Framework and Complexity Results for Coordinating Non-cooperative Planning Agents

J. Renze Steenhuisen[1], Cees Witteveen[1], Adriaan W. ter Mors[1,2], and Jeroen M. Valk[2]

[1] 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, A.W.terMors}@tudelft.nl

[2] Almende, Westerstraat 50, 3016 DJ Rotterdam, The Netherlands, tel.: +31 10 404 9444, fax +31 10 404 7773, {Adriaan, Jeroen}@almende.nl

Abstract. In multi-agent planning problems agents are requested to jointly solve a complex task consisting of a set of interrelated tasks. Since none of the agents is capable to solve the whole task on its own, usually each of them is assigned to a subset of tasks. If agents are dependent upon each other via interrelated tasks they are assigned to, moderately coupled teams of agents are called for. Such teams solve the task by coordinating during or after planning and revising their plans if necessary. In this paper we show that such complex tasks also can be solved by loosely coupled teams of agents that are able to plan independently, although the computational complexity of the coordination problems involved is high. We also investigate some of the factors influencing this complexity.

Keywords: Multi-agent system, complex tasks, task assignment, planning, coordination, computational complexity.

1 Introduction

A multi-agent planning problem requires a set of autonomous planning agents to come up with a *joint plan* for achieving a set of tasks. Usually, none of the participating agents is capable of solving all tasks by itself. Therefore, each agent is assigned a subset of tasks to carry out and each agent has to construct a plan to carry out the tasks assigned to it. Obviously, it is required that these individual plans are compatible with each other in the sense that together they should constitute a feasible plan for the complete set of tasks. Therefore, we need some form of *coordination* between the agents.

Like in [1,2], we classify multi-agent planning problems according to the type of coordination needed. This classification is based on a schematic partitioning of multi-agent planning problems into three phases: A *task-allocation* phase where it is decided who does what, a *planning* phase where it is decided how to do it, and a *task-execution* phase where it is done.

K. Fischer et al. (Eds.): MATES 2006, LNAI 4196, pp. 98–109, 2006.

Problems where coordination is only needed in the task-allocation phase are problems that can be solved by *loosely-coordinated agents* that are able to plan and act autonomously, but need to coordinate with respect to the task allocation (e.g., negotiating about the subtasks to be assigned). Typical tasks that can be solved in this way are reconnaissance tasks and simple pick-up and delivery tasks. If coordination is also needed in the planning phase, such problems are said to be solvable by *moderately-coordinated agents*. Such agents need to coordinate in the pre-execution phase, but are not dependent upon each other if the plan is executed. Typical problems in this category are monitoring and multi-modal transportation tasks. Finally, problems where coordination between agents is needed in all three phases are problems that can be solved by *tightly-coordinated agents*. These agents need coordination not only in the preparation but also in the execution phase. Examples are platooning vehicles and moving in formation.

Most approaches to coordination in multi-agent planning, like [3, 4, 5], stress the intertwining of planning and coordination processes allowing the agents to revise their plans by exchanging information during the planning process. Other approaches, like [6, 7, 8], consider coordination as an after-planning process to either remove conflicts between independently developed plans or to improve such plans by exploiting positive interactions between them.

In both these approaches it is assumed that due to coordination requests – either during or after planning– individual agents are prepared to adapt and revise their current (partial) plans in order to obtain a feasible joint plan.

In this paper, we concentrate on solving multi-agent planning problems where agents are self-interested and non-cooperative. Typical problems we are interested in are solving complex tasks that require the joint effort of a set of agents, but where interaction between the agents during planning and execution is absent. Examples of such problems are multi-modal transportation problems that have to be solved by the joint effort of competitive transportation organizations and patient planning in hospitals where self-interested and independent parties are involved in scheduling health-care resources. In all these applications, we have to assume that *(i)* the agents do not want to be interfered during planning and *(ii)* agents do not want to revise their plans afterwards. We will call the problem of ensuring that a feasible joint plan will be created, while taking into account *(i)* and *(ii)*, the *plan-coordination problem*. Clearly, both approaches mentioned above are not suitable to solve this plan-coordination problem.

