Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

A Distributed Approach to Summarizing Spaces of Multiagent Schedules

James C. Boerkoel Jr. and Edmund H. Durfee
Computer Science and Engineering, University of Michigan
Ann Arbor, MI 48109
{boerkoel,durfee}@umich.edu

Abstract

We introduce the Multiagent Disjunctive Temporal Problem (MaDTP), a new distributed formulation of the widely-adopted Disjunctive Temporal Problem (DTP) representation. An agent that generates a summary of *all* viable schedules, rather than a single schedule, can be more useful in dynamic environments. We show how a (Ma)DTP with the properties of *minimality* and *decomposability* provides a particularly efficacious solution space summary. However, in the *multiagent* case, these properties sacrifice an agent's strategic interests while incurring significant computational overhead. We introduce a new property called *local decomposability* that exploits loose-coupling between agents' problems, protects strategic interests, and supports typical queries. We provide and evaluate a new *distributed* algorithm that summarizes agents' solution spaces in significantly less time and space by using local, rather than full, decomposability.

1 Introduction

Computational scheduling agents can assist people in managing and coordinating their activities in environments in which tempo, a limited (local) view of the overall problem, and complexity can outstrip people's cognitive capacity. As an example, imagine the scheduling operations at three manufacturing plants whose scheduling considerations interact due to a truck that must make deliveries to each location by a predetermined deadline. Not only might each manufacturing plan have complex internal scheduling considerations, but the truck must also determine the order in which to visit the three locations, where each order may have different implications on travel and processing time due to things like traffic congestion and the overhead involved in reshuffling inventory inside the truck.

Scheduling agents that dispatch advice based on a single schedule, however, may be brittle to the dynamics involved in the problem (due to durational uncertainty, exogenous events, additional planning by other agents, etc.). A more robust approach for dealing with dynamism in scheduling applications is to instead consider the set of *all* feasible schedules. As we will show, a representation with the properties of *minimality* and *decomposability* helps an agent maintain the solution space so as to efficiently answer queries. An agent using a minimal representation can efficiently respond to queries of the form "When can I start activity A?" or "How much time do I have to complete activity B before I need to start activity C?" with the exact set of feasible values. An agent using a decomposable representation can efficiently respond to queries involving *sets* of activities and also propagate newly arriving dynamic constraints (e.g., that specify the actual start time or duration of an activity) so that minimality, and thus the integrity of the advice it dispatches, is retained.

Often, the schedules of multiple agents interact. For example, the scheduling agent of the delivery truck may need to coordinate with agents responsible for scheduling operations at each location. Representing the set of all feasible *joint* schedules becomes much more complex. Not only does the number of joint schedules grow exponentially with each additional agent, but generating joint schedules for every eventuality that could arise may also compromise the strategic interests (privacy, autonomy, etc.) of individual scheduling agents and introduce significant computational overhead.

In this paper, we define the Multiagent Disjunctive Temporal Problem (MaDTP), which is a multiagent, distributed generalization of the Disjunctive Temporal Problem (DTP) (Stergiou and Koubarakis 2000), and is capable of capturing general types of multiagent scheduling problems. We extend the properties of minimality and decomposability to the more general (Ma)DTP and introduce *local decomposability*, an approximation of decomposability that exploits the idea that, for many loosely-coupled problems, only an exponentially small portion of an agent's local solution space will affect the global problem. We empirically show that our distributed algorithm for computing locally decomposable solution spaces yields significant speedup over centralized algorithms that compute the globally decomposable solution space.

2 Background

In this section, we present a family of constraint-based scheduling problem formulations from which our definition of the Multiagent Disjunctive Temporal Problem inherits. Dechter, Meiri, and Pearl (1991) defined the ***Simple Temporal Problem (STP)***, $\mathcal{S} = \langle V, C_{\mathcal{S}} \rangle$, as a set of timepoint variables, V, and a set of temporal difference constraints, $C_{\mathcal{S}}$. Each temporal difference constraint $c_{ij} \in C_{\mathcal{S}}$ is of the form

$v_j - v_i \in [-b_{ji}, b_{ij}]$, where v_i and v_j are distinct timepoints and $b_{ji}, b_{ij} \in \mathbb{R}$ form real (possibly infinite) lower and upper bounds on the difference between v_j and v_i. Each timepoint variable, v_i, represents an event, and has a continuous numeric domain of times (e.g., clock times) formed implicitly by a constraint between v_i and z, a special zero timepoint denoting the start of time. An STP instance is *consistent* if it contains at least one *solution*, which is an assignment of specific time values to all timepoint variables that respects all constraints to form a *schedule*. To exploit extant graphical algorithms and efficiently reason over the *simple temporal network* (STN), each STP is associated with a *distance graph*, where each variable, $v_i \in V$, is represented by a vertex and each constraint, $c_{ij} \in C_{\mathcal{S}}$, is represented by a directed edge from v_i to v_j weighted by its associated constraint bounds. The ***Multiagent Simple Temporal Problem (MaSTP)*** establishes how an STP representation can be distributed among n agents using n local STP subproblems, one per agent, and a set of external constraints that establish relationships between subproblems of different agents (Boerkoel and Durfee 2010).

