Decoupling the Multiagent Disjunctive Temporal Problem

James C. Boerkoel Jr.
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology, Cambridge, MA
boerkoel@csail.mit.edu

Edmund H. Durfee
Computer Science and Engineering
University of Michigan, Ann Arbor, MI
durfee@umich.edu

Abstract

The Multiagent Disjunctive Temporal Problem (MaDTP) is a general constraint-based formulation for scheduling problems that involve interdependent agents. Decoupling agents' interdependent scheduling problems, so that each agent can manage its schedule independently, requires agents to adopt additional local constraints that effectively subsume their interdependencies. In this paper, we present the first algorithm for decoupling MaDTPs. Our distributed algorithm is provably sound and complete. Our experiments show that the relative efficiency of using temporal decoupling to find solution spaces for MaDTPs, compared to algorithms that find complete solution spaces, *improves* with the interconnectedness between agents schedules, leading to orders of magnitude relative speeedup. However, decoupling by its nature restricts agents' scheduling flexibility; we define novel flexibility metrics for MaDTPs, and show empirically how the flexibility sacrificed depends on the degree of coupling between agents' schedules.

Introduction

In multiagent scheduling, each agent has to schedule its activities to respect its local (internal) temporal constraints, and also to satisfy external constraints between its activities and activities of other agents. External constraints capture key relationships between different agents' activities, such as the ordering or synchronization of tasks. However, they also introduce *coupling* between agents' local problems in that local reasoning, such as schedule refinement or revision, can require coordination between agents. For example, to modify the completion time for a deliverable (e.g., a data analysis or query answer), an agent needs to check that others can adjust their schedules to accommodate the change, which can trigger a further cascade of adjustments by other agents to their schedules. Hence, coupling limits an agent's ability to make rapid (unilateral) modifications to its local schedule in response to changing circumstances.

A scheduling problem is *decoupled* if each agent can unilaterally form a solution to its local problem such that agents' combined solutions are guaranteed to satisfy all the external constraints (Hunsberger 2002). The Multiagent Temporal Decoupling Problem (MaTDP), then, is the problem of decoupling a coupled problem. This involves agents adopting additional temporal constraints within their local problems (e.g., bounds on the timings of activities) such that, if these constraints are met, the external constraints must also be met. Decoupling thus sacrifices some flexibility in each agents' local scheduling options in favor of independent and rapid local scheduling. We have previously presented a formal definition of the MaTDP and algorithms for solving it for Multiagent Simple Temporal Problems (MaSTPs), in which temporal constraints are strictly conjunctive (Boerkoel and Durfee 2011). MaSTPs thus exclude problems where, for instance, two activities are forbidden to overlap (they might both need the same unsharable resource) but could be scheduled relative to each other in either order. Including disjunction raises a host of challenging computational issues, introduces complications in characterizing concepts like flexibility, and expands the space of decoupling options available to agents. The focus of this paper is on addressing these challenges by leveraging shared information as early and as often as possible to efficiently find decouplings for more general Multiagent Disjunctive Temporal Problems (MaDTPs).

Preliminaries

Simple Temporal Problem (STP). The STP, $\mathcal{S} = \langle V, C \rangle$ (Dechter, Meiri, and Pearl 1991), consists of a set of n timepoint variables, V, and a set of m temporal difference constraints, C. Each timepoint variable represents an event and has a continuous domain of values (e.g., clock times) that can be expressed as a constraint relative to a special ***zero reference timepoint*** variable, $z \in V$, which represents the start of time. Each temporal difference constraint c_{ij} is of the form $v_j - v_i \leq b_{ij}$, where v_i and v_j are distinct timepoints, and $b_{ij} \in \mathbb{R}$ is a real number bound on the difference between v_j and v_i. Often, as notational convenience, two constraints, c_{ij} and c_{ji} can be represented as a single constraint using a ***bound interval*** over the difference, $v_j - v_i \in [-b_{ji}, b_{ij}]$. A ***schedule*** is an assignment of specific time values to timepoint variables. An STP is ***consistent*** if it has at least one ***solution***, which is a schedule that respects all constraints.

Each STP is associated with a Simple Temporal Network (STN). The bottom row of Figure 1a is an example of an STN containing the start (ST) and end (ET) times for two tasks ($T3, T4$), where the differences between some pairs

of timepoints are constrained to be within certain bounds intervals. For instance the edge from $T3^B_{ST}$ to $T3^B_{ET}$ represents the constraint $T3^B_{ET} - T3^B_{ST} \in [50, 80]$, that is, the duration of task T3 is between 50 and 80 minutes. An edge between a node and itself represents constraints relative to z; for example, all timepoints in Figure 1a must occur between 7AM and noon. For efficient algorithmic manipulation, each STN can be represented by a weighted, directed graph called a distance graph where each variable appears as a graph node and each constraint $v_j - v_i \leq b_{ij}$ is represented with an edge e_{ij} from v_j to v_i with initial weight $w_{ij} = b_{ij}$. Shortest path algorithms, such as Floyd-Warshall (1962), can efficiently calculate and represent the entire ***solution space*** of an STP instance by finding the tightest possible path between every pair of variables in the fully-connected distance graph.

