Plan Coordination for Durative 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 domains, agents can be given planning or scheduling autonomy through coordination. However, plan coordination discards available scheduling information, while schedule coordination possibly over-constrains the problem from a planning point of view. In an attempt to attack this problem, we included more temporal information used in schedule coordination into the plan-coordination framework. We discovered that plan coordination can be achieved when reasoning with qualitative temporal constraints and durative tasks. Additionally, we defined this new coordination problem and studied its complexity. The relevance of this work is that it studies plan coordination in a wider context, such that the agents can–through coordination–be provided with both planning and scheduling autonomy.

Introduction

Agents are being introduced in a wide variety of task domains, because they promise to increase *agility*. These agent systems are emerging in such diverse domains as multi-modal transportation (Chiu *et al.* 2005), firefighting with unmanned aerial vehicles and air traffic control (Léauté & Williams 2005), and crisis response (Harrald 2005). In general, the tasks that need to be performed in these domains are interdependent, require more than one agent to execute them, require a task-planning process for every individual agent, and need to be completed on time. Obviously, due to the task interdependencies, some form of *coordination mechanism* is needed to ensure that the results of the task-planning processes are jointly feasible. In general, such a coordination mechanism (Christodoulou, Koutsoupias, & Nanavati 2004) should enable individual agents to choose their preferred way to solve their part of the task, thereby (minimally) reducing the initial (planning) autonomy of the agents. Therefore, the quality of a coordination mechanism depends on both the severeness of the restrictions imposed and the overall performance quality it allows. In this paper, we propose and analyse the (computational complexity) properties of such a coordination mechanism for interrelated tasks with time constraints. Before we introduce the basics of our coordination framework, we will place it in a more general perspective.

Background First of all, the reader should be aware that there are several existing approaches to solve this *plan-coordination problem*. For example, in (Smith *et al.* 2007) the authors propose to manage the coordinated planning and execution of the tasks by letting the agents keep each other informed about any changes (e.g., completed, new, or rescheduled tasks). It is clear that such an approach requires the agents to be *collaborative*, each agent willing to inform other agents about details of its individual plan and cannot be used if the agents are competitive or self-interested. Moreover, this mechanism will fail when communication is impossible or difficult to establish.

In this paper, we advocate a more principled approach to the plan-coordination problem. First of all, we distinguish different *phases* in a multi-agent planning process each requiring some form of coordination. Second, we distinguish some specific *interrelations* between tasks and agents that determine which form of coordination is required in the (multi-agent) planning process. Together, these phases and interrelationships determine which form of coordination is required in which phase of the planning process (see also (Zlot & Stentz 2006)).

To start with the four main phases in the planning process, in the *allocation phase*, tasks are assigned to agents that are 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. Last, we have the *execution phase* in which the tasks are executed according to the constructed schedule.

The interrelationship between tasks and agents determine in which phase plan coordination is needed and which form of plan coordination should be applied. First, a set of tasks M is called *loosely coupled* with respect to a set of agents $\mathcal{A}$ when the tasks occurring in M can be assigned to the agents $A \in \mathcal{A}$ such that there exist no dependency relations between tasks assigned to different agents. Clearly, if we have a loosely-coupled system, each agent is able to construct an independent plan for its subset of tasks. Therefore, coordination is not needed in the planning, scheduling, and execution phase and reduces to solving a *task-allocation* problem. Examples of this category are tasks that are totally independent of each other, such as searching for casualties in different parts of a city.

Second, if the set of tasks M is partially ordered by some dependency relation, and it is impossible to assign the tasks such that tasks assigned to different agents in $\mathcal{A}$ have no dependency relation between them, then M is said to be *moderately coupled*. For these complex tasks, plan coordination is required before or during the planning phase in order to ensure that the induced partially-ordered dependency relation between agents is preserved by the individual planning processes. There is, however, no a-priori need to provide a coordination mechanism in the scheduling or execution 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 of tasks M is moderately coupled and, moreover, requires the satisfaction of constraints (e.g., time constraints) when scheduling and executing the tasks, the set of tasks is said to be *tightly coupled*. Here, coordination is required in the planning, scheduling, and execution phase. 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.

Plan coordination: previous and current work In previous work (Steenhuisen *et al.* 2006; Buzing *et al.* 2006), we concentrated on the the plan-coordination problem for moderately-coupled tasks. Basically, in these papers, we showed that there exists a plan-coordination mechanism (although difficult to design) that is able to reduce a moderately-coupled task to a loosely-coupled task. In other words, such a plan-coordination mechanism is able to guarantee the participating agents that they can plan completely independent from the other agents while still guaranteeing the feasibility of the joint plan. This approach is robust against failing communication, allows individual replanning, and additional tasks to be done by the agents while not violating global constraints. Moreover, this approach is applicable in many other domains where agents are unwilling or unable to revise their plans.

In more recent work (Steenhuisen & Witteveen 2007), we have extended this approach to tightly-coupled tasks with dependency constraints and synchronisation constraints. We showed that there exist plan-coordination mechanisms that guarantee independent planning by the individual agents, but require information exchange after planning to establish the exact time of scheduling synchronisation tasks.

In this paper, we will extend the planning approach, showing that we can also provide coordination mechanisms for *durative tasks* with time constraints. The coordination mechanisms developed ensure the participating agents that whatever feasible plan they provide for their own set of tasks, there always exists a schedule for the joint plan obtained by assembling the individual plans. However, this does not mean that all locally feasible schedules are guaranteed to form a feasible joint schedule. In this paper, we reduce the construction from previously developed coordination mechanisms to those with durative tasks and time constraints, which implies that the latter are at least as hard to design as the former mechanisms.