The ***Disjunctive Temporal Problem (DTP)*** (Stergiou and Koubarakis 2000), $\mathcal{D} = \langle V, C_{\mathcal{D}} \rangle$, specifies a more general set of disjunctive constraints, $C_{\mathcal{D}}$, where $c_y \in C_{\mathcal{D}}$ takes the form $d_1 \vee d_2 \vee \cdots \vee d_k$, and each $d_z = v_{j_z} - v_{i_z} \in [-b_{j i_z}, b_{i j_z}]$. These constraints represent a *disjunctive* choice among k possible temporal difference constraints, each with its own bounds expressed over (possibly *different*) pairs of timepoints. A *labeling*, ℓ, of a DTP, is the *component STP* formed by selecting a disjunct (temporal difference) for each disjunctive constraint. A schedule s, then, is a solution to a consistent DTP instance iff it is the solution to at least one of its component STPs. For general DTPs with $|C_{\mathcal{D}}|$ constraints of arity k, there are $\mathcal{O}(k^{|C_{\mathcal{D}}|})$ possible labelings, each of which must be explored in the worst case, which, as the number of disjunctive constraints grows, makes the DTP an NP-hard problem. Each component STP, however, can be evaluated in polynomial time, putting the DTP in the class of NP-complete problems. The ***Temporal Constraint Satisfaction Problem (TCSP)*** (Dechter, Meiri, and Pearl 1991), $\mathcal{T} = \langle V, C_{\mathcal{T}} \rangle$, is a well-studied special case of a DTP where all disjuncts of a given constraint are expressed over the *same* pair of variables. The STP is also a special case of the DTP (and TCSP) where $k = 1$. Thus, $C_{\mathcal{S}} \subseteq C_{\mathcal{T}} \subseteq C_{\mathcal{D}}$ and so, STP $\subseteq$ TCSP $\subseteq$ DTP.

All of these constraint-based scheduling representations share the principle that consistent problem instances implicitly represent a *space* of solutions. There are two properties of problems' corresponding temporal constraint networks that are particularly useful for representing spaces of solutions explicitly. A ***minimal constraint*** c_{ij} is one whose interval(s) *exactly* specify the set of *all* feasible values for the difference $v_j - v_i$. A temporal network is ***minimal*** iff all of its constraints are minimal and establishes the exact space of values for each timepoint and constraint that can lead to solutions. An important complement to minimality, decomposability facilitates the maintenance of minimality by capturing constraints that, if satisfied, will lead to global solutions. A temporal network is ***decomposable*** if any assignment of values to a subset of timepoint variables that is locally consistent (satisfies all constraints involving only those variables) can be extended to a solution (Dechter, Meiri, and Pearl 1991).

The STN represents a special case where both minimality and decomposability can be established efficiently (in $\mathcal{O}(|V|^3)$) by applying an all-pairs-shortest-path algorithm, such as Floyd-Warshall (1962), to the distance graph. This finds the tightest possible path between every pair of timepoints, forming a fully-connected graph that explicitly represents the STP's solution space. *Partial path consistency* approximates decomposability by calculating minimality for a subset of constraints that form a chordal (multiagent) temporal network; this sparser representation is more efficient to maintain, but contains less information and limits decomposability to subsets of variables belonging to the same clique (Xu and Choueiry 2003; Planken, de Weerdt, and van der Krogt 2008; Boerkoel and Durfee 2010). Generally, establishing minimality and decomposability for the TCSP, and thus DTP, is NP-hard (Dechter, Meiri, and Pearl 1991).

3 Multiagent Disjunctive Temporal Problem

While we could solve scheduling problems like the one introduced in Section 1 as a single DTP, each business may have computational and other strategic reasons for maintaining and reasoning over its information independently such that each location has its own scheduling agent to ensure scheduled delivery times align with internal operations. Next, we define a variation of the DTP that captures the distributed, multiagent nature of our example problem.

3.1 Problem Formulation

Our definition of the Multiagent Disjunctive Temporal Problem (MaDTP) parallels the definition of the MaSTP (Boerkoel and Durfee 2010). The MaDTP is informally composed of n local DTP subproblems, one for each of n agents, and a set of external constraints, C_X, which are disjunctive temporal constraints that relate the local subproblems of different agents. An agent i's ***local*** DTP subproblem is defined as $\mathcal{D}_L^i = \left\langle V_L^i, C_L^i \right\rangle$, where V_L^i is agent i's set of ***local variables*** and partitions all timepoints into the subset assignable by agent i (and may include agent i's reference to z), and C_L^i is agent i's set of ***local constraints***, where each $c_y \in C_L^i$ is specified exclusively over local variables.

Agent i is also aware of its ***external constraints*** C_X^i, where each disjunctive temporal constraint $c \in C_X^i$ is specified over at least one variable $v_i \in V_L^i$ and one variable $v_j \in V_L^j$, $i \neq j$, and of its ***external variables*** V_X^i, where each $v_j \in V_X^i$ appears in at least one of agent i's external constraints, but is local to some other agent $j \neq i$. Agent i's set of ***known variables*** is $V^i = \left\{ V_L^i \cup V_X^i \right\}$ and agent i's set of ***known constraints*** is $C^i = \left\{ C_L^i \cup C_X^i \right\}$.

