.2 The RTD Algorithm

When generating an ordinary temporal decoupling, constraints must be added to the network to ensure that each inter-subnetwork *edge* is made redundant, as illustrated in Figure 7. As already observed, this ensures that each inter-subnetwork *path* is also made redundant. When generating a relative temporal decoupling, inter-subnetwork paths must similarly be made redundant. As will be seen, it suffices to deal with *tight, proper, mixed (TPM) paths.*

DEFINITION 15 (TIGHT, PROPER, MIXED PATH). *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be an STN. Let $\mathcal{T}_1, \ldots, \mathcal{T}_m$ and $\mathcal{T}_W$ be sets of time-points such that $(\mathcal{T}_1, \ldots, \mathcal{T}_m; \mathcal{T}_W)$ is a relative z-partition of $\mathcal{T}$. Let r and s be arbitrary such that $1 \leq r, s \leq m$ and $r \neq s$. Let $t_i \in \mathcal{T}_r \setminus \{z\}$ and $t_j \in \mathcal{T}_s \setminus \{z\}$ be arbitrary. A shortest path P of the form $(t_i, w_1, \ldots, w_k, t_j)$ is called a* tight, proper, mixed path *in $\mathcal{S}$ relative to $\mathcal{T}_W$ if: $k \geq 0$ and $\{w_1, \ldots, w_k\} \subseteq \mathcal{T}_W$.*

A sample TPM path is depicted in Figure 9. In general, TPM paths may travel through points in $\mathcal{T}_W$, but do not begin or end in $\mathcal{T}_W$. The following theorem shows that the ordinary TD algorithm can be tricked into generating a relative temporal decoupling simply by adding a set of *redundant* constraints to the network, one for each TPM path.

THEOREM 16. *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be a consistent STN whose time-points are z-partitioned by $\mathcal{T}_1, \ldots, \mathcal{T}_m$ relative to $\mathcal{T}_W$. If to each TPM path P in $\mathcal{S}$ there corresponds an explicit constraint of the form $t_j - t_i \leq |P|$ in $\mathcal{C}$ (where t_i and t_j are the first and last points in the path P), then running the* ordinary *TD algorithm on $\mathcal{S}$ will produce a* relative *temporal decoupling of $\mathcal{S}$ with respect to the relative z-partition $(\mathcal{T}_1, \ldots, \mathcal{T}_m; \mathcal{T}_W)$.*

The reason this works is illustrated in Figure 10 (which should be compared to Figure 9). First, the TPM path from t_1 to t_2 causes the redundant *edge* from t_1 to t_2 (shown as a dashed arrow) to be added to the network. Next, the ordinary TD algorithm adds constraints to ensure that this edge is *dominated by a path through zero* by adding new edges from t_1 to z, and from z to t_2 (recall Figure 7), thereby ensuring that the TPM *path* from t_1 to t_2 is made redundant.

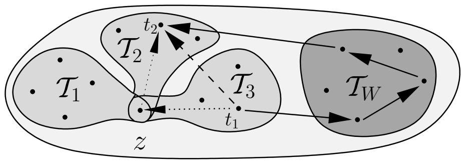


Figure 10: Making a TPM path redundant

Given: A consistent STN $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ whose time-points are z-partitioned by $\mathcal{T}_1, \ldots, \mathcal{T}_m$ relative to $\mathcal{T}_W$.

(0) Let $\mathcal{E} = \emptyset$.

(1) Let $\mathcal{C}_{Wx} = \{(t_j - t_i \leq \delta) \in \mathcal{C} : t_j \neq z$ and $t_i \in \mathcal{T}_W\}$.

(2) Let $\mathcal{D}_{Wx}$ be the distance matrix for $(\mathcal{T}, \mathcal{C}_{Wx})$.

(3) `FOR` each $\mathcal{T}_r, \mathcal{T}_s,\ t_p \in \mathcal{T}_r \setminus \{z\}$, and $t_q \in \mathcal{T}_s \setminus \{z\}$:
If there is an edge E from t_p to t_q in $\mathcal{C}$ such that $\mathcal{D}(t_p, t_q) = |E|$ and $\mathcal{D}(t_p, z) + \mathcal{D}(z, t_q) > |E|$,
Then add E to $\mathcal{E}$;
Otherwise, `FOR` each $w \in \mathcal{T}_W$:
If: $(w - t_p \leq \mathcal{D}(t_p, w)) \in \mathcal{C}$
and: $\mathcal{D}(t_p, w) + \mathcal{D}_{Wx}(w, t_q) = \mathcal{D}(t_p, t_q)$
and: $[\mathcal{D}(t_p, z) + \mathcal{D}(z, t_q)] > \mathcal{D}(t_p, t_q)$
then: add $(t_q - t_p \leq \mathcal{D}(t_p, t_q))$ to $\mathcal{E}$ and break out of the inner `FOR` loop.

Return: $\mathcal{E}$.

