Coordinating Non Cooperative Planning Agents: Complexity Results

Adriaan ter Mors Cees Witteveen*
Faculty of Electrical Engineering, Mathematics and Computer Science,
Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands,
{a.w.termors, c.witteveen}@ewi.tudelft.nl

Abstract

Whenever independent, non-cooperative actors jointly have to solve a complex task, they need to coordinate their efforts. Typical examples of such task coordination problems are supply chain management, multi-modal transportation and patient-centered health care management. Common elements in such problems are a complex task, i.e., a set of interdependent subtasks, and a set of competitive actors. Solving a task coordination problem first of all requires to solve a task allocation problem (how to assign competitive actors to the subtasks). As a result, each of the actors will receive a set of subtasks to complete and will need to make a plan for this set of tasks. Therefore, also a plan coordination problem has to be solved (how to ensure that a joint plan always can be composed, whatever plan is chosen by the individual actors). The aim of this paper is twofold: first of all to present a general formal framework to study some computational aspects of this non-cooperative coordination *problem, and secondly to establish some complexity results and to identify some of the factors that contribute to the complexity of this problem.*

2. Introduction

As the result of a significant shift of focus in artificial intelligence from single agent to multi-agent systems, task coordination has become one of the central topics in AI-research [10, 12, 13, 20]. The task coordination problem itself is easy to state: *How to coordinate autonomous actors (agents) to jointly solve a complex task they are not able to solve individually*. Typical application areas where such complex tasks have to be solved are e.g., automated supply chain management, seamless multi-modal transportation provided by independent transportation companies, and patient-centered health care management systems. Usually, a complex task is specified as a set of interdependent subtasks and for every subtask some specific abilities that are required to perform the task. The autonomous actors involved each have specific abilities enabling them to solve specific subtasks, but not the complete task. Therefore, solving the complete problem first of all requires to solve a *task allocation* problem (how to assign competitive actors to the subtasks distinguished). Besides their abilities, agents may also differ in the costs they associate with performing a subtask. *Task allocation methods* aim to minimize the cost of individual agents or the total cost of performing all tasks. As the result of a task allocation process, each actor receives a set of subtasks to perform. It is important to realize that due to differences in capabilities, interdependent tasks might be allocated to different actors. For example, a task (order) in a supply chain consists of composing parts where some suppliers assemble some parts that in turn are assembled by other actors. As a result, two actors (suppliers) might become dependent upon each other. In the applications we have in mind, each actor will need to make a *plan* for the set of tasks allocated to it, e.g., each supplier will have to make a plan for performing its set of assembling orders. Due to the dependency constraints (one subtask being dependent upon the completion of another), however, not every plan feasible for an individual actor might be compatible with plans of the other actors. Therefore, besides a task allocation problem, also a *plan coordination* problem has to be solved (i.e., how to ensure that the plans constructed by each of the individual actors constitute a feasible conflict-free plan for the complete task). The outcome of such a plan coordination process is the specification (implicitly or explicitly) of a set of additional *plan constraints* that, once satisfied, guarantees the existence of a conflict-free and feasible joint plan.[1] Accepting such plan constraints, however, might incur some additional costs on performing the tasks

* Also affiliated with the Centre for Mathematics and Computer Science, P.O. Box 94079, NL-1090 GB Amsterdam, The Netherlands, C.Witteveen@cwi.nl

1 Take the following simple example: Suppose actor X has to perform task x1 and x2 and actor Y has to perform task y1 and y2. Let task y1 require task x1 to be completed first and let task x2 require y2 to be completed first. Then a plan where X will perform x2 before x1 is not compatible with a plan where Y will perform y1 before y2, since then a circular dependency will be introduced.

already accepted by an actor. For example, plans satisfying such an additional constraint might be more costly than plans without. Therefore, we should also provide a method to allocate plan constraints to actors that takes into account these cost factors. Finally, we have to take into account that in some cases both processes (task allocation and plan coordination) cannot be solved independently from each other: a (nearly) optimal task allocation process might induce such large costs on the acceptance of plan coordination that the total costs are more than the total costs based on an inferior task allocation. Therefore, a careful integration of task allocation and plan coordination methods is called for in order to minimize the costs for the actors involved.