Framework

We consider a set of agents $\mathcal{A} = \{A_1, \ldots, A_n\}$ that have to complete a complex task $\mathcal{T}$. Such a complex task consists of a set $M = \{m_1, \ldots, m_k\}$ of methods m_i, together with a set of constraints on the execution of these methods. These constraints can be partitioned into a set of *dependency* constraints, determining for each method the set of methods it is dependent upon and a set of *synchronisation* constraints determining which methods m_j should be executed concurrently with a given method m.

For a uniform representation also suitable for temporal planning aspects, we use a set of time points T and binary relations between them to represent methods and relations between them. First of all, each $m_i \in M$ is represented by an ordered pair $(t_{i,s}, t_{i,e})$ where $t_{i,s}$ is the time point indicating the starting time of m_i and $t_{i,e}$ its ending time. Dependencies and synchronisation relations between methods can now be represented by relations between time points as follows. We distinguish a partial order $\prec \subseteq (T \times T)$ representing the dependency relation and an equivalence relation $\equiv \subseteq (T \times T)$ for synchronisation. More precisely, if a method m_j depends on method m_i, there is a dependency constraint $t_{i,e} \prec t_{j,s}$. Furthermore, for every method m_i, $t_{i,s} \prec t_{i,e}$. If m_i is synchronised with m_j, then $t_{i,s} \equiv t_{j,s}$ and $t_{i,e} \equiv t_{j,e}$. Note that since $\equiv$ is an equivalence relation, each time point $t \in T$ is synchronised with itself.

Finally, the composition of $\prec$ and $\equiv$ satisfies the following two natural inclusion properties.

1. $(\prec \circ \equiv) \subseteq \prec$ and $(\equiv \circ \prec) \subseteq \prec$, which means that $(t \prec t' \wedge t' \equiv t''$ implies $t \prec t'')$ and $(t \equiv t' \wedge t' \prec t''$ implies $t \prec t')$, and
2. $(\prec \cap \equiv) = \emptyset$, meaning that $\prec$ and $\equiv$ are orthogonal relations.

Temporal relations and classification of complex tasks In many planning domains (e.g., airport planning, manufacturing, and supply-chain management), we have to account for temporal constraints that constrain the execution of a method relative to the execution of other methods. Seven[1] of such *qualitative temporal constraints* have been identified for (qualitative) time intervals: `before`, `overlaps`, `during`, `meets`, `starts`, `finishes`, and `equals` (Allen 1983). In previous work (Steenhuisen & Witteveen 2007), we showed that all these qualitative temporal constraints can be represented in a task framework with time points, together with the above precedence and synchronisation constraints.

Also, a clear distinction is emerging on the degree to which methods can be coupled using these constraints. On the one hand, we have the constraints that require both precedence and synchronisation constraints (i.e., `meets`, `starts`, `finishes`, and `equals`), while others only need precedence constraints (i.e., `before`, `overlaps`, and `during`). We call the complex tasks in which the end points of methods are constrained by *both* precedence and synchronisation constraints *tightly-coupled* tasks, when *only* prece-

[1]Neglecting the converse of each of these relations.

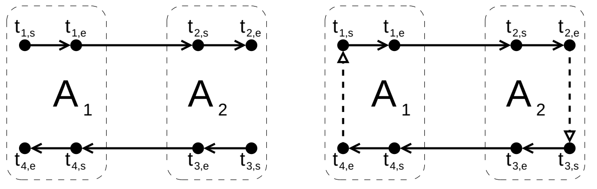


Figure 1: An uncoordinated moderately-coupled task.

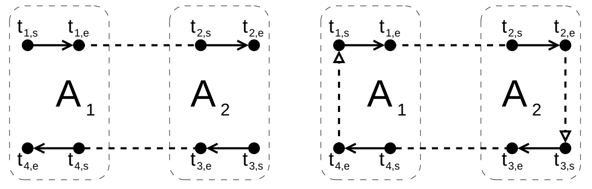


Figure 2: An uncoordinated tightly-coupled task.

dence constraints are used *moderately-coupled* tasks, and *loosely-coupled* tasks when methods are not constrained at all. This classification corresponds to the distinction made in the introduction.

Before we discuss an extension of this framework with time windows and durations, we first give an overview of the results already obtained for coordinating moderately and tightly-coupled tasks.

Plan coordination for moderately-coupled tasks

Note that in moderately-coupled tasks, we have a set of time points T and a partially-ordered precedence relation $\prec$ between time points. Each method $m_i \in M$ is represented by an ordered pair $(t_{i,s}, t_{i,e})$ of time points. We assume the original set M to be partitioned into n disjoint sets M_i, representing the subset[2] of methods to be executed by agent A_i. Therefore, the complex task assigned to A_i can be represented by a partial order $\langle T_i, \prec_i \rangle$, where T_i is the subset of time points associated with M_i, and $\prec_i$ the partial order $\prec$ restricted to T_i.

The agents are assumed to be independent and self-interested planners, able to reason with all given information. Therefore, every agent A_i is allowed to come up with an individually chosen plan, ordering the methods it received, as long as it is compatible with the original constraints $\prec_i$. That implies that every plan $\langle T_i, \prec_i^* \rangle$ could be put forward by an agent A_i as long as $\prec_i^*$ is a partial order *extending* $\prec_i$.

