The Mathematics of Dispatchability Revisited

Paul Morris
NASA Ames Research Center
Moffett Field, CA 94035, U.S.A.

Abstract

Dispatchability is an important property for the efficient execution of temporal plans where the temporal constraints are represented as a Simple Temporal Network (STN). It has been shown that every STN may be reformulated as a dispatchable STN, and dispatchability ensures that the temporal constraints need only be satisfied locally during execution. Recently it has also been shown that Simple Temporal Networks with Uncertainty, augmented with wait edges, are Dynamically Controllable provided every projection is dispatchable. Thus, the dispatchability property has both theoretical and practical interest.

One thing that hampers further work in this area is the underdeveloped theory. The existing definitions are expressed in terms of algorithms, and are less suitable for mathematical proofs. In this paper, we develop a new formal theory of dispatchability in terms of execution sequences. We exploit this to prove a characterization of dispatchability involving the structural properties of the STN graph. This facilitates the potential application of the theory to uncertainty reasoning.

Introduction

The concept of Dispatchability was introduced by (Muscettola, Morris, and Tsamardinos 1998) in the context of execution of temporal plans. The work was motivated by the needs of the Remote Agent experiment (Muscettola et al. 1998), where an AI system controlled the NASA Deep Space I spacecraft for several days.

Dispatchability is a property of a Simple Temporal Network (STN) (Dechter, Meiri, and Pearl 1991) that allows the network to be correctly executed even when propagation is limited to neighboring timepoints. Another way of looking at this (Morris et al. 2013) is that temporal constraints need only be checked locally when deciding whether to execute a procedure. It was shown that every consistent STN can be reformulated as an equivalent *minimum dispatchable* network, which improves execution efficiency while providing real-time guarantees (Muscettola, Morris, and Tsamardinos 1998). A more efficient version of the reformulation algorithm was presented in (Tsamardinos, Muscettola, and Morris 1998). Dispatchability algorithms have also been extended to disjunctive temporal problems (Tsamardinos, Pollack, and Ganchev 2001; Shah and Williams 2008).

Previous papers (e.g. (Shah et al. 2007)) have noted a relationship between dispatchability and a property involving temporal uncertainty called Dynamic Controllability (Vidal and Fargier 1999; Morris, Muscettola, and Vidal 2001; Hunsberger 2009). In particular, it has been shown (Morris 2014) that Simple Temporal Networks with Uncertainty (STNU), augmented with wait edges, are Dynamically Controllable provided every projection is dispatchable. This suggests that an increased understanding of dispatchable networks may benefit studies of temporal uncertainty. A number of authors, e.g. (Rossi, Venable, and Yorke-Smith 2006; Tsamardinos and Pollack 2003; Moffitt 2007; Moffitt and Pollack 2007; Hunsberger 2002), have studied temporal uncertainty in wider contexts than STNUs.

In this paper, we develop an updated theory of dispatchability oriented towards mathematical proofs, and use it to provide characterizations of dispatchability in terms of the structural properties of the STN graph. In particular, we show that an STN is dispatchable if and only if each temporal path constraint is enforced by a particular type of path. This provides insight into "reduction rules" used in previous work (Shah et al. 2007; Nilsson, Kvarnström, and Doherty 2013; Morris 2014) that can be seen to create such paths.

Preliminaries

In this section, we review relevant properties of the Simple Temporal Network formalism (originally called Simple Temporal Problem) introduced by (Dechter, Meiri, and Pearl 1991).

A *Simple Temporal Network* (STN) is a pair <N,C> where N is a set of temporal events or *timepoints*, and C is a set of direct constraints of the form $a \leq y - x \leq b$ where x and y are time variables corresponding to the timepoints and a and b are extended[1] real numbers. The network has a graphical representation where the above constraint is represented in the form $\mathrm{x} \xrightarrow{[\mathrm{a,b}]} \mathrm{y}$.

Note that $a \leq y - x$ can be rewritten as $x - y \leq -a$. Thus, all the constraints can be re-expressed as upper-bound constraints with simple numerical values, and the graph in

[1] Negative or positive infinity can be used to indicate the absence of a lower or upper bound, respectively.

this form is called the *distance graph*. In this case, the example constraint corresponds to the two edges $\mathrm{x} \xrightarrow{\mathrm{b}} \mathrm{y}$ and $\mathrm{x} \xleftarrow{-\mathrm{a}} \mathrm{y}$. It is sometimes convenient to mix the two forms of graphical notation in a single diagram.

The upper-bound value on an edge in the distance graph is referred to as the edge *weight* (also known as the edge *length* or edge *distance*). Note that a constraint with a negative weight such as $\mathrm{x} \xleftarrow{-5} \mathrm{y}$ implies timepoint x precedes timepoint y. Observe that from constraints $y - x \leq b$ and $z - y \leq c$ we can infer $z - x \leq b + c$; thus, paths in the distance graph give rise to derived constraints.

An STN is consistent if and only if the distance graph does not contain a *negative cycle*, i.e., a cycle where the weights add up to a negative value. A consistent *schedule* is a mapping from the timepoints to real numbers that satisfies the constraints; it exists if and only if the STN is consistent. This can be determined by a single-source shortest path distance computation such as with the Floyd-Warshall or Bellman-Ford algorithms (Cormen, Leiserson, and Rivest 1990). If the STN is consistent, such computations may also determine *propagated* upper and lower bounds for a timepoint x relative to a source q as $\mathrm{upper}(x) = \mathrm{distance}(q, x)$ and $\mathrm{lower}(x) = -\mathrm{distance}(x, q)$.

A *partial schedule* is a mapping from a subset of the timepoints to the real numbers. A partial schedule S is *locally consistent* if it satisfies the constraints between pairs of nodes in its domain. The (Dechter, Meiri, and Pearl 1991) paper shows that any STN can be represented in a particular form called the *d-graph*, also known as the *minimal* or *All-Pairs* network, where the derived shortest-path distance constraints are included as direct constraints. It further shows that every partial schedule that is locally consistent with respect to the d-graph can be extended to a consistent complete schedule. As a more convenient terminology, we will say a partial schedule is *path-consistent* if it is locally consistent with respect to the d-graph, since this means it satisfies the shortest-path distance constraints.