Although quite a number of studies have concentrated on approaches to task allocation in multi-agent systems (cf.[15, 16, 19]) and its relations to combinatorial optimization problems (c.f. [9]), in most approaches the complex tasks described are rather simple: often they consist of a set of independent subtasks and each of the actors receives one single subtask or a subset of subtasks. Even if the task description is more elaborate like in the Traderbots architecture [4], it is assumed that the set of subtasks does not require an elaborate planning process to execute. Therefore, the problem of identifying planning constraints and the problem of allocating them does not occur in this approach. Furthermore, almost all current approaches to plan coordination assume that tasks already have been allocated and consider plan coordination with respect to already individually completed plans (coordination after planning, cf. [1, 2, 14]), or coordination of partially completed plans (coordination during planning, cf. [3, 5, 6, 11]). In both these coordination approaches, it is taken for granted that the individual agents are prepared to share information about their plans and, if necessary, to adapt and revise their individual plans after they have constructed their plans.

In this paper we concentrate on the coordination of *non-cooperative autonomous planning agents*: How to coordinate autonomous planning agents that, although aiming to solve a common planning problem, do not want to be interfered by other agents in *planning* their part of the problem. In particular, this implies that to solve this problem, the individual planning products should allow to compose a joint coordinated solution, *whatever plans will be chosen by the individual agents*. This means that coordination during or after planning is simply not possible: in these cases every non-trivial coordination would imply a proposal for revision of a partial or complete plan of an individual agent and every such a proposal would be rejected by the agent involved. Hence, the only possibility for such non-cooperative planning agents to coordinate is to do so *before* they start to plan.We therefore concentrate on such a coordination *before planning* (cf. [5]) approach.

Our main goals are (i) to present a formal and general framework to discuss task allocation and coordination problems, (ii) to point out the complexity of the resulting coordination problems, and (iii) to identify some of the factors that influence the complexity of the problem.[2]

Choosing a pre-planning approach to multi-agent planning also enables us to separate the *task-allocation* problem (which agent performs which subtask) from the *planning* problem and to relate it to the plan coordination problem to be discussed. In this sense our approach can be seen to extend current task allocation research approaches as e.g. in [16].

3. Introducing the framework

The formal framework we will introduce is intended to capture the general aspects of task-based planning and coordination of non-cooperative[3] agents. Using this framework, we are able to distinguish the main components of the coordination problem –and their interactions– we are interested in: *(i)* a *complex task* requiring the joint effort of several agents to complete it; *(ii)* a *task assignment* process by means of which each agent obtains a subset of tasks to solve, *(iii)* a *planning process* enabling each agent solves its subset of tasks to complete *(iv)* a *coordination mechanism* by means of which a joint solution (if possible) to the original complex task can be ensured.

Complex tasks We distinguish a set of non-cooperative agents $A = \{A_1, A_2, \ldots, A_n\}$ and a structured set of tasks $\mathcal{T}$, called a complex task. Such a complex task $\mathcal{T} = (T, \rho, \prec)$ consists of a set of tasks $T = \{t_1, t_2, \ldots t_k\}$ and two relations ρ and $\prec$ whose transitive closures specify a partial order on T.[4] The relation $\prec$ specifies a *precedence*[5] relation between tasks in T, $t \prec t'$ expressing that t' cannot start until t has been completed, i.e., in every plan t has to be planned to occur before t'. The relation ρ is a *refinement* relation that specifies a hierarchical task *decomposition* relation between tasks in the *task network* (T, ρ) associated with $\mathcal{T}$, closely resembling the way HTN-plans are constructed (see [7]). Specifically, let $\rho(t) = \{t' \mid t\rho t'\}$, then $\rho(t)$ is the set of (sub)tasks that might be used to complete t.

[A few notes on terminology: (i) we will use σ^c to denote the converse of a relation σ, σ^+ to stand for the *transitive closure* of σ, and σ^- to denote its *transitive reduction*; (ii) two relations σ and τ are called *equivalent*, de-

2 Elsewhere, we have discussed some approximation algorithms for achieving pre-planning coordination.

3 That is, non cooperative with respect to planning

4 We do not require $\prec$ and ρ to be transitively closed and make a distinction between the relation as specified and the relation as induced by its transitive closure.

5 Note that such a precedence relation can be induced by various other dependency relations like resource dependencies, organizational regulations, etc.

noted as $\tau \equiv \sigma$, iff $\tau^+ = \sigma^+$; (iii) a relation τ is said to *extend* σ, denoted by $\sigma \ll \tau$, iff $\sigma^+ \subseteq \tau^+$.]

