

Plan Decoupling of Agents with Qualitatively Constrained 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, tel. +31 15 278 7486, fax +31 15 278 6632,
{J.R.Steenhuisen,C.Witteveen}@tudelft.nl

July 21, 2008

Abstract

In many task-planning domains, assemblies of autonomous agents need to construct plans and schedules for executing their assigned sets of tasks in order to complete them. Often, there will be dependencies between tasks to be executed by different agents. In such a situation, a plan-coordination problem arises when the joint plan is required to be feasible, whatever locally-feasible plans the individual agents come up with. This problem can be solved by plan decoupling, which is to add a minimum number of constraints such that each agent can make a plan for its set of tasks independently of the others while still joint-plan feasibility is guaranteed. Previous work on plan decoupling concentrated on a coordination framework where the only dependencies between tasks are precedence constraints. In this paper an extension of the framework is discussed where not only precedence constraints, but also synchronisation constraints can be used in order to express qualitative temporal constraints between tasks. It is shown that adding synchronisation constraints does not add any complexity to the plan-decoupling problem, and that a previously-developed approximation algorithm for plan decoupling can be extended to cope with synchronisation constraints as well.

1 Introduction

A *plan-coordination problem* arises whenever a complex task has to be solved by a number of agents, each having to solve a part of this task. To achieve its part of the overall task, each agent has to make a plan. The plan-coordination problem now consists of how to ensure that these individual plans are jointly feasible (i.e., conflict free) and achieve the total task specified. Examples of applications where plan coordination plays an important role are such diverse as building a house, preparing for rescue operations [8] after a disaster, and servicing planes at an airport [11].

In general, this plan-coordination problem is a non-trivial problem. The reason is that often the overall (complex) task consists of an interdependent set of elementary tasks, and elementary tasks that are dependent upon each other are assigned to different agents. Hence, although the plans of the agents might be individually feasible, the feasibility of the plan of an agent might become dependent upon the structure of plans developed by other agents and the result can be an infeasible joint plan. To illustrate such a plan-coordination problem, a simple example is discussed.

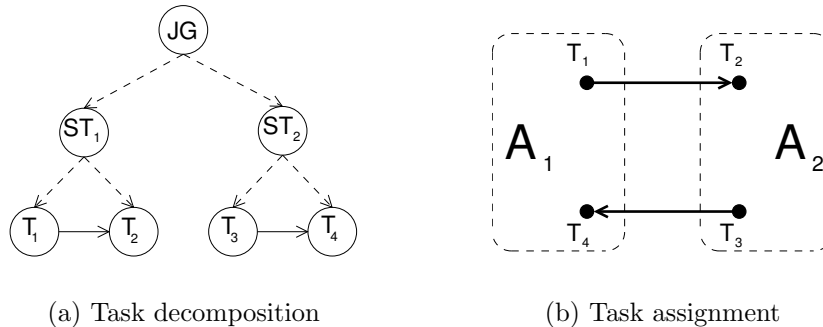


Figure 1: A task allocation can cause agents to become interdependent.

Example 1 Consider a set of agents which have agreed on a common goal, and have identified a high-level task JG (i.e., joint goal) which achieves this goal. This complex task can be decomposed into subtasks ST_1, ST_2 , which can further be decomposed into a set of elementary tasks $\tau_1, \tau_2, \tau_3, \tau_4$. In Figure 1(a), a very basic task decomposition is shown, although more elaborate decomposition frameworks exist (e.g., *TÆMS* [9]). The set of elementary

tasks is partitioned and allocated to the agents. In Figure 1(b), the resulting situation is depicted after assignment of tasks to agents together with the inherited dependency constraints. Here, it can be seen that agent A_1 has to make a plan for task τ_1 and τ_4 , while A_2 has to make a plan for τ_2 and τ_3 . The problem now is how we can prevent a combination of infeasible individual plans to occur, for example, how to prevent that agent A_1 chooses to complete τ_4 before τ_1 and agent A_2 chooses a plan where τ_2 is completed before τ_3 , because the combination of those two plans will create a deadlock in the combined plan.

In the remainder of this work, this decomposition phase is assumed to be completed and the focus will be on solving the plan-coordination problem that arises. Basically, there are two main approaches to solve this plan-coordination problem: (i) by requiring collaborative actions from the agents themselves, or (ii) by designing a suitable coordination mechanism that ensures feasible plans.

Quite some studies have taken the first *coordination-by-collaboration* approach. For example, in SHAREDPLANS [7], the collaborating agents must be committed to both planning and doing the activity together. In GPGP [12], the coordination mechanisms require the agents to cooperatively search for commitments in environments where communication is always and sufficiently available during task execution. Others, like [15], even propose to manage the coordinated planning and execution of the tasks by letting the agents keep each other informed about any changes (e.g., re-scheduled tasks). It is clear that such approaches require *cooperative* agents that are willing to inform other agents about details of their individual plans and, if necessary, are prepared to revise parts of their plans in striving to construct a feasible joint solution. As a consequence, however, when communication is unavailable, combining the individual chosen plans can easily lead to an infeasible joint solution, even when these plans are locally feasible.

Example 2 Consider the situation depicted in Figure 1 when communication is lost. Adhering to the local constraints, agent A_1 might decide to execute τ_4 before τ_1 , and agent A_2 might decide to execute τ_2 before τ_3 . Clearly, this situation leads to an infeasible joint plan due to the circular dependency of the tasks (i.e., τ_1 before τ_2 before τ_3 before τ_4 before τ_1).

In contrast, the *coordination-by-design* approach does neither assume information sharing nor collaborative actions to take place in order to achieve a

feasible joint solution. This approach ensures the existence of such a feasible joint solution by specifying a *coordination mechanism* [3]. In general, such a coordination mechanism should enable individual agents to choose their preferred way of solving their part of the task, thereby (minimally) reducing the initial (planning) freedom of the agents. In particular, in designing a plan-coordination mechanism we need to specify for each agent a set of commitments that are minimally restricting, such that the combined plans are guaranteed to be jointly feasible. Note, that this approach is especially useful in unpredictable environments because it distributes the responsibility for the tasks to the agents, that must somehow come up with a solution to their subproblem such that the commitments (i.e., constraints) are met. Moreover, this approach is robust against failing communication, allows individual replanning, and additional tasks to be done by the agents while not violating global constraints.

In this paper, we study the *plan-decoupling problem*, which can be described as a coordination-by-design coordination mechanism. Basically, this problem is to find a minimum number of constraints to be imposed on the tasks given to the agents in order to prevent all possible conflicts before the agents construct their plans individually.