More formally, then, an MaDTP, $\mathbf{D}$, is defined as the set of agent DTP subproblems, $\mathbf{D} = \left\{ \bigcup_i \mathcal{D}^i \right\}$, where $\mathcal{D}^i = \left\langle V^i, C^i \right\rangle$. $V = \left\{ \bigcup_i V_L^i \right\}$ is the set of *all* variables and $C = \left\{ \bigcup_i \left\{ C_L^i \cup C_X^i \right\} \right\}$ is the set of *all* constraints. A schedule s is a solution to $\mathbf{D}$ iff it is a solution to the *component MaSTP* corresponding to one of $\mathbf{D}$'s labelings. There are $\mathcal{O}(k^{|C|})$ ($\approx \mathcal{O}(k^{a \cdot |C_L|})$ where a is the number of agents) possible labelings, each of which must be explored in the

	EST	Deadline	Min. Duration	Nonconcurrency	Transition (external)
Loc. A	$z - T^A_{ST} \leq -60$;	$T^A_{ET} - z \leq 300$;	$T^A_{ST} - T^A_{ET} \leq -30$;	$T^A_{ET} - M^A_{ST} \leq 0 \vee$	$T^A_{ET} - T^B_{ST} \leq -60 \vee$
	$z - M^A_{ST} \leq 0$;	$M^A_{ET} - z \leq 480$;	$M^A_{ST} - M^A_{ET} \leq -300$;	$M^A_{ET} - T^A_{ST} \leq 0$;	$T^B_{ET} - T^A_{ST} \leq -90$;
Loc. B	$z - T^B_{ST} \leq -75$;	$T^B_{ET} - z \leq 360$;	$T^B_{ST} - T^B_{ET} \leq -30$;	$T^B_{ET} - M^B_{ST} \leq 0 \vee$	$T^B_{ET} - T^C_{ST} \leq -90 \vee$
	$z - M^B_{ST} \leq 0$;	$M^B_{ET} - z \leq 480$;	$M^B_{ST} - M^B_{ET} \leq -120$;	$M^B_{ET} - T^B_{ST} \leq 0$;	$T^C_{ET} - T^B_{ST} \leq -120$;
Loc. C	$z - T^C_{ST} \leq -90$;	$T^C_{ET} - z \leq 420$;	$T^C_{ST} - T^C_{ET} \leq -30$;	$T^C_{ET} - M^C_{ST} \leq 0 \vee$	$T^A_{ET} - T^C_{ST} \leq -120 \vee$
	$z - M^C_{ST} \leq 0$;	$M^C_{ET} - z \leq 480$;	$M^C_{ST} - M^C_{ET} \leq -240$;	$M^C_{ET} - T^C_{ST} \leq 0$;	$T^C_{ET} - T^A_{ST} \leq -150$

Table 1: Summary of the example logistics problem.

worst case. While each component MaSTP can be evaluated in polynomial-time, the number of possible labelings grows exponentially as the number of disjunctive constraints (or agents with local constraints) grows, putting the MaDTP in the class of NP-complete problems. Note, multi- (and single-) agent versions of the STP and TCSP are special cases of the MaDTP definition, making it a generalization of the approaches discussed in Section 2.

3.2 Example Problem

We illustrate how to populate the MaDTP formulation with a detailed version of the example problem introduced in Section 1. We give the problem specification in Table 1, and its graphical rendering in Figure 1. The problem involves transportation activities (T) involving a single truck that needs to make deliveries to three locations, A, B, and C, each of which also must perform some manufacturing activity M. In addition to the zero timepoint z (where $z = 0$ represents the start time of the journey), there are timepoint variables for the start time (ST) and end time (ET) of each activity. As represented in the EST and Deadline columns, each delivery and manufacturing activity has constraints dictating an earliest start time and particular deadline, which is influenced by the start and end of the work day and also transportation time to and from the truck's depot. The fourth column specifies minimum duration constraints for all activities. We use disjunctive constraints (Nonconcurrency column) to enforce that activities at a location do not overlap. Notice that because each disjunct is specified over a different pair of variables, these non-concurrency constraints cannot be represented in the TCSP framework. Finally, note that the disjunctive constraints over the truck's transition time (last column), which include transportation time, are neither reflexive nor transitive due to the directionality and traffic congestion of available roads and overhead of reshuffling inventory on the truck. Agent A, B, and C's local timepoints and constraints in Table 1 are in rows Loc. A, Loc. B, and Loc. C, respectively. The external constraints are those appearing in the right subtable, while in Figure 1, external timepoints and constraints are denoted with dashed lines, and local constraints are denoted with solid lines. This particular example problem has $2^6 = 64$ possible labelings, but only two (Figure 1 (a) and (b)) satisfy all scheduling constraints.

