The Mathematics of Dispatchability, Revisited

Paul Morris

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, dispatchability has considerable theoretical as well as practical significance.

One thing that hampers further work in this area is the underdeveloped theory. Moreover, the existing foundation is inadequate in certain respects. In this paper, we develop a new mathematical theory of dispatchability and its relationship to execution. We also provide several characterizations of dispatchability, including characterizations in terms of the structural properties of the STN graph. This facilitates the potential application of the theory to other areas.

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). Recently, it has been shown (Morris 2014) that Simple Temporal Networks with Uncertainty, augmented with wait edges, are Dynamically Controllable provided every projection is dispatchable. Thus, dispatchability has theoretical as well as practical significance.

However, there are some problems with the foundations of dispatchability theory, as we will see. Also, a greater understanding of the nature of dispatchability would be helpful in proving further connections of dispatchability to other concepts. In this paper, we develop a new formal theory of dispatchability, and provide characterizations of dispatchability in terms of the structural properties of the STN graph.

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:
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 already 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 always "correctly executed" by a dispatching execution. (Note 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.)

This captures the intuitive concept but is unsatisfactory as a formal definition for several reasons:

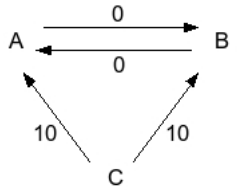
1. If this is to be independent of practical (computer) limitations, it must assume an idealized situation where the time required to execute steps 2-5 is negligible compared to the granularity of the time steps.

   For example, if a TP x is constrained ([0,0] constraint) to occur at the SAME time as a TP y, then after executing one of those, say x, the algorithm must race through steps 2-5 in order to get to y within the same time tick.

   However, we would like the formal definition to represent time in real numbers, in which case steps 2-5 would have to take 0 time.

2. The terminology "correctly executed" is not precisely defined in the paper.

3. The paper introduces a process where an edge CB can be removed as not needed for dispatchability, provided it satifies a *triangle rule* CA+AB=CB with respect to another edge AB, said to *dominate* it. However, the analysis in MMT 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.[1] The MMT analysis relies heavily on dominance arguments.

Ideally, we would like to define "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

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*[2] upper and lower bounds from the timepoint, and repeating for some subset of the timepoints.

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

[2]i.e., only to neighboring timepoints

In the following discussion, we assume the STNs are consistent unless there is an explicit statement otherwise. By *local consistency* of a partial schedule S, we mean S satisfies the direct constraints between every pair of timepoints in its domain.

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

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[3] 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.

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 negative edge 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) $S_{i+1} \geq S_i$, for each i, and
(2) each direct predecessor of a timepoint in E is also in E.

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.

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.

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)*.

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

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).

We also use the following notations:

```
MIN_LOWER(E)
  = min{lower(E,x) | x in ready(E)}

MIN_UPPER(E)
  = min{upper(E,x) | x in ready(E)}

latest(E) = S(a_n).
```

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

We are almost ready to define the concept of a dispatching execution in a way that captures more formally the result of the TIME DISPATCHING ALGORITHM. We proceed by considering and then simplifying various candidate definitions. Our first candidate is a literal transcription of the criterion in the TIME DISPATCHING ALGORITHM for selecting a timepoint for execution.

Candidate Definition 1: A *dispatch-sequence* is an execution-sequence E = $<\{a_i\}, S>$ such that for $i > 1$,

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

Note, however, the left-hand inequality is satisfied by every execution-sequence because of the local consistency requirement, so it is unnecessary. Furthermore, because of monotonicity, if x is in ready(E_{i-1}) and upper(E_{i-1}, x) $< S(a_i)$, this will continue to hold for larger i, since S(a_i) cannot decrease, upper bounds cannot increase, and thus x cannot get into E. In effect, x will be "left behind" as the latest time passes it by, and $S(a_i) \leq$ MIN_UPPER(E_{i-1}) will continue to be violated. Thus, we can simplify the definition further by requiring it only for the greatest value of i. These simplifications lead to the following:

Candidate Definition 2: A *dispatch-sequence* is an execution-sequence E = $<\{a_1, ..., a_n\}, S>$ such that upper(E_{n-1}, x) $\geq S(a_n)$ for all x in ready(E_{n-1})

We can express this more succinctly in terms of a "past completeness" property. As we will see later, it is useful to define several versions of this property of varying strengths.