In some previous work [2, 18], the main features of this approach have been studied in a framework that only allowed precedence constraints as dependencies between elementary tasks. The main outcomes of these studies concerned the computational complexities associated with constructing minimum plan-coordination sets.

In this paper, we will extend this previous work on plan decoupling in the following ways. First, the framework will be extended by representing tasks as a tuple of time points and allowing both precedence and synchronisation constraints on these time points. This will allow us to represent all qualitative-temporal constraints on the tasks. Second, the computational complexity of the plan-decoupling problems in this extended framework are determined. Finally, a polynomial-time approximation algorithm suitable to deal with the plan-decoupling problem for tasks with qualitative temporal constraints is provided.

This paper is organised as follows: In Section 2, we describe the extended framework in which we will study the plan-decoupling problem. In Section 3, we analyse the possible conflicts that need to be prevented from occurring by decoupling in more detail. In Section 4, we determine the complexity and provide approximation algorithms for different plan-decoupling problems. In

Section 5, an approximation algorithm is given for constructing coordination sets in polynomial time. In Section 6, our plan-decoupling approach will be related to other work that has been done. Finally, in Section 7, we conclude and give some directions for future research.

2 Framework

We consider a high-level (complex) task that has been decomposed into a set of elementary tasks $\mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_k\}$ that need to be completed in order to complete the high-level task. These elementary tasks, called *methods* in the TÆMS [9] framework, are constrained by a set of constraints \mathcal{C} . In the original description of the framework [18], this set of constraints consisted of *precedence constraints* between elementary tasks, each precedence constraint $\tau \prec \tau'$ specifying that task τ should be completed before task τ' could start. Together these constraints were assumed to induce a partial ordering on the set of elementary tasks \mathcal{T} .

In this paper, we extend the set of task constraints to deal with the familiar *qualitative temporal constraints* [1]: **before**, **overlaps**, **during**, **meets**, **starts**, **finishes**, and **equals**. Note that the precedence relation \prec defined on pairs of tasks corresponds to the **before** relation defined on them. Reasoning on networks with such temporal constraints $\langle \mathcal{T}, \mathcal{C} \rangle$ has been studied extensively [1, 13]. If we represent tasks τ as tuples of *time-point variables* (t^s, t^e) , with t^s being the start and t^e being the end of the execution of task τ , it is well-known that these seven qualitative temporal constraints between tasks can be expressed by using a combination of precedence and synchronisation constraints on these time-point variables [4].

Therefore, we represent a set of elementary tasks \mathcal{T} together with its set of qualitative temporal constraints \mathcal{C} as a tuple $\langle T, \prec, \equiv \rangle$, where T is the set of time points associated with the elementary tasks in \mathcal{T} , $\prec \subseteq (T \times T)$ is a partially-ordered precedence relation, and $\equiv \subseteq (T \times T)$ denotes an equivalence relation expressing the synchronisation constraints.

Note that for every task τ , the associated time points are constrained by $t^s \prec t^e$. Moreover, \prec and \equiv together satisfy the following two properties:

1. $(\prec \circ \equiv) \subseteq \prec$ and $(\equiv \circ \prec) \subseteq \prec$,
that is, $(t \prec t' \wedge t' \equiv t'' \text{ implies } t \prec t'')$ and $(t \equiv t' \wedge t' \prec t'' \text{ implies } t \prec t'')$, and

2. $(\prec \cap \equiv) = \emptyset$
that is, \prec and \equiv are orthogonal relations.

Summarising the properties of this extension of the framework, we have the following result:

Proposition 1 *All qualitative temporal relations between tasks τ in a set of tasks \mathcal{T} can be expressed in the extended task framework $\langle T, \prec, \equiv \rangle$, where T is the set of time point variables representing \mathcal{T} .*

2.1 Agents and Tasks Assignments

We consider a set of agents $\mathcal{A} = \{A_1, A_2, \dots, A_n\}$ that are jointly capable of completing the set \mathcal{T} of all elementary tasks. We assume that each elementary task τ in \mathcal{T} is assigned to at least one agent A_i that has the capability of completing that task.¹

If a task τ must be assigned to multiple agents it is simply duplicated, giving the duplicates unique names $\tau_1, \tau_2, \dots, \tau_k$ adding appropriate synchronization constraints between the start and ending time points associated with these duplicates. Then, the duplicates can be assigned to separate agents such that every task is assigned to exactly one agent.

Example 3 *The task of simultaneously lifting a table with two agents becomes two elementary tasks that both have to start and end at the same time (i.e., duplicate task τ into tasks τ_1, τ_2 and add the constraint τ_1 equals τ_2).*

The task τ of conducting a certain scientific experiment three times becomes three tasks τ_1, τ_2, τ_3 without any additional constraints.

Therefore, an assignment of tasks to agents can simply be conceived as a partitioning $\{\mathcal{T}_i\}_{i=1}^n$ of the set of elementary tasks \mathcal{T} , where \mathcal{T}_i is the set of tasks assigned to agent A_i .

As a result, every agent A_i is assigned a disjoint subset $\mathcal{T}_i \subseteq \mathcal{T}$ of tasks together with the local subset $\prec_i = (\prec \cap (T_i \times T_i))$ of precedence constraints and the local subset $\equiv_i = (\equiv \cap (T_i \times T_i))$ of synchronisation constraints induced by the set of time points T_i associated with \mathcal{T}_i . We represent such an assigned subset of elementary tasks as the tuple $\langle T_i, \prec_i, \equiv_i \rangle$.

¹How to find a suitable assignment for a set of agents is a separate problem [14, 19], and is beyond the scope of this paper.

Note that the original relations (sets of constraints) \prec and \equiv are partitioned into the set of constraints between time points belonging to the same partition, and the set of constraints between time points belonging to different partitions. We call the sets of constraints \prec_i, \equiv_i belonging to the same partition *intra-agent constraints*, and the constraints between time points of different partitions *inter-agent constraints* $\prec_{inter}, \equiv_{inter}$. Together, the sets of inter and intra-agent constraints make up the complete sets of constraints, i.e., $\prec = (\prec_{inter} \cup \bigcup_{i=1}^n \prec_i)$ and $\equiv = (\equiv_{inter} \cup \bigcup_{i=1}^n \equiv_i)$.