3.3 Representational Properties

The MaDTP formulation allows the distributed representation of scheduling problems that span multiple agents, which potentially yields strategic (e.g., privacy) and computational (e.g., concurrency) advantages. The extent of these advantages relies, in large part, on the level of independence inherent in the problem, where two timepoints are *independent* if there is no path that connects them in the constraint network corresponding to *any* labeling, and *dependent* otherwise. Notice that all dependencies between agents flow through the set of external variables, $V_X = \cup_i V^i_X$. In fact, we later formalize this idea by defining agent i's set of *interface variables* as the set $V^i_I = \{V^i_L \cap V_X\}$, which encapsulate agent i's influence on other agents. The implication is that, outside its interface variables, each agent i can independently (and thus concurrently, asynchronously, privately, autonomously, etc.) reason over its local subproblem $\mathcal{D}^i_L$.

As discussed in Section 2, a minimal, decomposable representation avoids requiring that agents solve an NP-hard problem to evaluate queries. For example, minimality allows an agent to quickly and exactly answer queries like "At which times can I start my manufacturing activity?" A scheduling agent can use a decomposable representation to pose 'what-if' queries involving subsets of variables, such as "If I start manufacturing at 8:00, at what times can the truck arrive?" Moreover, as new constraints arrive dynamically (e.g., the actual start time or duration of an activity is determined), an agent can use a decomposable representation to directly compute how these constraints affect the domains of future events so as to keep dispatching consistent scheduling advice.

Tsamardinos, Pollack, and Ganchev's (2001) approach to establishing and maintaining the solution space of a DTP suggests that these properties can always be established. Their approach calculates the minimal, decomposable STN associated with each of the DTP's (exponentially many) feasible labelings $\ell \in \mathcal{L}$, and then as new constraints arise, it tightens each STN accordingly, discarding all inconsistent STNs. We exploit this observation to formally prove that minimal representations of DTPs always exist, which follows as a corollary of Dechter, Meiri, and Pearl's Theorem 1 (1991).

Corollary 1. *A minimal representation of a consistent DTP always exists.*

Proof Sketch. The minimal network, $\mathcal{M}$, of a given DTP, $\mathcal{D}$, satisfies $\mathcal{M} = \cup_{\ell \in \mathcal{L}} M_\ell$, where M_ℓ is the minimal network of the STP defined by labeling ℓ, and the union is over the set of all possible labelings $\mathcal{L}$. Thus, the minimal network of $\mathcal{D}$ is the TCSP, $\mathcal{T} = \langle V, C_\mathcal{M} \rangle$, where the set of constraints, $C_\mathcal{M}$, is composed of constraints $C_{ij} \in C_\mathcal{M}$ defined as $v_j - v_i \in \cup_{\ell \in \mathcal{L}} (\mathcal{M}_\ell)_{ij}$, where $(\mathcal{M}_\ell)_{ij}$ corresponds to the bound interval on the difference between v_j and v_i in the minimal network of the STP corresponding to label ℓ. □

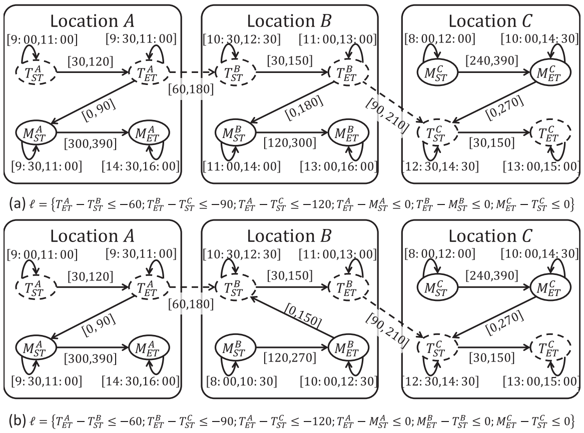

Figure 1: The minimal STN distance graphs corresponding to two feasible labelings of the problem in Table 1.

So, to generate a single, minimal temporal constraint network, we can merge the set of all consistent, minimal STNs (e.g., those in Figure 1) by labeling each edge with the union over all bound intervals. Similarly, if each of the consistent STNs is separately decomposable (the STNs in Figure 1 can be made decomposable trivially by adding explicitly the implicit constraints between *every* pair of timepoints, e.g., $T^B_{ST} - T^A_{ST} \in [90, 330]$), then any assignment of variables that is locally-consistent will lead to a global solution.

Theorem 2. *A minimal, decomposable representation of a consistent DTP always exists.*

Proof Sketch. If a DTP is consistent, its set of solutions can be represented as the set of minimal, decomposable STNs for the feasible labelings, $\ell \in \mathcal{L}$. Given this representation, any assignment to a set of variables that is locally consistent with respect to at least one of these STNs is, by definition, guaranteed to be extensible to a global solution. □

Since an MaDTP can be centralized into a DTP and a component MaSTP can be centralized into a component STP, these theorems and corollaries hold, *mutatis-mutandis*, for the MaDTP. Unfortunately, this remains an NP-hard problem and relying on centralized approaches mitigates the potential advantages of a distributed MaDTP representation (e.g., concurrency, privacy, etc.). Next, we introduce an approximate method that leads to increased concurrency, independence, and efficiency in finding a compact summary of the joint solution space by exploiting the limited interaction of agents in multiagent problems.

4 Local Decomposability

In this section, we introduce a new property called *local decomposability* that exploits loose-coupling between agents' problems, protects their strategic interests, and supports typical queries all by compactly summarizing the impact an agent has on others as an *influence space*. We provide and evaluate a new *distributed* algorithm that summarizes agents' solution spaces in significantly less time and space by using local, rather than full, decomposability.

