Distributed Algorithms for Solving the Multiagent Temporal Decoupling Problem

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

ABSTRACT

Scheduling agents can use the Multiagent Simple Temporal Problem (MaSTP) formulation to efficiently find and represent the complete set of alternative consistent joint schedules in a distributed and privacy-maintaining manner. However, continually revising this set of consistent joint schedules as new constraints arise may not be a viable option in environments where communication is uncertain, costly, or otherwise problematic. As an alternative, agents can find and represent a temporal decoupling in terms of locally independent sets of consistent schedules that, when combined, form a set of consistent joint schedules. Unlike current algorithms for calculating a temporal decoupling that require centralization of the problem representation, in this paper we present a new, provably correct, distributed algorithm for calculating a temporal decoupling. We prove that this algorithm has the same theoretical computational complexity as current state-of-the-art MaSTP solution algorithms, and empirically demonstrate that it is more efficient in practice. We also introduce and perform an empirical cost/benefit analysis of new techniques and heuristics for selecting a maximally flexible temporal decoupling.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Theory

Keywords

Multiagent Scheduling, Temporal Decoupling Problem

1. INTRODUCTION

A scheduling agent is often responsible for independently managing the scheduling constraints of its user, while also ensuring that its user's schedule coordinates with the schedules of other agents' users. In many scheduling environments, agents must also react to new constraints that arise over time, either due to volitional decisions by the agents (or their users) or due to the dynamics of the environment. Agent coordination is further challenged by desires for privacy and uncertain, costly, or otherwise problematic communication. Fortunately, scheduling agents can use the Multiagent Simple Temporal Problem (MaSTP) formulation to, in a distributed and efficient manner, find and represent sets of alternative consistent joint schedules to these types of complex, multiagent scheduling problems [3, 2].

As an example of this type of problem, suppose three student colleagues, Ann, Bill, and Chris, have each selected a tentative morning schedule (from 8:00 to noon) and have each tasked a personal computational scheduling agent with maintaining his/her schedule. Ann will have a 60 minute run with Bill before spending 90 to 120 minutes on a group project (after picking up deliverables that Chris will leave in the lab); Bill will have a 60 minute run with Ann before spending 60 to 180 minutes working on homework; and finally, Chris will work on the group project for 90-120 minutes and drop it off in the lab before attending a lecture from 10:00 to 12:00. This example is displayed graphically as a distance graph (explained in Section 2.1) in Figure 1(a).

One approach for solving this problem is to represent the set of *all* possible joint schedules that satisfy the constraints, as displayed in Figure 1(b). In this approach, if a new, non-volitional constraint arrives (e.g., Chris' bus is late), the agents can easily recover by simply eliminating inconsistent joint schedules from consideration. However, doing so may still require communication (e.g., Chris' agent should communicate that her late start will impact when Ann can start, and so on). In fact, this communication must continue (e.g., until Chris actually completes the project, Ann does not know when she can start), otherwise agents could make inconsistent, concurrent decisions. For example, if Ann decides she wants to run at 8:00, while Bill simultaneously decides he wants to run at 9:00 (both allowable possibilities), Ann and Bill's agents will inadvertently introduce an inconsistency. An alternative to this approach, displayed graphically in Figure 1(d), is for agents to simply select one joint schedule from the set of possible solutions. However, as soon as a new, non-volitional constraint arrives (e.g., Chris' bus arrives late by even a single minute), this exact, joint solution may no longer be valid. This, in turn, can require agents to regenerate a new solution *every* time a new constraint arrives, unless that new constraint is consistent with the selected schedule. Due to either a lack of robustness or lack of independence, neither of these two approaches is likely to perform well in time-critical, highly-dynamic environments.

Fortunately, there is a third approach that balances the

Cite as: Distributed Algorithms for Solving the Multiagent Temporal Decoupling Problem, James C. Boerkoel Jr. and Edmund H. Durfee, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 141-148.

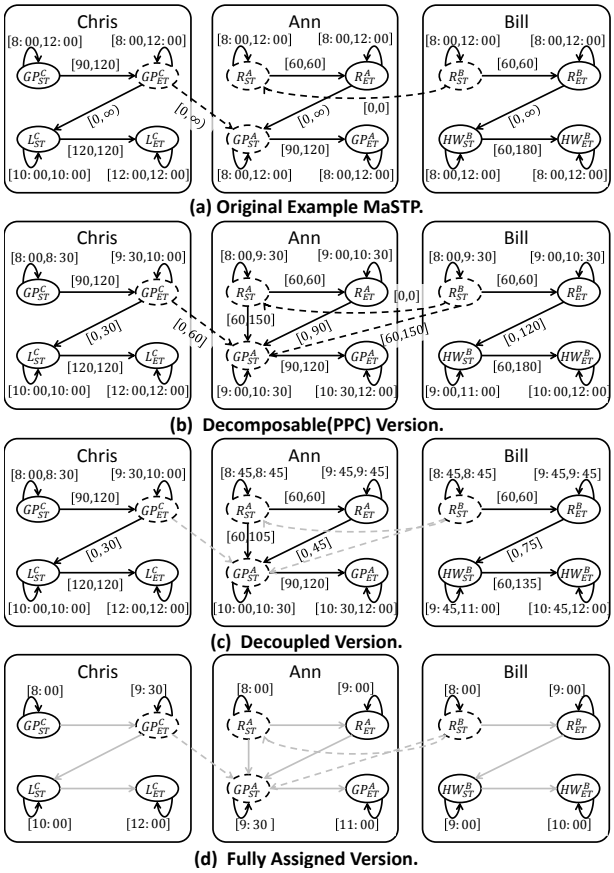

Figure 1: The distance graph corresponding to the (a) original, (b) decomposable, (c) decoupled, and (d) fully assigned, versions of our example MaSTP.

robustness of Figure 1(b) with the independence of Figure 1(d). Agents can find and maintain a temporal decoupling, which is composed of *independent* sets of locally consistent schedules that, when combined, form a set of consistent joint schedules [4]. An example of a temporal decoupling is displayed in Figure 1(c), where, for example, Chris' agent has agreed to complete the group project by 10:00 and Ann's agent has agreed to wait to begin work until after 10:00. Not only are agents' schedules no longer interdependent, but agents also still maintain sets of locally consistent schedules. Now when Chris' bus is late by a minute, Chris' agent can "absorb" this new constraint by independently updating its local set of schedules, without requiring any communication with any other agent. The advantage of this approach is that once agents establish a temporal decoupling, there is no need for further communication unless (or until) a new (series of) non-volitional constraint(s) render the chosen decoupling inconsistent. It is only if and when a temporal decoupling does become inconsistent (e.g., Chris' bus is more than a half hour late, violating her commitment to finish the project by 10:00) that agents must calculate a new temporal decoupling (perhaps establishing a new hand-off deadline of 10:15), and then once again independently react to newly-arriving constraints, repeating the process as necessary.