At first sight seems that the plan-coordination problem can only be used to solve task planning problems that require loosely-coordinated agents, since in that case the planning can be achieved by the agents independently. In such cases, it is often sufficient to deal with coordination in the task-assignment (and task decomposition) process, such that the plan-coordination problem is solved in a trivial way.

In this paper, however, we will show that the plan-coordination problem can also be used to solve tasks that require moderately-coordinated agents. We will develop a general framework for representing and solving the task assignment and plan-coordination problem. The main idea we apply here is that a set of

tasks requiring moderately-coupled agents to solve them can be *reduced* to a set of tasks requiring loosely-coupled agents. This reduction ensures that the set of independently developed plans constructed for the latter set is also a solution for the former set of tasks. We will use this framework to describe this reduction and to establish some relations between properties of tasks, properties of agents, and the complexity of the resulting plan-coordination problems.

Significance and Perspectives The approach we propose can be viewed upon from different perspectives. First, it is an attempt to partially integrate the research areas of task assignment, coalition formation, and multi-agent planning. In particular, we will show that in solving moderately-coupled tasks, one has to recognize the interaction effects between individual planning methods of agents and the task-assignment methods employed, since the outcome of one agent is not only determined by the set of tasks it receives but also by the plans developed by other agents. We will show that such interaction effects have consequences for the computational complexity of the plan-coordination problem.

Second, our approach can be viewed as an attempt to connect other research on coordinating agents like *social laws* [9,10] and *cooperation protocols* [11] with multi-agent task-based planning research. Social laws are general rules that govern agent behavior. If a set of agents abide by these rules, then their behavior will be coordinated without the need for any problem-specific information exchange between the agents. In many situations, however, coordination cannot be achieved (or not efficiently) through general, problem-independent rules alone. In such cases, cooperation protocols can be applied that require simple forms of problem-specific information exchange before the agents can start planning. These protocols guarantee that if the agents adhere to the protocol, then the individual plans can easily be assembled into a feasible joint plan for the overall task. Our approach to solve plan-coordination problems for moderately-coupled agents comes down to reducing these problems to plan-coordination problems for loosely-coupled agents that agree upon cooperation protocols.

Finally, our approach can be viewed upon from a broader computational perspective. Note that tasks requiring loosely-coupled coordination allow the agents to plan independently. Hence, they allow the multi-agent planning problem to be partitioned into independent planning subproblems. For moderately-coupled plan-coordination problems, such a partitioning is not possible. As we will show, solving moderately-coupled plan-coordination problems for self-interested agents comes down to transforming moderately-coupled plan-coordination problems into loosely-coupled plan-coordination problems in such a way that the original problem is changed in a minimal way. In other words, our approach is an example of applying a *minimal change* and a *task decomposition* approach to solve a plan-coordination problem.

2 Framework

The formal framework we present is intended to capture the basic aspects of task-based planning and coordination of non-cooperative agents. It enables us

to distinguish the main components of the plan-coordination problem –and their interactions– that we are interested in:

1. a *complex task* that requires the joint effort of several agents to complete it,
2. a *task-assignment* process by means of which each agent obtains a subset of tasks to complete,
3. a *planning* process enabling each agent to complete its subset of tasks, and
4. a *plan-coordination* mechanism by means of which the completion (if possible) of the original complex task can be ensured.

Complex Task. We consider a set of agents $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ that have to complete a complex task $\mathcal{T}$. Such a complex task is defined as a set of tasks $T = \{t_1, \ldots, t_k\}$ together with two binary relations between them. One of these relations is the *decomposition relation* ρ, relating an abstract task t to a set of more primitive subtasks.[1] Therefore, we define an OR-decomposition relation $\rho_\vee$ to model this type of relation between an abstract task t and its (more primitive) subtasks t_i. On the other hand, running a factory requires multiple tasks to be done, such as the purchasing of goods and hiring of personnel, and optimizing production lines. To model this relation, we define an AND-decomposition relation $\rho_\wedge$ between an abstract task t and a set of subtasks t_i.[2]

The second relation is a precedence relation $\prec$ among tasks, where $t \prec t'$ means that t has to be completed before t' can start. The set of precedence constraints induces a *partial order* on the tasks in T.