For our purposes (as in (Muscettola, Morris, and Tsamardinos 1998)) we require a **local** version of propagation. We will say two nodes are *neighbors* if there is a direct constraint between them (in either direction). A node is a neighbor of a partial schedule S if it is a neighbor of some node in the domain of S. Similar to the shortest path distance computation we can propagate a time bound from a node x in the domain of a partial schedule S to a neighboring node y not in the domain of S; we refer to this as *local propagation*. For example, given a constraint $y - x \leq u$, we can infer an upper bound $S(x) + u$ on y and similarly we can infer a lower bound $S(x) - u$ in the case where $x - y \leq u$. We can also compute tightest such bounds taking into account all the nodes in the domain of S and their constraints to neighboring nodes. Thus, for a neighbor node y not in domain(S), we define

$$\mathrm{upper}(S, y) = \min\{S(x) + u \mid x \text{ in } S \text{ and } \mathrm{x} \xrightarrow{\mathrm{u}} \mathrm{y}\}$$

$$\mathrm{lower}(S, y) = \max\{S(x) - u \mid x \text{ in } S \text{ and } \mathrm{x} \xleftarrow{\mathrm{u}} \mathrm{y}\}$$

where x in S indicates x is in the domain of S.

Time Dispatching Algorithm

The KR'98 paper (Muscettola, Morris, and Tsamardinos 1998) "Reformulating Temporal Plans For Efficient Execution" (hereafter called MMT) defines dispatchability. First it defines an algorithm for a "dispatching execution" as follows.

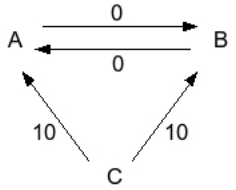
```
TIME DISPATCHING ALGORITHM (TDA):
1. Let
     A = {start_time_point}
     current_time = 0
     S = {}
2. Arbitrarily pick a time point TP in
   A such that current_time belongs to
   its time bound;
3. Set TP execution time to
   current_time and add TP to S;
4. Propagate the time of execution of
   TP to its IMMEDIATE NEIGHBORS in
   the distance graph;
5. Put in A all time points TPx such
   that all negative edges starting
   from TPx have a destination in S;
6. Wait until current_time has advanced
   to some time between
      min{lower_bound(TP) : TP in A}
   and
      min{upper_bound(TP) : TP in A}
7. Go to 2 until every time point is
   in S.
```

Then it defines a network to be dispatchable if it is correctly executed by a dispatching execution. We may think of the dispatching algorithm as performing a number of local checks of consistency to guide the execution. For example, it requires a timepoint to be executed within its locally propagated time bounds. Also, the step 5 condition prevents a TP x from being executed until it is "enabled," i.e., all the TPs that are directly constrained to precede x have already been executed.

The network is correctly executed if the local checks are sufficient to guarantee global consistency; if not, the dispatching execution will be "blocked" at some point, for example by `A` containing a TP whose upper bound is in the past (so step 6 waits forever). The (Muscettola, Morris, and Tsamardinos 1998) paper gives examples of this blocking for general STNs, but goes on to show that any STN can be reformulated to an equivalent *minimal dispatchable* STN in which blocking does not occur. The reformulation involves starting with the All-Pairs network, which is shown to be dispatchable. The paper then introduces a process where an edge can be removed as not needed for dispatchability, provided it satisfies a *triangle rule* with respect to another edge of the same sign, said to *dominate* it.

This captures the intuitive concept and leads to effective algorithms that have been of great practical value. However, the formal treatment in MMT has certain characteristics that make it less suitable for proving results about dispatchability, as we seek to do in this paper. These are:

1. In general, it is inconvenient to work mathematically with properties defined in terms of an algorithm. Note that the TDA needs to be interpreted as an *idealized* algorithm. For example, steps 2-5 must take zero time in the case where a TP x is executed and another TP y is constrained ([0,0] constraint) to occur at the SAME time as x.
2. The blocking concept is a complicated one that may be awkward to use in formal proofs.
3. The reformulation algorithm in (Muscettola, Morris, and Tsamardinos 1998) has a minor problem because the dominance analysis overlooks the following pathological case.



In MMT terms, AB strictly dominates CB and BA strictly dominates CA. Thus, both CA and CB would be removed (see the code in Figure 5 in MMT). However, the resulting network would not be equivalent to the original. Note that CA and CB are NOT mutually dominating.[2]

For these reasons, we redefine "dispatching execution" without reference to an algorithm, and clarify how it relates to dispatchability. In the following sections, we consider sequences where timepoints are "instantiated" (i.e., set to occur at a specific time) and where the instantiated times are propagated to neighboring timepoints in the STN graph. We will then consider executions to be instantiations that proceed forward in time. In our analysis, dominance does not play a central role so we avoid the issue in item 3 above.

Instantiation/Execution Sequences

In the following discussion, we assume the STNs are consistent unless there is an explicit statement otherwise. Recall from the preliminaries section that a *partial schedule* for an STN is a mapping from a subset of the timepoints to time values, *local consistency* of a partial schedule S means S satisfies the direct constraints between every pair of timepoints in its domain, and *local propagation* involves propagating lower and upper bounds to neighboring timepoints.

Intuitively, an instantiation-sequence may be viewed as an attempt to find a partial solution to an STN by scheduling a timepoint at a fixed time within its bounds, locally propagating bounds from the timepoint, and repeating for some subset of the timepoints.

Definition 1 *Given an STN, an* instantiation-sequence *E is a pair <A, S> where A = $\{a_1, ..., a_n\}$ is a sequence*[3] *of distinct timepoints and S is a locally consistent partial schedule defined on the timepoints in A.*

Thus, an instantiation-sequence has two parts: a partial schedule part that determines the times of the timepoints, and a sequence part that specifies the order of propagation.

