Partitioning the Multiagent Simple Temporal Problem For Concurrency and Privacy (Extended Abstract)

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

ABSTRACT
In this paper, we describe the how a multiagent Simple Temporal Problem can be partitioned into private and shared components with important implications for privacy and concurrency.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *Multiagent Systems*

General Terms
Algorithms, Theory

Keywords
Simple Temporal Problem, Multiagent Scheduling

1. INTRODUCTION
A person often develops a tentative schedule, possibly implicitly, that organizes her daily goals in such a way that is likely to lead to their successful achievement. This task becomes more challenging as her goals rely more heavily on coordination with other individuals (joint meetings, projects, etc), since this requires reacting to their tentative schedules, which may fluctuate in ways beyond her control or knowledge. We discuss a way to partition scheduling problems into shared and private components, as well as approaches to exploit this partitioning, to allow scheduling agents to coordinate schedules on behalf of human users, without unnecessarily sacrificing the privacy users have over their personal schedules.

A Simple Temporal Problem (STP) is composed of a set of timepoint variables, V, and a set of temporal difference constraints, C, of the form $v_i - v_j \in [-B_{ji}, B_{ij}]$, where $v_i, v_j \in V$, and B_{ji} ($\geq v_j - v_i$) and B_{ij} ($\geq v_i - v_j$) are real numbers representing the minimum and maximum differences between v_i and v_j, respectively. The STP is a popular formulation for many scheduling and planning applications because an STP instance can be efficiently (in time polynomial in V) compiled to be *decomposable* [3]. A decomposable STP establishes the tightest bounds on all constraints such that (1) no valid schedule is eliminated from consideration, and (2) any assignment of a time to a timepoint variable that respects these bounds is guaranteed to be part of a feasible schedule. While the advantages of decomposable STP instances (representing the entire set of possible schedules) have long been realized in many centralized, or single agent settings [2], with few exceptions (e.g., [4,6]) little research has taken advantage of decomposability in the Multiagent STP (MaSTP).

Cite as: Partitioning the Multiagent Simple Temporal Problem for Concurrency and Privacy (Extended Abstract), Boerkoel, J. and Durfee, E., *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 1421-1422

Figure 1 shows an example of a MaSTP, displayed as a distance graph. Each timepoint variable is represented as a vertex. This STP includes start and end times (ST,ET) for a study session (SS), a take-home exam (EXAM), a group meeting (GM1), and a research paper (RP) for agent 1, and a programming assignment (PA), homework assignment (HW), group meeting (GM2), and a run (RUN) for agent 2. Each temporal difference constraint ($v_i - v_j \in [-B_{ji}, B_{ij}]$) is represented by an edge from v_j to v_i with label $[-B_{ji}, B_{ij}]$. For example, that the duration of the study session should last between 30 minutes and four hours is represented by a line from SS.ST to SS.ET with label [30,240]. In this example, solid edges represent minimum/maximum duration constraints, dashed edges represent precedence constraints, dotted edges represent an overall makespan constraint, and bold edges represent inter-agent synchronization constraints.

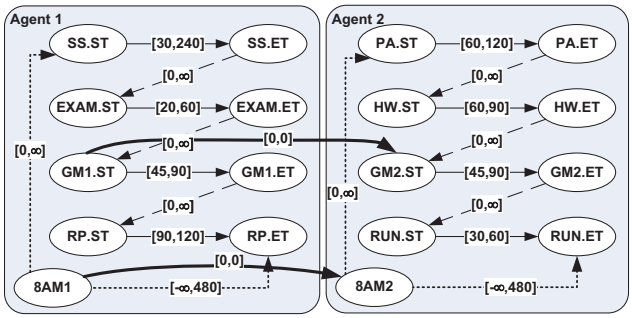

Figure 1. An example multiagent STP.

Traditionally, decomposability is established by applying an all-pairs-shortest-path algorithm to establish full path consistency [3]. However, more recent advances have shown that *partial* path consistency (PPC) sufficiently establishes decomposability over the *subset* of constraints that results from a triangulation of the distance graph [7]. As a result, PPC algorithms, such as ΔSTP [7] and P3C [5], exploit sparse constraint structure to find a decomposable STP instance more efficiently than fully path consistent algorithms, in expectation. Our work exploits the loosely-coupled structure that many MaSTPs (such as in Figure 1) have by intro-

ducing a partitioning of MaSTPs that allows individual agents to maintain privacy over, and concurrently apply PPC algorithms to, many of their timepoint variables.

2. PARTITIONING THE MaSTP

In a multiagent setting with n agents, we can partition the set of timepoint variables, V, into n+1 sets: V_P^i, for each agent i, is the set of i's *private* timepoint variables, which are involved in no inter-agent constraints, and V_S, the set of *shared* timepoint variables, which are involved in one or more inter-agent constraints. In Figure 1, V_S = {GM1.ST, GM2.ST, 8AM1, 8AM2} and V_P^1 and V_P^2 are the remaining timepoint variables in each agent's respective shaded box. Similarly, we can partition the set of constraints, C, into C_P^i, the set of constraints involving at least one variable in the set V_P^i, and C_S, the set of constraints in which both timepoint variables involved are in the set V_S.