Furthermore, ρ consists of two disjoint subsets: $\rho_\vee$, defining an OR-relation between the subtasks of a task, and $\rho_\wedge$, defining an AND-relation. We require that for any task t, $\rho(t)$ is either completely in $\rho_\vee(t)$, or completely in $\rho_\wedge(t)$. Intuitively, if $\rho(t) \subseteq \rho_\wedge$, task t itself is an abstract task. An agent that has to achieve t might choose to complete t in its own way without taking notice to its set of subtasks, or by completing every subtask $t' \in \rho(t)$; analogously, if $\rho(t) \subseteq \rho_\vee$, t can be completed by performing t (choosing the agents own method to solve it) or by completing one of the subtasks $t' \in \rho(t)$. Finally, we require that for any pair of tasks t, t', $\rho(t) \cap \rho(t') = \emptyset$, i.e. refinements are unique.

The relations ρ and $\prec$ are related as follows: First of all, ρ

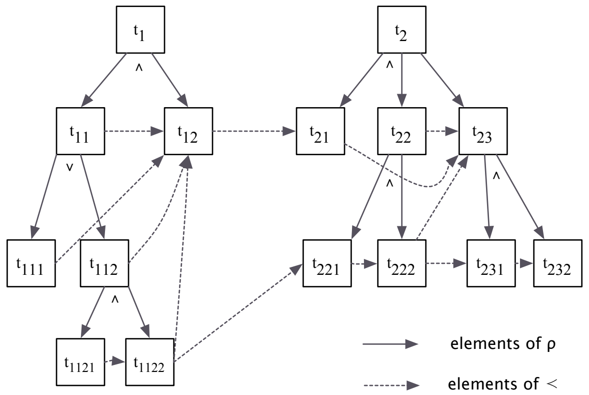

Figure 1. **A complex task with refinement (ρ) and precedence ($\prec$) relations between tasks.** $T_0 = \{t_1, t_2\}$ **is the set of initial tasks. Other tasks are refinements of these tasks. Note that** $\{t_{21}, t_{22}, t_{23}\} = \rho_\wedge(t_2)$**, while** $\rho_\vee(t_{11}) = \{t_{111}, t_{112}\}$**.**

and $\prec$ are orthogonal, i.e., $\rho^+ \cap (\prec \cup \prec^c)^+) = \emptyset$: precedence relations only exist between tasks that are not refinements of each other. Secondly, precedences are inherited via refinements, that is, if $t \prec t'$ then for all $t_1 \in \rho(t)$ and for all $t_2 \in \rho(t')$ we have $t_1 \prec^+ t'$, $t \prec^+ t_2$ and $t_1 \prec^+ t_2$. See Figure 1 for an example of a complex task specification.

We now inductively define task completion in a task network (T, ρ) as follows:

Definition 3.1 (Task completion) *A task t in (T, ρ) is said to be* completed *if exactly one of the following conditions holds:*

1. *t has been performed directly;*
2. *$\emptyset \neq \rho(t) \subseteq \rho_\vee$ and there is a task $t' \in \rho(t)$ that has been completed;*
3. *$\emptyset \neq \rho(t) \subseteq \rho_\wedge$ and all tasks $t' \in \rho(t)$ have been completed;*

In Figure 1 for instance, task t_{11} is completed if either t_{11} is performed directly, or if t_{111} is performed, or if t_{112} is completed by either performing t_{112} directly, or by performing both t_{1121} and t_{1122}.

Definition 3.2 (Task network completion) *A task network (T, ρ) is said to be completed if every task t in the set of initial tasks $T_0 = \{t \mid \rho^c(t) = \emptyset\}$ has been completed.*

A task network is thus completed if all 'root' tasks have been completed; in Figure 1, the task network has been completed if both t_1 and t_2 have been completed and these tasks can be completed by e.g. performing the tasks $t_{111}, t_{12}, t_{21}, t_{221}, t_{222}$ and t_{23}. Note that the model presented here differs from most other hierarchical task frameworks in the sense that we do not restrict the tasks to be performed to the set of leaf-tasks (tasks t for which $\rho(t) = \emptyset$).