We will say an instantiation-sequence is *complete* if the $\{a_i\}$ sequence[4] includes every timepoint in the STN. Note that we do NOT require instantiation-sequences to be complete in general. This allows us to define a prefix relationship between them. In the definition, a sequence $\{a_1, ..., a_m\}$ is a *prefix* of a sequence $\{b_1, ..., b_n\}$ if $m \leq n$ and $a_i = b_i$ for $1 \leq i \leq m$. A partial schedule Q *coincides* with a partial schedule P on a sequence A if $Q(a_i) = P(a_i)$ for each a_i in A.

Definition 2 *An instantiation-sequence <A, P> is a* prefix *of an instantiation-sequence <B, Q> if the sequence A is a prefix of the sequence B and the partial schedule Q coincides with P on A.*

Intuitively, an execution is an instantiation that proceeds forward in time. In the following, a timepoint y is a *direct predecessor* of a timepoint x if there is an explicit edge with a negative weight from x to y in the distance graph of the STN.

Definition 3 *An* execution-sequence *for an STN is an instantiation-sequence E = <$\{a_i\}, S$> such that*
(1) $S(a_{i+1}) \geq S(a_i)$, for each i, and
(2) each direct predecessor of a timepoint in E is also in E.[5]

Condition (2) may be viewed as a reasonable strengthening of local consistency for executions, since if it is false, an extension to a complete execution-sequence is always impossible.

Note that even though an execution proceeds forward in time, the partial schedule alone does not determine the order of propagation when timepoints are simultaneous. The sequence part makes that unambiguous, which is useful for proving results about executions.

Execution sequences bear a resemblance to Real Time Execution Decisions (Hunsberger 2009), though they differ in detail and setting.

It is easy to see that any prefix of an execution-sequence is itself an execution-sequence since local consistency implies that $S(y) < S(x)$ if y is a direct predecessor of x.

NOTATION The following notations assume E = <$\{a_1, ..., a_n\}, S$> is an execution sequence.

We write E_i to denote the prefix of E that contains the timepoints up to a_i for $i < n$. By convention, E_0 is the empty execution sequence.

[2] It should be noted that the algorithm in the later paper (Tsamardinos, Muscettola, and Morris 1998) fortuitously handles this case correctly because there the AB rigid component would be contracted to a single node.

[3] We follow old-school mathematics (e.g., (Hille 1965)) in using braces to denote sequences as well as sets; it should be clear from the context which is intended.

[4] We may sometimes write $\{a_1, ..., a_n\}$ as $\{a_i\}$ when we don't need to refer to a_n.

[5] i.e., x in E and y $\xleftarrow{-u}$ x implies y in E. We take the liberty of saying a timepoint is *in* E to mean it is one of the a_i.

A timepoint x is *enabled* by E if every direct predecessor of x is in E. We write *enabled(E)* to denote the set of timepoints that are enabled by E. Note that E is contained in enabled(E) by condition (2) of the definition. However, there may be timepoints in enabled(E) that are not in E; we denote this set by *ready(E)*.

As an example, a_n must be in enabled(E_{n-1}) by (2), and hence is in ready(E_{n-1}) since it is not in E_{n-1}.

We write lower(E, x) to denote the lower bound on x obtained by local propagation from the $\{a_i\}$ in E with respect to their scheduled times in S, and similarly for upper(E, x).[6]

We set latest(E) = $S(a_n)$ and also use the following notations:

$$\text{MIN\_LOWER (E)} = \min\{\text{ lower (E, } x) \mid x \text{ in ready (E) }\}$$

$$\text{MIN\_UPPER (E)} = \min\{\text{ upper (E, } x) \mid x \text{ in ready (E) }\}$$

OBSERVATION: Note the monotonicity of key concepts with respect to an execution sequence. For example, lower(E_i, x) is non-decreasing with respect to i, and upper(E_i, x) is non-increasing. Similarly, latest(E_i) = $S(a_i)$ can only increase or stay the same.

Dispatchability

Our goal is to define a dispatching execution sequence, which we will call a dispatch sequence, in such a way that it captures the same local checks as the TIME DISPATCHING ALGORITHM (TDA). As we have seen, the definition of execution sequence already incorporates the requirement that a timepoint be enabled before it can enter the sequence, and it cannot be assigned a time earlier than the timepoints already in the sequence.

The following is a literal transcription of the remaining criterion in the TDA:

$$\text{MIN\_LOWER}(E_{i-1}) \leq S(a_i) \leq \text{MIN\_UPPER}(E_{i-1}).$$

It turns out that every execution sequence must already satisfy the left-hand inequality. To see this, suppose otherwise. Then

$$\begin{aligned} S(a_i) &< \text{MIN\_LOWER}(E_{i-1}) \\ &\leq \text{lower}(E_{i-1}, a_i) \\ &\leq S(a_j) - u \text{ for some } \mathrm{a_i} \xrightarrow{\mathrm{u}} \mathrm{a_j} \end{aligned}$$

where $j < i$. But this implies $S(a_j) - S(a_i) > u$, which would violate local consistency of the execution sequence.

The right-hand inequality is an interesting check that is not already captured by execution sequences. It is phrased as a requirement on the timepoint currently being executed that its execution time should not exceed the upper bounds of the other timepoints eligible and waiting for execution.[7]

[6] i.e. lower(E, x) = lower(S, x) and upper(E, x) = upper(S, x) where lower(S, x) and upper(S, x) are as defined in the preliminaries section.

[7] The apparent intent is to indirectly enforce deadlines. For example, if some eligible timepoint x is at its deadline, then x would have to be executed before the current time could advance further.

Again, this is a local check that, if violated, makes a complete consistent execution impossible. As an example, consider

$$\mathrm{a_3} \xleftarrow{5} \mathrm{a_1} \xrightarrow{10} \mathrm{a_2}$$

Note that $<\{a_1, a_2\}, S>$ where $S(a_1) = 0$ and $S(a_2) = 10$ is a valid execution sequence but it cannot be extended to a_3 without violating local consistency or monotonicity.