Unfortunately, the current temporal decoupling algorithms [4, 7] require centralizing the problem representation at some "coordinator" who sets the decoupling constraints for all. The computational, communication, and privacy costs associated with centralization may be unacceptable in multiagent planning and scheduling applications, such as military, health care, or disaster relief, where agents specify problems in a distributed fashion, expect some degree of privacy, and must provide unilateral, time-critical, and coordinated scheduling assistance. In this paper, we contribute new, *distributed* algorithms for calculating a temporal decoupling, prove the correctness, privacy implications, and runtime properties of these algorithms, and perform an empirical comparison that shows that these algorithms calculate a temporal decoupling that approaches the best centralized methods in terms of flexibility, but with less computational effort than current MaSTP solution algorithms.

2. PRELIMINARIES

In this section we provide definitions necessary for understanding our contributions, using and extending terminology from the literature.

2.1 Simple Temporal Problem

As defined in [3], the Simple Temporal Problem (STP), $\mathcal{S} = \langle V, C \rangle$, consists of a set of timepoint variables, V, and a set of temporal difference constraints, C. Each timepoint variable represents an event, and has an implicit, continuous numeric domain. 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. To exploit extant graphical algorithms and efficiently reason over the constraints of an STP, each STP is associated with a weighted, directed graph, $\mathcal{G} = \langle V, E \rangle$, called a *distance graph*. The set of vertices V is as defined before (each timepoint variable acts as a vertex in the distance graph) and E is a set of directed edges, where, for each constraint c_{ij} of the form $v_j - v_i \leq b_{ij}$, we construct a directed edge, e_{ij} from v_i to v_j with an initial weight $w_{ij} = b_{ij}$. As a graphical short-hand, each edge from v_i to v_j is assumed to be bi-directional, compactly capturing both edge weights with a single label, $[-w_{ji}, w_{ij}]$, where $v_j - v_i \in [-w_{ji}, w_{ij}]$ and w_{ij} is initialized to ∞ if there exists no corresponding constraint c_{ij} in P. All times (e.g. 'clock' times) can be expressed relative to a special ***zero*** timepoint variable, $z \in V$, that represents the "start of time". Bounds on the difference between v_i and z are expressed graphically as "unary" constraints specified over a timepoint variable v_i. Moreover, w_{zi} and w_{iz} then represent the earliest and latest times, respectively, that can be assigned to v_i, and thus implicitly define v_i's domain. In this paper, we will assume that z is always included in V and that, during the construction of $\mathcal{G}$, an edge e_{zi} is added from z to every other timepoint variable $v_i \in V$.

An STP is ***consistent*** if there exist no negative cycles in the corresponding distance graph. A consistent STP contains at least one ***solution***, which is an assignment of specific time values to timepoint variables that respects all constraints to form a ***schedule***. A ***decomposable*** STP represents the entire set of solutions by establishing the tightest bounds on timepoint variables such that: (1) no solutions are eliminated and (2) any assignment of a specific time to a timepoint variable that respects these bounds can be extended to a solution with a backtrack-free search using constraint propagation. ***Full-Path Consistency*** (FPC) works by establishing de-

composability of an STP instance in $O(|V|^3)$ by applying an all-pairs-shortest-path algorithm, such as Floyd-Warshall, to the distance graph to find the tightest possible path between every pair of timepoints, v_i and v_j, forming a fully-connected graph, where $\forall i, j, k, w_{ij} \leq w_{ik} + w_{kj}$. The resulting graph is then checked for consistency by validating that there are no negative cycles, that is, $\forall i \neq j$, ensuring $w_{ij} + w_{ji} \geq 0$ [3].

An alternative for checking STP consistency is to establish ***Directed Path Consistency*** (DPC) [3] on its distance graph. DPC *triangulates* the distance graph by visiting each timepoint, v_k, in some ***elimination order***, $o = (v_1, v_2, \ldots, v_n)$, tightening (and when necessary, adding) edges between each pair of its not yet eliminated neighboring timepoints, v_i, v_j (connected to v_k via an edge), using the rule $w_{ij} \leftarrow \min(w_{ij}, w_{ik} + w_{kj})$, and then "eliminating" that timepoint from further consideration. The quantity ω_o^* is the *induced graph width* relative to o, and is defined as the maximum, over all v_k, of the size of v_k's set of not yet eliminated neighbors at the time of its elimination. The edges added during this process, along with the existing edges, form a ***triangulated*** (also called chordal) ***graph*** — a graph whose largest non-bisected cycle is of size three. The complexity of DPC is $O(|V| \cdot \omega_o^{*2})$, but instead of establishing decomposability, it establishes the property a solution can be recovered from a DPC distance graph in a backtrack-free manner if variables are assigned in reverse elimination order. ***Partial Path Consistency*** (PPC) [1] is sufficient for establishing decomposability on an STP instance by calculating the tightest possible path for *only* the subset of edges that exists within a triangulated distance graph. As a result, PPC may establish decomposability much faster than FPC algorithms in practice ($O(|V| \cdot \omega_o^{*2}) \subseteq O(|V|^3)$) [8, 6]. The PPC representation of our example is displayed in Figure 1(b).

2.2 Multiagent Simple Temporal Problem

The Multiagent Simple Temporal Problem (MaSTP) is informally composed of n local STP subproblems, one for each of n agents, and a set of constraints C_X that establish relationships between the local subproblems of different agents. Our definition of the MaSTP improves on our original MaSTP specification [2]. An agent i's ***local*** STP subproblem is defined as $\mathcal{S}_L^i = \langle V_L^i, C_L^i \rangle$[1], where:

- V_L^i is defined as agent i's set of ***local variables***, which is composed of all timepoints *assignable* by agent i and also includes agent i's reference to z;
- C_L^i is defined as agent i's set of intra-agent or ***local constraints***, where a local constraint, $c_{ij} \in C_L^i$ is defined as a bound on the difference between two local variables, $v_j - v_i \leq b_{ij}$, where $v_i, v_j \in V_L^i$.