2.2 Plans

In order to complete its set of tasks \mathcal{T}_i , an agent A_i needs to construct a plan for it. Since we want to deal with coordination mechanisms that could be used for a variety of planning agents, we do not want to be dependent upon the details of the planning tools used by the individual agents. The only thing that we need to bother about is that each plan satisfies the task constraints given to the agents. This means that each agent should be capable of representing its plan as an abstract plan $\pi_i = \langle T_i, \prec_i^*, \equiv_i^* \rangle$, specifying constraints on the tasks induced by its concrete plan. Such a plan π_i then meets the task constraints iff $\prec_i \subseteq \prec_i^*$ and $\equiv_i \subseteq \equiv_i^*$, i.e., the plan π_i is a *refinement* of $\langle T_i, \prec_i, \equiv_i \rangle$.

From now on, we will simply assume that each such a local plan π_i meets the task constraints specified on the tasks in \mathcal{T}_i . Given the local plans π_i , we define the *joint plan* $\{\pi_i\}_{i=1}^n$ of all agents as the task that results from taking the union of all individual plan refinements: $\{\pi_i\}_{i=1}^n = \langle T, \prec_{inter} \cup \bigcup_{i=1}^n \prec_i^*, \equiv_{inter} \cup \bigcup_{i=1}^n \equiv_i^* \rangle$. However, the question is whether this joint plan meets the set of constraints specified on the global task \mathcal{T} . As the following example illustrates, this is not always the case, since the local plans only satisfy all intra-agent constraints and not necessarily the inter-agent constraints.

Example 4 In Figure 2(a), two agents are shown where agents A_1 and A_2 have the freedom of planning $t_4^s \prec t_1^e$ and $t_2^s \prec t_3^e$, respectively. But when these plans are joined, a cycle $\langle t_1^e, t_2^s, t_3^e, t_4^s, t_1^e \rangle$ is introduced (see Figure 2(b)). Such a cycle indicates an infeasible joint plan, since it implies that t_1^e precedes t_4^s , but also vice versa.

In order to define what additional criteria need to be satisfied, such that the joint plan $\{\pi_i\}_{i=1}^n$ is guaranteed to meet the global task constraints, we will use the notion of a *dispatching* satisfying the task constraints.

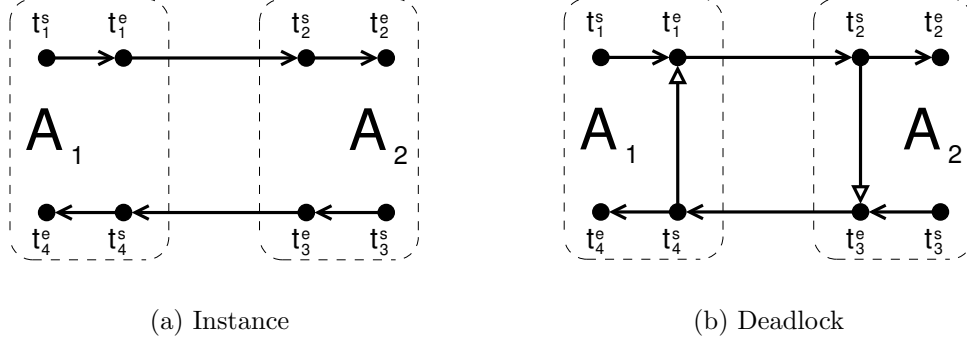


Figure 2: Two plan-uncoordinated moderately-coupled agents.

Definition 1 Given a plan $\pi = \langle T, \prec^*, \equiv^* \rangle$, a dispatching δ for π is a function $\delta : T \rightarrow \mathbb{Z}$ that respects the constraints in π , that is

1. $\forall t, t' \in T: t \prec^* t'$ implies $\delta(t) < \delta(t')$, and
2. $\forall t, t' \in T: t \equiv^* t'$ implies $\delta(t) = \delta(t')$.

A plan π is said to be *feasible* if there exists a dispatching δ for it that satisfies the plan. Since we have assumed a plan π for a task $\langle T, \prec, \equiv \rangle$ to be a refinement of it, if π has a dispatching δ that satisfies it, δ will also satisfy all the task constraints.

We assume that the local plans π_i developed by an agent A_i are feasible, i.e., for every π_i there exist a local dispatching δ_i satisfying it. Now, the (in)feasibility can be expressed of the joint plan $\{\pi_i\}_{i=1}^n$ as follows:

Definition 2 Given a task $\langle T, \prec, \equiv \rangle$, a joint plan $\{\pi_i\}_{i=1}^n$ for it is feasible if there exist dispatchings δ_i for the local plans π_i such that the joint dispatching δ , defined as

$$\forall t: \delta(t) = \delta_i(t) \text{ iff } t \in T_i, i = 1, 2, \dots, n$$

satisfies all the task constraints, that is

1. $\forall t, t' \in T: t \prec t'$ implies $\delta(t) < \delta(t')$, and
2. $\forall t, t' \in T: t \equiv t'$ implies $\delta(t) = \delta(t')$.

Note that this definition immediately implies that $\{\pi_i\}_{i=1}^n$ is feasible iff $\prec_{inter} \cup \bigcup_{i=1}^n \prec_i^*$ is an acyclic relation (i.e., a partial order). As the example above illustrated, it is not guaranteed on beforehand that such a joint plan is feasible, even when all local plans are guaranteed to be feasible. Therefore, a plan-coordination problem arises, which provides this guarantee while allowing the agents to construct local plan refinements independently from the others.

3 Plan Decoupling

A task is said to be *plan coordinated* when every joint plan, of individually constructed plan refinements, is feasible. In that case, for every set of individually-chosen feasible plans π_i , the joint plan $\{\pi_i\}_{i=1}^n$ allows a joint dispatching δ . In general, plan coordination can be achieved by ensuring that cyclic ordering of tasks due to independent planning cannot occur.

In this section, a *plan-decoupling* method to plan coordination is taken, where a minimum number of constraints is added in order to construct plan-coordinated tasks.

Before we discuss this plan decoupling method in detail, let us relate this form of plan decoupling to a well-known classification of plan-coordination instances based on the types of inter-agent constraints occurring in them. Like in [19], we consider a plan-coordination instance as *loosely coupled* when there are no inter-agent constraints, as *moderately coupled* when only precedence constraints are used, and *tightly coupled* when both precedence and synchronisation constraints occur as inter-agent constraints.

Clearly, when agents are loosely coupled, there are no constraints between time points t, t' that are assigned to different agents. In other words, both sets of inter-agent constraints \prec_{inter} and \equiv_{inter} are empty sets. Hence, such tasks are always plan-coordinated and there is no real plan-coordination problem to solve.

