An Asynchronous Algorithm to Improve Scheduling Quality in the Multiagent Simple Temporal Problem

(Extended Abstract)

Vinicius De Antoni
Universidade Federal do Rio Grande do Sul
Porto Alegre, Brasil
vinicius@deantoni.com.br

Alvaro Moreira
Universidade Federal do Rio Grande do Sul
Porto Alegre, Brasil
afmoreira@inf.ufrgs.br

ABSTRACT

Even though modeling and solving multiagent scheduling problems seem completely understood and several algorithms can be found in the literature, one limitation still stands up: How to find a compatible time slot for an activity shared by many users without requiring the users themselves to spend time going through their calendar and choosing time slots until everybody agrees. This extended abstract introduces an algorithm called Asynchronous Time Finder (ATF) based on the Asynchronous Backtracking (ABT) that enables applications to find compatible times when scheduling shared activities among several users while requiring minimal user interaction.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Multiagent systems, MaSTP, Scheduling, DCSP

1. INTRODUCTION

Recently, with the advent of intelligent personal assistants capable of managing users' activities, the attention dedicated to multiagent scheduling problems and to algorithms for checking their consistency has increased among researchers of the field and the focus has been in algorithms for checking the (mutual) consistency of their schedules.

One approach for that is to gather all the agents' activities and constraints into a single STP [**?**] and solve it. Centralized approaches like this though, force users to reveal their full schedule (including information that they would prefer to keep private). Besides, relying on a central solver leads to performance issues as the number of agents grows.

In [**?**], the Multiagent Simple Temporal Problem (MaSTP) and the distributed algorithm DΔP3C for solving its consistency are presented in order to address the issues mentioned above. The MaSTP can be seem as a collection of local STPs, one for each agent in a multiagent system, with each local STP having a set of intra-agent and inter-agent time constraints for private and shared activities, respectively.

As far as we know one problem still remains: how to deal with the problem of finding a compatible time for every agent involved in a shared activity. Hence, the main contribution of this word is a novel algorithm, named Asynchronous Time Finder, (ATF) that finds a compatible time for a given activity in a distributed fashion.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

2. BACKGROUND

The *Simple Temporal Problem (STP)* was first introduced in [**?**], being defined as a set of binary constraints over these variables. Constraints can be written as $x_j - x_i \in [t, t']$ meaning that time difference between events x_j and x_i is constrained to be no less than t and no more than t'. If the time point variables x_j and x_i represent the *end* and *start* event of a given activity, the values t and t' are the minimum and maximum duration of the activity, respectively.

An STP is consistent if it has at least one solution. An STP solution is an assignment of specific time values to time point variables that respects every constraint. In [**?**] the authors present an algorithm for solving the STP in a centralized fashion called ΔP3C, which is a more efficient version of the algorithm P3C [**?**].

The *Multiagent Simple Temporal Problem (MaSTP)* can be informally defined as being composed of n local STP instances, each one assigned to one agent, and a set of constraints that allow relationships between local instances.

The MaSTP is also an STP and it could also be solved by one of the STP algorithms but by doing so, no advantage from the partitioning would be taken. Distributed algorithms such as the DΔP3C [**?**] verify the consistency of the MaSTP exploring the partitioning improving privacy, flexibility and performance.

The problem that remains is that whenever an instance is inconsistent, being a STP or a MaSTP, is up to the user to make changes to the schedule so it can be consistent again. The following scenario is adopted to explain the problem: Jane, Matt and Annie are college classmates sharing several classes during the week. They need to meet some time in the next few days to work on a paper they must deliver by the end of the month. They do not know about each other's activities other than the classes they share. Matt is responsible for finding a time for their meeting.

When using algorithms, like the DΔP3C, Matt would need to try and add the activity several times with different times until nobody's agenda is inconsistent, in other words the agent would have to run the algorithm several times with different inputs from the user (activities in different times) until one is consistent. This is impracticable in a multiagent environment where communication is not cheap and we want to avoid human interaction where it's not really needed.

3. ASYNCHRONOUS TIME FINDER

Our problem can be easily seen as DCSP, where the people involved in a shared activity (agents) have to reach an agreement (respect constraints) regarding the time of the activity (variables). Besides that, our problem come with two special requirements: we need a reliable way to determine when the algorithm is finished and we need to support the sets of available times from which the values are taken.