A stronger version of this local check would take into consideration the upper bounds of all the unexecuted timepoints, not just those currently eligible for execution. Intuitively, a violation of these requirements means that the current execution is broken in terms of timepoints with missed deadlines. This leads us to define the following properties.

Definition 4 *An execution-sequence E = $<\{a_1, ..., a_n\}, S>$ is* weakly dispatching *if x in ready(E_{n-1}) implies upper(E_{n-1}, x) $\geq$ latest(E).*

Definition 5 *An execution-sequence E = $<\{a_i\}, S>$ is* strongly dispatching *if upper(E,x) $\geq$ latest(E) for all timepoints x not in E.*

It is not hard to see from the monotonicity observation that if E satisfies one of these properties, then the prefixes of E must satisfy the same property. (If a timepoint misses a deadline, it can never thereafter meet its deadline.)

As expected, the strong property implies the weak one.

Lemma 1 *Strongly dispatching implies weakly dispatching.*

Proof: Suppose an execution-sequence E is strongly dispatching where E = $<\{a_1, ..., a_n\}, S>$. If x is in ready(E_{n-1}) then either $x = a_n$ or x is not in E.

If x is not in E then

$$\text{latest}(E) \leq \text{upper}(E, x) \leq \text{upper}(E_{n-1}, x)$$

by strong dispatching, while if $x = a_n$ then latest(E) = $S(a_n) \leq$ upper(E_{n-1}, x) by local consistency. In both cases, upper(E_{n-1}, x) $\geq$ latest(E) so E is weakly dispatching. □

The criterion in the TDA corresponds to the weak property and we incorporate this in the definition of a dispatch sequence:[8]

Definition 6 *A* dispatch-sequence *is an execution-sequence that is weakly dispatching.*

With some additional notation, there is a "super" version of the dispatching property that involves a global rather than local condition. This will be useful for our proofs.

NOTATION We will use $d(x, y)$ to denote the shortest-path distance between two timepoints x and y in the distance graph of an STN. (See preliminaries section.). By convention $d(x, y) = \infty$ if there is no path from x to y.

Definition 7 *An* execution-sequence *E = $<\{a_i\}, S>$ is* super dispatching *if whenever x is in E and y is not in E, then $S(x) + d(x, y) \geq$ latest(E).*

[8] We will see later that we could have used the strong property instead, in the sense that it leads to an equivalent definition of dispatchability. We use the weak property for continuity with MMT and it may have a practical advantage of checking fewer timepoints.

Note that if x is a node in E that determines upper(E,y), then there is an edge of some length u from x to y. Since $d(x,y) \leq u$ we have upper$(E,y) = S(x) + u \geq S(x) + d(x,y)$. Thus the super version implies the strong version.

Observe that any solution to an STN can have its timepoints sorted according to the scheduled time (placing simultaneous timepoints in arbitrary order). Thus, every solution schedule S can be associated with a complete execution-sequence E = $<\{a_i\}, S>$. We will see below (in Theorem 1) that all the prefixes of E are super dispatching, hence weakly dispatching, and thus are dispatch-sequences.

We are almost ready to define dispatchability. The definition essentially says that for every dispatch-sequence E = <A,S>, the schedule S can be extended to a solution schedule S' where the added timepoints occur no earlier than those in E.

For an example of an STN that is not dispatchable, consider

$$\text{B} \xrightarrow{+5} \text{C} \xrightarrow{-10} \text{D}$$

and E = $<\{B\}, S>$ where $S(B) = 1$. This is a dispatch sequence. (Local propagation only reaches C, not D.) This could be extended to D without violating monotonicity; for example, we could set $S(D) = 1$ also. Note that this does not violate local consistency because C is not in the sequence. However, it clearly violates global consistency so the sequence cannot be completed. As another example, consider

$$\text{B} \xrightarrow{+10} \text{C} \xrightarrow{-5} \text{D}$$

and E = $<\{B, D\}, S>$ where $S(B) = 1$ and $S(D) = 7$. Again E is a dispatch sequence that does not extend to C.

Definition 8 *An STN is* dispatchable *if every dispatch-sequence is a prefix of a complete execution-sequence.*

The usefulness of the super dispatching property lies in the following result. Recall that a partial schedule S is *path-consistent* if $S(y) - S(x) \leq d(x,y)$ for each x and y in the domain of S.

Theorem 1 *An execution sequence E = <A,S> is a prefix of a complete execution-sequence if and only if (1) E is super dispatching and (2) S is path-consistent.*

Proof: First suppose E = $<a_1, ..., a_n$,S> is a prefix of a complete execution-sequence E$'$ = $<\{A', S'\}>$. Then S' extends S to a solution, so (2) holds. Suppose x is in E but y is not in E. Then y comes after a_n in A' so

$$S(x) + d(x,y) \geq S(x) + S'(y) - S(x) = S'(y) \geq S(a_n)$$

so (1) holds.

Conversely, suppose (1) and (2) both hold. Consider a modified STN where for each z not in E we add a new edge from z to a_n of length 0 (i.e., a new constraint that requires $z \geq a_n$).

