Coordinating Planning Agents for Moderately and Tightly-Coupled Tasks

J. Renze Steenhuisen and Cees Witteveen
Faculty of Electrical Engineering, Mathematics and Computer Science,
Delft University of Technology
P.O. Box 5031, 2600 GA Delft, The Netherlands

Abstract—In multi-agent planning, teams of agents need to jointly solve a complex task that consists of a set of interrelated tasks. A plan-coordination mechanism can be used to guarantee feasible joint plans by combining arbitrary locally constructed plans. Commonly, a set of tasks is considered that is constrained by precedence constraints, while often more complex constraints are required. We analysed whether the qualitative temporal relations can be represented using only precedence constraints. We found that, besides the precedence constraint, a synchronisation constraint is needed to represent all these qualitative temporal relations. We defined moderately and tightly-coupled tasks based on the absence or use of synchronisation constraints, and found that the related pre-planning coordination problems are both \( \Sigma^p_5 \)-complete.

Index Terms—Theory, algorithms, coordination, planning, qualitative temporal constraints

I. INTRODUCTION

Autonomous agents are being introduced in a wide variety of domains, because they promise to increase agility. Example domains where these systems are emerging are such diverse domains as multi-modal transportation [1], crisis response [2], firefighting with unmanned aerial vehicles and air traffic control [3]. In general, the tasks that need to be performed in these domains are interdependent, require more than one agent to execute them, and often require a careful task-planning process for each individual agent. Obviously, due to the task interdependencies, some form of coordination is needed between the individual agents to ensure that the individually constructed plans are jointly feasible. However, if we assume the participating agents to be self-interested and requiring planning autonomy, it is often not desirable or feasible to use approaches that either severely restrict the autonomy of the participating agents or require intensive negotiation and plan revision in order to reach a joint solution.

In these cases, coordination mechanisms in the sense of [4] provide a framework to ensure feasibility of any joint solution obtained by enabling the individual agents to choose their preferred way to solve their part of the task. However, this promise of agility does not come for free. A coordination mechanism has to (minimally) reduce the initial autonomy of the agents.\(^1\) The quality then of a coordination mechanism is dependent on both the severeness of the restrictions imposed and the overall performance quality it ensures.

\(^1\) In most cases, agents are willing to reduce their autonomy when they are compensated or get a guarantee of feasibility in return.

In task-based multi-agent systems, at least four different phases can be distinguished in or before which some form of coordination between the agents might be required. First, in the allocation phase, each task is assigned to some agent that is capable of completing it. Second, the order in which the tasks are to be executed is determined in the planning phase. Third, a time schedule is constructed for the tasks in the scheduling phase that is compatible with the plan. Finally, we have the execution phase in which the tasks are executed according to the constructed schedule.

Task instances are often classified with respect to whether or not they require coordination in these phases [5], [6]. First, a set \( T \) of tasks is called loosely coupled when the tasks occurring in \( T \) can be assigned to the agents independently. Here, each agent is able to construct an independent plan for its subset of tasks, and coordination reduces to solving the task allocation problem (i.e., who will do which tasks). Therefore, we do not need to bother about coordination for the planning or execution phase when dealing with loosely coupled sets of tasks. Examples of this category are tasks that are totally unrelated to each other, such as searching for casualties in different parts of a city.

Second, if the set \( T \) of tasks is partially ordered, \( T \) is said to be moderately coupled. Obviously, for these task instances, coordination is required before or during planning in order to ensure that the partial-order relation between tasks allocated to different agents is not broken. There is, however, no need to coordinate the scheduling phase if plan coordination can be guaranteed. Typical problems in this category are monitoring tasks, patient scheduling, and multi-modal transportation tasks.

Finally, if the set \( T \) of tasks is not only partially ordered, but also requires the satisfaction of constraints when executing the tasks, such as simultaneity constraints, the set of tasks is said to be tightly coupled. Here, not only plan coordination, but also schedule coordination is required. Examples of tightly-coupled tasks are (i) extinguishing a large fire that requires simultaneous action of multiple firefighters from different angles, and (ii) simultaneously lifting a patient onto a bed.

In our approach to designing coordination mechanisms, we do not address coordination issues that have to do with either task allocation or task scheduling. We assume the task allocation phase to have been completed and we concentrate exclusively on coordination issues that have to be dealt with before the agents start to plan and schedule the set of tasks.
allocated to them. The reason is that we want to deal with coordination mechanisms for self-interested and autonomous agents that are assumed (i) to require planning autonomy, and (ii) are not willing to adapt their previously constructed plan in order to ensure feasibility of the joint plan and the joint schedule. The coordination mechanisms we are aiming at will provide such planning autonomy and ensure that the joint plan composed of the individually constructed plans is feasible.