Task allocation To perform a certain task $t \in T$, an agent A_i must have the capabilities required to perform it. We assume that in the entire multi-agent system, m distinct capabilities $c_1, c_2, \ldots, c_m$ can be distinguished. We represent the capabilities of agent A_i by the vector $\vec{c}(A_i) = (c_1(A_i), \ldots, c_m(A_i)) \in (\mathbb{N} \cup \{\infty\})^m$, where $c_j(A_i)$ specifies how much agent A_i can *offer* of capability c_j (we will assume integral quantities). Similarly, the vector $\vec{c}(t_j) = (c_1(t_j), \ldots, c_m(t_j)) \in \mathbb{N}^m$ specifies how much of each capability task $t_j \in T$ *requires*. An agent A_i is said to be able to perform a subset of tasks $T_i \subseteq T$ iff $\vec{c}(A_i) \geq \Sigma_{t \in T_i} \vec{c}(t)$ (where $\vec{x} \geq \vec{y}$ iff for all $i = 1, \ldots, m$, $x_i \geq y_i$). Note that if $c_j(A_i)$ is finite, the capability is considered to be a *consumable resource* (i.e., fuel, time, or money). If $c_j(A_i) = \infty$, we are dealing with a *non-consumable resource* capability (i.e., knowledge or a skill).[6] In the following, we will abbreviate the set of agent capability vectors and the set of task capability vectors by $\vec{c}(A)$ and $\vec{c}(T)$, respectively.

A typical *free task instance* is specified as a tuple $(T, \rho, \prec, A, \vec{c}(A), \vec{c}(T))$. Such an instance specifies the tasks, their refinement relation, dependencies, and the task as well as the agent capabilities.

To complete the set of tasks T, individual tasks $t \in T$ have to be assigned to agents. Given a task network (T, ρ), first of all we have to define which (subsets of) tasks can be assigned to agents in order to complete (T, ρ). Such a set

6 Such resources are also called *infinite* resources.

$T' \subseteq T$ we call a *candidate assignment set* and is defined as follows:

Definition 3.3 *$T' \subseteq T$ is a* candidate assignment set *of (T, ρ) if T' satisfies the following requirements:*

1. *T' is a ρ^+-independent subset of T, i.e. if $t, t' \in T'$ then neither $t\rho^+ t'$ nor $t'\rho^+ t$ should hold; (it is not allowed to perform both a task and one of its (indirect) subtasks);*
2. *If $t \in \rho_\vee(t')$ for some $t' \in T$ then $\rho_\vee(t') \cap T' = \{t\}$) (a unique choice has to be made to complete a task by OR-subtasks);*
3. *(T, ρ) is completed by performing the tasks in T' (cf. Definition 3.1) .*

Referring to Figure 1, the set $T' = \{t_{111}, t_{12}, t_{21}, t_{221}, t_{222}, t_{23}\}$ is a candidate assignment set[7]. An *assignment set* is a candidate assignment set T' where every task $t \in T'$ can be assigned to an agent capable of performing it:

Definition 3.4 (assignment set) *$T' \subseteq T$ is an* assignment set *for a free task instance $(T, \rho, \prec, A, \vec{c}(A), \vec{c}(T))$ if* (i) *T' is a candidate assignment set and* (ii) *there exists a partitioning*[8] *$[T'] = [T_1, T_2, \ldots, T_n]$ of T' such that for $i = 1, \ldots, n$, agent A_i is able to perform T_i, i.e., $\vec{c}(A_i) \geq \Sigma_{t \in T_i} \vec{c}(t)$.*

Applying an assignment to a free task instance $(T, \rho, \prec, A, \vec{c}(A), \vec{c}(T))$ results in a *fixed task instance* $([T_i]_{i=1}^n, \prec, A, \vec{c}(A), \vec{c}(T))$. Since now the refinement relation and the capabilities are no longer needed[9] and agents are characterized by the partition blocks of a ρ-independent set $T' \subseteq T$, we often abbreviate fixed task instances by the tuple $([T_i]_{i=1}^n, \prec)$. Without loss of generality we assume every block T_i to be non-empty.

Intuitively, a complex task $\mathcal{T}$ specifies both alternative ways and minimal restrictions for organizing the completion of T. We mention that these complex tasks can be seen to extend the notion of a *task tree* as introduced by [21].

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

1. the set $\prec_{intra} = \bigcup_{i=1}^n \prec_i$ of *intra-agent* constraints, where $\prec_i = (\prec^+ \cap \bigcup_{i=1}^n (T_i \times T_i))^-$ is the set of precedence constraints between tasks assigned to the same agent A_i and
2. the set of *inter-agent* constraints, i.e., the set of constraints that hold between tasks assigned to different agents: $\prec_{inter} = (\prec^+ \cap \bigcup_{i \neq j} (T_i \times T_j))^-$.