PPC algorithms operate by establishing consistency on triangles of triangulated graphs (graphs where the largest non-bisected cycle is of size three). Conceptually, a graph is triangulated by the process of considering vertices and their adjacent edges, one-by-one, adding edges between parents of the vertex if no edge previously existed, and then eliminating that vertex from further consideration, until all vertices are eliminated. The order in which these vertices are eliminated is known as an elimination order and is typically heuristically chosen to try to minimize the number of resulting triangles. If we restrict the elimination order of the MaSTP so that private timepoint variables are eliminated *before* shared timepoint variables, each agent can triangulate its private timepoint variables (and subsequently can establish the consistency of the resulting "private" triangles) completely privately, asynchronously, and independently of other agents. This leaves the relatively small shared portion of the MaSTP that requires coordination to triangulate and establish consistency. It is worth noting that the private-before-shared restriction may interfere with the efficacy of a timepoint variable elimination ordering heuristic, though our preliminary studies indicate that the advantages of this restriction outweigh its costs.

Figure 2 shows this process in action (without the weights). Here (left), agent 1 triangulates by eliminating its private timepoint variables in the order indicated by the alphabetically labeled diamond tags. Meanwhile, agent 2 can asynchronously establish the same (or a different) elimination order over its private timepoint variables. Agents then coordinate to triangulate the shared timepoint variables (Figure 2 right). Notice that, given this combined triangulation, each agent is independently responsible for establishing consistency over seven triangles, while they only need to coordinate to establish consistency for two triangles. This reduces the non-concurrent triangle processing cost from 16 (in a centralized approach) to nine (in a distributed approach).

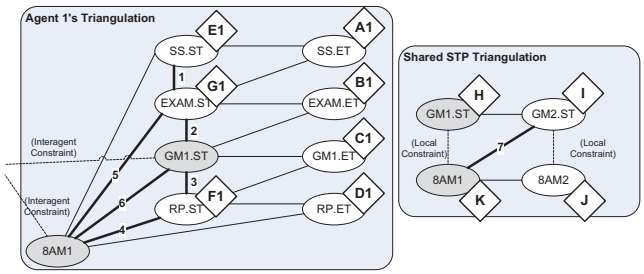

Figure 2. Agent 1's triangulation of its private timepoint variables (left) and the triangulation of the shared STP (right).

3. DISCUSSION

As shown in our running example, our MaSTP partitioning decreases the amount of nonconcurrent computation that agents must perform. Additionally, since agents only need to coordinate over the relatively smaller set of shared timepoint variables, the non-local view of the problem of any given agent is necessarily limited to the shared timepoint variables and the edges between them (Shared STP in Figure 2, right). Our partitioning does *not*, however, guarantee the privacy of a constraint, c, between two shared timepoints variables ($c \in C_S$), even if both timepoint variables belong to the same agent. As an example, in Figure 2, Agent 2 will know the earliest and latest time Agent 1 can start the group meeting (represented by Agent 1's local edge between GM1.ST and 8AM1) because it is part of the shared STP. However, Agent 2 will *not* know which, how many, or the durations of private commitments of Agent 1 (to study, take an exam, etc.) that influence these earliest and latest start times.

By limiting all necessary coordination to a relatively small, shared portion of a MaSTP, a scheduling agent can retain privacy and independence over its private timepoint variables. We have formalized these approaches into a suite of algorithms with varying degrees of decentralized computation and empirically tested the performance of our algorithms, quantifying the trade-off between the costs of the private-before-shared restriction on elimination ordering with the concurrent computation gains [1]. In the future, we plan to extend these approaches to adjust for dynamics during schedule execution, such as the addition or removal of (inter and intra-agent) constraints, events (timepoint variables), and agents.

4. ACKNOWLEDGMENTS

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

5. REFERENCES

[1] Boerkoel, J. and Durfee, E. 2010. A Comparison of Algorithms for Solving the Multiagent Simple Temporal Problem. To appear in *ICAPS 10*.

[2] Cesta, A., and Oddi, A. 1996. Gaining efficiency and flexibility in the simple temporal problem. In *TIME-96*, pp. 45-50.

[3] Dechter, R., Meiri, I., and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 29 (1-3) pp. 61-95.

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

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

[6] Smith, S. F., Gallagher, A., Zimmerman, T., Barbulescu, L., and Rubinstein, Z. 2007. Distributed management of flexible times schedules. In *AAMAS 07*, pp. 472-479.

[7] Xu, L., and Choueiry, B. Y. 2003. A new efficient algorithm for solving the simple temporal problem. In *TIME-ITCL 03*, pp. 210-220.