In previous work [7], [8], we have concentrated on designing coordination mechanisms for task-based planning domains. In particular, we have investigated the computational complexity of finding such coordination mechanisms for tasks that are moderately coupled (i.e., tasks where the only dependency relation is a partially-ordered precedence relation). The computational complexity turned out to be rather high, but we identified a polynomial-time approximation algorithm. For some domains, like the logistics domain, this algorithm offered an effective coordination mechanism for constructing multi-modal transportation plans [9] by autonomous agents.

In this paper, we extend this approach to coordination mechanisms for tasks that are constrained by qualitative temporal relations. Using a set of basic temporal constraints as distinguished by Allen [10], we first show that, in order to represent such relations in a task-based framework, it suffices to use a synchronisation relation \( \equiv \) between tasks besides a precedence relation \( \prec \). We show that a set of tasks with both these relations a tightly-coupled task instance. Surprisingly, it turns out that from a computational point of view designing coordination mechanisms for tightly-coupled tasks does not differ essentially from designing coordination mechanisms for moderately-coupled tasks: The decision variants of both problems are \( \Sigma^P_2 \)-complete. Although this problem is most probably intractable, we show that there exists a polynomial-time approximation algorithm that provides a (non-optimal) coordination mechanism for tightly-coupled tasks enabling agents to plan completely autonomous their part of the task while ensuring feasibility of the joint plan thus produced. From this joint plan, a joint schedule can be derived without forcing any of the agents to revise any part of their plan.

The remainder of this paper is structured as follows. In Section II, we describe a framework for coordinating moderately-coupled tasks. In Section III, we define tightly-coupled tasks and divide qualitative temporal constraints into two categories (i.e., requiring moderately and tightly-coupled task relations respectively). In Section IV, we describe a way to represent tightly-coupled tasks and analyse the problem of solving the related coordination problem. In Section V, we relate our approach to coordinating temporally constrained tasks to other work and give some directions for future research.

II. A FRAMEWORK FOR MODERATELY-COUPLED TASKS

In this section, a brief but necessary discussion of the existing framework for moderately-coupled tasks is given. We introduce the associated coordination problem and present some of the results obtained in previous work.

We consider a set of agents \( A = \{A_1, \ldots, A_n\} \) that have to complete a set of tasks \( T = \{t_1, \ldots, t_k\} \). We allow tasks to be interdependent by specifying a partially-ordered set of precedence constraints \( \prec \), where \( t \prec t' \) indicates that task \( t \) has to be completed before task \( t' \) might start. We assume that each task \( t \in T \) is assigned to at least one agent \( A_i \), and that, as a result, every agent \( A_i \) has obtained its disjoint subset \( T_i \subseteq T \) of tasks together with the local subset \( \prec_i \subseteq \prec \) of precedence constraints induced by \( T_i \). In order to execute its partially-ordered subset of tasks \( (T_i, \prec_i) \), agent \( A_i \) needs to construct a plan \( P_i \) for it. Such a plan can be simply conceived as a refinement of the partially-ordered set \( (T_i, \prec_i) \), i.e., \( P_i = (T_i, \prec_i^\ast) \) where \( \prec_i^\ast \) is a partially-ordered extension of \( \prec_i \).

If we assume the agents to plan independently, we should ensure that whatever feasible plans \( P_i \) are chosen by the individual agents \( A_i \), the simple joining of these plans constitutes a feasible global plan for the original set of tasks \( T \) satisfying all the precedence constraints (i.e., for every possible extension \( \prec_i^\ast \) of \( \prec_i \), we should require \( \bigcup_{i=1}^n \prec_i^\ast \cup \prec \) to constitute a partially-ordered extension of \( \prec \)). It is easy to see that cycles occurring in such an extension imply a deadlock when trying to execute the resulting joint plan.

Note that, due to the interdependencies between the tasks in \( T \), tasks assigned to different agents might be dependent upon each other and that, therefore, independent planning of the individual agents might lead to a deadlock as illustrated in Figure 1.

In Figure 1(a), a task instance is shown where agents \( A_1 \) and \( A_2 \) can plan \( t_4 \prec_1^* t_1 \) and \( t_2 \prec_2^* t_3 \) (see Figure 1(b)). But when these plans are joined, a cycle \( \langle t_1, t_2, t_3, t_4, t_1 \rangle \) is introduced. Such a cycle indicates an infeasible global plan since it implies \( t_1 \) to precede \( t_2 \), but also vice versa. Since such a combination of individual plans is possible, we call this instance uncoordinated.

We can now define the pre-planning coordination problem for tasks with precedence constraints as follows: Given a task instance \( \{T_i\}_{i=1}^n \), how to guarantee the feasibility of a global plan for all tasks respecting the precedence constraints when every agent \( A_i \) is allowed to autonomously construct a plan for its set \( T_i \) of subtasks, respecting only the set \( \prec_i \) of local constraints?