In addition, an agent can only complete a task when it is capable of carrying it out at all. Therefore, capabilities need to be associated with both tasks and agents. These capabilities are incorporated by defining capability vectors for both the agents $\boldsymbol{c}(\mathcal{A}_i)$ and the tasks $\boldsymbol{c}(t_j)$. We postpone their detailed description to a separate paragraph about Task Assignment below.

We now formally define a complex task as $\mathcal{T} = (T, \rho, \prec, \boldsymbol{c}(T))$ thereby extending the concept of a *task tree* [12, 13]. In this tuple, T represents the set of abstract and primitive tasks, ρ is the decomposition relation, $\prec$ a precedence order, and $\boldsymbol{c}(T)$ is the abbreviation for the set of task capability vectors.

In Figure 1, an example is shown of a complex task and its decomposition into subtasks with precedence relations

In a complex task $\mathcal{T} = (T, \rho, \prec, \boldsymbol{c}(T))$, we require that the decomposition $\rho(t)$ of a task t is unique and that it is either completely in $\rho_\vee$, or completely in $\rho_\wedge$ (i.e., $\rho_\vee \cap \rho_\wedge = \emptyset$). Note that this does not limit the expressiveness of our framework with respect to tasks that can be decomposed by a combination of OR and AND decomposition (e.g., in Figure 1, we have $\rho(t_1) = \{t_{11}, t_{12}\} \subseteq \rho_\vee$ with $\rho(t_{11}) = \{t_{111}, t_{112}\} \subseteq \rho_\wedge$).

[1] For instance, when goods need to be transported from a harbor to a factory we have an abstract task *transport*. Such an abstract task could be completed by carrying out one of the more primitive subtasks *transport_by_train*, *transport_by_truck*, and *transport_by_ship*.

[2] In [12], a similar decomposition of complex tasks is used.

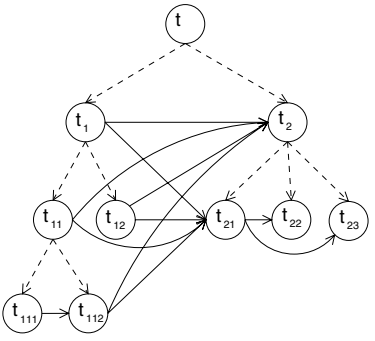

Fig. 1. An example complex task decomposed (dashed arcs) into subtasks with precedence relations (normal arcs) between them

Moreover, the decomposition and precedence relation are related in two ways. First, the relations are *orthogonal*, which means that precedence relations only exist between tasks that have no decomposition relation between them (i.e., $\rho^+ \cap (\prec \cup \prec^c)^+ = \emptyset$).[3] Second, precedence relations are inherited via decomposition (i.e., if $t_1 \prec t_2$, then for all $t_1' \in \rho(t_1)$ and for all $t_2' \in \rho(t_2)$ we have that $t_1' \prec^+ t_2$, $t_1 \prec^+ t_2'$, and $t_1' \prec^+ t_2'$).

Having defined the hierarchical decomposition of a set of tasks, we now define a *task network* (T, ρ) with $\rho = \rho_\vee \cup \rho_\wedge$, to be able to express when a task and a set of tasks has been completed. We say that a task $t \in T$ in a task network (T, ρ) is completed if exactly one of the following conditions holds:

1. t has been completed directly,
2. a task $t' \in \rho(t) \subseteq \rho_\vee$ has been completed, or
3. all tasks $t' \in \rho(t) \subseteq \rho_\wedge$ have been completed.

This notion naturally extends to the completion of a task network. A task network (T, ρ) is said to be completed if all *initial tasks* in (T, ρ) have been completed, that is if all tasks in the set $\{t \mid \rho^c(t) = \emptyset\}$ have been completed. Note that this framework differs from others in the sense that it does not restrict completion to completing the set of leaf tasks $\{t \mid \rho(t) = \emptyset\}$.

Task Assignment Now that we have defined when a task network (T, ρ) associated with a complex task $\mathcal{T}$ is completed, we can deal with the problem of which agent is going to complete which task.