In Figure 1(a), the boxes labeled Chris, Ann, and Bill represent each person's respective local STP subproblem from our running example. Notice, the sets V_L^i partition the set of all non-reference timepoint variables and the sets C_L^i partition the set of all local constraints.

Moreover, C_X is the set of inter-agent or ***eXternal constraints***, where an external constraint is defined as a bound on the difference between two variables that are local to different agents, $v_i \in V_L^i$ and $v_j \in V_L^j$, where $i \neq j$. Further, V_X is defined as the set of ***external timepoint variables***, where a timepoint is external if it is involved in at least one external constraint. In Figure 1(a), external constraints and variables are denoted with dashed edges. It then follows that:

- C_X^i is agent i's set of external constraints that each involve exactly one of agent i's assignable timepoints;
- V_X^i is the set of timepoint variables known to agent i due their involvement in some constraint from C_X^i, but that are local to some other agent $j \neq i$.

More formally, then, an MaSTP, $\mathcal{M}$, is defined as the STP $\mathcal{M} = \langle V_\mathcal{M}, C_\mathcal{M} \rangle$ where $V_\mathcal{M} = \{\bigcup_i V_L^i\}$ and $C_\mathcal{M} = \{C_X \cup \bigcup_i C_L^i\}$. Note, the definition of the corresponding distance graph is defined as before, where the definition of agent i's local and external edges, E_L^i and E_X^i, follows analogously from the definition of C_L^i and C_X^i, respectively.

2.3 Multiagent Temporal Decoupling Problem

Given the previous definitions, we adapt the definition of temporal decoupling in [4] to apply to the MaSTP. Agents' ***local STP subproblems*** $\{\mathcal{S}_L^1, \mathcal{S}_L^2, \ldots, \mathcal{S}_L^n\}$ form a ***temporal decoupling*** of an MaSTP $\mathcal{M}$ if:

- $\{S_L^1, S_L^2, \ldots, S_L^n\}$ are consistent STPs; and
- Merging *any* locally consistent solutions to the problems in $\{\mathcal{S}_L^1, \mathcal{S}_L^2, \ldots, \mathcal{S}_L^n\}$ yields a solution to $\mathcal{M}$.

Alternatively, when $\{\mathcal{S}_L^1, \mathcal{S}_L^2, \ldots, \mathcal{S}_L^n\}$ form a temporal decoupling of $\mathcal{M}$, $\{\mathcal{S}_L^1, \mathcal{S}_L^2, \ldots, \mathcal{S}_L^n\}$ are said to be ***temporally independent***. The Multiagent Temporal Decoupling Problem (MaTDP), then, is defined as, for each agent i, finding a set of constraints C_Δ^i such that if $\mathcal{S}_{L+\Delta}^i = \langle V_L^i, C_N^i \cup C_\Delta^i \rangle$, then $\{\mathcal{S}_{L+\Delta}^1, \mathcal{S}_{L+\Delta}^2, \ldots, \mathcal{S}_{L+\Delta}^n\}$ is a temporal decoupling of MaSTP $\mathcal{M}$. Figure 1(c) represents a temporal decoupling of our example, where new unary decoupling constraints, in essence, replace all external edges (shown faded). A ***minimal decoupling*** is one where, if the bound of any decoupling constraint $c \in C_\Delta^i$ for some agent i is relaxed, then $\{\mathcal{S}_{L+\Delta}^1, \mathcal{S}_{L+\Delta}^2, \ldots, \mathcal{S}_{L+\Delta}^n\}$ is no longer a decoupling (e.g., Figure 1(c) is an example of a minimal decoupling whereas the decoupling in (d) is not minimal). The original TDP algorithm [4] executes on a centralized representation of the MaSTP and iterates between proposing new constraints to decouple agent subproblems with respect to a particular external constraint (until all external constraints have been decoupled) and reestablishing FPC on the corresponding global distance graph, so that subsequently proposed decoupling constraints are guaranteed to be consistent.

3. ALGORITHMS

In this section, we introduce new *distributed* algorithms for calculating a temporal decoupling and prove their correctness, computational complexity, and privacy properties.

3.1 A Distributed MaTDP Algorithm

The goal of our Multiagent Temporal Decoupling Problem (MaTDP) algorithm, presented as Algorithm 1, is to find a set of decoupling constraints C_Δ that render the external constraints C_X moot. Agents accomplish this goal by coordinating both to establish DPC and also to consistently assign external variables in reverse elimination order. First, we use DΔP3C-1 (an efficient, distributed DPC algorithm corresponding to lines 1-22 of the DΔP3C algorithm presented in [2]) to triangulate and propagate the constraints,

[1]Throughout this paper we will use superscripts to index agents and subscripts to index variables and edges.

Algorithm 1 Multiagent Temporal Decoupling Problem (MaTDP) Algorithm

Input: $\mathcal{G}^i$, agent i's known portion of the distance graph corresponding an MaSTP instance $\mathcal{M}$.
Output: C^i_Δ, agent i's decoupling constraints, and $\mathcal{G}^i$, agent i's PPC distance graph w.r.t. C^i_Δ.

1: $\mathcal{G}^i, o^i_L, o_X = (v_1, v_2, \ldots v_n) \leftarrow$D$\Delta$P3C-1($\mathcal{G}^i$)
2: Return INCONSISTENT if DΔP3C-1 does
3: $C^i_\Delta = \emptyset$
4: **for** $k = n \ldots 1$ such that $v_k \in V^i_L$ **do**
5: $\quad w^{DPC}_{zk} \leftarrow w_{zk}$, $w^{DPC}_{kz} \leftarrow w_{kz}$
6: $\quad$ **for** $j = n \ldots k+1$ such that $\exists e_{jk} \in E^i_L \cup E^i_X$ **do**
7: $\quad\quad$ **if** $e_{jk} \in E^i_X$ **then**
8: $\quad\quad\quad w_{zj}, w_{jz} \leftarrow$ Block until receive updates from $(Agent(v_j))$
9: $\quad\quad$ **end if**
10: $\quad\quad w_{zk} \leftarrow \min(w_{zk}, w_{zj} + w_{jk})$
11: $\quad\quad w_{kz} \leftarrow \min(w_{kz}, w_{kj} + w_{jz})$
12: $\quad$ **end for**
13: $\quad$ Assign v_k // tighten w_{zk},w_{kz} to ensure $w_{zk} + w_{kz} = 0$
14: $\quad$ Send w_{zk}, w_{kz} to each $Agent(v_j)$ s.t. $j < k, e_{jk} \in E^i_X$
15: $\quad C^i_\Delta \leftarrow C^i_\Delta \cup \{(z - v_k \in [-w_{zk}, w_{kz}])\}$
16: **end for**
17: **if**(RELAX)**then** $\mathcal{G}^i, C^i_\Delta \leftarrow$ MaTDR($\mathcal{G}^i, w^{DPC}$)
18: **return** P3C-2($\mathcal{G}^i_{L+\Delta}, o^i_L$), C^i_Δ