Disjunctive Temporal Problem (DTP). In the DTP, $\mathcal{D} = \langle V, C \rangle$, each of the m constraints in C takes the form of a *disjunctive* temporal difference $d_1 \vee d_2 \vee \cdots \vee d_k$, where each $d_w = v_{j_w} - v_{i_w} \in [-b_{ji_w}, b_{ij_w}]$ is itself a temporal difference constraint (Stergiou and Koubarakis 2000). The STP is thus a special case of the DTP where every constraint has exactly one disjunct. This more general disjunctive temporal difference constraint represents a choice among its k constituent temporal difference constraints, where each constituent has its own bounds expressed over its own pair of timepoints, which could differ from the timepoints of the other disjunctive constituents of the same constraint. Taken as a whole, Figure 1a is a DTP, where the edges corresponding to the disjuncts of disjunctive constraints are represented with double lines and where the edges belonging to a single disjunctive constraint intersect (e.g., T1 must follow T2 by 60 minutes or precede T2 by 45).

A ***component STP*** is formed by selecting a disjunct (temporal difference) with which to label each disjunctive constraint. A schedule s, then, is a solution to a DTP instance if and only if it is the solution to at least one of the DTP's component STPs. The DTP is known to be an NP-hard problem, where for general DTPs with m disjunctive temporal constraints, each of arity k, each of the $\mathcal{O}(k^m)$ possible component STPs must be explored in the worst case (Stergiou and Koubarakis 2000). However, each component STP can be evaluated in polynomial time, putting the DTP in the class of NP-complete problems. The DTP is often solved using a ***meta-CSP*** formulation (Tsamardinos and Pollack 2003), where each constraint $c \in C$ forms a ***meta-variable*** with a domain of ***meta-values*** comprised of the set of possible disjuncts. This leads to a backtracking search algorithm that interleaves the STP forward-checking procedure with an assignment of a meta-value to a meta-variable, and can incorporate CSP techniques such as no-good recording and back-jumping to decrease the expected DTP search runtime.

Multiagent Disjunctive Temporal Problem (MaDTP). The Multiagent Simple Temporal Problem (MaSTP) is an STP where the set of timepoint variables are partitioned among a agents, resulting in a STP subproblems that are related through a set of ***external*** (interagent) constraints (Boerkoel and Durfee 2010; 2011; 2013). Following a parallel construction, the MaDTP is composed of a local DTP subproblems, one for each of the a agents, and a set of external constraints, C_X, which are disjunctive temporal constraints that relate the local subproblems of different agents (Boerkoel and Durfee 2012). An agent i's ***local DTP subproblem*** is defined as $\mathcal{D}^i_L = \langle V^i_L, C^i_L \rangle$, where: V^i_L is agent i's set of ***local variables***, and is the partition of timepoints assignable by agent i (and includes agent i's reference to z); and C^i_L is agent i's set of ***local constraints***, where each $c_y \in C^i_L$ is specified over exclusively local variables. In addition to its local problem, agent i is also aware of: its ***external constraints***, C^i_X, where each disjunctive temporal constraint $c \in C^i_X$ is specified over at least one local variable $v^i_k \in V^i_L$ and at least one variable belonging to another agent $v^j_l \in V^j_L$, $i \neq j$; and its ***external variables***, V^i_X, where each $v^j_l \in V^i_X$ appears in at least one of agent i's external constraints, but is local to some other agent, $v^j_l \in V^j_L$, $i \neq j$. Since the DTP can be viewed as a specialization of the MaDTP (and *vice-versa*), the MaDTP, like the DTP, falls into the class of NP-complete problems (Boerkoel and Durfee 2012).