Plans are feasible when the tasks are partially ordered, which graphically boils down to not containing any directed cycles. Note that referring to Figure 1 this task instance becomes coordinated if we add, e.g., the constraint \( t_1 \prec t_3 \) to the set of local constraints of agent \( A_1 \). Since then the
only plan $A_1$ can produce is the plan where $t_1$ is executed before $t_3$, while agent $A_2$ has three options: executing $t_3$ and $t_4$ concurrently, $t_3$ before $t_4$, or vice-versa. Clearly, no combination of the individual plans creates a cycle.

It is not difficult to prove that, in general, for every uncoordinated task instance, there is a set $\Delta$ of local precedence constraints that when added to the existing set of constraints and taking the transitive closure, transforms the uncoordinated instance into a coordinated one [7]. In order to minimize the loss of freedom for the individual agents, we have to identify a minimum set of additional precedence constraints. We call this the coordination problem for moderately-coupled task.

**COORDINATING MODERATELY-COUPLED TASKS**

INSTANCE: A moderately-coupled task $\{\{T_i\}_{i=1}^n, \prec\}$ and a positive integer $K$.

QUESTION: Does there exist a coordination set $\Delta$ such that the task instance $\{\{T_i\}_{i=1}^n, (\prec \cup \Delta)^+\}$ is coordinated?²

In previous work [8], this problem has been studied extensively. It turns out that this problem is $\Sigma_2^p$-complete in general, and NP-complete when the number of agents is bounded by some constant. In addition, it was shown that this coordination problem is APX-hard and that a constant-ratio approximation algorithm is not likely to exist [11].

There is at least one planning domain, the logistics domain, where this framework applies. Here, we have a number of precedence constraints for multi-modal transportation tasks to be executed by autonomous transportation agents. Applying the above stated approach, we can show that there exists no polynomial algorithm for either a cooperative or a selfish multi-agent system solving the multi-modal transportation task with approximation ratio $\epsilon < 1.2$ unless some generally accepted conjecture in complexity theory is violated [9].

There is, however, a polynomial-time algorithm that finds an approximate but sufficient coordination set for distributed tasks with precedence constraints. This algorithm is based on the following idea: In constructing a local plan for $(T_i, \prec_i)$, each agent $A_i$ can safely start with the subset $T_i^1 \subseteq T_i$ of tasks that are not dependent upon other tasks, the so-called prerequisite-free tasks. Therefore, every agent sends its set of prerequisite-free tasks (according to $\prec_i$) to a blackboard managing the inter-agent precedence constraints. The blackboard checks which of these tasks is also globally prerequisite free and accepts the resulting set to the agent and removes all constraints pertaining to these tasks. Each agent now stores the set of tasks obtained from the blackboard in the set $T_i^1$ and removes this set from the original set $T$. As a result, other tasks in $T$ will become prerequisite free in the next round and each agent again selects its subset of prerequisite-free tasks, sends it to the blackboard and stores the result in $T_i^2$, etc. After at most $k = |T|$ iterations, all tasks have been selected exactly once as a prerequisite-free task. For each agent $A_i$, we now have a set of disjoint subsets $T_i^k$ for $k = 1, \ldots, |T|$, where $T_i^k$ denotes the (possibly empty) set of prerequisite-free tasks selected by agent in iteration $k$. The resulting coordination set $\Delta$ is constructed as follows: $\Delta$ is the union of the sets $\Delta_i$ for agent $A_i$. These sets $\Delta_i$ are obtained as follows:

1) Remove all empty subsets $T_i,k$ and let $m$ be the number of remaining subsets;
2) A precedence constraint $t \prec t'$ is added to $\Delta_i$ for every pair of tasks $t \in T_i^j$ and $t' \in T_i^{j+1}$, for every $j = 1, \ldots, m - 1$.

It can be shown that, although not creating a minimum coordination set, the resulting set $\Delta$ is sufficient for coordination [11]. Pseudo code for this distributed algorithm delivering the sets $T_i^k$ is given below.

**COORDINATION-BY-PARTITIONING$(T_i, \prec_i)$**

```
1   k ← 1
2   while $T_i \neq \emptyset$
3     do
4       Get the set $F_i$ of prerequisite-free tasks in $T_i$
5       Send $F_i$ to the blackboard $B$
6       Wait for the response $T_i^k \subseteq F_i$ from $B$
7       $T_i ← T_i \setminus T_i^k$
8       $k ← k + 1$
9   return $(T_1^1, \ldots, T_n^k)$
```

Previously, we showed that this algorithm is a 1.25-approximation algorithm for coordinating problem instances in the logistics domain [12], where—as stated above—1.25-approximation algorithms are the current lower bound.