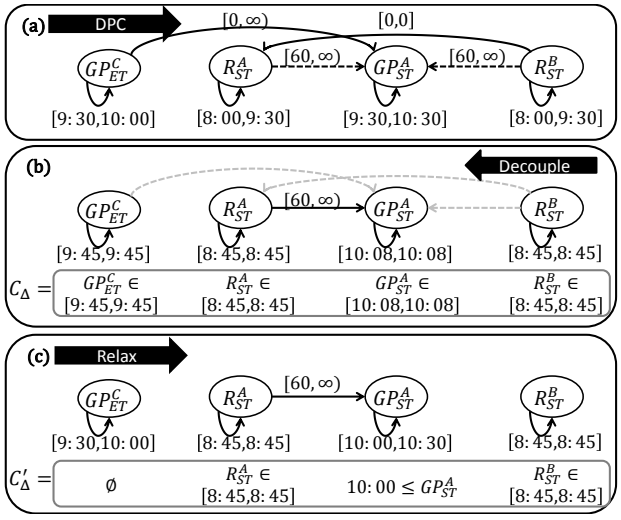

Figure 2: Applying the MaTDP algorithm to the example scheduling problem.

where external timepoints V_X are eliminated last. Figure 2(a) shows V_X after all other local variables have been eliminated. Notice that local constraints are reflected in the tighter domains. The external variables are eliminated in order, from left to right ($o_X = (GP^C_{ET}, R^A_{ST}, GP^A_{ST}, R^B_{ST})$), which introduces the new edges, shown with dotted lines, and their weights. If DΔP3C-1 propagates to an inconsistent graph, then our algorithm returns INCONSISTENT.

Otherwise, we initialize an empty C_Δ and then step through vertices in inverse elimination order, starting with R^B_{ST}. We skip over the inner loop (lines 6-12) because there are no vertices later in o_X than R^B_{ST}. In line 13, we use a heuristic that decouples by "assigning" the timepoint to the midway point between its upper and lower bounds. In this case we add the constraint that R^B_{ST} happens at 8:45 to C_Δ (line 15). In line 14, this is sent to Ann's agent, because R^B_{ST} shares external edges with Ann's timepoints. The next vertex is GP^A_{ST}. Note, Ann's agent would consider processing this variable right away, but the inner loop (lines 6-12) forces Ann's agent to wait for the message from Bill's agent. When it gets there, Ann's agent updates its edge weights accordingly (lines 10-11). In this case, given that GP^A_{ST} is at least 60 minutes after R^B_{ST}, GP^A_{ST}'s domain is tightened to [9:45, 10:30]. Then in line 13, Ann's agent chooses the decoupling point by splitting the difference, thus adding the constraint that G^A_{ST} occurs at 10:08. This same process is repeated until all timepoints in V_X have been assigned; the result is shown in Figure 2.

As mentioned, our default heuristic is to assign v_k to the midpoint of its path consistent domain (which corresponds to using the rules $w_{zk} \leftarrow w_{zk} - \frac{1}{2}(w_{zk} + w_{kz}); w_{kz} \leftarrow -w_{zk}$ for line 13). In general, however, assigning variables is more constraining than necessary. Fortunately, agents can optionally call a *relaxation* algorithm (introduced in Section 3.2) that replaces C_Δ with a set of *minimal* decoupling constraints. Later in this paper, we will explore and evaluate other assignment heuristics for line 13 (other than our default midpoint assignment procedure) that, when combined with the relaxation algorithm, could lead to less constraining decoupling constraints.

To avoid inconsistency due to concurrency, before calculating decoupling constraints for v_k, an agent blocks in line 8 until it receives the fresh, newly computed weights w_{zj}, w_{jz} from v_j's agent ($Agent(v_j)$, as sent in line 14) for each external edge $e_{jk} \in E^i_X$ where $j > k$. While this implies some sequentialization, it also allows for concurrency whenever variables do not share an external edge. For example, in Figure 2(b), because GP^C_{ET} and R^A_{ST} do not share an edge, after Ann's agent has assigned GP^A_{ST}, both Ann and Chris' agents can concurrently and independently update and assign R^A_{ST} and GP^C_{ET} respectively. Finally, each agent establishes PPC in response to its new decoupling constraints, by executing P3C-2 (which refers to the second phase of the single-agent P3C algorithm presented in [6] as Algorithm 3).

THEOREM 1. *The MaTDP algorithm has an overall time complexity of $O(|V|\omega^{*2}_o)$ and requires $O(|E_X|)$ messages.*

PROOF. The MaTDP algorithm calculates DPC and PPC in $O(|V|\omega^{*2}_o)$ time. Unary, decoupling constraints are calculated for each of $|V_X|$ external variables $v_k \in V_X$ (lines 4-16), after iterating over each of v_k's $O(\omega^*_o)$ neighbors (lines 6-12). Thus decoupling requires $O(|V|\omega^*_o) \subseteq O(|V|\omega^{*2}_o)$ time, and so MaTDP has an overall time complexity of $O(|V|\omega^{*2}_o)$. The MaTDP algorithm sends exactly one message for each external constraint in line 14, for a total of $O(|E_X|)$ messages. □

THEOREM 2. *The MaTDP algorithm is sound.*

PROOF. Lines 1-2 return INCONSISTENT whenever the input MaSTP $\mathcal{M}$ is not consistent. By contradiction, assume that there exists some external constraint c_{xy} with bound b_{xy} that is *not* satisfied when the decoupling constraints c_{xz} and c_{zy}, calculated by MaTDP with bounds b_{xz} and b_{zy} respectively, are (that is $b_{xz} + b_{zy} > b_{xy}$). WLOG, let $x < y$ in o_X. Notice, line 1 (DPC) implies $w_{xy} \leq b_{xy}$. Line 11 then implies $w_{xz} + w_{zy} \leq w_{xy} \leq b_{xy}$ (since inductively $w_{yz} + w_{zy} = 0$). Notice that after line 11, all other possible updates to w_{xz} that occur before c_{xz} is constructed in line 15 (e.g., in lines 10-11, 13) only tighten (never relax) w_{xz}, and

so $b_{xz} + b_{zy} \leq w_{xz} + w_{zy} \leq w_{xy} \leq b_{xy}$. However, this is a contradiction to our assumption that $b_{xz} + b_{zy} > b_{xy}$, so the decomposable distance graph and constraints C_Δ calculated by MaTDP form a temporal decoupling of $\mathcal{M}$. □