When agents are moderately coupled, pairs of time points t, t' that are assigned to different agents can be constrained by an inter-agent precedence constraint (e.g., see Figure 2). The plan-decoupling problem for moderately-coupled agents has been studied thoroughly in previous work [2, 17, 18].

Here, we will concentrate on the tightly coupled case, where both precedence and synchronisation constraints occur as inter-agent constraints.

3.1 Plan Decoupling of Tightly-Coupled Agents

In tightly-coupled task, the sets \prec_{inter} and \equiv_{inter} are both non-empty. The presence of synchronisation constraints causes some additional problems for plan decoupling, compared to plan decoupling of moderately-coupled agents. In Figure 3, an example is depicted of two tightly-coupled agents with four time points each. There are two pairs of synchronised time points: $\{t_1^s, t_2^s\}$ and $\{t_4^s, t_3^s\}$.

Note that independent planning might violate these synchronisation constraints even if no inter-agent cycles are created. In Figure 3(b), time point t_1^s is planned before time point t_4^s , while t_3^s is planned before t_2^s . This clearly violates the given synchronisation constraints $t_1^s \equiv t_2^s$ and $t_4^s \equiv t_3^s$, since every dispatching δ satisfying the precedence constraints has $\delta(t_1^s) < \delta(t_4^s)$ and $\delta(t_3^s) < \delta(t_2^s)$. However, satisfying the synchronisation constraint $t_1^s \equiv t_2^s$ requires $\delta(t_1^s) = \delta(t_2^s)$, which implies $\delta(t_3^s) < \delta(t_4^s)$ and thereby violating $t_3^s \equiv t_4^s$.

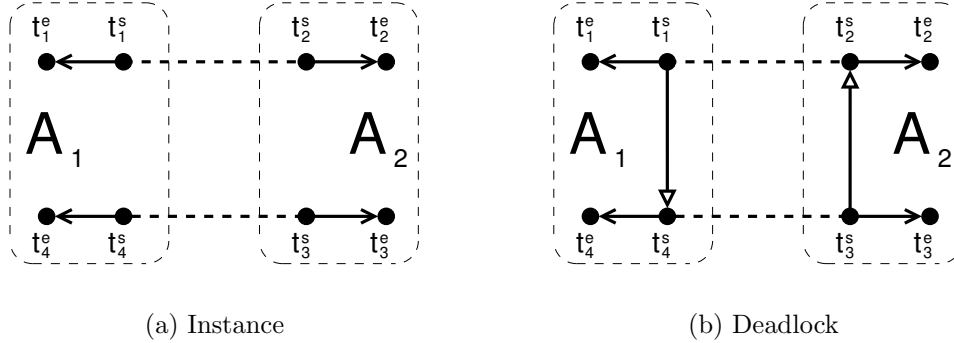


Figure 3: Two plan-uncoordinated tightly-coupled agents.

To characterise exactly when a tightly-coupled task is plan coordinated, we need the following definitions. A *path* in a tightly-coupled task $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$ is a sequence $\langle t_1, t_2, \dots, t_m \rangle$ of time point variables such that for each $h = 1, 2, \dots, m - 1$, either $t_h \equiv t_{h+1}$ or $t_h \prec t_{h+1}$ holds. A path is called *directed* if, for at least one h , it holds that $t_h \prec t_{h+1}$. A *directed cycle* is a directed path $\langle t_1, t_2, \dots, t_m, t_1 \rangle$. Note that a plan $\pi_i = \langle T_i, \prec_i^*, \equiv_i^* \rangle$ is a refinement of the task $\langle T_i, \prec_i, \equiv_i \rangle$ such that π_i does not contain any directed cycle. Now, we can obtain the following characterisation of a plan-coordinated task.

Proposition 2 *A tightly-coupled task $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$ is plan coordinated iff there exists no set $\{\pi_i\}_{i=1}^n$ of plans $\langle T_i, \prec_i^*, \equiv_i^* \rangle$ for the subtasks $\langle T_i, \prec_i \equiv_i \rangle$ such that the joint plan $\{\pi_i\}_{i=1}^n = \langle \{T_i\}_{i=1}^n, \prec_{inter} \cup \bigcup_{i=1}^n \prec_i^*, \equiv_{inter} \cup \bigcup_{i=1}^n \equiv_i^* \rangle$ contains a directed cycle.*

Proof (\Rightarrow) By definition, a tightly-coupled task is plan coordinated when a joint dispatching δ is guaranteed to exist for every set $\{\pi_i\}_{i=1}^n$ of individually-chosen feasible plans π_i . Such a joint dispatching is an assignment $\delta : T \rightarrow \mathbb{Z}$ of values to time points satisfying the constraints in $\{\pi_i\}_{i=1}^n$. This immediately excludes the occurrence of any directed cycle $c = (t, t', \dots, t)$ as such a cycle would imply $\delta(t) < \delta(t)$.

\Leftarrow If no joint plan contains a directed cycle, a dispatching δ can be easily constructed for the joint plan $\{\pi_i\}_{i=1}^n$ by determining the depth of the time-points t in the graph associated with $\{\pi_i\}_{i=1}^n$. Assuming that \prec is right- and left-closed under composition with \equiv (See Section 2), the depth $depth(t)$ of t is defined inductively as: $depth(t) = 1$ if t does not have predecessors in \prec and $depth(t) = 1 + \max\{depth(t') : t' \prec t\}$, else. Since there is no directed cycle, $depth()$ is well-defined. Now the dispatching δ defined by $\delta(t) = depth(t)$ for all $t \in T$ is a dispatching satisfying all constraints. \square

It is not difficult to show that every feasible (tightly-coupled) task can be changed into a plan-coordinated task by adding intra-agent precedence and/or synchronisation constraints. The idea is to extend every local relation $\prec_i \cup \equiv_i$ to a total ordering of T in such a way that their union, together with the inter-agent constraints, constitutes an acyclic relation (i.e., does not contain a directed cycle).

This can be accomplished as follows: Consider an arbitrary (tightly-coupled) task $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$. Since $\prec \cup \equiv$ is assumed to be acyclic, there exists an extension \prec^* of \prec and an extension \equiv^* of \equiv such that $\equiv^* \cup \prec^*$ is a total ordering of the set of time points T .

For every agent A_i , consider the subtask $\langle T_i, \prec_i^*, \equiv_i^* \rangle$. Here, both \prec_i^* and \equiv_i^* are extensions of \prec_i and \equiv_i , respectively, and $\prec_i^* \cup \equiv_i^*$ constitutes a total acyclic relation on T_i . Hence, there is no further refinement of this task. Therefore, every agent A_i constructs the plan $\pi_i = \langle T_i, \prec_i^*, \equiv_i^* \rangle$. The joint plan $\{\pi_i\}_{i=1}^n$ now is a feasible plan since $\bigcup_{i=1}^n (\prec_i^* \cup \equiv_i^*) \cup (\prec \cup \equiv) \subseteq \equiv^* \cup \prec^*$ is acyclic.