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

Intuitively, the weak past-completeness[4] condition requires that each timepoint chosen for execution must be executed at a time no greater than the minimal upper bound among the timepoints currently eligible for execution.

[4]The definition is equivalent to a requirement that x in enabled(E_{n-1}) and upper(E_{n-1}, x) $<$ latest(E) implies x is in E_{n-1}, which explains the "completeness" terminology.

This allows a more succinct version of Candidate Definition 2 as our final definition of a dispatching execution.

Definition 5 *A* dispatch-sequence *is an execution-sequence that is weakly past-complete.*

The next (strong) version of past-completeness is useful as an intermediate property in our later results.

Definition 6 *An execution-sequence E = $<\{a_i\}, S>$ is* strongly past-complete *if upper(E,x) $\geq$ latest(E) for x not in E.*

Intuitively, strong past-completeness ensures that each timepoint chosen for execution is executed at a time no greater than the minimal upper bound among *all* non-executed timepoints. Thus, no timepoint will be "left-behind" as the execution advances. The following result formally justifies the strong/weak terminology.

Lemma 1 *Strongly past-complete implies weakly past-complete.*

Proof: Suppose an execution-sequence E is strongly past-complete 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 past-completeness, 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 past-complete. □

Finally, with some additional notation, we have a super version that will be useful in a later proof.

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

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

Note that if x is a node in E that determines upper(E,y), then there is an edge of 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 past-complete, hence weakly past-complete, and thus are dispatch-sequences.

We are now 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 additional timepoints occur later.

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

By Lemma 1 every strongly past-complete execution-sequence is weakly past-complete. The converse is not true in general, but we will see later that for the purpose of defining dispatchability of an STN, there is a sense in which either the strong or weak past-completeness property could have been used. The strong property has the virtue of being simpler while still involving only local propagation. However, we have expressed the definition in terms of the weak property to maintain continuity with the earlier MMT work.

The usefulness of the super past-completeness property lies in the following result.

Theorem 1 *An execution sequence E = <A,S> is a prefix of a complete execution-sequence if and only if (1) E is super past-complete and (2) S is path-consistent with respect to the STN, i.e., $S(y) - S(x) \leq d(x, y)$ for each x and y in E.*

Proof: First suppose E = $<a_1, ..., a_n$,S> is a prefix of a complete execution-sequence. Then S can be extended to a solution S' so (2) clearly holds.

Suppose x is in E but y is not in E. Since S' is a solution, $S'(y) - S'(x) \leq d(x, y)$. Thus, $S'(x) + d(x, y) \geq S'(y)$. Note that $S'(y) \geq S'(a_n)$ since y comes after a_n in the complete execution sequence. Since S' coincides with S on E, we have $S(x) + d(x, y) \geq S(a_n)$ 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. Suppose not. Then there is a cycle-free shortest path

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

such that $S(y) - S(x) > d'(x, y)$, for some x and y in E and z not in E, where d' is the shortest-path distance in the modified STN. Note that the path must pass through one (and only one since the path is cycle-free) of the new edges. Thus, $d'(z, a_n) = 0$, while $d'(x, z) = d(x, z)$ and $d'(a_n, y) = d(a_n, y)$.

By (1) we have $S(x) + d(x, z) \geq S(a_n)$. This can be rewritten as

$$S(a_n) - S(x) \leq d(x, z) = d'(x, z) = d'(x, a_n)$$

since $d'(z, a_n) = 0$ (and the path is a shortest path).

By (2) we have

$$S(y) - S(a_n) \leq d(a_n, y) = d'(a_n, y).$$

Adding the inequalities, we get

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

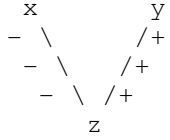
which contradicts our assumption. We conclude that S is path-consistent with respect to the modified STN. This is the same thing as saying that S is locally consistent with respect to the minimal (AllPairs) network. By (Dechter, Meiri, and Pearl 1991), S can be extended to a solution schedule S' for 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. These determine whether particular path constraints are enforced or not in a dispatching execution.