4.1 Definition

Intuitively, a scheduling agent should be reasonably expected to answer queries about combinations of timepoint variables that it must know about (ones that it can assign, or ones involved in known constraints with ones it can assign). We call such queries *typical*. In contrast, it would be counter-intuitive and unreasonable to ask an agent to answer queries concerning variables and constraints that it does not know about; such queries are *atypical*, and are generally unanswerable by an agent. For example, the scheduling agent at location B should support queries over any (subset) of the activities that will occur at location B, but not over timing between A and C's manufacturing activities, the details of which companies A and C would likely keep private.

Generally, an agent may have strategic reasons for keeping the number of local variables involved in external constraints (and thus known by other agents) to a minimum. Our idea is to exploit this *loosely-coupled* structure of the network to efficiently establish sufficient decomposability to answer typical queries, rather than much more expensive (and privacy destroying) full decomposability that can also answer additional queries that may never arise. Local decomposability extends the idea of partial path consistency, which approximates decomposability by assuming only queries over the original constraints will arise. Local decomposability instead allows queries over *any* locally-known variables or constraints.

Definition 1. *An MaDTP is* **locally decomposable** *if, for any agent* i*, any locally consistent assignment of values to any subset of agent* i*'s* known *timepoint variables can be extended to a* joint *solution.*

Local decomposability enables an agent i to maintain minimality and full decomposability over its locally known timepoint variables, $V^i = \left\{V^i_L \cup V^i_X\right\}$, but does not require that it maintain any information over unknown timepoints, $\{v | v \in V^{j \neq i}_L, v \notin V^i_X\}$. Thus agent B can support any query over its local activities and queries involving external variables (e.g., "How long before the truck arrives here from location A?"), but not queries regarding A or C's local manufacturing activities. A challenge of local decomposability that we explore next is that care must be taken to ensure that local constraints are globally minimal — that locally feasible variable assignments will lead to *joint* solutions.

4.2 Influence Space

A key insight of our approach is that not all local labelings will lead to STNs that qualitatively change how an agent's problem will impact other agents. For example, regardless of what other activities the scheduling agent at location A is responsible for scheduling, coordinating with other agents only requires communicating the set of feasible times that T^A_{ST} and T^A_{ET} can occur. Thus, instead of enumerating *all* joint labelings, an agent i can instead focus on enumerating labelings that lead to distinct STNs over its *interface timepoint variables*, $V^i_I = \{V^i_L \cap V_X\}$, those variables that are local to agent i, but involved in one of agent i's external constraints, C^i_X. We call this smaller space of labelings agent i's local ***influence space***, motivated by the work of Witwicki and Durfee (2010). We represent agent i's influence space as a set $\mathbf{S}_I^i$ of minimal, decomposable STNs expressed over agent i's interface variables, V_I^i. An alternative view of an influence space is a set of constraints that summarize how an agent's local constraints impact other agents, and *vice-versa*.

The upside is that all coordination, including all communication and jointly represented aspects of the problem, is limited to these smaller influence spaces. The joint solution space, then, is represented in a distributed fashion as a cross-product of local solution spaces. This distributed representation allows agents to better protect their strategic interests such as privacy, autonomy, etc., with easier to maintain local solution spaces. For example, an agent scheduling m manufacturing activities can primarily spend its time managing the $m!$ possible orderings and, at the extreme, may only have to coordinate over the bounds of the single time-window during which a delivery can occur. The main disadvantage of local decomposability is that arbitrary queries cannot be answered, but as previously argued, typical ones can. An additional disadvantage is that communication between agents (e.g., to maintain minimality) becomes slightly more complicated as it must be expressed in terms of influence space constraints (constraints among influence variables), rather than directly communicating assignments.

4.3 MaDTP-LD Algorithm

Our MaDTP local decomposability (MaDTP-LD) algorithm is presented as Algorithm 1. The algorithm uses a *findSolution* function, which allows *any* solution algorithm (e.g., (Stergiou and Koubarakis 2000; Tsamardinos and Pollack 2003; Dutertre and Moura 2006)) to be used for finding decomposable STNs corresponding to feasible labelings. Each agent i uses this function to independently populate its solution space representation as a set, $\mathbf{S}^i$, of minimal, decomposable STNs, $\mathcal{S}^i$. Of course, before agents can compute their locally decomposable STNs in a globally-consistent manner, they must first coordinate, which is done by calculating and exchanging their influence spaces. An *extractSubNetwork* function assists in this process by taking an already decomposable STN instance and extracting only the decomposable subnetwork associated with the interface variables and all constraints between them.

The algorithm begins with each agent i initializing its set of interface variables V_I^i (line 1) and both its local solution space $\mathbf{S}^i$ and its influence space $\mathbf{S}_I^i$ (line 2). Each agent then independently calculates its influence space by finding a minimal, decomposable STN labeling (line 3), extracting the subnetwork formed by its interface variables, V_I^i (line 4), incorporating this subnetwork in its influence space $\mathbf{S}^i$ (line 5), and then adding this subnetwork as a no-good (line 5) so that the loop will terminate once all consistent labelings have been enumerated. Notice, the process of computing the influence space does not grow an agent's set of interface variables. So if only one of an agent's many timepoints is involved in an external constraint, each extracted subnetwork will contain just the locally-consistent time window for that variable. In this case, the agent will only need to communicate this variable's domain of locally-consistent time windows.