### III. TASKS AND TEMPORAL CONSTRAINTS

The framework discussed above has been used to investigate pre-planning coordination problems for moderately-coupled tasks, i.e., sets of tasks with precedence constraints.

Many planning domains, however, such as airport planning, manufacturing, and supply-chain management require the ability to use temporal relations to constrain the execution of a set of tasks, especially with respect to the time intervals certain tasks should be executed. Seven³ of such qualitative temporal constraints have been identified for constraining time intervals in Allen’s time interval algebra [10]. As we will briefly show, all these qualitative temporal constraints can be represented in a task-based framework using precedence constraints and an additional type of constraints, the synchronisation constraints.

#### A. Representing Allen’s temporal relations in a task-based framework

Allen identifies seven temporal relations between time intervals: before, overlaps, during, meets, starts, finishes and equals [10]. Because these relations are defined on intervals instead of tasks, we assume that every task $t$ can be represented as an interval $[t_s, t_e]$ with $t_s < t_e$ representing the starting point of task $t$ and the end of start $t$.

We recall that the overlaps-relation constrains two tasks to partially overlap (like running in steeple chase), and that

²For any relation $\rho$, the transitive closure of $\rho$ is denoted by $\rho^+$. ³Neglecting the converse of each of these relations.
the during-relation constrains a task to be executed between the starting and ending of another task’s execution. Therefore, this new representation for tasks enables us to translate three of the qualitative temporal constraints: before, overlaps, and during as follows: We start by splitting the tasks \( t_1 \) and \( t_2 \) into time intervals with end points \( t_{1,s}, t_{1,e} \) and \( t_{2,s}, t_{2,e} \), respectively, and constrained by \( t_{1,s} \prec t_{1,e} \) and \( t_{2,s} \prec t_{2,e} \). Now, we can rewrite \( t_1 \) before \( t_2 \) as \( t_{1,s} \prec t_{1,e} \prec t_{2,s} \prec t_{2,e} \) (see Figure 2(a)), \( t_1 \) overlaps \( t_2 \) as \( t_{1,s} \prec t_{2,s} \prec t_{1,e} \prec t_{2,e} \) (see Figure 2(b)), and change \( t_1 \) during \( t_2 \) into \( t_{1,s} \prec t_{2,s} \prec t_{1,e} \prec t_{2,e} \) (see Figure 2(c)). Note that these three constraints have in common that the endpoints of the time intervals are not allowed to coincide (i.e., for every pair of time points a precedence relation is defined). We conclude that the basic task framework with precedence constraints suffices to represent these three temporal relations.

Contrary, to the previous three constraints, using the remaining constraints meets, starts, finishes, and equals end points of tasks need to coincide. Such synchronisation cannot be represented by precedence constraints between the end points. Therefore, we have to introduce the notion of synchronised events colliding two synchronised time points \( t_1, t_2 \) into one \( t_{1/2} \) and then constrain the tasks with precedence constraints. In this way we can rewrite \( t_1 \) meets \( t_2 \) as \( t_{1,s} \prec t_{1,e/2,s} \prec t_{2,e} \) (see Figure 2(d)), \( t_1 \) starts \( t_2 \) as \( t_{1,s/2,s} \prec t_{1,e} \prec t_{2,e} \) (see Figure 2(e)), \( t_1 \) finishes \( t_2 \) as \( t_{2,s} \prec t_{1,s} \prec t_{1,e/2,e} \) (see Figure 2(f)), and \( t_1 \) equals \( t_2 \) as \( t_{1,s/2,s} \prec t_{1,e/2,e} \) (see Figure 2(g)).

**Remark 1:** Notice that in \( t_1 \) starts \( t_2 \) it is correct to include \( t_{1,e} \prec t_{2,e} \) as translated from Allen’s formalism [10]. However, we could leave out this additional precedence constraint between the end points without any problem. In this way, we are actually representing, in terms of Allen, the disjunction relation \( t_1 \) starts \( t_2 \lor t_2 \) starts \( t_1 \). A similar remark can be made for the finishes constraint.

Summarising, it turns out to be possible to represent all basic qualitative temporal constraints as used by Allen into a task-based framework if we are prepared to introduce the notion of a synchronised task or event. Tasks that are subject to temporal relations and can be represented by precedence constraints are moderately-coupled tasks, while tasks that use temporal relations requiring synchronisation are tightly-coupled tasks. This corresponds to the distinction made in Section I, which was based on whether or not coordination was needed during plan execution. This exactly is covered by the synchronisation constraints, because the synchronised tasks not only need to have the same place in the partial order but need to be scheduled and executed synchronously.