THEOREM 3. *The MaTDP algorithm is complete.*

PROOF (SKETCH). The basic intuition for this proof is provided by the fact that, in some sense, the MaTDP algorithm is simply a distributed version of the basic backtrack-free assignment procedure that can be applied to a DPC distance graph. We show that when we choose bounds for new, unary decoupling constraints for v_k (effectively in line 13), w_{zk}, w_{kz} are path consistent with respect to all other variables. This is because not only is the distance graph DPC, but also the updates in lines 10-11 guarantee that w_{zk}, w_{kz} are path consistent with respect to v_k for all $j > k$ (since each such path from v_j to v_k will be represented as an edge e_{jk} in the distance graph). So the only proactive edge tightening that occurs, which happens in line 13 and guarantees that $w_{zk} + w_{kz} = 0$, is done on path consistent edges and thus will never introduce a negative cycle (or empty domain). □

3.2 A Minimal Temporal Decoupling Relaxation Algorithm

The goal of the Multiagent Temporal Decoupling Relaxation (MaTDR) algorithm, presented as Algorithm 2, is to replace the set of decoupling constraints produced by the MaTDP algorithm, C_Δ, with a set of *minimal* decoupling constraints, C'_Δ. Recall that a minimal decoupling is one where, if the bound of any decoupling constraint $c \in C^i_\Delta$ for some agent i is relaxed, then $\{\mathcal{S}^1_{L+\Delta}, \mathcal{S}^2_{L+\Delta}, \ldots, \mathcal{S}^n_{L+\Delta}\}$ is no longer a decoupling. Clearly the temporal decoupling produced when running MaTDP using the default heuristic on our example problem, as shown in Figure 2(b), is not minimal. The basic idea of the MaTDR algorithm is to revisit each external timepoint v_k and, while holding the domains of all other external timepoint variables constant, relax the bounds of v_k's decoupling constraints as much as possible.

The MaTDR works in original o_X order, and thus starts with GP^C_{ET}. First, Chris' agent removes GP^C_{ET}'s decoupling constraints and restores GP^C_{ET}'s domain to [9:30,10:00] by updating the corresponding edge weights to their stored, DPC values (lines 1,3). Notice that lines 3-16 are similar to backwards execution of lines 6-12 in the MaTDP algorithm, except that a separate, "shadow" δ bound representation is used and updated only with respect to the original external *constraint bounds* (not edge weights). Also, in lines 17-24, a decoupling constraint is *only* constructed when the bound of the potential new constraint (e.g. δ_{kz}) is tighter than the already implied edge weight (e.g. when $\delta_{kz} < w_{kz}$). So in the case of GP^C_{ET}, the only constraint involving GP^C_{ET} is that it should occur before GP^A_{ST}. However, GP^A_{ST} is currently set to occur at 10:08 (δ=10:08), and since GP^C_{ET} is already constrained to occur before 10:00 (w =10:00), $\delta \not< w$, and so no decoupling constraints are added to the set C'_Δ for GP^C_{ET}. The next variable to consider is R^A_{ST}, whose domain relaxes back to [8:00,9:30]. However, since R^A_{ST} shares a synchronization constraint with R^B_{ST}, whose current domain is [8:45,8:45], Ann's agent will end up re-enforcing the original decoupling constraints of $R^A_{ST} \in$ [8:45,8:45]. On the other hand, after Ann's agent recovers GP^A_{ST}'s original DPC domain of [9:30,10:30], it then needs to ensure that GP^A_{ST} will always occur after GP^C_{ET}'s new domain of [9:30,10:00]. In this case, decoupling from GP^C_{ET} requires only a lower bound of 10:00 for GP^A_{ST} and results in a more flexible domain of [10:00,10:30]. The minimal decoupling constraints and corresponding distance graph that MaTDR calculates for the running example are presented in Figure 2(c) and Figure 1(c).

Algorithm 2 Multiagent Temporal Decoupling Relaxation (MaTDR)

Input: $\mathcal{G}^i$, and the DPC weights, $w^{DPC}_{zk}, w^{DPC}_{kz}$, for each $v_k \in V^i_X$
Output: C'^i_Δ, agent i's *minimal* decoupling constraints, and $\mathcal{G}^i$, agent i's PPC distance graph w.r.t. C'^i_Δ.

1: $C'^i_\Delta \leftarrow \emptyset$
2: **for** $k = 1 \ldots n$ such that $v_k \in V^i_L$ **do**
3: $\quad w_{zk} \leftarrow w^{DPC}_{zk}$, $w_{kz} \leftarrow w^{DPC}_{kz}$
4: $\quad \delta_{zk} \leftarrow \delta_{kz} \leftarrow \infty$
5: $\quad$ **for** $j = 1$ to n such that $\exists e_{jk} \in E^i_L \cup E_X$ **do**
6: $\quad\quad$ **if** $e_{jk} \in E^i_X$ **then**
7: $\quad\quad\quad$ **if** $j < k$ **then** $w_{zj}, w_{jz} \leftarrow$ Block receive from $Agent(v_j)$
8: $\quad\quad\quad$ **if** c_{jk} exists **then** $\delta_{zk} \leftarrow \min(\delta_{zk}, b_{jk} - w_{jz})$
9: $\quad\quad\quad$ **if** c_{kj} exists **then** $\delta_{kz} \leftarrow \min(\delta_{kz}, b_{kj} - w_{zj})$
10: $\quad\quad$ **else if** $j < k$ **then**
11: $\quad\quad\quad$ $w_{zk} \leftarrow \min(w_{zk}, w_{zj} + w_{jk})$
12: $\quad\quad\quad$ $w_{kz} \leftarrow \min(w_{kz}, w_{kj} + w_{jz})$
13: $\quad\quad$ **end if**
14: $\quad$ **end for**
15: $\quad$ **if** $\delta_{kz} < w_{kz}$ **then**
16: $\quad\quad$ $w_{kz} \leftarrow \delta_{kz}$
17: $\quad\quad$ $C'^i_\Delta \leftarrow C'^i_\Delta \cup \{(z - v_k \leq \delta_{kz})\}$
18: $\quad$ **end if**
19: $\quad$ **if** $\delta_{zk} < w_{zk}$ **then**
20: $\quad\quad$ $w_{zk} \leftarrow \delta_{zk}$
21: $\quad\quad$ $C'^i_\Delta \leftarrow C'^i_\Delta \cup \{(v_k - z \leq \delta_{zk})\}$
22: $\quad$ **end if**
23: $\quad$ Send w_{zk}, w_{kz} to each $Agent(v_j)$ s.t. $j > k, e_{jk} \in E^i_X$
24: **end for**
25: **return** $\mathcal{G}^i, C'^i_\Delta$