Therefore, we have shown the following proposition to hold.

Proposition 3 *Finding a sufficient coordination set $\Gamma = \Gamma^{\prec} \cup \Gamma^{\equiv}$, consisting of precedence Γ^{\prec} and synchronisation constraints Γ^{\equiv} , for a tightly-coupled task $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$, such that the resulting task $\langle \{T_i\}_{i=1}^n, \prec \cup \Gamma^{\prec}, \equiv \cup \Gamma^{\equiv} \rangle$ is plan coordinated, can be done in polynomial time.*

4 Complexity

In the previous section, the patterns were identified that cause a set of tightly-coupled agents to be plan uncoordinated. In order to plan coordinate a set of agents, a plan-coordination problem needs to be solved such that these patterns do not occur.

Pre-planning plan coordination, as in [18], can be seen as a *plan-decoupling* approach, because it decouples the plan dependencies between the agents. Then, the PLAN-DECOUPLING PROBLEM (PDP) is to find a minimum set of additional constraints that transform a plan-coordination instance into a plan coordinated one. The idea is that such a set of constraints affects the planning freedom of the agents in a minimal way. Note that the PDP does not ask for an arbitrary set of additional constraints to make the instance plan coordinated. The problem of constructing an arbitrary coordination set is, as we have seen in the previous section, solvable in polynomial time.

In previous work [2, 16, 17, 18], PDP for moderately-coupled agents, as well as some of its variants, has been studied quite extensively. It was shown that this problem is Σ_2^p -complete in general [18]. The proof is based on a polynomial reduction of PLAN-COORDINATION RECOGNITION (PCR) (i.e., the verification problem of the PDP) to PATH WITH FORBIDDEN PAIRS [6], and PDP itself to a generalised version thereof. Although solving the PDP is intractable in general, subclasses exist that are easier to solve. In fact, two sources of complexity have been determined [16]: (i) the allowed number of time points per agent, and (ii) the number of agents.

In this section, the complexity of the PDP for tightly-coupled agents will be studied. In fact, it will be shown that the PDP for tightly-coupled agents is not harder—from a complexity point of view—than the PDP for moderately-coupled agents.

4.1 Plan Decoupling of Tightly-Coupled Agents

Analogous to the moderately-coupled case, plan decoupling of tightly-coupled agents can be achieved by finding a (smallest) set of additional constraints that allow agents to plan independently while guaranteeing the feasibility of the joint plan. Let us consider the following decision-variant of the problem of plan-decoupling tightly-coupled agents.

PDP for tightly-coupled agents

INSTANCE: Given a tightly-coupled task $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$ and a positive integer K .

QUESTION: Does there exist a coordination set $\Gamma = \Gamma^{\prec} \cup \Gamma^{\equiv}$ with $|\Gamma| \leq K$ such that $\langle \{T_i\}_{i=1}^n, \prec \cup \Gamma^{\prec}, \equiv \cup \Gamma^{\equiv} \rangle$ is plan coordinated?

Using the obtained results on the computational complexity of the PDP for moderately-coupled agents, the complexity of the PDP for tightly-coupled agents can be determined.

Proposition 4 *PDP for tightly-coupled agents is Σ_2^p -complete.*

Proof To establish the complexity of this problem, we know that it should be Σ_2^p -hard, because the (contained) PDP for moderately-coupled agents is known to be Σ_2^p -complete [18]. Moreover, if we guess a set Γ of additional constraints, the complexity of verifying coordination of the extended task $\langle \{T_i\}_{i=1}^n, \prec \cup \Gamma^{\prec}, \equiv \cup \Gamma^{\equiv} \rangle$ can be accomplished by verifying the condition stated in Proposition 2. Verifying whether it holds, for this extended task, that every set of individually-chosen feasible plans $\{\pi_i\}_{i=1}^n$ does not contain a directed cycle is in coNP [18]. Here, a directed cycle can be detected in polynomial time after guessing a set of locally-feasible plans. \square

Therefore, rather surprisingly, viewing the plan-decoupling problems from a complexity point of view, moderately-coupled and tightly-coupled variants do not differ significantly.

5 Approximation

In the previous section, the computational complexity of the PDP for tightly-coupled agents was determined which turned out to be intractable. Therefore, we should look for polynomial-time approximation algorithms to solve

the PDP. Concerning the approximability, it was shown previously that PDP for moderately-coupled agents is APX-hard, and that a constant-ratio approximation algorithm is not likely to exist [18]. The proof for this is based on a polynomial reduction from FEEDBACK VERTEX SET [6] to the subclass of the PDP for moderately-coupled agents, where each agent has exactly two time points.

There exists a simple polynomial-time approximation algorithm that succeeds in finding a sufficient—but not necessarily minimum—coordination set for moderately-coupled agents [17]. This algorithm uses the fact that the time points are partially ordered by a precedence relation, which is necessarily acyclic. Therefore, there is at least one agent A_i that has some subset $T_i^1 \subseteq T_i$ of *prerequisite-free* time points that do not depend on any other time points. Now, when all these agents complete these subsets of time points, a new subset $T_i^2 \subseteq T_i$ of time points that only depend on time points in T_i^1 becomes prerequisite free. Note that applying this approach recursively results in a partitioning $\{T_i^d\}_{d=1}^{\text{depth}(T)}$ of the time points, where $\text{depth}(T)$ is the depth of the partial order defined on T . More precisely, the depth $\text{depth}(t)$ of a time point t is defined as follows: If t does not have predecessors in \prec then $\text{depth}(t) = 1$, else $\text{depth}(t) = 1 + \max\{\text{depth}(t') \mid t' \prec t\}$. Then, the depth $\text{depth}(T)$ of the set of time points T is the maximum value of $\text{depth}(t)$ for a $t \in T$. After at most $\text{depth}(T) \leq |T|$ iterations, all time points have been accepted exactly once as prerequisite-free. For each agent A_i , we now have a set of disjoint subsets T_i^d for $d = 1, 2, \dots, |T|$. Here, T_i^d denotes the (possibly empty) set of time points selected for agent A_i in iteration d . The resulting coordination set Γ is then constructed by taking the union of the sets Γ_i for agent A_i , where each set Γ_i is obtained as follows:

1. Remove all empty subsets T_i^d and let m be the number of remaining subsets, and
2. add a precedence constraint $t \prec t'$ to Γ_i for every pair of time points $t \in T_i^j$ and $t' \in T_i^{j+1}$, for every $j = 1, 2, \dots, m - 1$.