We will now extend our coordination framework to include synchronisation constraints in order to be able to represent qualitative temporal constraints.

IV. COORDINATING THE PLANNING OF TIGHTLY-COUPLED TASKS

We want to extend the task-based framework we discussed above to deal with tightly-coupled tasks. To represent such instances we need to include synchronisation information.

Therefore, we define a tightly-coupled task as a triple \((\{T_i\}_{i=1}^n, \prec, \equiv)\), where \( \prec \subseteq (T \times T) \) is a precedence relation and \( \equiv \subseteq (T \times T) \) represents the synchronisation relation (i.e., \( t \equiv t' \) holds if \( t \) and \( t' \) are tasks that have to be synchronised).

We consider \( \equiv \) to be an equivalence relation on \( T \) and \( \prec \) to be a partial order on \( T \). The combination of \( \prec \) and \( \equiv \) satisfies the following natural properties:

1) \((\prec \circ \equiv) \subseteq \prec \land (\equiv \circ \prec) \subseteq \prec\), i.e., \((t \prec t') \land (t' \equiv t'') \implies t \prec t''\) and \((t \equiv t' \land t' \prec t'') \implies t \prec t'\).
2) \(\prec \) and \(\equiv\) are orthogonal relations, i.e., \(\prec \cap \equiv = \emptyset\).

Tightly-coupled tasks are said to be coordinated if the combination of individually feasible plans always results in a joint plan that respects both the precedence constraints and the synchronisation constraints.

Therefore, we say that an instance \((\{T_i\}_{i=1}^n, \prec, \equiv)\) is coordinated if for every set \(\{P_i\}_{i=1}^n\) of plans that (locally) respect \(\prec_i\) and \(\equiv_i\), there exists a function \(s: T \rightarrow \mathbb{Z}^+\) assigning a time point \(s(t) \in \mathbb{Z}^+\) to each task \(t \in T\) such that for each \(i = 1, \ldots, n\), and for all \(t, t' \in T\), (i) \(t \prec t'\) implies \(s(t) < s(t')\), (ii) \(t \equiv t'\) implies \(s(t) = s(t')\), (iii) \(t \prec t'\) implies \(s(t) < s(t')\), and (iv) \(t \equiv t'\) implies \(s(t) = s(t')\).

That is, any plan \(P_i\) is a tightly-coupled task \((T_i, \prec_i, \equiv_i)\) where \(\prec_i\) is a partial order extending \(\prec_i\) and \(\equiv_i\) is an equivalence relation extending \(\equiv_i\).
Think of \( s(\cdot) \) as a feasible schedule for the total set of tasks that satisfies all the precedence and synchronisation constraints. The existence of such a schedule \( s(\cdot) \) ensures both the feasibility of the joint plan and the existence of a feasible schedule that can be obtained without revision of any individual plan.

The presence of synchronisation constraints causes some additional problems above coordination of moderately-coupled tasks. In Figure 3, an example is depicted of a tightly-coupled task with two agents having each four tasks. There are two pairs of synchronised tasks: \((t_1, t_3)\) and \((t_3, t_7)\). Notice that independent planning might violate the synchronisation constraints even if no inter-agent cycles are created. In Figure 3(b), task \( t_1 \) is planned before task \( t_3 \), while \( t_7 \) is planned before \( t_5 \). This clearly violates the given synchronisation constraints \( t_1 \equiv t_5 \) and \( t_3 \equiv t_7 \), since every function \( s \) satisfying the precedence constraints will imply \( s(t_1) < s(t_3) \) and \( s(t_7) < s(t_5) \). Then trying to satisfy the synchronisation constraint \( t_1 \equiv t_5 \) requires \( s(t_1) = s(t_5) \), implying \( s(t_7) < s(t_3) \) thereby violating \( t_7 \equiv t_3 \).

**Remark 2:** Notice that, in general, violations of the synchronisation constraints in a tightly-coupled task can easily be detected if we consider a pair of synchronised tasks as a single task. Given an instance \( ((T_i)_{i=1}^n, \prec, \equiv) \) of a tightly-coupled task-planning instance, let us define the associated moderately-coupled task-planning instance by the tuple \( ((T_i)_{|i|=1}^n, \prec, \equiv) \), where \( ((T_i)_{|i|=1}^n, \prec, \equiv) \equiv \) consists of the equivalence classes modulo \( \equiv \) of the set of original tasks and \( \prec, \equiv \) is the associated precedence relation defined on \( T/\equiv \) where the representative \( t \) of each equivalence class \( [t]_{\equiv} \) inherits all the precedence relations occurring in \( \prec \).