We will show that S is still path-consistent with respect to the modified STN. Consider a cycle-free shortest path

$$x, ..., z, a_n, ..., y$$

that passes through one (and only one since the path is cycle-free) of the new edges. Thus, $d'(x,z) = d(x,z)$, $d'(a_n,y) = d(a_n,y)$, and $d'(z,a_n) = 0$, where d' is the shortest-path distance in the modified STN. From (1) we get $S(x) + d(x,z) \geq S(a_n)$, which can be rewritten as $S(a_n) - S(x) \leq d(x,z)$. We then have

$$\begin{aligned} S(y) - S(x) &= S(a_n) - S(x) + S(y) - S(a_n) \\ &\leq d(x,z) + S(y) - S(a_n) \\ &\leq d(x,z) + d(a_n,y) \\ &= d'(x,a_n) - d'(z,a_n) + d(a_n,y) \\ &= d'(x,a_n) - 0 + d'(a_n,y) \\ &= d'(x,y) \end{aligned}$$

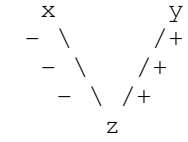
We conclude that S is path-consistent with respect to the modified STN.[9] By (Dechter, Meiri, and Pearl 1991), S can be extended to a solution schedule S' for the modified STN. Because of the added edges, $S'(z) \geq S(a_n)$ for each z not in E. Thus, by sorting S', we can form a complete execution sequence E$'$ that is an extension of E. □

Structural Characterizations

We will now prove some characterizations of dispatchability in terms of structural properties of the STN distance graph. We consider shortest paths in the distance graph, called vee-paths, where all negative edges, if any, come before all non-negative edges, if any. We also consider shortest paths, called hinge-paths, where this condition is violated. Our strategy will be to show that certain configurations of hinge-paths prevent dispatchability while vee-paths enable it. In other words, the vee-paths determine whether path constraints are enforced or not in a dispatching execution. We also consider intermediate properties, interesting in their own right, that facilitate the proof of a chain of equivalences.

DISCUSSION Intuitively, a path constraint between x and y of length $u - v$ can be enforced by propagating a lower bound of u to x and an upper bound of v to y from some common precursor z.

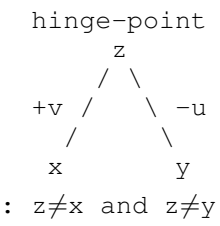
It is shown in MMT that, in a dispatching execution of the STN Distance Graph, upper bounds are propagated through non-negative edges in the forward direction, while lower bounds are propagated through negative edges in the backward direction. Thus, enforcement can be achieved by a chain of negative edges from x to to z and a chain of non-negative edges from z to y, as illustrated by this cartoon (edges directed from x to y):



This suggests that constraint enforcement in a dispatching execution might involve paths that consist of some number of negative edges followed by some number of non-negative edges. In the following, we develop concepts that lead to a rigorous derivation of this result.

[9] Recall this is the same thing as saying it is locally consistent with respect to the d-graph of the STN.

Definition 9 *A* hinge point *with respect to a pair $<x, y>$ of timepoints is an interior*[10] *point z on a shortest path from x to y such that d(x,z)*[11] *is non-negative and d(z,y) is negative, as illustrated (edges directed from x to y):*



Intuitively, the significance of this is that there can be a dispatching execution where x and y are executed before the hinge point z. Any propagation through z will occur after x and y have already been executed. Consequently, that path will not enforce the constraint between x and y in all dispatching executions. This leads to the following lemma.

Lemma 2 *Given a consistent STN and timepoint pair $<x, y>$, suppose there is a set H of timepoints such that every shortest path from x to y passes through at least one of the points, z, in H, and that point, z, is a hinge point for the path. Then there is a strongly dispatching execution-sequence whose partial schedule is not path-consistent.*

Proof: Consider a modified STN where, on each shortest path from x to y, the edge immediately following the point z in H has its weight increased by some amount $\delta > 0$. Thus, the shortest paths from x to y have their total distance increased by some positive multiple of delta. (A shortest path may pass through more than one hinge point.)

Since distances are only increased, the STN is still consistent. (There are no negative cycles.) Note that δ can be chosen to be sufficiently small such that (i) the shortest paths in the modified STN are also shortest paths in the original STN, and (ii) the distances from z in H to y are still negative.

Define a partial schedule S by setting $S(x)$ to an arbitrary value, and $S(z) = S(x) + d'(x, z)$ for each z in H, where d' gives the shortest path distance in the modified STN.. Also set $S(y) = S(x) + d'(x, y)$. By construction, S is locally consistent with respect to the d-graph for the modified STN.[12] Thus, it can be extended to a solution S' (Dechter, Meiri, and Pearl 1991). However, S is not path-consistent with respect to the *original* STN because $S(y) - S(x) \geq d(x, y) + \delta$ where $d(x, y)$ is the shortest path distance from x to y in the original STN. (Each shortest path in the modified STN passes through at least one enlarged edge.) Note that $S(z) \geq S(x)$ and $S(z) > S(y)$ for each z in H.

Now let E′ be a complete execution sequence for the modified STN that corresponds to the above solution S'. Since $S(z) \geq max(S(x), S(y))$ for each z in H, we can choose E′ so that z comes after both x and y in the sequence. Consider the prefix E of E′ up to and including x or y, whichever comes later. Thus, latest(E) = max(S(x),S(y)). Since E is a prefix of a solution, it is locally consistent and strongly dispatching for the modified STN.

Note that E does not include any of the timepoints z in H, and the modified constraints emanate from those timepoints. Thus, the constraints that determine both local consistency of E and upper(E,w), for any w, are the same for the original as for the modified STN.[13] Consequently, E is also locally consistent and strongly dispatching with respect to the original STN. However, the partial schedule of E coincides with S on x and y, and we have shown this violates path-consistency with respect to the original STN. □

This prepares the way for the following.

Definition 10 *Given an STN a* vee-path *is a shortest path in the distance graph that consists of a subpath (possibly empty) of negative edges followed by a subpath (possibly empty) of non-negative edges.*

Thus, the negative edges, if any, in a vee-path all come before the non-negative edges, if any, in the path.

The intuition behind the name is that a chain of negative edges points backward in time while a chain of active non-negative edges points forward in time, where time is visualized as proceeding upwards. Note that every subpath of a vee-path is itself a vee-path since the negative edges, if any, will precede the non-negative edges, if any.

The following properties will also be useful.

Definition 11 *Given an STN a* vee-zero-path *is a shortest path in the distance graph that consists of a subpath (possibly empty) of non-positive edges followed by a subpath (possibly empty) of non-negative edges.*

Definition 12 *Given timepoints x and y, a* hook-path *is either a non-negative shortest path that* ends *with a non-negative edge, or a negative shortest path that* starts *with a negative edge.*

It is not hard to see that every vee-path must also be a hook-path.