It can easily be shown that, although not creating a minimum coordination set, the resulting set Γ is sufficient for plan decoupling [17]. A description of this *depth-partitioning algorithm*, delivering the coordination sets Γ_i , is given in Algorithm 1.

In fact, there is at least one planning domain, the LOGISTICS domain from the AIPS competition, where this algorithm produces optimal solu-

Algorithm 1 Depth-Partitioning Algorithm.

1. Take the moderately-coupled task $\langle \{T_i\}_{i=1}^n, \prec \rangle$ and consider the subsets $T^d = \{t \in \bigcup_{i=1}^n T_i \mid \text{depth}(t) = d\}$ of time points having the same depth in T .
 2. Consider the partitioning $\{T_i\}_{i=1}^n$ of T induced by the task allocation to the agents $\{A_i\}_{i=1}^n$ and let $T_i^d = T^d \cap T_i$.
 3. For all i , let $(T_i^{d_1}, T_i^{d_2}, \dots, T_i^{d_k})$ be the sequence of all (non-empty) sets T_i^d sorted in increasing values of the depth value d_j .
 4. For all i , let Γ_i contain a constraint $t \prec t'$ for all the pairs $(t, t') \in (T_i^{d_j} \times T_i^{d_{j+1}})$ for $j = 1, 2, \dots, k - 1$ (i.e., all time points of an agent occurring at a lower depth are required to precede all time points occurring at a higher depth).
 5. Return $\Gamma = \bigcup_{i=1}^n \Gamma_i$.
-

tions. There, a multi-modal transportation task needs to be executed by transportation agents where a number of precedence constraints hold on the subtasks. Besides optimal coordination sets, it turns out that the quality of the resulting joint plans are comparable to the ones that are a result of centralised planning while taking much less time [17]. It can be shown that there exists no polynomial algorithm for either a cooperative or a selfish multi-agent system solving this multi-modal transportation task with an approximation ratio $\epsilon < \frac{10}{9}$ unless some generally accepted conjecture in complexity theory is violated [17]. Moreover, the algorithm above together with a 2-approximation algorithm for the local planning problems can be shown to be a $\frac{11}{9}$ -approximation algorithm for the planning of this multi-modal transportation domain [17].

5.1 Plan Decoupling of Tightly-Coupled Agents

In this section, the depth-partitioning algorithm for moderately-coupled tasks will be slightly modified to coordinate tightly-coupled agents as well. The depth-partitioning algorithm can be used to solve the contained problem for moderately-coupled tasks. Due to the synchronisation constraints, new

conflicts can arise that need to be prevented. In this section, the depth-partitioning algorithm will be adapted to coordinate tightly-coupled agents as well.

The depth-partitioning algorithm needs to be adapted such that two synchronised time-point variables $t \equiv t'$ have the same depth value. This is required by the inclusion properties of the composition of \prec and \equiv (see Section 2). Now note that using the depth-partitioning approach (i.e., partitioning the time points and ordering the time points from different partitions) results in a set of additional constraints which breaks all directed cycles that traverse through precedence constraints, but also partitions the time points that are involved in synchronisation constraints. In fact, because the remaining conflicts reside within each depth partition, the involved time-point variables must have the same depth. Thus, the (inter-agent) conflicts that remain are concerned with synchronisation constraints only (e.g., see Figure 3). To prevent such conflicts, such synchronised time points in the same partition are synchronised to prevent any conflicts from happening.

The approximation algorithm for plan-decoupling tightly-coupled agents can then be described as shown in Algorithm 2. In the following proposition, it is proven that generating a coordination set based on this algorithm results in a sufficient, albeit not necessarily minimum, coordination set Γ for tightly-coupled tasks.

Proposition 5 *Given a tightly-coupled task $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$. Let $\Gamma = \Gamma^\prec \cup \Gamma^\equiv$ be the set of additional precedence constraints induced by Algorithm 2, the depth-partitioning algorithm for tightly-coupled tasks. Then, the resulting tightly-coupled task $\langle \{T_i\}_{i=1}^n, \prec \cup \Gamma^\prec, \equiv \cup \Gamma^\equiv \rangle$ is plan coordinated.*

Proof Note that we can assume \prec to be an acyclic relation. Suppose, on the contrary, that the resulting tightly-coupled task is not plan coordinated. Then, by Proposition 2, some directed cycle $c = \langle t_1, t_2, \dots, t_m, t_1 \rangle$ must exist. Such a cycle consists of subsequences of tasks belonging to a single agent (intra-agent paths) and subsequences of tasks belonging to different agents (inter-agent paths).

First, note that, due to the construction of every set Γ_i , for every subsequence $\langle t_h, t_{h+1} \rangle$ of an intra-agent path $\langle t_j, \dots, t_k \rangle$ of agent A_i , it must hold that

$$\text{depth}'(t_h) \leq \text{depth}'(t_{h+1}). \quad (1)$$

Algorithm 2 Depth-Partitioning Algorithm for Tightly-Coupled Tasks.

1. Take the tightly-coupled task $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$ and consider the subsets $T^d = \{t \in \bigcup_{i=1}^n T_i \mid \text{depth}'(t) = d\}$ of time points having the same depth in T . Here, the depth $\text{depth}'(t)$ is defined as: If t does not have any predecessors in \prec then $\text{depth}'(t) = 1$, else $\text{depth}'(t) = \max(1 + \max\{\text{depth}'(t') \mid t' \prec t\}, \max\{\text{depth}'(t') \mid t' \equiv t\})$.
 2. Consider the partitioning $\{T_i\}_{i=1}^n$ of T induced by the task allocation to the agents $\{A_i\}_{i=1}^n$ and let $T_i^d = T^d \cap T_i$.
 3. For all i , let $(T_i^{d_1}, T_i^{d_2}, \dots, T_i^{d_k})$ be the sequence of all (non-empty) sets T_i^d sorted in increasing values of the depth value d_j .
 4. For all i , let Γ_i^{\prec} contain a constraint $t \prec t'$ for all the pairs $(t, t') \in (T_i^{d_j} \times T_i^{d_{j+1}})$ for $j = 1, 2, \dots, k-1$ (i.e., all time points of an agent occurring at a lower depth are required to precede all time points occurring at a higher depth).
 5. For all i , let Γ_i^{\equiv} contain a constraint $t \equiv t'$ for all the pairs $(t, t') \in (T_i^{\equiv, d_j} \times T_i^{\equiv, d_j})$, for all $j = 1, 2, \dots, k$ where $t \neq t'$ and $T_i^{\equiv, d_j} = \{t \in T_i^{d_j} \mid \exists t' \notin T_i : t \equiv t'\}$.
 6. Return $\Gamma = \bigcup_{i=1}^n \Gamma_i^{\prec} \cup \bigcup_{i=1}^n \Gamma_i^{\equiv}$.
-

Otherwise, $depth'(t_h) > depth'(t_{h+1})$ would imply that $t_{h+1} \prec t_h \in \Gamma_i^{\prec}$, and, therefore, together with the plan refinements $t_h \prec^* t_{h+1}$ and $t_h \equiv^* t_{h+1}$ would create a local directed cycle.