Figure 11: Generating the initial set of (redundant) constraints to trick the ordinary TD algorithm into generating a relative temporal decoupling

Figure 11 presents an algorithm for generating the initial set of constraints needed to trick the ordinary TD algorithm into "doing the right thing". The "Otherwise" clause in Step (3) of the algorithm is illustrated in Figure 12, where squiggly arcs represent paths. An edge from t_p to t_q is added to $\mathcal{E}$ if a shortest path from t_p to t_q through some $w \in \mathcal{T}_W$ is shorter than any shortest path from t_p to t_q through z.

6.3 Lambda Bounds

In a relative temporal decoupling, the agents controlling the subnetworks $\mathcal{S}_1, \ldots, \mathcal{S}_m$ are allowed to operate independently. As a result, the agent controlling the rest of the time-points (i.e., those in $\mathcal{T}_W$) must be certain not to do anything that would impose any additional constrainedness on those subnetworks. For example, for the STN shown in Figure 13, tightening the edge from w_k to w_l must not introduce a new, strictly shorter path from x_p to x_q. In other words, the following inequality must be maintained:

$$\mathcal{D}(x_p, x_q) \ \leq\ \mathcal{D}(x_p, w_k) + \delta + \mathcal{D}(w_l, x_q)$$

Equivalently, δ must be greater than or equal to

$$\mathcal{D}(x_p, x_q) - \mathcal{D}(x_p, w_k) - \mathcal{D}(w_l, x_q).$$

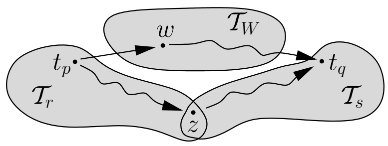


Figure 12: The "Otherwise" clause from Step (3) of the algorithm in Figure 11.

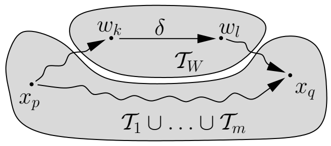


Figure 13: Adding an edge involving points in $\mathcal{T}_W$

The Lambda Bound for the edge from w_k to w_l is the maximum value of all such lower bounds on δ taken over all possible paths from x_p to x_q via w_k and w_l, where x_p and x_q can be any time-points within $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_m$. As will be seen below, the Lambda Bounds may be computed without considering *all* such paths. In addition, it turns out to be possible for the agent controlling the time-points in $\mathcal{T}_W$ to add edges linking $\mathcal{T}_W$ to the other subnetworks without imposing any additional intra- or inter-subnetwork constrainedness among $\mathcal{S}_1, \ldots, \mathcal{S}_m$. Thus, although the agent controlling the time-points in $\mathcal{T}_W$ is subject to the tighter Lambda Bounds, its control extends slightly beyond $\mathcal{T}_W$.

DEFINITION 17 (LAMBDA BOUNDS). *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C})$ be a consistent STN. Let $(\mathcal{T}_1, \ldots, \mathcal{T}_m; \mathcal{T}_W)$ be a relative z-partition of $\mathcal{T}$. Let $\mathcal{T}' = \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_m$. Let $\mathcal{C}_{xW}$ and $\mathcal{C}_{Wx}$ be given by:*

$$\begin{array}{rcl} \mathcal{C}_{xW} & = & \{(t_j - t_i \leq \delta) \in \mathcal{C} : t_j \in \mathcal{T}_W\} \\ \mathcal{C}_{Wx} & = & \{(t_j - t_i \leq \delta) \in \mathcal{C} : t_i \in \mathcal{T}_W\} \end{array}$$

Let $\mathcal{D}_{xW}$ and $\mathcal{D}_{Wx}$ be the respective distance matrices for the STNs, $\mathcal{S}_{xW} = (\mathcal{T}, \mathcal{C}_{xW})$ and $\mathcal{S}_{Wx} = (\mathcal{T}, \mathcal{C}_{Wx})$.

Let λ be as follows, where $w_k, w_l \in \mathcal{T}_W$ and $x_p, x_q \in \mathcal{T}'$ are arbitrary:

$$\lambda(w_k, w_l) = \max_{x_p, x_q \in \mathcal{T}'} [\mathcal{D}(x_p, x_q) - \mathcal{D}_{xW}(x_p, w_k) - \mathcal{D}_{Wx}(w_l, x_q)]$$

$$\lambda(x_p, w_l) = \max_{x_q \in \mathcal{T}'} [\mathcal{D}(x_p, x_q) - \mathcal{D}_{Wx}(w_l, x_q)]$$

$$\lambda(w_k, x_q) = \max_{x_p \in \mathcal{T}'} [\mathcal{D}(x_p, x_q) - \mathcal{D}_{xW}(x_p, w_k)]$$

Then a constraint E of the form $t_j - t_i \leq \delta$, where $t_i \in \mathcal{T}_W$ or $t_j \in \mathcal{T}_W$, is said to satisfy the appropriate Lambda Bound *for $\mathcal{S}$ relative to $\mathcal{T}_W$ if and only if $\delta \geq \lambda(t_i, t_j)$.*

The following theorem shows that, in the context of a relative temporal decoupling, if the agent controlling the time-points in $\mathcal{T}_W$ adds any constraint E satisfying both the NTC and the appropriate Lambda Bound, then the relative temporal decoupling will be preserved. Thus, the Lambda Bounds are necessary and sufficient bounds for preserving a relative temporal decoupling and, hence, the Relative Mergeable Solutions Property.