Each agent A_i then has to solve a subtask $(T_i, \prec_i)$ generated by the tasks T_i allocated to it. We assume that in order to complete T_i each agent has to construct a *plan* (or schedule) for it. We do not make any assumptions about the planning tools used by the agents. Whatever plan/schedule representation the agents (internally) employ, we assume that the plan A_i develops for T_i can be represented as a structure $P_i = (T_i, \pi_i)$ *extending*[10] the dependency structure $(T_i, \prec_i)$, i.e., π_i^+ is a partial order such that $\prec_i \ll \pi_i$.

Joint plans From the perspective of an individual agent, it should be completely autonomous in choosing its plan, i.e. the exact extension π_i of $\prec_i$. Due to the presence of the *inter-agent* constraints, however, not every combination of individually developed plans will result in a feasible joint plan. The coordination problem now can be stated as follows: How can we guarantee that every combination of individually generated plans can be combined into a feasible joint plan, without revising them? The answer lies in imposing, prior to planning, additional constraints on the agents' subtasks.

Before we state our coordination problem formally, we first define a *joint plan* of the agents A_i in a fixed task instance:

Definition 3.5 *A plan P is a joint plan for the task instance $([T_i]_{i=1}^n, \prec)$ if $P = (T', \pi)$, where $T' = \bigcup_{i=1}^n T_i$, and π^+ is a partial order extending $\prec$, i.e., $\prec \ll \pi$.*

We need to guarantee that individual agents do not need to revise their individual plans (T_i, π_i) when assembling a joint plan from them. That is, the joint plan should *respect* each individual plan:

Definition 3.6 *A joint plan $P = (T', \pi)$ respects the individual plan $P_i = (T_i, \pi_i)$ of agent A_i if $T_i \subseteq T'$ and $\pi_i \ll (\pi \cap (T_i \times T_i))$.*

We don't need to specify exactly how the individual plans are assembled to construct the overall plan. It suffices to consider the case of just joining the individual partial orderings together with the inter-agent constraints:

Definition 3.7 (Simple joining) *Given a fixed task instance $([T_i]_{i=1}^n, \prec)$ and a set $\{\ P_i = (T_i, \pi_i)\ \}_{i=1}^n$ of individual plans, the* simple joining *of them is the structure $J = (\bigcup_{i=1}^n T_i, \pi_J)$, where $\pi_J \equiv (\prec_{inter} \cup (\bigcup_{i=1}^n \pi_i))$.*

Now it is not difficult to see that the simple joining exhibits a tell-tale property w.r.t. respecting individual plans: Given a fixed task instance $([T_i]_{i=1}^n, \prec)$, there exists a joint plan P for it respecting the individual plans $P_i = (T_i, \pi_i)$ of the

7 Observe that a candidate assignment set does not have strict supersets or strict subsets that also are candidate assignment sets. Moreover, if $\rho = \emptyset$, there is only one unique candidate assignment set: $T' = T$

8 Since the agents are planning independently, we only consider *single-agent task assignments* (cf. [9]).

9 Since it is assumed that each agent is able to complete the tasks assigned to it and no two or more tasks in an assignment set are ρ-related.

10 Since a plan P_i at least has to satisfy all intra-agent constraints $\prec_i$.

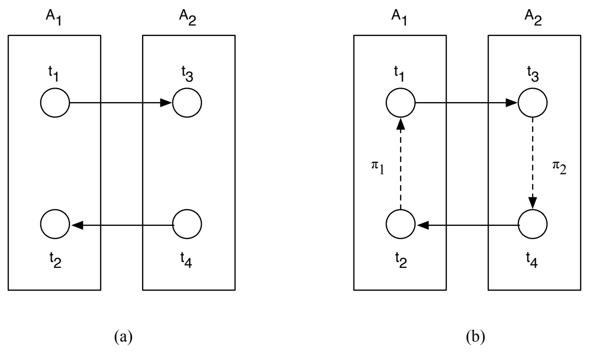

Figure 2. A set of interdependent tasks $T = \{t_1, t_2, t_3, t_4\}$ and two agents A_1 and A_2 each assigned to a part of T (a). If agent A_1 decides to make a plan where t_2 precedes t_1 and A_2 makes a plan where t_3 precedes t_4 (see b), these plans cannot be combined.

agents iff the simple joining $J = (\bigcup_{i=1}^{n} T_i, \pi_J)$ of them induces a partial ordering of $T' = \bigcup_{i=1}^{n} T_i$, i.e., if π_J^+ is acyclic. So it suffices to concentrate on simple joinings.