It is easy to see that, in many cases, not every combination of such autonomously chosen plans will result in a jointly feasible plan. For example, in Figure 1(a), a complex task is shown where agents A_1 and A_2 can plan $m_4 \prec m_1$ and $m_2 \prec m_3$ (see Figure 1(b)). But when these plans are joined, a cycle $\langle m_1, m_2, m_3, m_4, m_1 \rangle$ is introduced. Such a cycle indicates an infeasible joint plan, since it implies m_1 to precede m_2, but also vice versa. Since such an infeasible combination of individually feasible plans is possible, we call this complex task *uncoordinated*.

Thus, a plan-coordination mechanism should ensure that whatever feasible plans are chosen by the individual agents A_i, the joining of these plans constitutes a feasible global plan for the original set of methods M, satisfying all dependencies. Such a plan-coordination mechanism then should prevent every potential inter-agent cycle.

[2]How to find a suitable assignment for a set of agents is a separate problem (Zlot & Stentz 2006; Shehory & Kraus 1998), and is beyond the scope of this paper.

Plan coordination can be achieved by *plan decoupling* (Valk 2005): Adding precedence constraints to the set of time points of each agent such that every agent A_i is allowed to autonomously construct a plan for its set M_i of methods, respecting only the set $\prec_i$ of local constraints and the existence of a feasible joint plan is always guaranteed.

The *plan-decoupling problem* (PDP) is to find a minimum set of such additional constraints.

PDP FOR MODERATELY-COUPLED TASKS
INSTANCE: A moderately-coupled task $\langle \{T_i\}_{i=1}^n, \prec \rangle$ and a positive integer K.
QUESTION: Does there exist a coordination set Γ with $|\Gamma| \leq K$ such that $\langle \{T_i\}_{i=1}^n, (\prec \cup \Gamma)^+ \rangle$ is coordinated?[3]

This problem, as well as some of its variants, has been studied quite extensively. It turns out that this problem is Σ_2^p-complete in general (Valk 2005), and NP-complete when the number of agents is bounded by some constant (Steenhuisen *et al.* 2006). In addition, it was shown that this decoupling problem is APX-hard, and that a constant-ratio approximation algorithm is not likely to exist (Valk 2005). There exists a simple polynomial-time algorithm that finds a sufficient—but not necessarily minimum—coordination set for distributed tasks with precedence constraints. For some restricted cases of plan coordination, this algorithm has even been shown to be a constant-ratio approximation algorithm (ter Mors, Valk, & Witteveen 2006).

Plan coordination for tightly-coupled tasks

Analogous to moderately-coupled tasks, the coordination problem for a tightly-coupled task $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$ occurs if methods in some set M are assigned to agents such that each agent A_i has to complete a part $\langle T_i, \prec_i, \equiv_i \rangle$ of it. However, since some methods need to be synchronised, the agents want to find individually suitable plans that they do not need to revise when agreeing upon a joint schedule for the set of methods. The coordination problem for tightly-coupled tasks then is how to ensure that, whatever individually feasible plan is chosen for an agent's complex task, there will always exist a joint schedule for the total set of methods that satisfies each of the individual plans.

Clearly, in order to complete his part $\langle T_i, \prec_i, \equiv_i \rangle$ each agent A_i can choose a plan $\pi_i = \langle T_i, \prec_i^*, \equiv_i^* \rangle$ for it, where $\prec_i \subseteq \prec_i^*$ and $\equiv_i \subseteq \equiv_i^*$. Such a plan can simply be conceived as a *refinement* of the partially-ordered set $\langle T_i, \prec_i, \equiv_i \rangle$.

[3]For any relation ρ, the transitive closure is denoted by ρ^+.

The *joint plan* for a set of agents $\mathcal{A}$ on a tightly-coupled task $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$ is a plan $\pi = \langle \{T_i\}_{i=1}^n, \prec^*, \equiv^* \rangle$ where *(i)* each plan $\pi_i = \langle T_i, \prec_i^*, \equiv_i^* \rangle$ of agent A_i is respected, i.e., $\prec_i^* \subseteq (\prec^* \cap (T_i \times T_i))$, *(ii)* $\prec \subseteq \prec^*$, and *(iii)* $\equiv \subseteq \equiv^*$.

An individual schedule $s_i : T_i \to \mathbb{Z}$ is said to satisfy the individual plan $\pi = \langle \{T_i\}_{i=1}^n, \prec^*, \equiv^* \rangle$ of agent A_i if the following conditions hold:

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

Finally, a set of individual schedules $\{s_i\}_{i=1}^n$ of individual plans π_i constitutes a joint schedule if the following holds:

1. $\forall t \in T_i, t' \in T_j$: $t \prec t'$ implies $s_i(t) < s_j(t')$, and
2. $\forall t \in T_i, t' \in T_j$: $t \equiv t'$ implies $s_i(t) = s_j(t')$.

Now, we say that a tightly-coupled task $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$ assigned to a set of agents $\mathcal{A}$ is coordinated if for every combination $\{\pi_i\}_{i=1}^n$ of individually chosen feasible plans there exist a set of schedules $\{s_i\}_{i=1}^n$ such that each s_i satisfies π_i and $\{s_i\}_{i=1}^n$ constitutes a joint schedule.

Analogous to the moderately-coupled case, coordination for tightly-coupled tasks can be achieved by plan decoupling, that is finding a (minimum) set of additional constraints that allow agents to plan autonomously while guaranteeing the existence of a joint schedule based on their individually chosen plans. The associated problem is the following plan-decoupling problem.