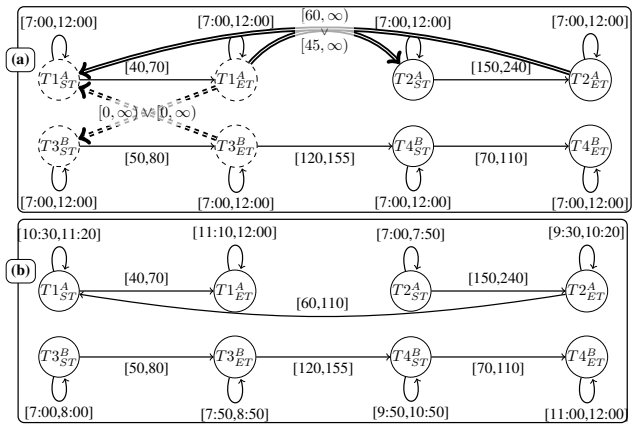

Figure 1: An example MaDTP (a) and corresponding temporal decoupling (b)

Agent i's set of ***known variables*** is $V^i = \{V^i_L \cup V^i_X\}$ and agent i's set of ***known constraints*** is $C^i = \{C^i_L \cup C^i_X\}$. ***Shared variables***, V_S, and similarly ***shared constraints***, C_S, are those that are known by more than one agent, $V_S = \{v_y | v_y \in V^i \cap V^j, i \neq j\}$ and $C_S = \{c_y | c_y \in C^i \cap C^j, i \neq j\}$, and together form the ***shared DTP***, $\mathcal{D}_S = \langle V_S, C_S \rangle$. In Figure 1, the top row represents agent A's DTP subproblem: it has the 4 local variables (associated with tasks $T1$ and $T2$) shown, along with reference timepoint z, and local constraints connecting just those 5 variables (recall constraints with z are shown as the variables' domains). A has one external constraint, the disjunctive constraint depicted with dotted lines on the left side of the figure, and thus A's external variables are $T3^B_{ST}$ and $T3^B_{ET}$, and these 2 variables along with its 5 local variables comprise its known variables. B's definitions follow analogously. Thus, the shared DTP is comprised of the temporal reference point z, the 4 leftmost nodes in Figure 1, and the constraints between them.

Multiagent Temporal Decoupling Problem (MaTDP). Here we extend the original definition of the MaTDP (Huns-

berger 2002; Boerkoel and Durfee 2011; 2013), which was previously defined for problems containing strictly conjunctive constraints (MaSTPs). Agents' local DTP subproblems $\{\mathcal{D}_L^1, \mathcal{D}_L^2, \ldots, \mathcal{D}_L^n\}$ form a ***temporal decoupling*** of a consistent MaDTP $\mathbf{D}$ if: $\{\mathcal{D}_L^1, \mathcal{D}_L^2, \ldots, \mathcal{D}_L^n\}$ are consistent DTPs; and *any* combination of local solutions that merges a solution from each of the local subproblems in $\{\mathcal{D}_L^1, \mathcal{D}_L^2, \ldots, \mathcal{D}_L^n\}$ yields a joint solution to $\mathbf{D}$.

The MaTDP is defined as, for each agent i, finding a set of constraints C_Δ^i such that if $\mathcal{D}_{L+\Delta}^i = \langle V_L^i, C_L^i \cup C_\Delta^i \rangle$, then $\{\mathcal{D}_{L+\Delta}^1, \mathcal{D}_{L+\Delta}^2, \ldots, \mathcal{D}_{L+\Delta}^n\}$ is a temporal decoupling of MaDTP $\mathbf{D}$. Figure 1b represents a decoupling of the MaDTP in Figure 1, where any solution to agent A's DTP in the top row can be combined with any solution to agent B's DTP in the bottom row to form a joint solution. Here, the critical constraints that allow the decoupling are that $T1_{ST}^A$ will not begin until after 10:30 while $T3_{ET}^B$ must complete before 8:50, making the external constraint between agents superfluous. Note that solving the MaTDP does not mean that the agents' subproblems have somehow become inherently independent of each other (with respect to the original MaDTP), but rather that the new decoupling constraints provide agents a way to perform sound reasoning completely independently of each other. A ***minimal decoupling*** is one where, if the bound of any decoupling constraint $c \in C_\Delta^i$ for some agent i is relaxed (or removed), then $\{\mathcal{D}_{L+\Delta}^1, \mathcal{D}_{L+\Delta}^2, \ldots, \mathcal{D}_{L+\Delta}^n\}$ is no longer a decoupling.

Decoupling a MaDTP is more challenging than decoupling a MaSTP. The first challenge is that imposing a valid temporal decoupling requires ensuring that at least one of the solutions to the MaDTP, if any exist, must survive the decoupling (Hunsberger 2002). Applying Planken, de Weerdt, and Witteveen's proof that temporal decoupling generally falls into the same complexity class as the underlying problem representation (2010), it follows that: (1) any solution to the MaDTP is a *defacto* temporal decoupling; and (2) any temporal decoupling of a MaDTP where each agent owns only a single timepoint is a solution to the MaDTP. Thus, finding a temporal decoupling of a MaDTP, like solving it, falls into the class of NP-complete problems and requires $\mathcal{O}(k^m)$ time-complexity in the worst case. A second challenge is that, unlike an MaSTP where a distance graph can summarize the solution space, the disjunctive constraints in a MaDTP may induce combinatorially many different distance graphs, making compact representation of the full solution space challenging. Disjunctive temporal constraints, on the other hand, involve arbitrarily many pairs of time-point variables, and may induce combinatorially many different network structures, making efficient representation of the set of these possible structures challenging.