First, an individual agent $\mathcal{A}_i$ must have the required *capabilities* to be able to carry out a certain task $t \in T$. We assume that in the entire multi-agent system, m distinct capabilities $c_1, \ldots, c_m$ can be distinguished. The capabilities of agent $\mathcal{A}_i$ are represented by the vector $\boldsymbol{c}(\mathcal{A}_i) = (c_1(\mathcal{A}_i), \ldots, c_m(\mathcal{A}_i)) \in (\mathbb{N} \cup \{\infty\})^m$, where $c_j(\mathcal{A}_i)$ specifies how much agent $\mathcal{A}_i$ can offer of capability c_j (we will

[3] The following notations are used on a binary relation σ: transitive closure σ^+, transitive reduction σ^-, and the converse σ^c.

assume integral quantities). Similarly, $\boldsymbol{c}(t_j) = (c_1(t_j), \ldots, c_m(t_j)) \in \mathbb{N}^m$ is the vector that specifies how much of each capability is required for carrying out task $t_j \in T$. An agent $\mathcal{A}_i$ is said to be able to carry out a subset of tasks $T_i \subseteq T$ iff $\boldsymbol{c}(\mathcal{A}_i) \geq \sum_{t \in T_i} \boldsymbol{c}(t)$ (where $\boldsymbol{x} \geq \boldsymbol{y}$ iff for all $i = 1, \ldots, m$, $x_i \geq y_i$).[4] In the following, the set of agent capability vectors and the set of task capability vectors are abbreviated by $\boldsymbol{c}(\mathcal{A})$ and $\boldsymbol{c}(T)$, respectively.

A task instance where the agents are not already assigned to tasks is called a *free task instance* and is specified as a tuple $(T, \rho, \prec, \mathcal{A}, \boldsymbol{c}(\mathcal{A}), \boldsymbol{c}(T))$. Such an instance specifies the tasks, their decomposition relation, their order dependencies, the task requirements, and the agent capabilities. To complete the task network (T, ρ) associated with the free task instance, individual tasks $t \in T$ have to be assigned to agents. Therefore, we need to define which sets of tasks can be assigned to agents in order to complete (T, ρ). Such a set $T' \subseteq T$ is called a *candidate-assignment set* and has to satisfy the following constraints:

1. T' is a ρ^+-independent subset of T: if $t, t' \in T'$ then neither $t \rho^+ t'$ nor $t' \rho^+ t$,
2. for all $t \in T$, if $t' \in \rho(t)$ and $\rho(t) \subseteq \rho_\vee$ then $\rho(t) \cap T' = \{t'\}$, and
3. (T, ρ) is completed by completing all tasks in T'.

In Figure 1, the set $T' = \{t_{111}, t_{112}, t_2\}$ is a candidate-assignment set. Note that a candidate-assignment set does not have strict supersets nor strict subsets that are candidate-assignment sets. Moreover, if $\rho = \emptyset$, there is only one unique candidate-assignment set (i.e., $T' = T$).

A partitioning $\{T_i\}_{i=1}^n$ of a candidate-assignment set T' with every $t \in T'$ assigned to an agent $\mathcal{A}_i$ capable of carrying it out is called an *assignment set* if

1. the set $\bigcup_{i=1}^n T_i$ is a candidate-assignment set, and
2. every agent $\mathcal{A}_i$ is capable of completing all tasks in its assigned partition T_i.

Applying an assignment set to a free task instance $(T, \rho, \prec, \mathcal{A}, \boldsymbol{c}(\mathcal{A}), \boldsymbol{c}(T))$ results in a *fixed task instance* $(\{T_i\}_{i=1}^n, \prec, \mathcal{A}, \boldsymbol{c}(\mathcal{A}), \boldsymbol{c}(T))$. The agents $\mathcal{A}_i$ are characterized by the blocks of the partitioning $\{T_i\}_{i=1}^n$ which are assumed, without loss of generality, to be non-empty. Additionally, the decomposition relation and the capabilities are no longer needed, because each agent is able to complete the tasks assigned to it which are ρ-independent. Therefore, fixed task instances will be abbreviate by the tuple $(\{T_i\}_{i=1}^n, \prec)$ in the remainder of this text.

Planning As the result of task assignment, the set of precedence constraints $\prec$ in a fixed task instance $(\{T_i\}_{i=1}^n, \prec)$ is split into two disjoint subsets:[5]

1. the set of *intra-agent* constraints $\prec_{intra} = \bigcup_{i=1}^n \prec_i = \bigcup_{i=1}^n (\prec^+ \cap (T_i \times T_i))^-$, contains all precedence constraints between tasks assigned to agent $\mathcal{A}_i$, and

[4] If $c_j(\mathcal{A}_i)$ is finite, the capability is said to be a *consumable* resource (e.g., fuel, time, or money). If $c_j(\mathcal{A}_i) = \infty$, it is a *non-consumable* capability (e.g., knowledge or a skill).