PDP FOR TIGHTLY-COUPLED TASKS
INSTANCE: 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 Γ with $|\Gamma| \leq K$ such that $\langle \{T_i\}_{i=1}^n, (\prec \cup \Gamma)^+, \equiv \rangle$ is coordinated?

Recently, we showed that PDP for tightly-coupled tasks is Σ_2^p-complete (Steenhuisen & Witteveen 2007). Furthermore, we showed that, with some minor modifications, the same approximation algorithm can be used as for moderately-coupled tasks.

Surprisingly, viewing the problem computationally, coordinating moderately or tightly-coupled tasks does not differ significantly.

Plan coordination of Durative Task Networks

In the previous section, we described our framework for studying PDP, and summarised some relevant achieved results. The most elaborate complex tasks on which PDP was studied, are the tightly-coupled tasks. Although the qualitative temporal constraints can be used in these tasks already, it is not possible to use any form of *quantitative* temporal constraint. In this section, we make a start at closing this gap by introducing both *time windows* on the methods and *durations* of methods to the framework.

We formally represent our extended tightly-coupled tasks as a tuple $\langle \{T_i\}_{i=1}^n, \prec, \equiv, I, \delta \rangle$, and refer to it as a *Durative Task Network* (DTN). Here, the first three entries are equal to those used in tightly-coupled tasks. Then, for each time point t, there is a time window (or temporal interval) $I(t) = (lb(t), ub(t))$, with $lb(t) < ub(t)$, $lb(t) \in \mathbb{Z} \cup \{-\infty\}$ and $ub(t) \in \mathbb{Z} \cup \{\infty\}$. The most relaxed time window is the *universal time window*, $(-\infty, \infty)$, which bounds are used when no lower or upper bound is provided. Furthermore, each method m_i has a certain *fixed* duration $\delta(m_i) \in \mathbb{Z}^+$ that represents the temporal distance from $t_{i,s}$ to $t_{i,e}$. Clearly, it must hold for method m_i that $ub(t_{i,e}) - lb(t_{i,s}) \geq \delta(m_i)$. Note that a time window $[x, y]$ can be rewritten to $(x - \epsilon, y + \epsilon)$, where ϵ is the smallest temporal distance between two time points.

Without loss of generality, we assume the following intuitive properties to hold for each DTN:

- If $t \prec t'$ then $lb(t') := \max(lb(t) + \epsilon, lb(t'))$ and $ub(t) := \min(ub(t), ub(t') - \epsilon)$,
- if $t \equiv t'$ then $I(t) := I(t') := [\max(lb(t), lb(t')), \min(ub(t), ub(t'))]$, and
- for each method m_i, $ub(t_{i,s}) := \min(ub(t_{i,s}), ub(t_{i,e}) - \delta(m_i))$ and $lb(t_{i,e}) := \max(lb(t_{i,s}) + \delta(m_i), lb(t_{i,e}))$.

A DTN for which these conditions hold, is called *normalised*. Henceforth, we assume each DTN to be normalised unless stated otherwise. Note that the time windows of $t_{i,s}$ and $t_{i,e}$ have the same width, and that this width is the available slack. We need $\epsilon \leq \min_i \delta(m_i)$, but we can assume, without loss of generality, that the smallest temporal distance is $\epsilon = 1$. Others have already reported on great gains in practise for scheduling by using these tighter bounds (Sultanik, Modi, & Regli 2006).

A feasible *schedule* $s : T \to \mathbb{Z}$ for a DTN $\langle \{T_i\}_{i=1}^n, \prec, \equiv, I, \delta \rangle$ is defined analogously to a schedule for tightly-coupled tasks with the additional requirement that for every $t \in T$, $lb(t) < s(t) < ub(t)$ and for every method m_i, $s(t_{i,e}) - s(t_{i,s}) = \delta(m_i)$.

A DTN $\langle \{T_i\}_{i=1}^n, \prec, \equiv, I, \delta \rangle$ is called *globally consistent* if there exists at least one schedule s for it. If the DTN is a single agent DTN $\langle T_i, \prec_i, \equiv_i, I, \delta \rangle$ it is called *locally consistent* if there is at least one feasible schedule for it. A DTN $\langle \{T_i\}_{i=1}^n, \prec, \equiv, I, \delta \rangle$ is called *coordinated* if it is globally consistent for every combination of extensions $\langle M_i, \prec_i^*, \equiv_i^* \rangle$ of the agent's DTNs that are locally consistent.

Reduced DTNs Remember that plan decoupling for moderately-coupled tasks in fact reduced a coordination problem for moderately-coupled tasks to loosely-coupled tasks, by allowing the agents to plan autonomously.

Before we analyse the decoupling problem for DTNs, in this paragraph, we show that every DTN instance can be reduced to a *reduced* DTN instance without time windows, obtaining a tightly-coupled task with durations. The idea is to introduce a new time agent A_0 that represents an absolute time line, which is a totally-ordered set of time points. Although more time points can be mentioned on that time line, we at least need the upper and lower bounds used in the time windows. Because we defined our time windows as $(lb(t), ub(t))$, precedence constraints can be used to constrain the occurrence of any time point t allowed during execution by $t_{lb(t)} \prec t \prec t_{ub(t)}$.

We now give a more formal reduction in which the time window constraints are replaced by a a set of precedence constraints using an additional time agent.