Influence-Based Temporal Decoupling

Our MaDTP-LD algorithm (Boerkoel and Durfee 2012) for computing the complete set of MaDTP solution spaces provides a basis for our decoupling algorithm. The key insight of MaDTP-LD is that not all local solutions qualitatively change how an agent's problem will impact other agents. Thus, instead of enumerating *all* joint component MaSTPs, an agent i can instead focus on enumerating its local component STPs that lead to distinct STP projections over its ***interface timepoint variables***, $V_I^i = \{V_L^i \cap V_S\}$, those variables that are local to agent i, but involved in one of agent i's external constraints, C_X^i. An agent's ***influence space*** (Oliehoek, Witwicki, and Kaelbling 2012) summarizes how its local constraints impact other agents so that all coordination can be limited to these smaller influence spaces. The MaDTP-LD operates in three distinct phases: (1) each agent independently enumerates its influence space; (2) then agents exchange their influence spaces, incorporating the influence spaces of other agents as new local constraints; and (3) finally, each agent independently enumerates its local solution space while respecting the influence space constraints of all agents. The joint solution space is represented in a distributed fashion as a cross-product of local solution spaces and allows agents to independently manage their local solution spaces.

MaDTP Temporal Decoupling Algorithm

Since the goal our MaDTP temporal decoupling (MaDTP-TD) algorithm (presented as Algorithm 1) is to compute a temporally decoupled space of solutions, it can incorporate information from the shared DTP as early and often as possible, rather than waiting for each agent to completely enumerate its local influence space before shared reasoning occurs as in the complete MaDTP-LD algorithm. Incorporating shared information has the effect of pruning globally infeasible schedules from an agents' local search space early on and then, once a temporal decoupling has been found, short-circuiting agents' reasoning by eliminating schedules that are no longer consistent with respect to the new decoupling coupling constraints.

Multiagent Singleton Consistency. The first key deviation of our MaDTP-TD algorithm (Algorithm 1) from MaDTP-LD is in lines 1-3, where we use an MaSTP relaxation of the MaDTP to efficiently propagate information between agents in order to prune provably infeasible portions of the MaDTP search space. Agent i starts (line 1) by extracting its local portion of the MaSTP abstraction, $\mathcal{S}^i = \langle V^i, C^{i,k=1} \rangle$, composed of agent i's set of variables V^i and agent i's ***singleton constraints***—the subset of agent i's temporal constraints that contain only a single disjunct (i.e., temporal difference), $C^{i,k=1} \subseteq C^i$. Agents then propagate the MaSTP constraints using the distributed, polynomial-time D$\triangle$PPC algorithm (Boerkoel and Durfee 2010) in line 2, which summarizes the space of solutions that is described by tightening the singleton constraints. Because only a subset of MaDTP constraints are considered, this represents a superset of the true solution space of the underlying MaDTP. These new and tighter singleton constraints, in turn, can be used to prune which meta-values (temporal differences) can be assigned to which meta-variables of the original MaDTP (Tsamardinos and Pollack 2003). The forward-checking procedure (line 3) prunes any disjunct that is inconsistent with the MaSTP compilation, since it is guaranteed to be provably inconsistent with the overall MaDTP, and also checks for ***subsumed constraints***, those that have an inherently-satisfied disjunct and thus can safely be ignored. Our empirical evaluation (discussed later) confirms that these preprocessing steps

Algorithm 1: MaDTP Temporal Decoupling Algorithm

Input: $\mathcal{D}^i = \langle V^i = \{V_L^i \cup V_X^i\}, C^i = \{C_L^i \cup C_X^i\}\rangle$.
Output: Agent i's temporally DTP $\mathcal{D}^i$.