[5] We would like to present these relations as concisely as possible. Therefore, we will use a transitive reduction of a transitively closed relation whenever it is possible.

2. the set of *inter-agent* constraints $\prec_{inter} = (\prec^+ \cap \bigcup_{i \neq j}(T_i \times T_j))^-$, contains all precedence constraints between tasks assigned to different agents.

Note that each agent $\mathcal{A}_i$ has to complete its part $(T_i, \prec_i)$ of the complex task, which is generated by the set of tasks T_i assigned to it. In order to complete $(T_i, \prec_i)$, each agent has to construct a plan (or schedule) for it. Such a plan can be represented as a partial ordering of the tasks T_i that satisfies the precedence constraints $\prec_i$. Irrespective of the (possibly domain-specific) planning tools employed by $\mathcal{A}_i$, we will therefore assume that the resulting plan always can be represented by a partial order $P_i = (T_i, \pi_i)$ with $\prec_i \subseteq \pi_i$.

From an individual agent's point of view, it wants to be completely autonomous in developing its plan P_i and unwilling to revise it afterwards. Therefore, we define a *joint plan* for a set of agents $\mathcal{A}_i$ on a fixed task instance $(\{T_i\}_{i=1}^n, \prec)$ as a plan $P = (\{T_i\}_{i=1}^n, \pi)$ where

1. π respects $\prec$, that is $\prec \subseteq \pi$, and
2. each individual plan $P_i = (T_i, \pi_i)$ of agent $\mathcal{A}_i$ is respected, that is, $\pi_i \subseteq (\pi \cap (T_i \times T_i))$.

Clearly, if such a joint plan exists, it implies that the current plans of all the agents are coordinated. There is no need for any revision of the individual plans in executing the joint plan.

Plan Coordination The existence of such a joint plan is by no means guaranteed: Due to the set of inter-agent precedence constraints, combinations of the individual plans together with this set can lead to breaking of the partial order as is shown in the following simple example.

Example 1. In Figure 2, a simple situation is depicted with four agents $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$, $\mathcal{A}_4$ being assigned two tasks each of two complex tasks. It is clear that multiple combinations of individual plans of agents lead to inter-agent cycles. For example, if $\mathcal{A}_1$ chooses a plan where $t_8 \prec t_1$ and $\mathcal{A}_3$ chooses a plan where $t_3 \prec t_6$, there exists such a cycle. Hence, the possibility of such plans prevents the existence of a joint plan respecting every individual plan.

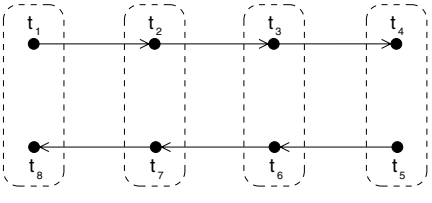

Fig. 2. Problems can occur when planning autonomously

The plan-coordination problem now can be stated as follows: How can we guarantee that *every* possible set of individual plans P_i can be easily combined into a feasible joint plan, without the need to revise the individual plans? In fact, this means that we want to have a solution to a moderately-coupled plan-coordination problem –there are constraints between the tasks assigned to different agents– without needing the agents to coordinate during planning.

A given fixed task instance $(\{T_i\}_{i=1}^n, \prec)$ is said to be *coordinated*, if it holds that for every set of individual plans $\{P_i = (T_i, \pi_i)\}_{i=1}^n$ constructed by the agents there exists a joint plan respecting the individual plans P_i. In [14], it was shown that every fixed task instance can be transformed into a coordinated one by adding a minimum number of intra-agent constraints $\Delta = \bigcup_{i=1}^n \Delta_i$ such that the resulting instance $(\{T_i\}_{i=1}^n, \prec \cup \Delta)$, is a coordinated instance.