Second, for every inter-agent subsequence $\langle t_h, t_{h+1} \rangle$ of the directed cycle c , it must hold that either $t_h \equiv t_{h+1}$ or $t_h \prec t_{h+1}$. According to Algorithm 2, this must imply that $depth'(t_h) = depth'(t_{h+1})$ or $depth'(t_h) < depth'(t_{h+1})$, respectively.

Hence, we conclude that the depths $depth'(t_i)$ of the tasks t_i in the directed cycle c starting with $depth'(t_1)$ constitute a monotonically non-decreasing sequence. But this immediately implies that $depth'(t_1) = depth'(t_2) = \dots = depth'(t_m)$. Therefore, since c is a directed cycle, there must exist a subsequence (t_h, t_{h+1}) such that $t_h \prec_{inter} t_{h+1}$ or $t_h \prec_i^* t_{h+1}$ such that $depth'(t_h) = depth'(t_{h+1})$. Clearly, $t_h \prec_{inter} t_{h+1}$ implies $depth'(t_h) < depth'(t_{h+1})$ since $\prec_{inter} \subseteq \prec$. Hence, there exists an agent plan π_i containing a tuple $t_h \prec_i^* t_{h+1}$ such that $depth'(t_h) = depth'(t_{h+1})$. Let $(t_j, \dots, t_h, t_{h+1}, \dots, t_{j'})$ be the maximal intra-agent path containing both t_h and t_{h+1} . Then, by construction of the set Γ_i^{\equiv} , this implies that not both t_j and $t_{j'}$ occur in $T_i^{\equiv, k}$ for some k , since this would imply that the depth of all tasks in between them would be equal to k preventing $t_h \prec_i^* t_{h+1}$ to occur. Hence, at least t_j has a \prec -predecessor t'_j or $t_{j'}$ has a \prec -successor $t'_{j'}$. In both cases, c contains an occurrence of an inter-agent constraint \prec_{inter} implying that either $depth'(t'_j) < depth'(t_j)$ or $depth'(t'_{j'}) < depth'(t_{j'})$, contradicting the fact that $depth'(t) = depth'(t')$ should hold for every pair t, t' occurring in the cycle. Hence, such a directed cycle cannot exist. Therefore the resulting task is plan coordinated. \square

Example 5 Applying Algorithm 2 to the plan-uncoordinated tightly-coupled agents depicted in Figure 3(a) gives the following. The result of the first step is the partitioning $T^1 = \{t_1^s, t_2^s, t_3^s, t_4^s\}$ and $T^2 = \{t_1^e, t_2^e, t_3^e, t_4^e\}$. In the next three steps, precedence constraints are added that result in the following coordination sets: $\Gamma_1^{\prec} = \{t_1^s \prec t_4^e, t_4^s \prec t_1^e\}$ and $\Gamma_2^{\prec} = \{t_2^s \prec t_3^e, t_3^s \prec t_2^e\}$. Then, in the fifth step, the potential conflict with synchronisation constraints in depth partition T^1 is detected and resolved by adding the synchronisation constraints $\Gamma_1^{\equiv} = \{t_1^s \equiv t_4^s\}$, and the corresponding $\Gamma_2^{\equiv} = \{t_2^s \equiv t_3^s\}$. Adding these constraints results in the tightly-coupled task depicted in Figure 4.

Note that, using only those time-point variables that are involved in an inter-agent constraint (i.e., $t_1^s, t_2^s, t_3^s, t_4^s$), the algorithm would have constructed

the coordination sets $\Gamma_1^{\equiv} = \{t_1^s \equiv t_4^s\}$ and $\Gamma_2^{\equiv} = \{t_2^s \equiv t_3^s\}$, which is also sufficient.

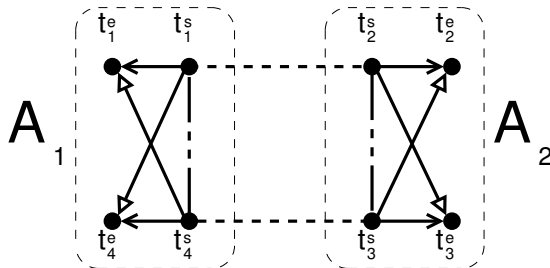


Figure 4: Two plan-coordinated tightly-coupled agents.

6 Related Work

In this paper, the plan-decoupling problem was studied as an approach to allow individual independent planning, while guaranteeing a jointly feasible plan. While previous work [2, 18] concentrated on the coordination problem in task-based planning where agents could make refinements of partial order constraints on tasks, in this paper we extended the framework to partial order and synchronisation constraints to deal with tightly coupled tasks.

This extension shows some relationships with current work on independent scheduling by autonomous agents: instead of individual planning, our agents might desire—or need—to independently choose the *dispatchings* themselves for their tasks. Although such a joint dispatching is guaranteed to exist after plan decoupling and individual planning, it does not hold that every combination of locally-feasible dispatchings is jointly feasible. Therefore, another coordination problem arises, because the combination of individually-chosen dispatchings are not guaranteed to be conflict free. When the agents are unwilling, or unable, to revise their constructed dispatchings, this *schedule-coordination* problem must be solved before the agents construct their dispatchings.

Similar to plan decoupling, a schedule-decoupling problem can be defined as a coordination mechanism for solving this coordination problem before individual dispatching. Here, instead of decoupling the agents' plans, the

agents' schedules are decoupled. Basically, this problem would be to minimally change the problem such that every combination of dispatchings that satisfies the local constraints is guaranteed to be globally conflict free.