1 $\mathcal{S}^i \leftarrow \langle V^i, C^{i,k=1}\rangle$
2 $D\triangle PPC(\mathcal{S}^i)$
3 $\mathcal{D}^i \leftarrow \mathcal{D}^i$.PRUNEINCONSISTENTANDSUBSUMED($\mathcal{S}^i$)
4 $V_I^i \leftarrow \{v \in V_L^i \cap V_S\}$
5 **while** STN $\mathcal{S}^i \leftarrow \mathcal{D}^i$.FINDSOLUTION() **do**
6 $\quad \mathcal{S}_I^i \leftarrow \mathcal{S}^i$.EXTRACTSUBNETWORK($V_I^i$)
7 $\quad \mathcal{D}^i$.ADDNOGOOD($\mathcal{S}_I^i$)
8 $\quad \mathcal{D}^i$.SENDUPDATETOCOORDINATOR($\mathcal{S}_I^i$)
9 $\quad$ **if** COORDINATORREPORTSSOLUTION() **break**
10 $\mathcal{S}_I^i \leftarrow$ BLOCKRECEIVESOLUTIONFROMCOORDINATOR()
11 $\mathcal{C}_\Delta^i \leftarrow$ MaTDP+R($\mathcal{S}_I^i$)
12 $\mathbf{S}_L^i \leftarrow \{\}$; $\mathcal{D}^i$.CLEARNOGOODS()
13 $\mathcal{C}_L^i \leftarrow C_L^i \cup \mathcal{C}_\Delta^i$
14 **while** STP $\mathcal{S}_L^i \leftarrow \mathcal{D}_L^i$.FINDSOLUTION() **do**
15 $\quad \mathbf{S}_L^i \leftarrow \mathbf{S}_L^i \cup \mathcal{S}_L^i$
16 $\quad \mathcal{D}_L^i$.ADDNOGOOD($\mathcal{S}_L^i$)
17 **return** $\mathbf{S}_L^i$

can yield up to a 3-fold reduction in the overall solve time of our MaDTP-TD algorithm in practice.

Incremental Influence Space Construction. The next deviation of our MaDTP-TD algorithm from the complete MaDTP-LD counter-part is in the way agents construct the shared DTP solution space. The shared DTP solution space can be thought of as the union or cross-product of agents' influence spaces. Thus as agents construct their local influence spaces, they can be simultaneously building the shared DTP solution space in a way that is provably sound and progressively more complete over time. Then, as soon as a solution to the shared DTP is found, it can be used to construct and install a temporal decoupling, which in turn saves each agent from computing its entire solution space, representing a potentially combinatorial savings.

Agent i first identifies its interface variables in line 4. Next, agent i loops to enumerate each local STN that leads to a distinct influence space (lines 5-9). The algorithm uses a generic FINDSOLUTION function in lines 5 and 14, which is a stand in for *any* solution algorithm (e.g., (Stergiou and Koubarakis 2000; Tsamardinos and Pollack 2003; Dutertre and Moura 2006)) that can find STN representations of component STP solutions. No-goods are constructed to avoid generating local solutions that have the same influences as previously generated solutions. Only generating local STNs that lead to distinct influences saves time over wastefully enumerating influence-subsumed local solution STNs. As each local STN that leads to a distinct influence space is computed, the incremental contribution to the overall, complete influence space is not only added to agent i's set of no-goods (line 7), but also sent to a coordinator (line 8) whose role is described shortly. Then during every loop, an agent checks to see if the coordinator has identified a solution to the shared DTP (line 9) and if so, breaks from enumerating its influence space.

Next, the agent receives the shared solution from the coordinator (line 10). At this point, each agent will possess its portion of a component STN that represents a solution to the shared DTP, to which agents apply the MaDTP+R algorithm as described by Boerkoel and Durfee (2011). MaDTP+R is a distributed algorithm that works by assigning all shared timepoint variables, which effectively renders all external constraints moot, and then revisits each timepoint to recover all possible slack for its domain while still maintaining a temporal decoupling. MaDTP+R computes a set of local decoupling constraints C_Δ^i for each agent i, which represents a *minimal* decoupling with respect to the chosen component STN; however, it may not necessarily be a globally optimal one. Due to the decoupling constraints, as an agent enumerates its local solution space (lines 14 - 16), it incurs the combinatorics of only its decoupled local solution space, rather the combinatorics of the entire joint solution space.

Shared DTP Reasoning. The agent responsible for solving the shared portion of the MaDTP, which we refer to as the "coordinator," executes Shared DTP Reasoning as shown in Procedure 1. The coordinator initializes its representation of the shared DTP by blocking until it has been seeded with each agent's initial influence space (line 2). Because each influence space is part of a local solution for each agent, this guarantees that any solution to the shared DTP that the coordinator finds must project to a global solution, and thus must contain a sound temporal decoupling. After this initial blocking communication, the coordinator loops through receiving other alternative influences from agents (with nonblocking communication in line 6) until it finds a solution to the shared DTP (line 4-7). Thus, until a solution is found, the coordinator progressively grows the shared DTP representation by merging any agent's newly-reported influence space with the current influence space representation for that agent.[1] Once a solution to the shared DTP is found, the coordinator sends each agent its portion of the solution STN to be decoupled.