Given an instance $\langle \{T_i\}_{i=1}^n, \prec, \equiv, I, \delta \rangle$, we first collect all the lower and upper bounds of the time windows $I(t)$. More precisely, for each time point t, the time window $I(t) = (lb(t), ub(t))$ is coded into two values s and s', where $s = t_{lb(t)}$ and $s' = t_{ub(t)}$. We collect the total set of all these time values in the set $T_0 = \{s_1, \ldots, s_p\}$ where $s_1 < \cdots < s_p$. Now, we construct the following coordination instance $\langle \{T_i\}_{i=0}^n, \prec', \equiv', \delta \rangle$, where

1. The partitioned set of tasks is extended to $\{T_0\} \cup \{T_i\}_{i=1}^n$.
2. The precedence relation $\prec'$ equals $\prec$ extended with the total ordering imposed on T_0. Moreover, for each task t, with associated time window $I(t) = (s, s')$, two additional precedence constraints $s \prec t$ and $t \prec s'$ are constructed. The result then is $\prec' = \prec \cup \{s_i \prec s_j \mid 1 \le i < j \le p\} \cup \{s \prec t, t \prec s' \mid t \in \bigcup_{i=1}^n T_i, s = lb(t), s' = ub(t)\}$.
3. The synchronisation relation $\equiv'$ equals the relation $\equiv$ extended with the tuples $t \equiv s_i$, for each time point t with $I(t) = (s_i - \epsilon, s_j + \epsilon)$ and $s_i = s_j$.

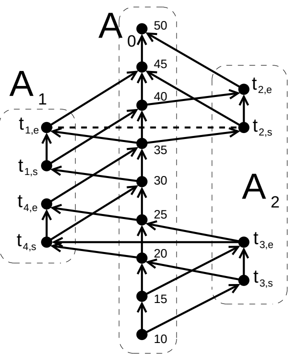


Figure 3: Reducing a DTN with time windows to a DTN tightly-coupled task.

Example 1 *As an illustration, consider the tightly-coupled task depicted in Figure 3. In fact, this complex task is the result of applying the above reduction to a DTN. Clearly, all time windows have been replaced by precedence constraints to the totally-ordered time line of agent* A_0.

Although the complex task in this example is already coordinated, it is not hard to come up with examples that are uncoordinated. For instance, such an uncoordinated task can be constructed by extending the moderately-coupled task given in Figure 1(a) as follows. Let all methods m_i *have a duration of* $\delta(m_i) = 3$, *and constrain each time point with time windows* $(17, 43)$. *Clearly, this task is uncoordinated because it contains the possible deadlock that is shown in Figure 1(b). However, by adding the constraint* $m_4 \prec m_2$, *all potential inter-agent cycles are prevented.*

In general, a coordination set Γ for a reduced DTN also is a coordination set for the original task with time windows. This can easily be seen by noting that a coordination set only contains precedence intra-agent precedence constraints, the time agent A_0 is totally ordered, and no time points are added to the set of agents $\mathcal{A}$.

It almost suffices to only use time points with precedence and synchronisation constraints. However, for each method assigned to an agent, we need an ordered pair of time points *and* its associated duration.[4] Unfortunately, this cannot be represented by using relations among time points alone, and need to label the durative arcs.

As an example, reconsider the moderately-coupled task in Figure 1(a), where all time points are constrained to time window $(17, 43)$. The possible coordination sets Γ are $\{t_{1,e} \prec t_{4,s}\}$, $\{t_{3,e} \prec t_{2,s}\}$, and $\{t_{1,e} \prec t_{4,s}, t_{3,e} \prec t_{2,s}\}$. For example, if we have $\delta(m_1) = 19$, $\delta(m_2) = 1$, $\delta(m_3) = 2$, and $\delta(m_4) = 3$, then $\{t_{1,e} \prec t_{4,s}\}$ does not coordinate the task, because agent A_2 can plan $m_2 \prec m_3$. Here, the task is inconsistent and, therefore, not coordinated, because task m_4 must not be scheduled to start earlier than $lb(t_{1,s}) + \epsilon + \delta(m_1) + \epsilon + \delta(m_2) + \epsilon + \delta(m_3) + \epsilon = 17+1+19+1+1+1+2+1 = 43$ (based on the durations and $lb(m_1)$), and be completed before 43. Obviously, there is no schedule in which m_4 meets these constraints.

The key problem with DTNs is the interaction of time windows and durations of the methods. As we have already seen, it is not possible to discard the method durations when solving the coordination problem. A partial solution to this problem might be to discover all implied constraints first, or otherwise to adapt the problem to prevent these incorrect coordination sets.

So, instead of discarding the duration information, we need to use it for temporal constraint propagation and elicitation. This can tighten time windows which can, in turn, result in new precedence constraints because time windows become non overlapping. As an example, we consider the normalisation of a DTN for an agent A_1, that is assigned methods m_2, m_5 that are constrained by time windows. In Figure 4, the result is shown after propagating the available temporal information. Here, the constraint $t_{5,s} \prec t_{2,e}$ emerges through non-overlapping time windows $(20, 25)$ and $(28, 65)$.

Note that these constraints are implied without making any assumptions on the degree of parallelism available to that agent. In terms of qualitative temporal constraints, this additional constraint means that m_2 `before` m_5 is excluded. Now, the agent can reason whether other constraints hold due to, for instance, the degree of parallelism available. If an agent is strictly sequential, than it also discards all constraints with (partially) overlapping execution, such that only m_5 `before` m_2 remains. Adding such precedence constraints requires time windows to be tightened again, and possibly resulting in new local precedence constraints.

Plan coordination with durative tasks Using DTNs, we can represent methods as (durative) intervals, makes it possible to use qualitative temporal constraints on methods, and allows time windows to constrain the time points to absolute

[4]This is also possible for our newly introduced time agent.