Generally, the influence space acts as a set of constraints over an agent's interface variables that implicitly summarizes an agent's many local constraints without revealing them, thus avoiding the *de facto* centralization required by full decomposability. Agent i communicates this set of constraints, as formed by $\mathbf{S}_I^i$, along with its external constraints, C_X^i, to all other agents (line 8), and incorporates other agents' interface constraints locally (line 9). Note, this exchange may grow the set of external variables that agent i is aware of, V_X^i, but guarantees the subsequent computations will be consistent with the constraints implied by other agents. While it is possible that agents could be more judicious in the information they exchange (e.g., agent i could send only the constraints that neighboring agent j is already aware of), this would represent a further approximation that sacrifices agents' support of typical queries over externally known variables. Finally, each agent concurrently computes its local solution space, $\mathbf{S}^i$, by finding and incorporating all local minimal, decomposable STNs into its solution space, adding each as a no-good, until all consistent labelings have been enumerated (lines 11-13). The algorithm terminates by returning agent i's locally decomposable representation, $\mathbf{S}^i$ (line 14).

Algorithm 1: MaDTP Local Decomposability

Input: $\mathcal{D}^i = \left\langle V^i = \left\{V_L^i \cup V_X^i\right\}, C^i = \left\{C_L^i \cup C_X^i\right\}\right\rangle$.
Output: A locally decomposable temporal network.

1 $V_I^i \leftarrow \{v \in V_L^i \cap V_X\}$;
2 $\mathbf{S}^i \leftarrow \{\}$; $\mathbf{S}_I^i \leftarrow \{\}$;
3 **while** *STN* $\mathcal{S}^i \leftarrow \mathcal{D}^i$.*findSolution()* **do**
4     $\mathcal{S}_I^i \leftarrow \mathcal{S}^i$.*extractSubNetwork*$(V_I^i)$;
5     $\mathbf{S}_I^i \leftarrow \mathbf{S}_I^i \cup \{\mathcal{S}_I^i\}$;
6     $\mathcal{D}^i.addNoGood(\mathcal{S}_I^i)$;
7 **foreach** *Agent* $j \neq i$ **do**
8   SEND(Agent j,$\mathbf{S}_I^i \cup C_X^i$);
9     $C_X^i \leftarrow C_X^i \cup$ RECEIVE(Agent j);
10 $\mathcal{D}^i.clearNoGoods()$;
11 **while** *STN* $\mathcal{S}^i \leftarrow \mathcal{D}^i$.*findSolution()* **do**
12     $\mathbf{S}^i \leftarrow \mathbf{S}^i \cup \mathcal{S}^i$;
13     $\mathcal{D}^i.addNoGood(\mathcal{S}^i)$;
14 **return** $\mathbf{S}^i$

Theorem 3. *The MaDTP-LD algorithm calculates local decomposability.*

Proof Sketch. By way of contradiction, assume that there exists some locally consistent assignment of values, a_β, to a subset of variables, $V_\beta \subseteq V^i$, for some agent i such that a_β is not part of any joint solution. Since a_β is locally consistent, it must have appeared as a solution to at least one of the feasible local component STNs, $\mathcal{S}_\beta^i$, generated by agent i in line 11. If a_β is consistent with a local component STN generated in line 11, it must also be simultaneously consistent with at least one constraint network, $\mathcal{S}_{I_\beta}^j$, for each agent $j \neq i$ (as collected in line 9). For each agent $j \neq i$, $\mathcal{S}_{I_\beta}^j$ is generated only if there is a corresponding feasible STN label $\mathcal{S}_\beta^j$ from which it was

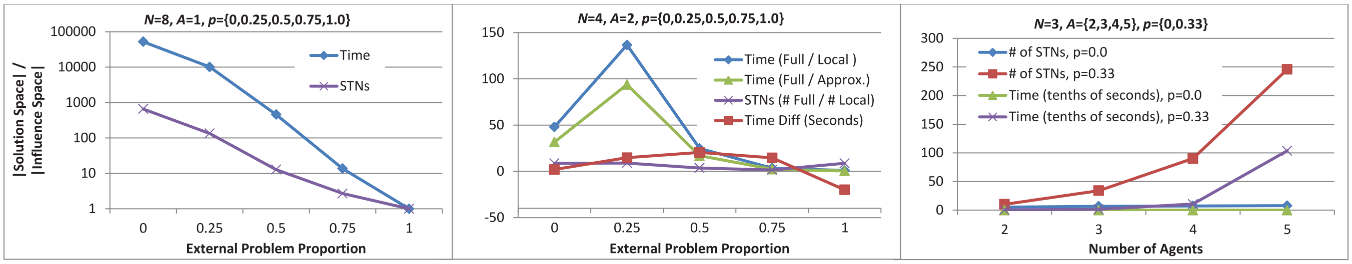

Figure 2: (a) Local solution space vs. Influence Space (b) Full vs. Local Decomposability (c) Scalability of Local Decomposability

extracted (lines 3-4). Hence, the MaSTP formed by the union, $\bigcup_i \mathcal{S}^i_\beta$, with which a_β is consistent, simultaneously satisfies all local C^i_L and external C^i_X constraints for all agents i. But this, by definition, is a joint solution, which violates our assumption. This implies that the MaDTP-LD algorithm does indeed calculate local decomposability. □