In Figure 2, the reader might check that $\Delta = \{t_7 \prec t_2, t_6 \prec t_3, t_5 \prec t_4\}$ is such a set of additional constraints that turns the fixed task instance into a coordinated one. The problem, of course, is to determine how difficult it is to find such a set of additional constraints.

3 The Computational Complexity of Coordination

In this section, we will analyze the computational complexity of plan coordination. We study three variants of the plan-coordination problem and some factors that influence their complexity. Some results have been published in [14]; the results about the complexity of coordination when the number of agents is kept fixed are new.

3.1 Variant I: Pure Coordination

We start with analyzing the complexity of coordination in multi-agent planning isolated from the task-assignment process and define the following decision problem for fixed task instances:

Pure Coordination Recognition (PCR)
INSTANCE: Given a fixed task instance $(\{T_i\}_{i=1}^n, \prec)$.
QUESTION: Is this instance coordinated?

We have shown that PCR is coNP -complete [15], by a non-trivial reduction from the complement of the Path With Forbidden Pairs (PWFP) problem.

While PCR only asks whether a fixed task instance is coordinated, it does ask for the existence of a bounded coordination set as in the following problem:

Pure Coordination (PC)
INSTANCE: Given a fixed task instance $(\{T_i\}_{i=1}^n, \prec)$ and integer $K \geq 0$.
QUESTION: Does there exist a coordination set Δ with $|\Delta| \leq K$ such that the fixed task instance $(\{T_i\}_{i=1}^n, \prec \cup \Delta)$ is coordinated?

Intuitively, guessing a coordination set Δ, we can verify in polynomial time using a PCR-oracle whether the instance $(\{T_i\}_{i=1}^n, \prec \cup \Delta)$ is coordinated. Since PCR is coNP -complete, it follows that PC $\in \Sigma_2^p$. It turns out that PC is Σ_2^p-complete, using a reduction from a quantified version of PWFP [15].

Factors Influencing the Complexity of Pure Coordination It seems reasonable to assume that one source of complexity of the plan-coordination problem can be attributed to the *number of tasks* each agent receives and –indirectly– to the complexity of the single-agent planning problems for these tasks. However, it turns out that the PC problems remain intractable even if the single-agent planning problems are trivial (see Table 1). Notice that the Σ_2^p-completeness of PC is still open for instances where each agent has at most $4, 5$, or 6 tasks.

Table 1. Complexity of PCR and PC with limited number of tasks per agent.

	$\|T_i\| = 2$	$\|T_i\| \leq 3$	$\|T_i\| \leq 4$	$\|T_i\| \leq 7$
PCR	P	P	coNP -complete	coNP -complete
PC	NP -complete	NP -complete	Σ_2^p	Σ_2^p-complete

Due to limited space, we will merely give hints on how these results are obtained. All proofs of these and other complexity results can be found in [16]. First, for PC with $|T_i| = 2$ a reduction can be made from Feedback Vertex Set. For PCR with $|T_i| \leq 3$, a reduction can be made to topological sorting. The completeness of PCR with $|T_i| \leq 4$, and PC with $|T_i| \leq 7$, is derived from the properties of the reductions used in the completeness proofs of the general variants.

Another source of complexity might be the number of agents involved. Indeed, if we limit the number of agents, it can be shown that the PCR problem is in P for any fixed number of agents. This can be proven by reducing the problem to simple inter-agent cycle-testing. Detecting such a cycle can be achieved in polynomial time [16]. As an easy consequence, the associated PC-problems are in NP . They turn out to be NP -complete for all fixed values $|\mathcal{A}| = n \geq 3$, which can be proven by reduction from 3-Partite Vertex Cover [16]. Although we suspect PC for $n = 2$ to be tractable, the complexity of this problem is still open.

Table 2. Complexity of PCR and PC with fixed number of agents.

	$\|\mathcal{A}\| = 2$	$3 \leq \|\mathcal{A}\|$
PCR	P	P
PC	NP	NP -complete

3.2 Variant II: Coordinated Assignment

If a task-assignment problem is part of the coordination problem, we define

Coordinated Assignment Recognition (CAR)
INSTANCE: Given a free task instance $(T, \rho, \prec, \mathcal{A}, \boldsymbol{c}(\mathcal{A}), \boldsymbol{c}(T))$.
QUESTION: Does there exist an assignment set $\{T_i\}_{i=1}^n$ such that the resulting fixed task instance $(\{T_i\}_{i=1}^n, \prec)$ is coordinated?