The type of update performed on timepoint v_k's actual domain edge weights, w_{zk} and w_{kz} (line 11-12), and shadow edge weights, δ_{zk} and δ_{kz} (line 8-9), differs based on whether the edge e_{jk} being considered in the inner loop is local or external respectively. For example, suppose v_k has a domain of [1:00,4:00], v_j has a domain of [2:00,2:30] (which already incorporates its new decoupling constraints, since v_j appears before v_k in o_X), and e_{jk} has the label [0,60] (e.g., $v_k - v_j \in [0, 60]$), which corresponds to bounds of original constraints. If e_{jk} is an external edge, the "shadow" domain of v_k would be updated by lines 8-9 to be [2:30,3:00]. Otherwise, if e_{jk} is a local edge, then the actual domain of v_k would be instead updated by lines 11-12 and result in the less restrictive domain [2:00, 3:30]. The difference between the two updates is that the updates in lines 8-9 guarantee that *all* possible assignments to the two variables will be consistent with respect to the external constraint, whereas the updates in lines 11-12 only guarantee that there exists *some* local assignment to the two variables that will be consistent. Finally, notice that if the domain of v_j had instead been assigned (e.g., to [2:30,2:30]), the updates in lines 8-9 and lines 11-12 would have resulted in the exact same update to the domain of v_k (e.g., [2:30,3:30]). Due to its similarity to the MaTDP algorithm, we forgo formally proving the correctness and computational complexity of the MaTDR sub-routine.

THEOREM 4. *The reference constraints calculated by the MaTDR algorithm form a minimal temporal decoupling of S.*

PROOF (SKETCH). The proof that C'_Δ form a temporal

decoupling is roughly analogous to the proof for Theorem 3.1. By contradiction, we show that if the bound b_{xz} of some decoupling constraint $c_{xz} \in C'_\Delta$ is relaxed by some small, positive value $\epsilon_{xz} > 0$, then C'_Δ is no longer a temporal decoupling. This is because lines 8-9 imply that there exists some y such that either, $b_{xz} = b_{xy} - b_{zy}$, and thus $b_{xz} + \epsilon_{xz} + b_{zy} > b_{xy}$ (and thus no longer a temporal decoupling), or that $b_{zy} = b_{xy} - (b_{xz} + \epsilon_{xz})$ (and so is either not a decoupling or requires us to also alter b_{zy} in order to maintain the temporal decoupling). □

3.3 Privacy

The natural distribution of the MaSTP representation affords a partitioning of the MaSTP into private and shared components [2]. Agent i's set of ***private variables***, V_P^i, is the subset of agent i's local variables that are involved in *no* external constraints, $V_P^i = V_L^i \setminus V_X$. Agent i's set of ***private constraints***, C_P^i, is the subset of agent i's local constraints C_L^i that include at least one of its private variables. Alternatively, the ***shared* STP**, $\mathcal{S}_S = \langle V_S, C_S \rangle$ is composed of the set of ***shared variables***, V_S, where $V_S = V_X \cup \{z\}$, and the set of ***shared constraints***, C_S, where shared constraints are defined exclusively over shared variables and by definition include all external constraints, $C_S = C_X \cup \left\{\bigcup_i C_N^i \setminus C_P^i\right\}$. In the example displayed in Figure 1, all shared variables and constraints are represented with dashed lines. $\mathcal{S}_S$ represents the maximum portion of the MaSTP that a set of colluding agents could infer, given only the joint MaSTP specification [2]. Hence, given the distribution of an MaSTP $\mathcal{M}$, if agent i executes a multiagent algorithm that does not reveal any of its private timepoints or constraints, it can be guaranteed that any agent $j \neq i$ will not be able to infer any private timepoint in V_P^i or private constraint in C_P^i by also executing the multiagent algorithm — at least not without requiring conjecture or ulterior (methods of inferring) information on the part of agent j. Additionally, it is not generally required that any agent knows or infers the entire shared STP. In our algorithms, agents attempt to minimize shared knowledge to increase efficiency.

COROLLARY 5. *The MaTDP and MaTDR algorithms never reveal any of agent i's private variables or private constraints (or edges) and hence maintain privacy over them.*

PROOF (SKETCH). Follows from proof of the properties of the MaSTP privacy partitioning, Theorem 1 in [2]. □

Together, the MaTDP and MaTDR algorithms calculate a minimal temporal decoupling for an MaSTP. In the next section, we empirically compare the performance of these algorithms with previous approaches.

4. EMPIRICAL EVALUATION

In the following subsections, we introduce the methodology we use to empirically evaluate the performance of our algorithm's computational effort and flexibility.

4.1 Methodology

To develop results comparable to those elsewhere in the literature, we model our experimental setup after [2] and [4], adapting the random problem generator described in [4] so that it generates MaSTP instances. Each problem instance has A agents each with start timepoints and end timepoints for 10 actions. Each action is constrained to occur within the time interval [0,600] relative to a global zero reference timepoint, z. Each activity's duration is constrained by a lower bound, lb, chosen uniformly from interval [0,60] and an upper bound chosen uniformly from the interval $[lb, lb + 60]$. In addition to these constraints, the generator adds 50 additional local constraints for each agent and N total external constraints. Each of these additional constraints, e_{ij}, is constrained by a bound chosen uniformly from the interval $[-B_{ji}, B_{ij}]$, where v_i and v_j are chosen, with replacement, with uniform probability. To confirm the significance of our results, we generate and evaluate the expected performance of our algorithms over 25 independently generated trials for each parameter setting. Since the novelty of our algorithms lies within the temporal decoupling aspects of the problem, we only generate consistent MaSTP problem instances to compare the computational effort of full applications of the various decoupling algorithms. We modeled a concurrently executing multiagent system by systematically sharing a 3 Ghz processor with 4 GB of RAM by interrupting each agent after it performed a single bound operation (either an update or evaluation) and a single communication (sending or receiving one message).