Theorem 4. *The MaDTP-LD algorithm calculates minimal constraints.*

Proof Sketch. Note, by Theorem 3, all values that appear in any interval that agent i calculated for any of its known constraints, $c \in C^i$, are part of at least one valid solution. By contradiction, assume that there exists some assignment a_β of a subset of known variables $V_\beta \subseteq V^i$ for some agent i such that a is part of a valid joint solution, but is not represented in the intervals that agent i calculated for its known constraints, C^i. Since line 11 results in only globally valid solutions (Theorem 3), agent i must never generate an STN $\mathcal{S}^i$ containing a_β. However, this is a contradiction, since line 11 is executed until *all* local, unique STN solutions are generated. Therefore, the MaDTP algorithm captures the exact set of feasible values within the intervals of each known constraint. □

Together, these two theorems prove that Algorithm 1 calculates a distributed joint solution space representation that is both sound (Theorem 3) and complete (Theorem 4). Note each agent i, in the worst case, will concurrently generate $\mathcal{O}(k^{|C^i|})$ unique labelings. This compares favorably to previous, centralized approaches (Tsamardinos and Pollack 2003; Shah, Conrad, and Williams 2009), which centrally generate $\mathcal{O}(k^{|C|})$ global labelings. While the exact runtime of both our approach and previous approaches depend on the performance of the solution algorithm used, only generating $\mathcal{O}(k^{|C^i|})$ labelings instead of $\mathcal{O}(k^{|C|})$, which, as the number of agents grows, implies $|C| >> |C^i|$, clearly represents a potentially exponential runtime savings. Similarly, the space (and analogously bandwidth) required of each agent to locally represent these local, fully-connected, STN labelings, $\mathcal{O}(|V^i|^2 \cdot k^{|C^i|})$, represents another potentially exponential reduction over previous approaches' worst case, $\mathcal{O}(|V|^2 \cdot k^{|C|})$. Of course, these exponential savings depend on the structure of the MaDTP. Our hypothesis, evaluated next, is that the relative performance of our algorithm will be at its best for loosely-coupled, evenly-distributed problems.

4.4 Empirical Evaluation

As described by Tsamardinos and Pollack (2003), the canonical random DTP generator for evaluating DTP algorithms (Stergiou and Koubarakis 2000) instantiates DTP instances using the parameters $\langle k, N, m, L\rangle$, where k is the number of disjuncts per constraint, N is the number of timepoint variables, m is the number of disjunctive temporal constraints, and L is a positive integer that specifies a range of values, $[-L, L]$, from which bounds over disjuncts are chosen with uniform probability, $v_j - v_i \leq b_{ij} \in [-L, L]$. We adapt this generator to be multiagent by adding two parameters: a, the number of agents, for each of which we generate a local DTP using the above specified random generator, and p, the proportion of constraints and timepoints that are external, e.g., $|C_X| = p \cdot m \cdot a$. In our experiments, we vary a, N, and p, and our default parameter settings are $k = 2$ and $L = 100$ (Tsamardinos and Pollack 2003). Two disjuncts per constraint naturally captures the kinds of constraints that typically appear in most scheduling problems (e.g., those in the right two columns of Table 1). We set m so that the ratio of constraints to timepoints is 4, $\frac{m}{N} = 4$.

For all parameter settings, we average over 100 randomly generated test cases. We use the state-of-the-art SMT solver YICES (Dutertre and Moura 2006) as the baseline implementation of $findSolution()$ in both our distributed MaDTP-LD algorithm and its centralized variant, which executes MaDTP-LD on a centralized, single-agent version of the problem. We record the maximum processing time across agents (i.e., the time the last agent completes execution) and the number of unique, decomposable STNs.

Our first experiment tests our hypothesis that the size of the influence space is smaller than the size of the corresponding local solution space. The calculation and relative size of the influence space *vs.* local solution space is specific to individual agents and is independent of the external constraints involved. As such, comparing the influence and local solution space sizes is done most straightforwardly and simply using a single agent by treating a portion of its variables as if they were interface variables (but without needing to explicitly add external constraints). Here, p determines the ratio of $|V^i_I|$to $|V^i_L|$, where $V^i_I = \emptyset$ when $p = 0$ and $V^i_I = V^i_L$ when $p = 1$. Figure 2 (a) shows that, when there are relatively few local variables in the interface (as dictated by parameter p), the influence space contains orders-of-magnitude fewer STNs and takes many orders-of-magnitude less time to find. However, when the interface contains all variables ($p = 1.0$), there is no advantage gained, which is to be expected since the local solution and influence spaces would be the same.