For example, in Figure 4, the moderately-coupled task associated with the tightly-coupled instance depicted in Figure 3(b) is given. In this case, the synchronised pair \((t_1, t_3)\) is represented by \( t_1 \) and the the synchronised pair \((t_3, t_7)\) by \( t_7 \). It is clear that synchronisation constraints are violated by the occurrence of an intra-agent cycle in the plan of agent \( A_1 \).

The example we discussed above suggests that in general, a tightly-coupled task \( ((T_i)_{|i|=1}^n, \prec, \equiv) \) should satisfy two conditions in order to be coordinated:

1) Tasks in shared synchronisation constraints should be ordered. Whenever there exist tasks \( t_{i_1}, t_{i_2} \in T_i \) and \( t_{j_1}, t_{j_2} \in T_j \) such that \( t_{i_1} \equiv t_{j_1} \) and \( t_{i_2} \equiv t_{j_2} \) then either \( t_{i_1} < t_{i_2} \) and \( t_{j_1} < t_{j_2} \) or \( t_{i_2} < t_{i_1} \) and \( t_{j_2} < t_{j_1} \). Clearly, if this condition is not satisfied, the order between \( t_{i_1} \) and \( t_{i_2} \) can be planned opposite to the order \( t_{j_1} \) and \( t_{j_2} \) might be planned by the other agent. Note that if this condition is satisfied, the orders between tasks in the synchronisation relation are compatible.

2) If we abstract from the synchronisation relation, the resulting moderately-coupled task should be coordinated. This guarantees that the joint plan with respect to the precedence relation is feasible.

These conditions together enable us to reduce the test for being (tightly) coordinated to the test for being moderately coordinated as expressed in the following proposition.

**Proposition 1:** A tightly-coupled task \( ((T_i)_{|i|=1}^n, \prec, \equiv) \) is (tightly) coordinated if the following conditions are satisfied:

1) whenever there exist tasks \( t_{i_1}, t_{i_2} \in T_i \) and \( t_{j_1}, t_{j_2} \in T_j \) such that \( t_{i_1} \equiv t_{j_1} \) and \( t_{i_2} \equiv t_{j_2} \) then either \( t_{i_1} < t_{i_2} \) and \( t_{j_1} < t_{j_2} \) or \( t_{i_2} < t_{i_1} \) and \( t_{j_2} < t_{j_1} \) holds;

2) the moderately-coupled task \( ((T_i)_{|i|=1}^n, \prec) \) is coordinated. Here, \( (T_i)_{|i|=1}^n \) consists of the representatives (in \( T_i \)) of the equivalence classes of \( \equiv_i \).

**Proof:** See Appendix A for a sketch of the proof.

Using the results we have obtained for establishing the computational complexity of designing coordination mechanisms for moderately-coupled tasks, this proposition can be used directly to establish the complexity of designing tightly-coupled tasks. Let us consider the following decision-variant of the problem of coordinating tightly-coupled tasks.

**COORDINATING TIGHTLY-COUPLED TASKS**

**INSTANCE:** A tightly-coupled task \( ((T_i)_{|i|=1}^n, \prec, \equiv) \) and a positive integer \( K \).

**QUESTION:** Does there exist a coordination set \( \Delta \) with \( |\Delta| \leq K \) such that the instance \( ((T_i)_{|i|=1}^n, (\prec \cup \Delta)^+, \equiv) \) is coordinated?
To establish the complexity of this problem, we know that it should be $\Sigma^p_2$-hard, because the (contained) coordination problem for moderately-coupled tasks is known to be $\Sigma^p_2$-complete [7]. Moreover, if we guess a coordination set $\Delta$, the complexity of verifying coordination of the extended instance $\{\{T_i\}_{i=1}^n, \prec \cup \Delta, \equiv\}$ can be accomplished by verifying the conditions stated in Proposition 1. Obviously, verification of the first condition can be done in polynomial time, while we have shown in [7] that verifying the second condition can be done in nondeterministic polynomial time.

**Proposition 2:** Coordinating tightly-coupled tasks is $\Sigma^p_2$-complete.

Surprisingly, viewing the problem computationally, coordinating moderately-coupled and tightly-coupled problems do not differ significantly.

A. Approximation algorithm for solving tightly-coupled tasks

The problem of coordinating tightly-coupled tasks is APX-hard, because APX-hard problem of coordinating moderately-coupled tasks is contained. However, we will show that with some minor modifications, the partitioning algorithm (see Section II) can be used also to solve a tightly-coupled task.

First of all, observe that whenever two tasks $t \in T_i$ and $t' \in T_j$ are synchronised, they will be selected in exactly the same round $k$ of the algorithm used by agent $A_i$ and by agent $A_j$. This is a trivial consequence of the fact that $t \equiv t'$ implies that $\{x \mid x \prec t\} = \{y \mid y \prec t'\}$, hence they have the same set of predecessors and, therefore, will be elected as prerequisite free in exactly the same round $k$. This implies that for local synchronisation constraints $t \equiv t'$, the existing algorithm does not need to be adapted, since no additional precedence constraints between these tasks will be added.