The CAR problem turns out to be Σ_2^p-complete [14]. Note that here task-assignment problems seem to constitute an independent factor of complexity as the total complexity goes up one step in the polynomial hierarchy compared to PCR. Note that checking a candidate-assignment set for a free task-instance is polynomially verifiable, both for the case where agents have consumable capabilities as well as the case where agents have non-consumable capabilities. However, the problem of deciding whether there exists a suitable task assignment for a free task instance is NP -hard for consumable capabilities[6] but polynomially solvable for non-consumable capabilities. Therefore, one might expect that the consumability of capabilities would influence the complexity of CAR. This turns out not to be the case: The CAR problem turns out to be Σ_2^p-complete for both assignment conditions.

Since adding the task-assignment problem resulted in moving one step up in the polynomial hierarchy for the coordination recognition problem, one would expect the same for the coordination problems themselves. However, this is not the case for the following problem, where the task-assignment problem is added to the pure coordination problem.

Coordinated Assignment (CA)
INSTANCE: Given a free task instance $(T, \rho, \prec, \mathcal{A}, \boldsymbol{c}(\mathcal{A}), \boldsymbol{c}(T))$ and a K.
QUESTION: Does there exist an assignment set $\{T_i\}_{i=1}^n$ and a coordination set $\Delta \subseteq \bigcup_{i=1}^n (T_i \times T_i)$ with $|\Delta| \leq K$ such that the resulting fixed task instance $(\{T_i\}_{i=1}^n, \prec \cup \Delta)$ is coordinated?

Note that it suffices to guess both an assignment and a coordination set Δ to verify in polynomial time using a PCR-oracle that the given instance is a yes-instance. Therefore, the problem cannot be harder than the PC problem. In fact, it turns out that the CA is Σ_2^p-complete. Hence, maybe contrary to expectation, adding a task-assignment problem does not increase the complexity of the coordination problem in an essential way.

3.3 Variant III: Complete Coordination

Obviously, the next step is to extend the notion of being coordinated from having *some* coordinated assignment to having *all* task assignments being coordinated. The associated coordination recognition problem can be stated as follows:

Complete Coordination Recognition (CCR)
INSTANCE: Given a free task instance $(T, \rho, \prec, \mathcal{A}, \boldsymbol{c}(\mathcal{A}), \boldsymbol{c}(T))$.
QUESTION: Is it true that for all assignment sets $\{T_i\}_{i=1}^n$ the resulting fixed task instance $(\{T_i\}_{i=1}^n, \prec)$ is coordinated?

It turns out that CCR is Π_2^p-complete [14].

Again we are interested in checking whether a bounded coordination set exists for this problem. This problem can be seen as guaranteeing that every possible assignment of tasks to agents results in a coordinated fixed task instance by

[6] This can be shown by reduction from PARTITION.

adding a limited number of additional constraints. We now define the most general variant of the coordination problem as

Complete Coordination (CC)
INSTANCE: Given a free task instance $(T, \rho, \prec, \mathcal{A}, \boldsymbol{c}(\mathcal{A}), \boldsymbol{c}(T))$ and a K.
QUESTION: Is it true that for all assignment sets $\{T_i\}_{i=1}^n$ there exists a coordination set $\Delta \subseteq \bigcup_{i=1}^n (T_i \times T_i)$ with $|\Delta| \leq K$ such that the resulting fixed task instance $(\{T_i\}_{i=1}^n, \prec \cup \Delta)$ is coordinated?

By guessing an assignment and using a Σ_2^p-oracle for the resulting PC problem, we can verify a counter-example in polynomial time. Hardness for this class can be proven by reducing a quantified version of PWFP to it, which is Π_3^p-complete. It turns out that the CC-problem is Π_3^p-complete.

Table 3. Complexity of three variants of the coordination problem

	Coordination Recognition	Coordination
Pure Coordination	coNP -complete	Σ_2^p-complete
Coordinated Assignment	Σ_2^p-complete	Σ_2^p-complete
Complete Coordination	Π_2^p-complete	Π_3^p-complete

To sum up, in Table 3, the complexity results of the discussed three variants of the coordination problem are given. It is clear that the general problems are intractable. Therefore, we have to rely on approximation algorithms for finding practical solutions to these problems. We refer the reader to [14] for some practical applications using such approximation algorithms for coordination.