4.2 Evaluation of Computational Effort

In the first set of experiments, we empirically compared:

- **MaTDP+R** – our MaTDP algorithm with the MaTDR subroutine,
- **Cent. MaTDP+R** – a single agent that executes MaTDP+R on a centralized version of the problem,
- **D-P3C** – our implementation of the DΔP3C distributed algorithm for establishing PPC for an MaSTP (but not a decoupling) [2], and
- **TDP** — our implementation of the fastest variation (the RGB variation) of the (centralized) TDP algorithm as reported in [4].

For the TDP approach, we used the Floyd-Warshall algorithm to initially establish FPC and the incremental update described in [5] to maintain FPC as new constraint were posted. We evaluated approaches across two metrics. The non-concurrent computation (***NCC***) metric is the number computational cycles before all agents in our simulated multiagent environment have completed their execution of the algorithm [2]. The other metric we report in this section is the total number of messages exchanged by agents.

In the first experiment set (Figure 3), $A = \{1, 2, 4, 8, 16, 32\}$ and $N = 50 \cdot (A - 1)$. In the second experiment set (Figure 4), $A = 25$ and $N = \{0, 50, 100, 200, 400, 800, 1600, 3200\}$. The results shown in both figures demonstrate that our MaTDP+R algorithm clearly dominates the original TDP approach in terms of execution time, even when the MaTDP+R algorithm is executed in a centralized fashion. When compared to the centralized version of the MaTDP+R algorithm, the distributed version has a speedup (centralized computation/distributed computation) that varies between 19.4 and 24.7. This demonstrates that the structures of the generated problem instances support parallelism and that the distributed algorithm can exploit this structure to achieve significant amounts of parallelism.

Additionally, notice that the MaTDP+R algorithm dominates the DΔP3C algorithm in both computation and number

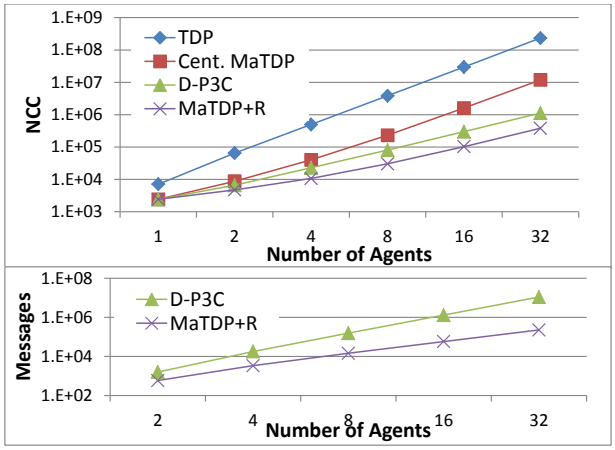

Figure 3: Computational effort as A grows.

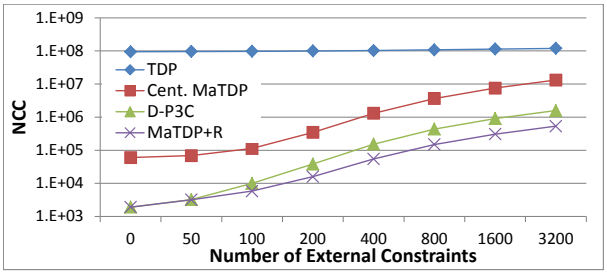

Figure 4: Computational effort as N increases.

of messages (which held true in both experiments, although messages are not displayed in Figure 4 due to space considerations), which means the MaTDP+R algorithm can calculate a temporal decoupling with less computational effort than the DΔP3C algorithm can calculate a decomposable, PPC representation of the MaSTP. This is due to the fact that, while the MaTDP+R is generally bound by the same runtime complexity as the DΔP3C, as argued in Theorem 1, the complexity of the actual decoupling procedure is less in practice, since the decoupling algorithm only calculates new bounds for reference edges, instead of calculating new bounds for *every* shared edge. This is important because if agents instead chose to try to maintain the complete set of consistent joint schedules (as represented by the decomposable, PPC output of DΔP3C), agents may likely perform *additional* computation and communication *every* time a new constraint arises, whereas the agents that calculate a temporal decoupling can perform all additional computation locally and independently, unless or until a new constraint arises that invalidates the temporal decoupling. The fact that MaTDP+R algorithm dominates the DΔP3C algorithm also implies that even if the original TDP algorithm were adapted to exploit the current state-of-the-art distributed PPC algorithm [2], our algorithm would still dominate the basic approach in terms of computational effort. Overall, we confirmed that we could exploit the structure of the MaSTP to calculate a temporal decoupling not only more efficiently than previous TDP approaches, but also in a distributed manner, avoiding centralization costs previously required, and exploiting parallelism to lead to impressive levels of speedup.

We next ask whether the quality of our MaTDP+R algorithm is competitive.

4.3 Evaluation of Flexibility

As mentioned earlier, one of the key properties of a decomposable MaSTP is that it can represent a set of consistent joint schedules, which in turn can be used as a hedge against scheduling uncertainty. In the following subsections we describe a metric for more generally quantifying the robustness of an MaSTP, and hence a temporal decoupling, in terms of a *flexibility* metric and perform an empirical evaluation of our algorithms with regards to flexibility.

4.3.1 Flexibility Metrics

Hunsberger introduced two metrics, ***flexibility*** ($Flex$) and conversely ***rigidity*** (Rig), that quantify the basic notion of robustness so that the quality of alternative temporal decouplings can be compared [4]. He defined the flexibility between a pair of two timepoints, v_i and v_j, as the sum $Flex(v_i, v_j) = B_{ij}+B_{ji}$ which is always positive for consistent MaSTPs. The rigidity of a pair of timepoints is defined as $Rig(v_i, v_j) = \frac{1}{1+Flex(v_i,v_j)}$, and the rigidity over an entire STP is the root mean square (RMS) value over the rigidity value of *all* pairs of timepoints:

$$Rig(\mathcal{S}) = \sqrt{\frac{2}{|V|(|V|+1)} \sum_{i<j} [Rig(v_i, v_j)]^2}.$$

This implies that $Rig(\mathcal{S}) \in [0, 1]$, where $Rig(\mathcal{S}) = 0$ when S has no constraints and $Rig(\mathcal{S}) = 1$ when S has a single solution [4]. Since $Rig(\mathcal{S})$ requires FPC to calculate, we only apply this metric as a post-processing evaluation technique by centralizing and establishing FPC on the temporal decouplings returned by our algorithms. There exists a centralized, polynomial time algorithm for calculating an optimal temporal decoupling [7], but it requires an evaluation metric that is a linear function of distance graph edge weights, which the aggregate rigidity function $R(\mathcal{S})$, unfortunately, is not.