Therefore, the only problem to solve is to prohibit that two pairs of synchronised tasks $t_i, t_j, t_k, t_l$ such that $t_i \equiv t_j$ and $t_k \equiv t_l$ are selected in the same round $k$, while $t_i, t_k \in T_i$ and $t_j, t_l \in T_j$ for some $i \neq j$. For then the first condition of Proposition 1 could be violated by independently constructed plans by agent $A_i$ and $A_j$.

One way then to adapt this algorithm to obtain a solution is to adapt the blackboard in such a way that it is aware of both precedence relations and synchronisation constraints. Upon receiving the locally prerequisite-free sets $F_i$ from the agents $A_i$, the blackboard selects maximal subsets $T_i^k$ from these sets $F_i$ such that (i) all tasks occurring in $T_i^k$ are globally prerequisite-free and (ii) these sets satisfy the property that whenever there are tasks $t_i \equiv t_j$ and $t_k \equiv t_l$ such that $t_i, t_k \in F_i$ and $t_j, t_l \in F_j$, only one of these pairs occurs in the resulting sets $T_i^k$ sent back to the agents. As a result, all such shared tasks will be totally ordered, since each of them will appear in a set returned in a new round $k$ and tasks occurring in round $k$ will precede any task occurring in a round $k' > k$ due to the additional ordering constraints.

Let us give a final example to solve the coordination problem for the task instance depicted in Figure 3(a) using this approximation algorithm. In the first round, agent $A_1$ will send $F_1^1 = \{t_1, t_3\}$, while agent $A_2$ will send $F_2^1 = \{t_5, t_7\}$ to the blackboard. The blackboard checks their prerequisite freeness and then detects that the first condition of Proposition 1 is violated. It selects $t_1, t_5$ as the single pair of synchronised constraints and sends back the sets $T_1^1 = \{t_1\}$ and $T_2^1 = \{t_5\}$. The agents adapt their set of local tasks and select again a set of prerequisite-free tasks: $F_1^2 = \{t_3, t_2\}$ and $F_2^2 = \{t_7, t_8\}$. Both sets are checked by the blackboard and returned: $T_1^2 = F_1^2$ and $T_2^2 = F_2^2$. The agents remove these tasks from the set of to be selected tasks and now $F_1^3 = \{t_4\}$ and $F_2^3 = \{t_8\}$. Both are prerequisite-free and returned to the agents. After removal of these tasks both sets of local tasks are empty. As a result of ordering the sets $T_i^k$, the following constraints are added to $T_1$: $t_1 \prec t_3$ and $t_2 \prec t_4$. Likewise, $t_5 \prec t_7$ and $t_6 \prec t_8$ are added to $T_2$. The reader might check that indeed the resulting task instance is coordinated.

V. Discussion and Conclusions

We introduced a framework for specifying task instances with qualitative temporal constraints, we analysed the coordination problems for these task instances, we analysed the computational complexity of this problem, and we provided a fast approximation algorithm to find a coordination set.

As we have remarked before, coordination is needed to guarantee that using local planning autonomy does not cause conflicts to the global goal. In coordination, a distinction can be made between *pre*, *interleaved*, and *post*-planning coordination. Both interleaved and post-planning coordination assume communication to be available during and after planning and thus during execution. Since it is not unlikely that communication is lost in a crisis situation, or that agents are unwilling to revise their plans, interleaved and post-planning coordination is not always applicable to all domains. Therefore, we used a pre-planning approach to coordination, while other approaches might be relevant to future extensions of this work. For example, although post-planning coordination techniques sometimes do not suffice, they might be used as an additional tool in case communication is available. In addition, techniques applied in post-planning coordination constitutes a source of inspiration for future research in pre-planning coordination. For instance, the Partial-Order Causal-Link (POCL) framework [13], that is used in post-planning coordination, allows symmetric concurrency and non-concurrency relations to constrain tasks. These constraints are very useful for describing, for instance, disaster plans. Note that the concurrency relation is more or less comparable to the synchronisation relation in our framework (see Section IV).

With respect to the use of temporal constraints, existing work on plan coordination has been limited. In schedule-coordination approaches often use a framework that allows more quantitative temporal information to be used. For example, in the Temporal Constrained Satisfaction Problem (TCSP) [14] framework, it is possible to represent arbitrary intervals of temporal distances between time points. Some work on distributed autonomous scheduling in this domain has been proposed by Hunsberger [15] introducing a Temporal
Decoupling Problem on STPs, a special subset of TCSP, which can be classified as a pre-scheduling coordination approach. Analogously, our coordination method could be called a Plan Decoupling Problem, because it resolves all interdependencies between agents on a plan level instead of a schedule level. In the future, it would be interesting to combine the pre-planning and pre-scheduling coordination approaches using an even more expressive framework including both qualitative and quantitative temporal information.