We are now ready for the main theorem containing several characterizations of dispatchability.

Theorem 2 *The following conditions are equivalent for a consistent STN:*

(i) The STN is dispatchable.

(ii) Every strongly dispatching execution-sequence is a prefix of a complete execution sequence.

(iii) For every pair $<x, y>$ of timepoints, if d(x,y) is finite, there is a hook-path from x to y.

(iv) For every pair $<x, y>$ of timepoints, if d(x,y) is finite, there is a vee-path from x to y.

Proof: (i) => (ii)

As we have noted, any strongly dispatching execution sequence is also weakly dispatching. Thus, it can be extended to a complete execution sequence if the STN is dispatchable.

(ii) => (iii)

We will show if (iii) does not hold then (ii) cannot hold.

[10] i.e., $z \neq x$ and $z \neq y$

[11] Notice that the $d(x, y)$ function can be used to measure distance along any shortest path.

[12] Note that $d'(x, y) - d'(x, z) \leq d'(z, y)$ by the triangle inequality, so $S(y) - S(z) \leq d'(z, y)$ for each z in H.

[13] The modified edges, which are negative, could potentially affect lower(E,w) but that does not matter for the conclusion.

Suppose (iii) does not hold. Thus, $d(x, y)$ is finite, but there is no hook path from x to y.

Consider first the case where $d(x, y)$ is non-negative. Then each shortest path from x to y ends with a negative edge. Thus, the points z immediately preceding y on each shortest path from x constitute a set of hinge points H. Moreover, each shortest path from x to y passes through at least one point in H. Thus, we can apply Lemma 2 to conclude (ii) does not hold.

Otherwise, $d(x, y)$ is negative. Then each shortest path from x to y starts with a non-negative edge. In this case, the points z immediately following x constitute a set of hinge points such that each each shortest path from x to y passes through at least one of them. Again we can apply Lemma 2 to conclude (ii) does not hold. The result follows.

(iii) => (iv)

We prove the result in two stages. First we show there is a vee-zero-path from x to y. Then we show there is a vee-path.

Consider a modified STN where all explicit all-zero cycles are contracted to a single timepoint. Thus, the modified STN is free of all-zero cycles. It is not hard to see that property (iii) is preserved in the modified network.[14]

Consider any two timepoints x_0 and y_0. We will define two sequences $\{x_i\}$ and $\{y_i\}$, starting with x_0 and y_0 respectively. By (iii), if the distance between x_0 and y_0 is negative, then there is a shortest path between them that starts with a negative edge; if non-negative, there is a shortest path that ends with a non-negative edge. In the first case define x_1 to be the point on the path after x_0 and leave the $\{y_i\}$ sequence unchanged. In the second case, define y_1 to be the point on the path before y_0 and leave the $\{x_i\}$ sequence unchanged.

Let x' and y' be the last elements in the $\{x_i\}$ and $\{y_i\}$ sequences, respectively. If $x' \neq y'$, we repeat the application of (iii) to the x', y' pair, to further extend either the $\{x_i\}$ or $\{y_i\}$ sequences, and continue in this way.

Note that there can be no repetition in the $x_0, x_1, x_2, ..$ sequence, since that would imply a negative cycle. Also, there cannot be a repetition in the $y_0, y_1, y_2, ...$ sequence, since that would imply an all-zero cycle.[15]

Since there are no repetitions (and the STN has a finite number of timepoints), the applications of (iii) must eventually terminate when some $x' = y'$. This implies a vee-path between x and y. In the original STN (where all-zero cycles have not been contracted) this will correspond to a vee-zero-path between any pair of timepoints. (Since one or more of the timepoints in the negative segment may have arisen from a contraction.)

Now let x and y be any two timepoints in the *original* STN. Among the shortest paths from x to y, there must be one that has the largest number of initial consecutive negative edges. Suppose x' is the end timepoint of that initial sequence of negative edges. If $x' = y$, we are done. Otherwise, there must be a vee-zero-path VZ from x' to y. If VZ contains a negative edge, it must be somewhere in the negative-or-zero segment. Thus, there must be an intermediate point y' between x' and y such that the path from x' to y' is negative. But then (iii) implies there is a shortest path from x' to y' that starts with a negative edge, which gives us a path from x to y with a greater number of initial negative edges, which is a contradiction. We conclude that VZ contains only non-negative edges, and the initial negative path to x' followed by VZ constitutes a vee-path from x to y.

(iv) => (i)

Suppose there is a vee-path between every pair of timepoints.[16] Our strategy will be to show each dispatch sequence E is (1) super dispatching and (2) path-consistent with respect to the STN. The result will then follow from Theorem 1.

Suppose otherwise for some E =$<\{a_1, ..., a_n\}, S>$. If condition (1) is violated then there is some x in E and y not in E such that $S(x) + d(x, y) < S(a_n)$. If condition (2) is violated then there is some x and y in E such that $S(x) + d(x, y) < S(y)$. Define $\widehat{S}(y) = S(y)$ if y is in E, and $\widehat{S}(y) = S(a_n)$ otherwise. Then both types of violation are captured by the expression $S(x) + d(x, y) < \widehat{S}(y)$. (For future reference, note that $\widehat{S}(y) \leq S(a_n)$.)

Without loss of generality, we can choose a violation node y that minimizes $d(x, y)$, i.e., such that

$$d(x, y) = \min\{d(x, q) : S(x) + d(x, q) < \widehat{S}(q)\}.$$

By hypothesis, there is a vee-path from x to y. We reiterate that since a vee-path is a shortest path, the d function can be used to measure local distance along the vee-path.