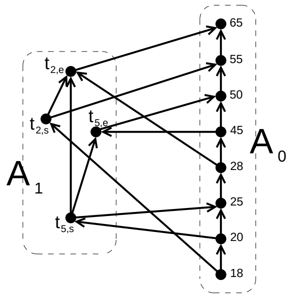


Figure 4: Retrieving implied precedence constraints explicit.

time. Note that DTNs have much in common with *Simple Temporal Networks* (STNs) (Dechter, Meiri, & Pearl 1991). Before continuing, we briefly describe STNs and show that DTNs are in fact a restriction of STNs.

Formally, an STN $\mathcal{S}$ is written as a tuple $\langle T, C\rangle$, where T is the set of time points $\{z, t_1, \ldots, t_m\}$, and C is a finite set of binary constraints on those time points. The time point z represents an arbitrary fixed reference point on the time line, commonly referred to as the zero time point. Each constraint $c \in C$ has the form $\delta_{lb} \leq t_j - t_i \leq \delta_{ub}$ for some $\delta_{lb} \in \mathbb{R} \cup \{-\infty\}, \delta_{ub} \in \mathbb{R} \cup \{\infty\}$, and commonly is represented as $\delta(t_i, t_j) = [\delta_{lb}, \delta_{ub}]$.

We show that DTNs are in fact restrictions of STNs, by giving a transformation of a DTN $\langle \{T_i\}_{i=1}^n, \prec, \equiv, I, \delta\rangle$ to its representation as an STN $\langle T, C\rangle$. First, the time points in an STN consist of the time points from the DTN together with the zero time point z: $T = \{z\} \cup \bigcup_{i=1}^n T_i$. Second, it is assumed that all pairs of time points are constrained by the universal temporal constraint $\delta(t, t') = (-\infty, \infty)$, but can be tightened in the following way.

1. $\forall i, j\, \forall t \in T_i, t' \in T_j : (t, t') \in \prec$ implies $\delta(t, t') = [1, \infty) \in C$,
2. $\forall i, j\, \forall t \in T_i, t' \in T_j : (t, t') \in \equiv$ implies $\delta(t, t') = [0, 0] \in C$,
3. $\forall i\, \forall t \in T_i : I(t) = (lb(t), ub(t))$ implies $\delta(z, t) = (lb(t), ub(t)) \in C$, and
4. $\forall m \in M : \delta(t_s, t_e) = [\delta(m), \delta(m)] \in C$.

In the previous paragraph, we showed that a DTN with time windows can be reduced to a tightly-coupled task with durations, without any new difficulties being introduced. In this paragraph, we analyse the consequences of the added duration to the reasoning framework.

We start by verifying whether such DTNs can be coordinated at all in a similar way as the PDP for moderately- and tightly-coupled tasks. Thereafter, we study the complexity of actually determining a coordination set, and look at the issues that need to be taken into account.

Proposition 1 *For any globally consistent DTN* $\mathcal{D} = \langle \{T_i\}_{i=1}^n, \prec, \equiv, I, \delta\rangle$*, there exists a coordination set* $\Gamma = (\Gamma_\prec \cup \Gamma_\equiv)$ *such that the resulting DTN is coordinated.*

Proof Since $\mathcal{D}$ is globally consistent, there exists at least one feasible schedule s for it. Define the set $\Gamma_\prec$ as follows: $\forall i$: if $t, t' \in T_i$, $s(t) < s(t')$ iff $(t, t') \in \Gamma_\prec$, and define $\Gamma_\equiv$ as follows: $\forall i$: if $t, t' \in T_i$, $s(t) = s(t')$ iff $(t, t') \in \Gamma_\equiv$. It is not difficult to see that $\mathcal{D}' = \langle \{T_i\}_{i=1}^n, (\prec \cup \Gamma_\prec)^+, (\equiv \cup \Gamma_\equiv)^+, I, \delta\rangle$ is a globally consistent DTN (since s satisfies all constraints). Moreover, it is not difficult to see that every agent's DTN $\mathcal{D}'_i = \langle T_i, (\prec_i \cup \Gamma_{\prec,i})^+, (\equiv_i \cup \Gamma_{\equiv,i})^+, I, \delta\rangle$ derived from $\mathcal{D}'$ has itself as its unique extension. Hence, $\mathcal{D}'$ is coordinated. □

In order to minimise the loss of freedom for the individual agents, we again have to identify a smallest set of additional constraints. We call this the PDP for DTNs, for which we define the recognition problem as follows.

DTN-COORDINATION RECOGNITION (DTN-CR)
INSTANCE: Globally consistent DTN $\mathcal{D}$ partitioned into n agent's DTNs.
QUESTION: Does it hold that the DTN is globally consistent for every set of locally-consistent extensions of individual DTNs $\mathcal{D}_i$?

Lemma 1 DTN-CR *is* CONP-*complete.*

Proof Membership is shown by noting that a no-certificate is verified in polynomial time. A no-certificate contains a set of precedence and synchronisation constraints that are added to the DTN in polynomial time. As shown above, a DTN can be transformed into an equivalent STN in polynomial time, whose consistency can be checked in polynomial time.