Our algorithm ATF is based on the Asynchronous Backtracking (ABT) [**?**], a well known algorithm for solving the DCSP, with the addition of functions to support extra requirements of our problem in particular.

The ATF algorithm is organized in three phases: during the *activity creation* phase, the owner agent sends a *create* message to every invitee with the activity identifier and its duration, and also a set containing all possible start times. Upon receiving the *create* message the invitee agent calculates the intersection of its available times and the set of possible times for the activity.

The *time negotiation* phase is fundamentally treated by the procedures of the ABT algorithm and it is led by the exchange of *ok?* and *nogood* messages between the invitees and the owner until everybody agrees with a time or someone gives up due to no being able to find a compatible time.

The *activity termination* phase consists of determining whether or not the negotiation was successful and informing all the invitees of the outcome. The owner agent sends a *snapshot?* message to all the other agents and waits for everyone to respond. Upon receiving a *snapshot?* message the invitee replies with its *current value* in a *snapshot!* message.

When the owner receives back a *snapshot!* it stores the agent identification paired with its current value in *snapshotview*. If everyone responds with the same value the owner then sends a *ok!* message to confirm the activity. If there is someone with a different value, the owner waits for a random period of time and re-sends the *snapshot?* messages. A maximum number of attempts can be set to avoid infinite waiting.

3.1 Evaluation and Results

In order to evaluate the computation effort and network usage of the ATF we use the random problem generator described in [**?**] to generate MaSTP instances. The instances are separated in seven different configurations each having 2, 4, 8, 16, 32, 64 and 128 agents. Each agent in each given configuration has from 10 to 20 private activities with duration varying from 1 to 4 hours separated from each other by 0 to 4 hours.

Our experiments consist in randomly selecting an agent as the owner of 10 new shared activities involving every agent and sending them the invitations. The experiment finishes when every agent has agreed to every activity start time and duration.

The Table 1 shows how many messages on average (2nd column) were exchanged between the agents (1st column) in order to reach an agreement for the 10 shared activities.

NO. OF AGENTS	AVG. NO. OF MESSAGES
2	55
4	163
8	378
16	757
32	1605
64	3286
128	6579

Table 1: Average no. of msgs. per configuration.

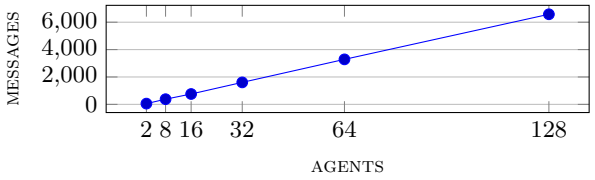

Figure 1: No. of messages grows linearly.

It is important to note that the complexity of the number of messages is $O(n)$, where n is the number of agents. In other words, the number of messages exchanged grows linearly as the number of agents increase (in Fig. 1). This is due to the fact that the agents involved in a shared activity exchange messages only with the owner of the activity and not with every other agent.

4. CONCLUSIONS

The purpose of ATF is to aid the management of shared activities within the MaSTP. The number of messages it generates in order to negotiate activities' time slots grows linearly with the number of agents, making it scalable and cost efficient. The main contribution of ATF is a reliable way to schedule shared activities with less user interaction.

5. REFERENCES

[1] J. C. Boerkoel and E. H. Durfee. A comparison of algorithms for solving the multiagent simple temporal problem. In *Proc. of the 20th International Conference on Automated Planning and Scheduling*, pages 26–33, Toronto, Canada, 2010.

[2] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence Magazine*, 49(1-3), May 1991.

[3] L. Hunsberger. Algorithms for a temporal decoupling problem in multi-agent planning. In *Proc. of the 18th National Conference on Artificial Intelligence*, pages 468–475, Sydney, Australia, 2002.

[4] L. R. Planken, M. M. de Weerdt, and R. van der Krogt. P3c: A new algorithm for the simple temporal problem. In *Proc. of the 18th International Conference on Automated Planning and Scheduling*, pages 256–263, Sydney, Australia, 2008.

[5] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673–685, 1998.