Our second experiment, shown in Figure 2 (b), tests how much speedup two agents using our MaDTP-LD algorithm achieve over a centralized approach calculating full decomposability. There are two contributing factors for why we would expect MaDTP-LD to outperform its centralized, full decomposability counter-part — (1) computational savings due to the approximation and (2) concurrency gained from load balancing. The second line (Full / Approx.) captures the gains made by the approximation alone by *centrally* calculating full decomposability and comparing to centrally calculating the local decomposability approximation. This demonstrates that, on average, 67% of the total speedup (Full/Local) is due to the savings generated from the approximation, while concurrency contributes the remaining 33% of the speedup. Unsurprisingly, it takes less time and fewer STNs to find local decomposability when some variables are not located in the interface ($p = 0.0 \ldots 0.75$), leading to up to 109.8 times speedup (and taking up to 21.3 fewer seconds) per problem instance in expectation. However, note that when $p = 1.0$, local decomposability is actually *less* efficient, taking 20.3 *more* seconds per problem instance in expectation. This is because there is overhead in first attempting to enumerate the local influence space, and when $p = 1.0$, many of the local solutions lead to unique global solutions. Finally, there is a decrease in speed-up from $p = 0.25$ to $p = 0.0$. This is because even the centralized approach can benefit from exploiting completely disjoint problem structure. Overall, local decomposability leads to significant speedup over calculating full decomposability, both due to the approximation being employed and the concurrency that it allows.

Finally, while the amount of relative speedup over the centralized approach is important, so is how well our MaDTP-LD algorithm scales. The results of our third experiment, shown in Figure 2 (c) show that when problems are completely disjoint, our approach scales well, as one would expect. However, even for relatively small amounts of coupling ($p = 0.33$), the effort and number of STNs that must be explored still grows exponentially (though at a significantly reduced rate due to a smaller base). These results indicate that applications that cannot afford substantial precompilation time will require exploiting additional local and interaction structures or employing additional forms of approximation to scale to larger problems containing more interacting agents.

5 Discussion

In this paper, we introduced the Multiagent Disjunctive Temporal Problem, a general constraint-based scheduling formulation that can capture the interactions of multiple agents in a distributed fashion. We demonstrated how the concepts of minimality and decomposability naturally extend to the MaDTP formulation, but that decomposability counters the computational and strategic objectives of an agent. We also contributed the idea of *local decomposability*, which eliminates significant computational overhead of centrally computing joint schedules, thus promoting agents' strategic interests while still supporting typical queries. We introduced an algorithm that exploits loose-coupling between agents and demonstrated significant speedup over a centralized algorithm that calculates full decomposability. In the future, we hope to follow the lead of Shah and Williams (2008) and exploit additional structure within DTPs to calculate more compact and efficient solution space representations of the MaDTP. We also intend to explore the trade-off between further approximation and efficiently handling (even atypical) queries. In this paper, we introduced an approach that sacrificed decomposability in favor of local decomposability so as to protect minimality. We would like to explore an alternative approach that instead sacrifices minimality (and thus the completeness of the joint solution space), in favor of a 'local minimality' that protects decomposability (Hunsberger 2002; Boerkoel and Durfee 2011).

6 Acknowledgments

We thank the anonymous reviewers for their comments and suggestions. This work was supported, in part, by the NSF under grant IIS-0964512.

References

Boerkoel, J., and Durfee, E. 2010. A Comparison of Algorithms for Solving the Multiagent Simple Temporal Problem. In *Proc. of ICAPS-10*, 26–33.

Boerkoel, J., and Durfee, E. 2011. Distributed Algorithms for Solving the Multiagent Temporal Decoupling Problem. In *Proc. of AAMAS 2011*, 141–148.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. In *Knowledge representation*, volume 49, 61–95.

Dutertre, B., and Moura, L. D. 2006. The YICES SMTsolver. Technical report.

Floyd, R. 1962. Shortest path. *Comm. of the ACM* 5(6):345.

Hunsberger, L. 2002. Algorithms for a temporal decoupling problem in multi-agent planning. In *Proc of AAAI-02*, 468–475.

Planken, L.; de Weerdt, M.; and van der Krogt, R. 2008. P3C: A new algorithm for the simple temporal problem. In *Proc. of ICAPS-08*, 256–263.

Shah, J., and Williams, B. 2008. Fast Dynamic Scheduling of Disjunctive Temporal Constraint Networks through Incremental Compilation. In *Proc. of ICAPS-08*, 322–329.

Shah, J.; Conrad, P.; and Williams, B. 2009. Fast distributed multi-agent plan execution with dynamic task assignment and scheduling. In *Proc. of ICAPS 2009*, 289–296.

Stergiou, K., and Koubarakis, M. 2000. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence* 120(1):81–117.

Tsamardinos, I., and Pollack, M. 2003. Efficient solution techniques for disjunctive temporal reasoning problems. *Artificial Intelligence* 151(1-2):43–89.

Tsamardinos, I.; Pollack, M.; and Ganchev, P. 2001. Flexible Dispatch of Disjunctive Plans. In *Proc. of ECP-06*, 417–422.

Witwicki, S., and Durfee, E. 2010. Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In *Proc. of ICAPS-10*, 185–192.

Xu, L., and Choueiry, B. 2003. A new effcient algorithm for solving the simple temporal problem. In *Proc. of TIME-ICTL-03*, 210–220.