Theorem 1. *The MaDTP-TD algorithm is sound, complete.*

Proof (sketch). It can be shown by contradiction that any solution s to the decoupled MaDTP must be a solution to the original MaDTP, because s must be consistent with decoupling constraints that each agent calculates, which are in turn constructed with respect to a solution to the shared DTP, which is in turn composed of influence spaces that are part of local solutions for each agent. Similarly, since a valid decoupling exists iff a solution s to the overall MaDTP exists, if s is a global solution, each agent will find its local component of s and communicate it to the coordinator, who finds the shared component of s, which leads to a valid decoupling for the entire problem. □

While our focus here is on generating a valid temporal decoupling as expediently as possible, as discussed in our future research aims, our algorithm could be adapted so that

[1] The disjunction involves taking the constraints implied by the growing set of influence spaces, and converting them to disjunctive constraints; that is, taking the union of an agents' influence space, which is inherently represented in CNF, and converting it to DNF.

Procedure Shared DTP Reasoning

foreach agent i **do**
  $\mathbf{S}_I^i \leftarrow$ BLOCKRECEIVEUPDATE(agent i)
$\mathcal{D}_S \leftarrow \mathbf{S}_I^1 \times \mathbf{S}_I^2 \times \cdots \mathbf{S}_I^n \cup C_X$
while $\mathcal{S}_S \leftarrow \mathcal{D}_S$.FINDSOLUTION() == null **do**
  foreach agent i **do**
    $\mathbf{S}_I^i \leftarrow \mathbf{S}_I^i \cup$ RECEIVEUPDATE(agent i)
  $\mathcal{D}_S \leftarrow \mathbf{S}_I^1 \times \mathbf{S}_I^2 \times \cdots \mathbf{S}_I^n \cup C_X$
foreach agent i **do**
  $\mathcal{S}_I^i \leftarrow \mathcal{S}^i$.EXTRACTSUBNETWORK($V_I^i$)
  SENDSHAREDSOLUTION(i, S_I^i)

agents generate candidate decouplings in heuristic best-first and anytime manner by using the metrics we describe in the next section to select a temporal decoupling whose expected quality progressively improves with allowed runtime.

Quantifying the Completeness Costs of Decoupling

One of the MaDTP-TD algorithm's major advantages is that as soon as a decoupling is installed, agents can immediately prune all local schedules that are inconsistent with the new decoupling constraints. This is in contrast to the MaDTP-LD algorithm, in which each agent generates *all* consistent STPs involving its known variables. Decoupling, then, represents a sacrifice in the completeness of the MaDTP solution space—new decoupling constraints may mean that a portion of local solutions are lost. For instance, in the decoupling in Figure 1b, any joint solutions where $T1$ is performed before $T3$ have been sacrificed. This begs the question: how do we empirically quantify the magnitude of this sacrifice?

A standard measure of completeness in the (Ma)STP literature is flexibility (Hunsberger 2002; Wilcox, Nikolaidis, and Shah 2012), which captures the amount of slack (i.e., $w_{ij} + w_{ji}$) for each edge e_{ij} in the temporal network. However, such measures do not account for the presence of disjunctive constraints. So to measure flexibility in MaDTPs, we start by generalizing the definition of ***flexibility*** for a particular edge, e_{ij}, to $Flex(i,j) = \sum_{\ell \in IL}(w_{ij}^{\ell} + w_{ji}^{\ell})$, by summing over each edge's set of disjunctive interval labellings (IL), where each interval label is the corresponding edge weights of a consistent component STN. We explicitly avoid double counting flexibility by counting only distinct (non-overlapping) labels. So for example, if an edge has labels $\{[0,20],[30,35],[5,15],[15,25],[0,5],[30,40]\}$, we aggregate to $\{[0,25],[30,40]\}$ for a total flexibility of $25 + 10 = 35$. The goal is for each agent to maintain as much scheduling flexibility between each pair of timepoints as possible, which can provide greater flexibility for future scheduling decisions.