4 Concluding Remarks

We discussed a general framework capturing the basic aspects of task-based planning and coordination for non-cooperative agents. One of the advantages of this framework is that it allows us to study several factors, such as task-assignment procedures and capabilities of agents that might affect the complexity of the plan-coordination problem. The general plan-coordination problems turned out to be intractable. Studying the change in complexity when bounding the number of tasks per agent or the number of agents, we showed that these subclasses are much easier to solve, especially when the number of agents is kept constant.

These results are not only of theoretical interest, but also have some practical implications. First, because we assumed self-interested non-cooperative agents, the proposed solution to the plan-coordination problem allows the agents to plan independently. This enables the (re)use of single-agent planners in a multi-agent planning setting. After the addition of the constraints, the moderately-coupled planning problem has been reduced to a loosely coupled one. It turns out that in this way we can solve the multi-agent planning problem more efficiently by decomposing it into smaller subproblems that can be solved independently.

Secondly, these results show that we can only hope for approximation algorithms to solve these problems. In fact, in [14], such approximations have successfully been applied to solve multi-modal logistic planning problems.

Finally, we note that one of the shortcomings of the current framework is that it lacks the notion of time. Currently, we are extending our framework to represent time intervals and time constraints on tasks to apply our plan-coordination methods on. This will enable us to generalize similar decoupling methods, like the temporal decoupling method [17], and to use them as part of the coordination of temporal planners.

References

1. Dias, M.B., Zlot, R.M., Kalra, N., Stentz, A.: Market-based multirobot coordination: A survey and analysis. Technical Report CMU-RI-TR-05-13, Robotics Institute, Carnegie Mellon University (2005).
2. Kalra, N., Stentz, A., Ferguson, D.: Hoplites: A market framework for complex tight coordination in multi-agent teams. Technical Report CMU-RI-TR-04-41, Robotics Institute, Carnegie Mellon University (2004).
3. Decker, K.S., Lesser, V.R.: Designing a family of coordination algorithms. In: Proc. of DAI. (1994) 65–84.
4. Durfee, E.H., Lesser, V.R.: Partial global planning: A coordination framework for distributed hypothesis formation. IEEE Transactions on Systems, Man, and Cybernetics **21**(5) (1991) 1167–1183.
5. Ephrati, E., Rosenschein, J.S.: Multi-agent planning as the process of merging distributed sub-plans. In: Proc. of DAI. (1993) 115–129.
6. Cox, J.S., Durfee, E.H.: Discovering and exploiting synergy between hierarchical planning agents. In: Proc. of AAMAS. (2003) 281–288.
7. Foulser, D.E., Li, M., Yang, Q.: Theory and algorithms for plan merging. Artificial Intelligence Journal **57**(2-3) (1992) 143–182.
8. von Martial, F.: Coordinating Plans of Autonomous Agents. Springer (1992).
9. Moses, Y., Tennenholtz, M.: Artificial social systems. Computers and AI **14**(6) (1995) 533–562.
10. Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: Off-line design. Artificial Intelligence **73**(1–2) (1995) 231–252.
11. Jennings, N.R.: Commitments and conventions: The foundation of coordination in multi-agent systems. The Knowledge Engineering Review **8**(3) (1993) 223–250.
12. Zlot, R.M., Stentz, A.: 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) (2006) 73–101.
13. Zlot, R.M., Stentz, A.: Market-based multirobot coordination using task abstraction. In: Proc. of FSR. (2003).
14. Buzing, P., ter Mors, A.W., Valk, J.M., Witteveen, C.: Coordinating self-interested planning agents. Autonomous Agents and Multi-Agent Systems **12**(2) (2006).
15. Valk, J.M.: Coordination among Autonomous Planners. PhD thesis, Delft University of Technology (2005).
16. Steenhuisen, J.R., Witteveen, C.: Complexity studies in coordinating non-cooperative planning agents. Technical report, Delft University of Technology (2006) Forthcoming.
17. Hunsberger, L.: Group Decision Making and Temporal Reasoning. PhD thesis, Harvard University (2002).