Hardness is proven by a reduction from the PDP-Coordination Recognition problem for tightly-coupled tasks. Here, the idea is to associate time windows with time points that are wide enough such that every partial order is consistent (e.g., set all time windows to $(0, \infty)$). Additionally, we fix all durations to $\delta(m_i) = 1$. □

As a corollary, we can say that the more general coordination variant of DTN-CR (i.e., minimally change the DTN such that it is a yes-instance for DTN-CR) is Σ_2^p-complete. Intuitively, guessing a yes-certificate can be verified in polynomial time using a DTN-CR-oracle, which proofs membership. Hardness, on the other hand, can be proven using the same reduction as used in the lemma.

Checking a DTN's consistency is an important issue in these problem instances. For this paper, STNs are sufficient to express the temporal constraints, because the agent's degrees of parallelism are not bounded. In the following example, we show a complex task where unions of time windows are needed when this parallelism is bounded. However, consistency checking for such temporal networks is intractable (Dechter, Meiri, & Pearl 1991), and its consequences are part of future research.

Consider the DTN depicted in Figure 5. Here, we have three methods that each have a duration $\delta(m_i) = 3$, methods m_1, m_2 need to be scheduled within the time window $(10, 19)$ and m_3 in $(0, 30)$. When the agent has unbounded parallelism, no problems occur when adding precedence constraints. However, when the agent is a unit-capacity resource, the temporal network basically becomes a disjunctive temporal network due to m_3 becoming constrained to $(0, 11) \cup (18, 30)$.

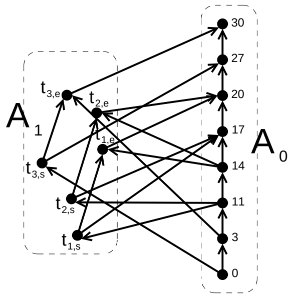


Figure 5: A union of available time windows can be implied when agent's parallelism is bounded.

Relation to Simple Temporal Networks

In the previous sections, we have taken a plan-decoupling approach to solving the plan-coordination problem. However, instead of decoupling agent's plans, we could as well decouple agent's schedules. This approach gives rise to the *Temporal-Decoupling Problem* (TDP) (Hunsberger 2002), which is defined on STNs.

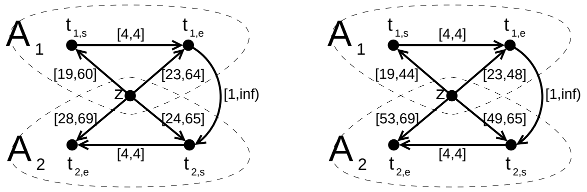
Basically, an STN is a set of time points and a set of binary constraints between those time points. Commonly, such an STN is represented as a directed graph with numerical values on its arcs. Here, the vertices represent the time points, while the arcs are upper and lower bounds on the temporal distance between two time points.

An STN can be viewed as the *set of schedules* for the task it represents. Therefore, a solution to an STN is a schedule for completing the methods (i.e., assignment of values to the time point variables such that all constraints are satisfied). An STN is *consistent* when at least one solution exists.

In Figure 6(a), we have two agents A_1, A_2 each having one method (i.e., two time points) that takes exactly 4 time units to complete, where m_1 needs to be scheduled in $[19, 64]$, and m_2 in $[19, 69]$. Moreover, the methods are constrained by m_1 `before` m_2, which translates to $t_{1,e} \prec t_{2,s}$. The constraints between the time points are given as temporal distance intervals $[\delta_{lb}, \delta_{ub}]$ on the arcs.

Note that tightening of temporal distances works in a similar way as the tightening of time windows in DTNs. Clearly, in this example, the agents cannot schedule independently due to the precedence constraint $1 \leq t_{2,s} - t_{1,e} < \infty$. In order to allow independent scheduling, we need to temporally decouple the agents. This is achieved when the constraint between $t_{1,e}$ and $t_{2,s}$ is implied by the constraints between z and $t_{1,e}$, and z and $t_{2,s}$, respectively. A typical solution for TDP on the STN of Figure 6(a) is depicted in Figure 6(b). Here, the inter-agent constraint has become implicit, because $23 \leq t_{1,e} - z \leq 48 < 49 \leq t_{2,s} - z \leq 65$ results in $1 = 49 - 48 \leq t_{2,s} - t_{1,e} \leq 65 - 23 = 42 < \infty$.

Basically, the difference between PDP and TDP is the following. In TDP, the problem is to tighten the time windows of the time points in such a way that the inter-agent temporal differences are guaranteed to hold (e.g., by making them implicit) and the joint schedule to be feasible, whatever *schedules* is constructed by the individual agents. In PDP, the problem is to reduce the local planning freedom such that the joint plan is guaranteed to be feasible and to be consistent, whatever *plans* is constructed by the individual agents.



(a) Partitioned (b) Decoupled

Figure 6: Example of temporally decoupling an STN with two agents.

Clearly, there is a close resemblance between STNs and DTNs. For instance, both frameworks use time points and binary constraints between them as basic constructs, and time windows on time points in DTNs are represented as constraints between z and that time point in STNs. The major difference between the two representations is that in STNs every pair of time points can be constrained by a temporal distance interval, where in DTNs only fixed temporal distances are allowed between time points within an agent.

Although the frameworks in which these problems are studied show much likeness, the problems themselves largely differ. Even the basic approach, as in a decoupling, is very similar, although one plan-decouples the agents by guaranteeing the existence of *at least one schedule*, while in temporal-decoupling each agent is left with its decoupled part of the STN which represents the *set of schedules*. From a complexity point of view, there is a big difference, because PDP is Σ_2^p-complete while solving a TDP is in P. In fact, we believe that TDP over-constrains complex tasks in order to coordinate them, in the sense that it reduces the number of possible plans more than needed.