4.3.2 Evaluation

In our second set of experiments, we compare the rigidity of the temporal decouplings calculated by:

- **Default** – a variant MaTDP algorithm that uses the the described, default heuristic, but without MaTDR,
- **Relaxation** – the default MaTDP with MaTDR,
- **Locality** – a variant of the MaTDP algorithm where, in line 13, agents heuristically bias how much they tighten w_{zk} relative to w_{kz} using information from applying the *full* DΔP3C algorithm in line 1 (no MaTDR),
- **All** – the MaTDP using both the locality heuristic and the MaTDR sub-routine,
- **Input** – the rigidity of the input MaSTP, and
- **TDP** – our implementation of Hunsberger's RLF variation of his TDP algorithm (where $r = 0.5$ and $\epsilon = 1.0$ which lead to a computational multiplier of approximately 9) that was reported to calculate the least rigid decoupling in [4].

In this experiment, $A = 25$ and $N = \{50, 200, 800\}$. Table 1 displays the rigidity of the temporal decoupling calculated by each approach. On average, as compared to the Default, the Relaxation approach decreases rigidity by 51.0% (while

Table 1: The rigidity values of various approaches.

	N=50	N=200	N=800
Input	0.418	0.549	0.729
All	0.508	0.699	0.878
Relaxation	0.496	0.699	0.886
Locality	0.621	0.842	0.988
Default	0.628	0.849	0.988
TDP	0.482	0.668	0.865

increasing computational effort by 30.2%), and the Locality approach decreases rigidity by 2.0% (while increasing computational effort by 146%). The Relaxation approach, which improves the output decoupling the most, offers the best return on investment. The locality heuristic, however, is very computationally expensive while providing no significant improvement in rigidity. We also explored combining these rigidity decreasing techniques, and while the increase in computational effort tended to be additive (the All approach increases effort by 172%), the decrease in rigidity did not. In fact, no heuristics or other combinations of techniques led to a statistically significant decrease in rigidity (as compared to the default, Relaxation approach) in the cases we investigated. The All approach decreased rigidity by only 49.9% in expectation.

The fact that the Relaxation approach alone decreases rigidity by more than any other combination of other approaches can be attributed to both the structure of an MaSTP and how rigidity is measured. First, the Relaxation improves the distribution of flexibility to the shared timepoints reactively, instead of proactively trying to guess good values. As the MaTDP algorithm tightens bounds, the general triangulated graph structure formed by the elimination order "branches out" the impact of this tightening. So if the first timepoint is assigned, this defers more flexibility to the subsequent timepoints that depend on the bounds of the first timepoint, of which there could be many. So by being proactive, other heuristics may steal flexibility from a greater number of timepoints, where as the MaTDR algorithm allows this flexibility to be recovered only after the (possibly many more) subsequent timepoints have set their bounds to maximize their local flexibility.

Notice from Table 1 that the TDP approach decreases the rigidity the most, representing on average a 20.6% decrease in rigidity as compared to the Relaxation approach. However, this additional reduction in rigidity comes at a significant computational cost — the TDP approach incurs, in expectation, over 10,000 *times* more computational effort than our Relaxation approach. While in some scheduling environments the costs of centralization (e.g. privacy) alone would invalidate this approach, in others the computational effort may be prohibitive if constraints arise faster than the centralized TDP algorithm can calculate a temporal decoupling. Further, in many scheduling problems, all temporal decouplings may be inherently rigid if, for example, many of the external constraints enforce synchronization (e.g. Ann's run start time), which requires fully assigning timepoints in order to decouple. Overall, the Relaxation approach, in expectation, outputs a *high-quality* temporal decoupling, approaching the quality (within 20.6%) of the state-of-the-art centralized approach [4], in a *distributed*, *privacy-maintaining* manner *faster* than the state-of-the-art MaSTP solution algorithms.

5. CONCLUSION

In this paper, we have presented a new, distributed algorithm that solves the MaTDP without incurring the costs of centralization like previous approaches. We have proven that the MaTDP algorithm is correct, and demonstrated both analytically and empirically that it calculates a temporal decoupling faster than previous approaches, exploiting sparse structure and parallelism when it exists. Additionally we have introduced the MaTDR algorithm for relaxing the bounds of existing decoupling constraints to form a *minimal* temporal decoupling, and empirically showed that this algorithm can decrease rigidity by upwards of 50% (within 20.6% of the state-of-the-art centralized approach) while increasing computational effort by as little as 20%. Overall, we have shown that the combination of the MaTDP and MaTDR algorithms calculates a temporal decoupling faster than state-of-the-art distributed MaSTP solution algorithms and the MaTDR algorithm reduces rigidity further than other heuristics we evaluated. In the future, we hope to evaluate the computational and communication costs of our algorithms in the context of a dynamic scheduling environment. We hope to extend the MaTDR algorithm to an anytime approach for recovering flexibility as new constraints arise and evaluate the computational effort in comparison with calculating a new temporal decoupling after an existing temporal decoupling becomes inconsistent. Additionally, we hope to develop additional flexibility metrics that can be evaluated in a distributed setting for heuristically guiding scheduling agents in dynamic scheduling environments.

6. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their comments and suggestions. This work was supported, in part, by the NSF under grants IIS-0534280 and IIS-0964512 and by the AFOSR under Contract No. FA9550-07-1-0262.

7. REFERENCES

[1] C. Bliek and D. Sam-Haroud. Path Consistency on Triangulated Constraint Graphs. In *Proc. of IJCAI-99*, pages 456–461, 1999.

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

[3] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. In *Knowledge representation*, volume 49, pages 61–95. The MIT Press, 1991.

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

[5] L. Planken. Incrementally solving the stp by enforcing partial path consistency. In *Proc. of PlanSIG-08*, pages 87–94, 2008.

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

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

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