THEOREM 18. *Given the same setup as in Definition 17, let E be a constraint of the form, $t_j - t_i \leq \delta$, where $t_i \in \mathcal{T}_W$ or $t_j \in \mathcal{T}_W$, and such that E satisfies the NTC for $\mathcal{S}$. Let $\mathcal{C}^+ = \mathcal{C} \cup \{E\}$ and $\mathcal{S}^+ = (\mathcal{T}, \mathcal{C}^+)$ (i.e., $\mathcal{S}^+$ is the (consistent) STN that results from adding E to $\mathcal{S}$). Let $\mathcal{S}_1, \ldots, \mathcal{S}_m$ be a temporal decoupling of $\mathcal{S}$ relative to $\mathcal{T}_W$. Then, E satisfies the appropriate Lambda Bound for $\mathcal{S}$ relative to $\mathcal{T}_W$ if and only if the subnetworks $\mathcal{S}_1, \ldots, \mathcal{S}_m$ also form a temporal decoupling of $\mathcal{S}^+$ relative to $\mathcal{T}_W$.*

Theorem 18 implies that if (1) the agents controlling the subnetworks $\mathcal{S}_1, \ldots, \mathcal{S}_m$ in a relative temporal decoupling maintain the consistency of their local subnetworks and (2) the agent controlling the time-points in $\mathcal{T}_W$ observes the NTC and the appropriate Lambda Bounds, then the consistency of the entire network is ensured. Thus, a relative temporal decoupling can serve as the basis for distributing the control of a temporal network among a group of agents.

6.4 Hierarchically Distributing Control

If the subnetworks $\mathcal{S}_1, \ldots, \mathcal{S}_m$ temporally decouple $\mathcal{S}$ relative to $\mathcal{T}_W$, then the agents controlling $\mathcal{S}_1, \ldots, \mathcal{S}_m$ can operate independently without fear of threatening the consistency of the global network. In contrast, the agent controlling the time-points in $\mathcal{T}_W$ cannot act independently; instead, it must observe the Lambda Bounds described above. Since nothing that happens within any of the independently controlled subnetworks can threaten the consistency of the global network (as long as each subnetwork remains consistent), each of the subnetworks $\mathcal{S}_1, \ldots, \mathcal{S}_m$ may be treated as a stand-alone STN and may thus be recursively decoupled, creating a hierarchical relative temporal decoupling of the global network, where the agent controlling the *left-over* time-points (i.e., $\mathcal{T}_W$) in any decoupling must observe the Lambda Bounds corresponding to that local subnetwork.

7. CONCLUSIONS

This paper makes the following contributions to the theory of Simple Temporal Networks. First, it introduces Augmented STNs that include an explicit representation of the ever-changing present moment and formally defines *Add-Constraint* and *Execute-Time-Point* operators that modify Augmented STNs. Augmented STNs provide a framework for understanding a wide range of control issues that arise when agents seek to use temporal networks to coordinate their activity in real time. Second, this paper presents a general conception of distributing the control of a temporal network among a group of agents. This conception captures existing work on the use of STNs in real time, including scenarios in which Nature controls some temporal durations. It also captures existing work on the use of STNs among a group of agents seeking to coordinate their temporally related activities. Third, this paper presents a new way of distributing control of a temporal network among a group of agents. The Relative Temporal Decoupling algorithm is a generalization of the existing Temporal Decoupling algorithm. Future research in this area will be directed toward combining the results on (1) dynamic controllability of networks involving a single agent sharing control with Nature and (2) multi-agent scenarios in which subnetworks can be arbitrarily complex.

8. REFERENCES

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.

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

[3] L. Hunsberger. Algorithms for a temporal decoupling problem in multi-agent planning. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, 2002.

[4] L. Hunsberger. *Group Decision Making and Temporal Reasoning*. PhD thesis, Harvard University, 2002. (Available as Harvard Technical Report TR-05-02).

[5] P. Morris and N. Muscettola. Execution of temporal plans with uncertainty. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 491–496, 2000.

[6] P. Morris, N. Muscettola, and T. Vidal. Dynamic control of plans with temporal uncertainty. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 494–499, 2001.

[7] I. Tsamardinos. Reformulating temporal plans for efficient execution. Master's thesis, University of Pittsburgh, 2000.

[8] I. Tsamardinos, N. Muscettola, and P. Morris. Fast transformation of temporal plans for efficient execution. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 254–261. The MIT Press, Cambridge, MA, 1998.

[9] T. Vidal and H. Fargier. Contingent durations in temporal CSPs: from consistency to controllabilities. In *Proceedings of Fourth Workshop on Temporal Representation and Reasoning (TIME-97)*, 1997.

[10] T. Vidal and M. Ghallab. Temporal constraints in planning: Free or not free? In *Proceedings of the International Workshop on Constraint-based Reasoning (CONSTRAINT-95)*, 1995.