Suppose z is the last node on the vee-path such that the nodes on the subpath from x to z are all in E. By local consistency of E, $S(x) + d(x, z) = S(z)$, so $z \neq y$. Let w be the node immediately following z in the vee-path (thus not in E). Since all the nodes in the negative portion of the vee-path constitute a chain of direct predecessors of x and so are in E, z must lie in or begin the non-negative portion. We have

$$\begin{aligned} S(z) &= S(x) + d(x, z) \\ &\leq S(x) + d(x, w) \\ &\leq S(x) + d(x, y) \\ &< \widehat{S}(y) \\ &\leq S(a_n). \end{aligned}$$

Thus, $z \neq a_n$ and z is in E_{n-1}. Since the vee-path has a direct edge from z to w whose weight equals $d(z, w)$, we get

$$\text{upper}(E_{n-1}, w) \leq S(z) + d(z, w) < S(a_n).$$

It follows that w cannot be in enabled(E_{n-1}); otherwise a_n would fail the MIN-UPPER condition for a dispatch sequence.

[14] For a shortest path in the contracted STN, consider a corresponding shortest path in the original STN that does not start or end with a contracted edge..

[15] If a shortest path of non-negative edges contains a cycle, the edges in the cycle must all have zero length; otherwise, the path could be further shortened by omitting the cycle.

[16] Within the framework of MMT, dispatchability would follow because the vee-paths would dominate other edges in the AllPairs network. The proof here does not depend on dominance.

Since w is not in enabled(E_{n-1}), it must have a direct predecessor v that is not in E_{n-1}. Thus, either $v = a_n$ or v is not in E. In either case, $\widehat{S}(v) = S(a_n)$. We also have $d(x, v) < d(x, w)$ since there is a negative edge from w to v. It follows that

$$\begin{aligned} S(x) + d(x, v) &< S(x) + d(x, w) \\ &< S(a_n) \\ &= \widehat{S}(v). \end{aligned}$$

Since $d(x, v) < d(x, w) \leq d(x, y)$, this contradicts the minimality of y.

This, we have established that each dispatch sequence E is both path-consistent and super dispatching. It then follows from Theorem 1 that the network is dispatchable. □

Temporal Uncertainty

The results of this paper apply directly to STNs. In this section we consider some implications for STNUs and temporal uncertainty. The discussion assumes a basic knowledge of Dynamic Controllability as contained for example in the Hunsberger tutorial (Hunsberger 2013).

An STNU may be regarded as a compact representation of all its projections, which are STNs.. From this point of view, a reduction rule that adds an edge may be regarded as simultaneously adding the edge in each of the projections. (Note that a *wait* or conditional edge reduces to an ordinary edge in each projection (Morris 2014).) Reduction rules in recent STNU work (Shah et al. 2007; Nilsson, Kvarnström, and Doherty 2013; Morris 2014; Nilsson, Kvarnström, and Doherty 2015) involve "plus-minus" rules where a non-negative AB edge followed by a BC negative edge may result in an added AC edge. It is not difficult to see that applying "plus-minus" transformation rules repeatedly to a shortest path will eventually result in a vee-path where all the remaining negative edges precede all the remaining non-negative edges.

The sign of an edge in the labelled distance graph (Morris 2006) of an STNU is the same as its sign in each of the projections. Thus, the reduction rules result in vee-paths between connected nodes in every projection. It is not hard to see that if any projection is inconsistent, the reduction rules will produce a negative cycle. Thus, if there no negative cycles, every projection is dispatchable by Theorem 2. It then follows that the STNU is Dynamically Controllable (Morris 2014). Thus, the reduction rules, which were originally formulated empirically, may be understood as determining Dynamic Controllability indirectly by establishing dispatchability of each projection.

We see that the "plus-minus" reduction rules establish dispatchability, although they may add more edges than needed for that purpose. More specifically, the resulting projections are generally not *minimum* dispatchable in the sense of MMT. Theorem 2 part (iii) shows that an edge from x to y in a dispatchable network is unneeded for dispatchability provided there remains an alternative hook path from x to y after the edge is deleted. This provides a counterpart to MMT dominance that could be used to prune unneeded edges. Notice that it allows pruning edges from any dispatchable network, not just the All-Pairs one.

These results may also potentially be helpful in generalizing notions of Dynamic Controllability to multi-agent interactions. For a problem with two agents, where one may observe the outcomes of the other, the consistent schedules of the observed agent are analogous to the projections of Dynamic Controllability. We conjecture that ensuring dispatchability of these generalized projections may constitute a dynamic strategy for the observing agent. Related temporal decoupling work (Hunsberger 2002; Wilson et al. 2014) is static and does not take advantage of observation.

For example, consider an STNU

$$\mathrm{C} \overset{[18,29]}{\Longleftarrow} \mathrm{A} \overset{[20,31]}{\Longrightarrow} \mathrm{B} \overset{[1,3]}{\longleftarrow} \mathrm{D}$$

where the AB and AC links are contingent. This is not Dynamically Controllable (DC). (Applying the reductions to the ABD subgraph would produce a negative cycle.)

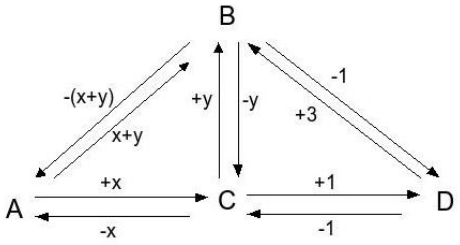
In an STNU, the contingent links are assumed to be independent. However, an observed agent may have a richer temporal plan that adds a cross-constraint

$$\mathrm{C} \overset{[2,4]}{\Longrightarrow} \mathrm{B}$$

to the above example. In the following discussion, we will extend the usage of the term *projection* to mean consistent schedule of the observed agent.

It is easy to see that the DB requirement can now be satisfied by executing D at 1 time unit after C is observed, so this new problem is intuitively DC. Note that the ABD reductions would produce the same negative cycle as before. This shows unmodified STNU methods break down when applied to this new type of problem.