Our generalized flexibility metric inherits the MaSTP emphasis on retaining as many different solutions as possible, captured in the ranges of timepoints' values. For instance, the decoupling in Figure 1b maintains significant flexibility (50 minutes of slack time per timepoint variable). However, unlike MaSTPs, MaDTPs also possess a second source of flexibility, because disjunctive temporal constraints allow alternative component (Ma)STPs to be adopted. Despite its flexibility, for example, the decoupling in Figure 1b completely breaks if agent A discovers or determines that T1 must precede T2, an alternative included in the original MaDTP. Agent A might have been better served sacrificing some flexibility in favor of a more diversified solution space to support critical alternatives in the face of such circumstances. Thus, as a second measure of completeness, we also count the number of distinct local STNs that are used to represent agents' local solution spaces in the decoupled *vs.* complete representations, where we separately count a local component STN if (1) it is a sound and complete representation of a component STP's solutions, and (2) it is not subsumed by (i.e., does not represent a subset of) another component STP's solution space. In the future, we would like to synthesize these metrics into a single, comprehensive metric of (Ma)DTP completeness, though the right balance of flexibility and diversity in a decoupled solution space will likely vary with application.

Empirical Evaluation

We adopt our experimental setup from our previous MaDTP evaluation (Boerkoel and Durfee 2012), which extends the canonical random DTP generator for evaluating DTP algorithms (Stergiou and Koubarakis 2000) by adding two parameters: a, the number of agents, where for each agent we generate a local DTP using the canonical random generator; and p, which establishes the proportion of the problem that is made external by adding $|C_X| = p \cdot a \cdot m$ constraints, which involve a total of $|V_X| = p \cdot a \cdot n$ variables. We also assume that each local agent problem contains a reference to the zero reference timepoint z along with makespan constraints of the form $v_i - z \in [0, \text{MAXPOSSIBLEMAKESPAN}]$. In these experiments, we vary a and p, and use the canonical generator to generate a agent problems each with $n = 5$ timepoint variables and $m = 20$ disjunctive constraints containing $k = 2$ disjuncts each. For all parameter settings, we report the average performance 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 the FINDSOLUTION() function in both our MaDTP-TD and MaDTP-LD algorithms. We record the maximum processing time across agents (i.e., the time the last agent completes execution) and count the number of distinct STP solutions generated by applying FINDSOLUTION.

Improved Scalability. In our first set of experiments, we investigate how our MaDTP-TD algorithm scales as the number of agents increases, $a \in \{2, 4, 8, 16, 32, 64\}$ for $p = \{0.0, 0.2, 0.4\}$. Using a 100 second timeout on loosely-coupled problems ($p = 0.2$), we compared the runtime performance of MaDTP-TD against the complete MaDTP-LD algorithm. Figure 2 gives the results of this set of experiments, showing that the MaDTP-LD algorithm does not scale well to problems containing more than just a few (no more than ten) agents (where nearly 100% of problems containing just 8 agents timed out). The MaDTP-TD, on the other hand, scales to problems with an order-of-magnitude more agents, executing in over 3 orders-of-magnitude less time than the MaDTP-LD algorithm for problems containing just 8 agents.

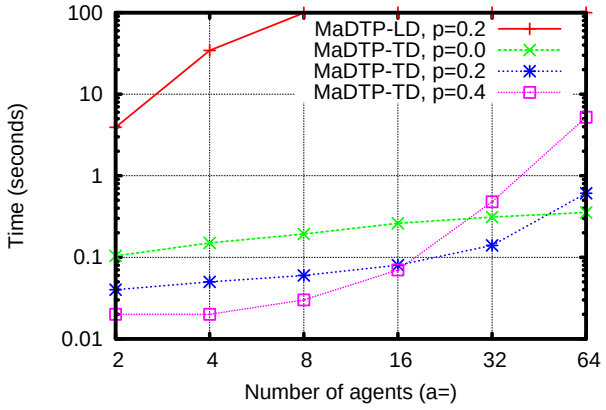

Figure 2: Scalability of MaDTP-TD vs MaDTP-LD.

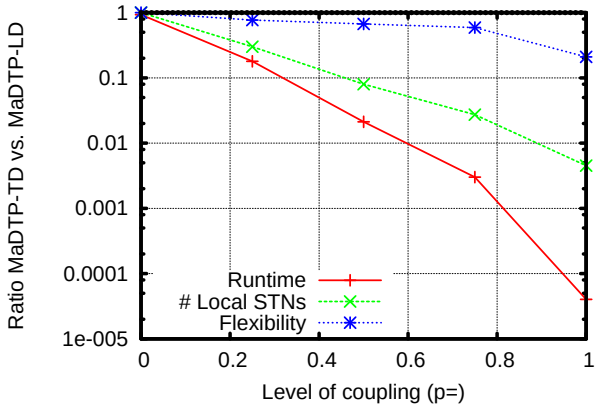

Figure 3: Flexibility of MaDTP-TD vs MaDTP-LD.