These schedule-coordination problems are naturally formulated in *temporal networks* [5], where the difference between each pair of time-point variables can be constrained by a set of intervals. Such a temporal network can be viewed as the allowed set of dispatchings for executing the tasks it represents. Therefore, each assignment of values to time-point variables satisfying all constraints is a consistent dispatching for completing these tasks. Solving the schedule-decoupling problem in such a framework, then, requires a tightening of the constraints, subject to a minimum reduction of the *dispatching freedom* of the agents.

A variant of this problem is the TEMPORAL-DECOUPLING PROBLEM (TDP) [10] which is the problem of decoupling a partitioned *Simple Temporal Networks* (STN) [5]. Such an STN is a special class of temporal networks, where only one interval is allowed between pairs of time-point variables. Although schedule decoupling should be achieved by finding a cardinal minimum reduction of the dispatching freedom of the agents, the TDP only guarantees a subset minimal solution. Consequently, the developed polynomial-time algorithms for it produce approximate solutions that have no performance quality guarantees in terms of cardinal minimum decouplings of the agents.

It would be interesting to investigate the complexity of this schedule-decoupling problem and related decoupling problems in STNs, as well as in more general temporal networks.

7 Conclusions and Future Work

In this work, the plan-decoupling problem was studied as an approach to solving the plan-coordination problem, such that each combination of individually constructed plans is conflict free. Building on previous work, a richer framework was used for studying this plan-decoupling problem. It is now possible to constrain tasks with qualitative-temporal constraints, by using an interval representation of the tasks and defining precedence and synchronisation relations on the end points. Using this framework allowed a classification into loosely, moderately, and tightly-coupled tasks, which was based on the type of constraints used.

After assigning such tasks to a set of agents, these agents themselves can become interdependent, which gives rise to a coordination problem. We showed that, due to these interdependencies, combinations of locally-feasible plans can be globally infeasible. Then, the patterns were identified that cause these joint plans to be infeasible and, as such, cause agents to have conflicting plans. Although such infeasible plans can also be resolved by minimal change afterwards, it was shown how these conflicts can be prevented (i.e., before individual planning) such that the agents do not need to revise their individually constructed plans.

Taking a coordination-by-design approach, the plan-decoupling problem was defined, which is to prevent all possible conflicts by adding a minimum number of constraints. In fact, this plan-decoupling problem was studied for loosely, moderately, and tightly-coupled agents, separately.

It turned out that loosely-coupled agents are already plan coordinated, due to a lack of interdependencies. For moderately-coupled agents, the problem was to prevent all inter-agent cycles of the precedence ordering to occur. In previous work, it was shown that solving this problem by adding a minimum number of additional constraints is Σ_2^P -complete. In tightly-coupled agents, additional types of conflicts can occur caused by the synchronisation constraints (possibly in combination with the precedence constraints). However, it was proven that the complexity of the plan-decoupling problem remains Σ_2^P -complete when also preventing these possible conflicts. Since this problem is intractable, we discussed a polynomial approximation algorithm for tightly-coupled agents to provide sufficient—but not necessarily minimum—coordination solutions.

In future research, the application of this decoupling approach should be studied in different contexts. In specific, we plan on studying the complexity of solving the schedule-decoupling problem by minimally reducing the dispatching freedom of the agents.

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

- [1] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, Nov. 1983.
- [2] P. C. Buzing, A. W. ter Mors, J. M. Valk, and C. Witteveen. Coordinating self-interested planning agents. *Autonomous Agents and Multi-Agent Systems*, 12(2):199–218, Feb. 2006.
- [3] G. Christodoulou, E. Koutsoupias, and A. Nanavati. Coordination mechanisms. In J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3142 of *Lecture Notes in Computer Science*, pages 345–357. Springer, July 2004.
- [4] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.
- [5] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, May 1991.
- [6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, CA, USA, 1979.
- [7] B. J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.
- [8] J. R. Harrald. Supporting agility and discipline when preparing for and responding to extreme events. In B. van de Walle and B. Carlé, editors, *Proceedings of the 2nd International Conference on Information Systems for Crisis Response and Management (ISCRAM)*, Apr. 2005. keynote presentation, available at <http://www.iscram.org/>.
- [9] B. Horling, V. Lesser, R. Vincent, and T. Wagner. The soft real-time agent control architecture. *Autonomous Agents and Multi-Agent Systems*, 12(1):35–91, Jan. 2006.
- [10] L. Hunsberger. *Group Decision Making and Temporal Reasoning*. PhD thesis, Harvard University, Cambridge, MA, USA, June 2002.

- [11] T. Léauté and B. Williams. Coordinating agile systems through the model-based execution of temporal plans. In B. J. Clement, editor, *Proceedings of the Workshop on Multiagent Planning and Scheduling*, pages 22–28, June 2005.
- [12] V. R. Lesser, K. S. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. NagendraPrasad, A. Raja, R. Vincent, P. Xuan, and X. Q. Zhang. Evolution of the GPGP/TÆMS domain-independent coordination framework. *Autonomous Agents and Multi-Agent Systems*, 9(1):87–143, July 2004.
- [13] B. Nebel and H.-J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of allen’s interval algebra. *Journal of the ACM*, 42(1):43–66, 1995.
- [14] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165–200, May 1998.
- [15] S. F. Smith, A. Gallagher, T. Zimmerman, L. Barbulescu, and Z. Rubinstein. Distributed management of flexible times schedules. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 472–479, May 2007.
- [16] J. R. Steenhuisen, C. Witteveen, A. W. ter Mors, and J. M. Valk. Framework and complexity results for coordinating non-cooperative planning agents. In K. Fischer, I. J. Timm, E. André, and N. Zhong, editors, *Proceedings of the 4th German Conference on Multi-Agent System Technologies (MATES)*, volume 4196 of *Lecture Notes in Artificial Intelligence*, pages 98–109, Berlin, Germany, Sept. 2006. Springer.
- [17] J. R. Steenhuisen, C. Witteveen, and Y. Zhang. Plan-coordination mechanisms and the price of autonomy. In *Proceedings of the 8th Workshop on Computational Logic in Multi-Agent Systems (CLIMA)*, Lecture Notes in Artificial Intelligence, Berlin, Germany, Sept. 2007. Springer-Verlag.
- [18] J. M. Valk. *Coordination among Autonomous Planners*. PhD thesis, Delft University of Technology, Delft, The Netherlands, Dec. 2005.
- [19] R. M. Zlot and A. Stentz. Market-based multirobot coordination for complex tasks. *International Journal of Robotics Research, Special Is-*

sue on the 4th International Conference on Field and Service Robotics,
25(1):73–101, Jan. 2006.