The conjecture provides a possible way forward. If true, then we should try to apply only enough plus-minus reductions to make every projection dispatchable. By Theorem 2 this means establishing vee-paths for every projection without making any projection inconsistent. In the present example, if we only apply reductions to the CBD subgraph, we can derive a $\mathrm{C} \overset{[1,1]}{\Longrightarrow} \mathrm{D}$ link and add it to the network.



As may be verified in the above distance graph for an arbitrary projection,[17] this results in a vee-path between every pair of timepoints. To see this, observe that the BD and DB edges are dominated by the BCD and DCB paths, respectively, in every projection, since $y \in [2, 4]$. Similarly,

[17] In a projection, the contingent links take on definite values; thus the resulting links are rigid. Note that $x + y \in [20, 31]$ so x and y are not fully independent.

the BA and AB edges are dominated by the BCA and ACB paths, respectively. If these dominated edges are omitted, every other cycle-free shortest path is a vee-path.

It may also be verified that no projection is inconsistent in the above diagram. Since the ABC subgraph is rigid and consistent, the only possible negative cycle would be in the BCD subgraph. There the clockwise cycle has length $y - 2$ while the counter-clockwise cycle has length $4 - y$. Since $y \in [2, 4]$, the cycles are non-negative in both cases.

Thus, the conjecture would say this agent problem is DC, which seems correct for this example. Moreover, the derived edges correspond to the intuitive dynamic strategy.

Closing Remarks

In this paper we introduced the formalism of execution sequences, and used it to prove a characterization of dispatchability in terms of vee-paths. The sigificance of this is that it provides an easy way of telling what needs to be done to make an STN dispatchable; we simply need to derive enough edges to create vee-paths.

We have also discussed potential applications of this result to temporal uncertainty. Previous methods of determining Dynamic Controllability can be interpreted as adding ordinary or wait edges to an STNU to make every projection dispatchable. We conjecture that essentially the same approach could be applied to richer temporal plans by generalizing the notion of a projection.

As a final remark, dispatchable networks have interesting properties from the point of view of constraint satisfaction. They have an extensibility property for partial schedules resembling that of the d-graph (Dechter, Meiri, and Pearl 1991) but with the added twist of an asymmetry with respect to the time parameter, i.e., an inherent "arrow of time."

References

Cormen, T.; Leiserson, C.; and Rivest, R. 1990. *Introduction to Algorithms*. Cambridge, MA: MIT press.

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

Hille, E. 1965. *Analytic Function Theory*. New York, Toronto, London: Blaisdell Publishing Company. Fourth Printing.

Hunsberger, L. 2002. Algorithms for a temporal decoupling problem in multi-agent planning. In *Proc. of Eighteenth Nat. Conf. on Artificial Intelligence (AAAI-02)*.

Hunsberger, L. 2009. Fixing the semantics for dynamic controllability and providing a more practical characterization of dynamic execution strategies. In *Proc. of the IEEE 16th International Symposium on Temporal Representation and Reasoning (TIME-2009)*, 155–162.

Hunsberger, L. 2013. Tutorial on dynamic controllability. http://icaps13.icaps-conference.org/wp-content/uploads/2013/06/hunsberger.pdf.

Moffitt, M. D., and Pollack, M. E. 2007. Generalizing temporal controllability. In *Proc. of IJCAI-07*.

Moffitt, M. D. 2007. On the partial observability of temporal uncertainty. In *Proc. of AAAI-07*.

Morris, P.; Schwabacher, M.; Dalal, M.; and Fry, C. 2013. Embedding temporal constraints for coordinated execution in habitat automation. In *International Workshop on Planning and Scheduling for Space*.

Morris, P.; Muscettola, N.; and Vidal, T. 2001. Dynamic control of plans with temporal uncertainty. In *Proc. of IJCAI-01*.

Morris, P. 2006. A structural characterization of temporal dynamic controllability. In *CP*, 375–389.

Morris, P. 2014. Dynamic controllability and dispatchability relationships. In *CPAIOR*.

Muscettola, N.; Nayak, P.; Pell, B.; and Williams, B. 1998. Remote agent: to boldly go where no AI system has gone before. *Artificial Intelligence* 103(1-2):5–48.

Muscettola, N.; Morris, P.; and Tsamardinos, I. 1998. Reformulating temporal plans for efficient execution. In *Proc. of Sixth Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*.

Nilsson, M.; Kvarnström, J.; and Doherty, P. 2013. Incremental Dynamic Controllability Revisited. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI Press.

Nilsson, M.; Kvarnström, J.; and Doherty, P. 2015. Efficient Processing of Simple Temporal Networks with Uncertainty: Algorithms for Dynamic Controllability Verification. *Acta Informatica*.

Rossi, F.; Venable, K. B.; and Yorke-Smith, N. 2006. Uncertainty in soft temporal constraint problems: A general framework and controllability algorithms for the fuzzy case. *Journal of Artificial Intelligence Research* 27:617–674.

Shah, J. A., and Williams, B. C. 2008. Fast dynamic scheduling of disjunctive temporal constraint networks through incremental compilation. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*, 322–329.

Shah, J. A.; Stedl, J.; Williams, B. C.; and Robertson, P. 2007. A fast incremental algorithm for maintaining dispatchability of partially controllable plans. In Boddy, M. S.; Fox, M.; and Thibaux, S., eds., *ICAPS*, 296–303. AAAI.

Tsamardinos, I., and Pollack, M. E. 2003. Efficient solution techniques for disjunctive temporal reasoning problems. *Artificial Intelligence* 151:43–89.

Tsamardinos, I.; Muscettola, N.; and Morris, P. 1998. Fast transformation of temporal plans for efficient execution. In *Proc. of Fifteenth Nat. Conf. on Artificial Intelligence (AAAI-98)*.

Tsamardinos, I.; Pollack, M. E.; and Ganchev, P. 2001. Flexible dispatch of disjunctive plans. In *Proceedings of the 6th European Conference on Planning*, 417–422.

Vidal, T., and Fargier, H. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *JETAI* 11:23–45.

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