VI. ACKNOWLEDGEMENTS

J. Renze Steenhuisen is supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024. The ICIS project is hosted by the DECIS Lab, the open research partnership of Thales Nederland, the Delft University of Technology, the University of Amsterdam, and the Netherlands Foundation of Applied Scientific Research (TNO).

REFERENCES


APPENDIX

Consider a tightly-coupled task \(\{T_i\}_{i=1}^{n} \), satisfying the conditions stated in the proposition. For every agent \(A_i\), let \(P_i = (T_i, \prec_i, \equiv_i)\) be an arbitrarily chosen local plan satisfying the conditions (i.e., \(\prec_i\) is a partial order relation extending \(\subseteq_i\), \(\equiv_i\) is an equivalence relation extending \(\subseteq_i\), \(\subseteq_i \cap \equiv_i = \emptyset\), and \(\equiv_i\) is both left- and right-closed under composition with \(\subseteq_i\)).

We have to show that there exists a function \(s : T \to \mathbb{Z}^+\) satisfying all local and global precedence and synchronisation constraints. Clearly, there exists such a function \(s_i : T_i \to \mathbb{Z}^+\) for every plan \(P_i\) satisfying all local precedence constraints and synchronisation constraints. We only have to prove that from these local functions \(s_i\) a function \(s\) can be constructed that also satisfies all inter-agent precedence and synchronisation constraints. First, we construct a total ordering of all these inter-agent constraints that satisfies the following conditions.

1) Whenever \((x, y)\) is an inter-agent precedence constraint, and \(x\) occurs in \(T_i\), then every inter-agent constraint \((u, v)\) (precedence or synchronisation constraint) such that \(v <_i x\) should occur before \((x, y)\) in the ordering.

2) Whenever \((x, y)\) is an inter-agent synchronisation constraint between \(T_i\) and \(T_j\), then every inter-agent constraint \((u, v)\) (precedence or synchronisation constraint) such that \(v <_i x\) or \(v <_j y\) should occur before \((x, y)\) in the ordering.

If such an ordering exists, a schedule \(s\) can be constructed satisfying all constraints as follows. We use the set of locally constructed schedules \(\{s_1, s_2, \ldots, s_n\}\) and process the inter-agent constraints one by one in the order as specified above. If all inter-agent constraints have been processed, we construct a joint schedule \(s(\cdot)\) by \(s(t) = s_j(t)\) if \(t \in T_j\).

The correctness of this procedure then follows from the fact that in this process of adapting to a new inter-agent constraint, we never invalidate earlier satisfied constraints. To see this we specify the adaptation procedure as follows: Let \((s_1, s_2, \ldots, s_n)\) be the current set of local schedules and let \((x, y) \in (T_i \times T_j)\) be the first inter-agent constraint in the ordering not yet processed.

If \(x < y\), then adapt \(s_j\) as follows: if \(s_j(y) \leq s_j(x)\) for \(z = y\) and all \(z \in \mathbb{T}_j\) such that \(y <_j z\), let \(s(z) := s_j(z) + (s_j(x) - s_j(y)) + 1\). If \(s_j(y) > s_j(x)\), no adaptation is necessary and the constraint is satisfied. This adaptation ensures that the set of local schedules also satisfies this precedence constraint \((x, y)\).

If \(x \equiv y\), then \(s_i(x), s_j(y) := \max\{s_i(x), s_j(y)\}\) and we adapt all successors of the updated function accordingly, i.e., if \(s_i(x) < s_j(y)\) then for all \(z\) such that \(x <_i z\) and \(s_i(z) := s_i(z) + (s_j(y) - s_i(x))\). This ensures that the resulting set of schedules also satisfies the synchronisation constraint.

The ordering between the constraints now ensures that when the \(s\) value of a task \(t\) is updated, all the values of all tasks 'below' \(t\) have obtained their definite value and do not need to be adapted. Moreover, although \(t\) might be involved in more than one update operation, its value \(s(t)\) will only increase and, therefore, never invalidates values assigned to tasks below \(t\). Therefore, at the end, the schedule \(s\) composed as \(s(t) = s_j(t)\) if \(t \in \mathbb{T}_j\) satisfies all constraints.

The only detail left is to prove that such an ordering as defined above exists. But this is easily seen with \(\bigcup_{i=1}^{n} \prec_i \cup \prec\) being acyclic and the conditions enforced on \(\equiv\).