Conclusions and Future Work

Plan coordination is needed to guarantee that using local planning autonomy does not cause conflicts to the global goal. Here, 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. In many domains, however, communication can be lost or difficult to establish or maintain, or agents are unwilling to revise their plans. Therefore, interleaved and post-planning coordination are not always applicable, and we chose to take a pre-planning approach to coordination.

In this paper, we presented a framework for modelling complex tasks. Using precedence and synchronisation constraints to constrain the relative execution of time points, all qualitative temporal constraints can be used to constrain two methods. This framework was further extended to allow methods to have a certain fixed duration, and to be executed within a certain time window. We discovered that plan coordination can be achieved when reasoning with qualitative temporal constraints and durative tasks, and defined and studied the complexity of this new coordination problem.

With respect to the use of temporal constraints, existing work on plan coordination has been rather limited. Although the described temporal-decoupling problem is a good start at coordinating problems with quantitative temporal information, its use is rather limited. First, it is a schedule-coordination approach that reduces the planning freedom to a greater extend than necessary. Second, the used STNs are a small subset of instances that can be defined in temporal constraint networks as used in general temporal networks (Dechter, Meiri, & Pearl 1991), where it is possible to represent arbitrary intervals of temporal distances between time points. 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 quantitative and qualitative temporal information.

In future research, some additional steps are needed to make our approach applicable to real-life problems in real-time. First, in many domains, uncertainties (e.g., on duration) need to be taken into account, for which an extension of STNs has been developed, called STNUs (Vidal & Fargier 1999). Second, until now, we have only been concerned with static problem instances. Reality, however, is much more dynamic than these situations, and a technique is needed to deal with this additional dynamism online. In (Hunsberger 2003), STNs are adapted to *Augmented STNs* (ASTNs) to cope with the passing of time, in which new methods and constraints can be inserted into an existing ASTN.

Finally, the plan-coordination problems should be formulated as optimisation problems, because this corresponds more closely to reality. In order to do this, we need to look at different criteria for the optimal coordination set. It is not hard to come up with examples where one coordination set tightens time windows, while another coordination set does not tighten any time window. Clearly, the latter solution reduces the scheduling freedom less than the first, in an absolute sense, and is likely to be preferred.

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

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11):832–843.

Buzing, P. C.; ter Mors, A. W.; Valk, J. M.; and Witteveen, C. 2006. Coordinating self-interested planning agents. *Autonomous Agents and Multi-Agent Systems* 12(2):199–218.

Chiu, D. K. W.; Lee, O. K. F.; Leung, H.-F.; Au, E. W. K.; and Wong, M. C. W. 2005. A multi-modal agent based mobile route advisory system for public transport network. In *Proc. of the 38th Annual Hawaii Int. Conf. on System Sciences*, volume 3, 92.2.

Christodoulou, G.; Koutsoupias, E.; and Nanavati, A. 2004. Coordination mechanisms. In *Proc. of the 31st Int. Coll. on Automata, Languages and Programming*, 345–357.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49(1-3):61–95.

Harrald, J. R. 2005. Supporting agility and discipline when preparing for and responding to extreme events. In *Proc. of the 2nd Int. Conf. on Information Systems for Crisis Response and Management*.

Hunsberger, L. 2002. *Group Decision Making and Temporal Reasoning*. PhD thesis, Harvard University, Cambridge, MA, USA.

Hunsberger, L. 2003. Distributing the control of a temporal network among multiple agents. In *Proc. of the 2nd Int. Joint Conf. on Autonomous Agents and Multiagent Systems*, 899–906.

Léauté, T., and Williams, B. 2005. Coordinating agile systems through the model-based execution of temporal plans. In *Proc. of the Workshop on Multiagent Planning and Scheduling*, 22–28.

Shehory, O., and Kraus, S. 1998. Methods for task allocation via agent coalition formation. *Artificial Intelligence* 101(1–2):165–200.

Smith, S. F.; Gallagher, A.; Zimmerman, T.; Barbulescu, L.; and Rubinstein, Z. 2007. Distributed management of flexible times schedules. In *Proc. of the 6th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, 472–479.

Steenhuisen, J. R., and Witteveen, C. 2007. Coordinating planning agents for moderately and tightly-coupled tasks. In *Proc. of the Int. Conf. on Foundation of Comp. Sci.*

Steenhuisen, J. R.; Witteveen, C.; ter Mors, A. W.; and Valk, J. M. 2006. Framework and complexity results for coordinating non-cooperative planning agents. In *Proc. of the 4th German Conf. on Multi-Agent System Technologies*, volume 4196 of *Lecture Notes in Art. Intel.*, 98–109.

Sultanik, E. A.; Modi, P. J.; and Regli, W. C. 2006. Constraint propagation for domain bounding in distributed task scheduling. In *Proc. of Principles and Practice of Constraint Programming*, volume 4204 of *Lecture Notes in Comp. Sci.*, 756–760.

ter Mors, A. W.; Valk, J. M.; and Witteveen, C. 2006. Task coordination and decomposition in multi-actor planning systems. In *Proc. of the Workshop on Software-Agents in Information Systems and Industrial Applications*, 83–94.

Valk, J. M. 2005. *Coordination among Autonomous Planners*. PhD thesis, Delft University of Technology, Delft, The Netherlands.

Vidal, T., and Fargier, H. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence* 11:23–45.

Zlot, R. M., and Stentz, A. 2006. Market-based multirobot coordination for complex tasks. *International Journal of Robotics Research, Special Issue on the 4th International Conference on Field and Service Robotics* 25(1):73–101.