Interestingly, even when agents' problems are completely decoupled ($p = 0.0$), the overall runtime of our algorithm still grows with the number of agents. This is because each execution of the MaDTP-TD algorithm terminates only when *all* agents have completed computing their local solution spaces, and so as the number of agents increases, the problem is increasingly likely to include an agent that has randomly drawn a particularly large solution space, and so requires more time to complete its execution. Similar trends can be noticed for problems containing fewer agents when $p = 0.2$ and $p = 0.4$. However, in both cases, there appears to be an elbow indicating where the exponential nature of the decoupling search overtakes the execution time of subsequently enumerating consistent local STPs. Additionally, we verified that using our MaSTP relaxation pruning technique (lines 1-3 in Algorithm 1) further improves the scalability of the MaDTP-TD algorithm in the presence of singleton constraints—the runtime of MaDTP-TD decreases by up to 41% when p=0.2 and 67% when p=0.4 as the number of agents grows to 64.

Trading Completeness for Efficiency. In our second experiment, shown in Figure 3, we report the ratio of our MaDTP-TD algorithm to the complete MaDTP-LD algorithm using both expected runtimes and our completeness metrics. We vary p while holding the number of agents constant at $a = 2$. As one would expect, when there are no external constraints ($p = 0$), the two algorithms have the same expected runtime. However, as p, and thus the level of coupling between agent problems, grows, the ratio of MaDTP-TD vs. MaDTP-LD runtime decreases, representing over *four* orders-of-magnitude decrease in runtime for a problem containing just *two* agents with five timepoints each. Comparing to the complete solution spaces output by the MaDTP-LD algorithm, we also report the average proportion of the number of local STP solutions that are maintained by the MaDTP-TD algorithm and the average ratio of flexibility over the local edges of all the decoupled agent problems. Generally, the trend across all completeness metrics is that, as coupling increases, the MaDTP-TD produces increasingly less complete solution spaces. The MaDTP-TD suffers the most in terms of the number of distinct local STNs metric, where it produces three orders-of-magnitude fewer local component STPs than MaDTP-LD as p approaches 1. This helps explain a significant source of its computational advantages. MaDTP-TD suffers to a much lesser extent with respect to the more traditional flexibility metric, leading to less than an order-of-magnitude reduction in flexibility. In expectation, gains in runtime outpace sacrifices in completeness, where in the extreme case when $p = 1.0$, MaDTP-TD achieves four orders-of-magnitude speedup while preserving 20% of the flexbility over complete approaches.

Discussion

In this paper, we addressed critical challenges and complications of decoupling general, disjunctive temporal constraints between agents' local scheduling problems. In our distributed MaDTP-TD algorithm, agents independently and incrementally build their influence spaces until a valid temporal decoupling can be found, which results in significant speed-up over approaches that calculate complete solution spaces, and extends the feasibility of using the MaDTP to at least an order-of-magnitude more agents in practice. Our new metrics of MaDTP completeness and flexibility allowed us to empirically demonstrate that the gains in runtime outpace losses in completeness in expectation. We also contributed a preprocessing technique that exchanged an abstraction of the shared DTP sooner, focusing search in more fertile areas of the search space and leading to up to an additional three-fold decrease in runtime. In the future, we would like to investigate optimal and heuristic variants of our MaDTP-TD algorithm where, for example, agents produce influence spaces in a best-first manner in an attempt to guide the coordinator to more flexible temporal decouplings in an anytime manner. We would also like to compare our approaches using data from real-world problems and further investigate the utility of our flexibility metrics in practice.

Acknowledgments

We thank the anonymous reviewers for their suggestions and Professor Julie Shah and the Interactive Robotics Group for their guidance. This work was supported, in part, by the NSF under grant IIS-0964512 and by a UM Rackham Fellowship.

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-11*, 141–148.

Boerkoel, J., and Durfee, E. 2012. A distributed approach to summarizing spaces of multiagent schedules. In *Proc. of AAAI-12*, 1742–1748.

Boerkoel, J. C., and Durfee, E. H. 2013. Distributed Reasoning for Multiagent Simple Temporal Problems. *Journal of Artificial Intelligence Research (JAIR),* To Appear.

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, SRI International. Available at http://yices.csl.sri.com/tool-paper.pdf.

Floyd, R. 1962. Shortest path. *Communications 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.

Oliehoek, F. A.; Witwicki, S. J.; and Kaelbling, L. P. 2012. Influence-based abstraction for multiagent systems. In *Proc. of AAAI-12*, 1422–1428.

Planken, L. R.; de Weerdt, M. M.; and Witteveen, C. 2010. Optimal temporal decoupling in multiagent systems. In *Proc. of AAMAS-10*, 789–796.

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.

Wilcox, R.; Nikolaidis, S.; and Shah, J. 2012. Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing. In *Proc. of RSS-12*, 56–63.