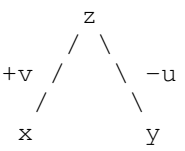
DISCUSSION Intuitively, a path constraint between x and y of length $u - v$ can be enforced by propagating a lower bound of u to x and an upper bound of v to y from some common precursor z.

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.

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

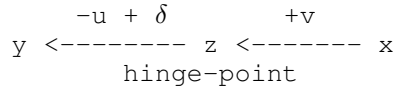


`z is distinct from x and y`

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 exactly one of the points in H, and that point is a hinge point for the path. Then there is a strongly past-complete execution-sequence whose partial schedule cannot be extended to a solution.*

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 length increased by some amount $\delta > 0$. Thus, the shortest paths to y following z have their total distance increased by δ, as illustrated.



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) every original shortest path from x to y is still a shortest path, and (ii) the $-\mathrm{u}+\delta$ values are still negative.

Define a partial schedule S by setting $S(x)$ to an arbitrary value and $S(q) = S(x) + d'(x,q)$ for each q on a shortest path from x to y, where $d'(x,q)$ is the shortest path distance from x to q in the modified STN. By construction, S is locally consistent with respect to the minimal (AllPairs) network for the modified STN; thus, it can be extended to a solution S' (Dechter, Meiri, and Pearl 1991). However, S is inconsistent with respect to the *original* STN because $S(x) - S(y) = -(d(x,y) + \delta)$ where $d(x,y)$ is the shortest path distance from x to y in the original STN. 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 past-complete 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.[5] Consequently, E is also locally consistent and strongly past-complete with respect to the original STN. However, the partial schedule of E coincides with S on x and y, and we have shown these values are inconsistent with respect to the original STN. Thus, the partial schedule of E cannot be extended to a solution. □

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.*

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.

The following propertties 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 past-complete execution-sequence can be extended to a solution.

(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 past-complete execution sequence is also weakly past-complete. 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.

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 exactly 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 exactly 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.

Consider any two timepoints $x = x_0$ and $y = y_0$. 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 define $y_1 = y_0$. In the second case define $x_1 = x_0$ and define y_1 to be the point on the path before y_0.

If x_1 is not equal to y_1, we can repeat the application of (iii) to the x_1, y_1 pair to define x_2 and y_2, and so on. 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. (If a shortest path of non-negative edges contains a cycle, the edges in the cycle must all have

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

zero length; otherwise, the path could be further shortened by omitting the cycle.)

Since there be no repetitions (and the STN has a finite number of timepoints), the applications of (iii) must eventually terminate when some $x_N = y_N$. 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. Within the framework of MMT, dispatchability would be then assured because the vee-paths would dominate all the edges in the AllPairs Shortest-Path network. However, within our formal framework, a somewhat different proof approach is needed.

Our strategy will be to show each dispatch sequence E is (1) super past-complete 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

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

By hypothesis, there is a vee-path from x to y. 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. 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. By local consistency of E, $S(x) + d(x, z) = S(z)$, so $z \neq y$. Also, $S(z) \leq S(x) + d(x, y) < \widehat{S}(y) \leq S(a_n)$, so $z \neq a_n$ and z is in E_{n-1}. Now consider the node w immediately following z in the vee-path. Combining the previous inequalities with $d(x, w) \leq d(x, y)$, and the vee-path direct edge from z to w, we get

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

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

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

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

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 past-complete. It then follows from Theorem 1 that the network is dispatchable. □

Closing Remarks

Reduction rules in previous work (Shah et al. 2007; Nilsson, Kvarnström, and Doherty 2013; Morris 2014) establish vee-paths between nodes, ensuring dispatchability, but may add more edges than needed. Note that 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. This provides an analogue of MMT dominance that can be used to prune unneeded edges, while avoiding the issues mentioned earlier.

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

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 minimal network (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

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

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

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. 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.

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.; 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)*.