4. Coordination problems

If, for a given fixed task instance, it holds that whatever individual plans are constructed, their simple joining is always acyclic, the instance is said to be *coordinated*. Clearly, coordinated instances guarantee independent planning without the need to revise individual plans. Unfortunately, not every fixed task instance is coordinated and the question arises how to induce this property for every task instance:

Example 4.1 *Consider two agents A_1 and A_2 and four tasks $T = \{t_1, t_2, t_3, t_4\}$ (see Figure 2 (a)): The precedence relation $\prec$ is given as $\prec = \{(t_1, t_3), (t_4, t_2)\}$. Suppose that t_1, t_2 are assigned to A_1 and t_3, t_4 to A_2. Then A_1 has to solve the subtask $(\{t_1, t_2\}, \emptyset)$, while A_2 has to solve $(\{t_3, t_4\}, \emptyset)$. Note that $\prec_{inter} = \prec$. Suppose now A_1 chooses a plan where t_2 will be performed before t_1 and A_2 chooses a plan where t_3 will be performed before t_4 (see Figure 2 (b)). Then there exists no feasible joint plan preserving $\prec$ and the individual plans since the combination of their plans with the inter-agent constraints constitutes a cycle: $t_1 \prec t_3 \; \pi_2 \; t_4 \prec t_2 \; \pi_1 \; t_1$, implying that t_1 has to be performed before t_1.*

The solution we propose is to add a *minimum* set of intra-agent precedence constraints (called a *coordination set*) such that the independence-threatening inter-agent constraints are made harmless: Looking back at Example 4.1, a possible solution is to add — prior to planning — an additional constraint, for instance $t_1 \prec t_2$, to the set of intra-agents constraints of agent A_1. Then, whatever plans the agents come up with (respecting their intra-agent constraints, of course), the results can always be combined into an acyclic joint plan: by adding such a coordination set the instance has become a coordinated instance. In general, the solution if to specify, for each agent A_i, a minimum set Δ_i of additional intra-agent constraints such that the resulting instance $([T_i]_{i=1}^n, \prec \cup \Delta)$, with $\Delta = \bigcup_1^n \Delta_i$, is a coordinated instance. It is not difficult to show that such a set Δ always exists:

Proposition 4.2 *Let $([T_i]_{i=1}^n, \prec)$ be a fixed task instance. Then there always exists a set $\Gamma \subseteq \bigcup_{i=1}^n T_i \times T_i$ such that $([T_i]_{i=1}^n, \prec \cup \Gamma)$ is a fixed task instance that is coordinated.*

Proof. Since $\prec^+$ is a partial order, there always exists a total ordering $\prec^*$ of T extending $\prec^+$. For each T_i, let Γ_i be a smallest set of precedence constraints such that $(\prec_i \cup \Gamma_i)^+ = \prec^* \cap (T_i \times T_i)$ and let $\Gamma = \bigcup \Gamma_i$. Clearly, $(\prec \cup \Gamma)^+ \ll \prec^*$ is a partial order, so $([T_i]_{i=1}^n, \prec \cup \Gamma)$ is a task instance. Moreover, for every $i = 1, \ldots, n$, $(\prec_i \cup \Gamma_i)^+$ totally orders T_i; hence, for every individual plan $P_i = (T_i, \pi_i)$ we must have $\pi_i \equiv (\prec_i \cup \Gamma_i)$. Hence, $\prec \cup \bigcup_{i=1}^n \pi_i \equiv \prec \cup \Gamma$ is acyclic and therefore the instance is coordinated. □

Given that the set of tasks and precedence constraints is finite, by Proposition 4.2, it follows that there always exists a minimum set $\Delta \subseteq \bigcup_{i=1}^n T_i \times T_i$ such that $([T_i]_{i=1}^n, \prec \cup \Delta)$ is a task instance that is coordinated.[11] In the following section we will analyze the computational complexity of some variants of the coordination problem and the factors that influence their complexity.

5. Complexity results

We will start with the easiest variant of the coordination problem: to *verify* for a *fixed* task instance whether individual plans always can be joined whatever plans may be composed by the participating agents, i.e., determining whether a task instance is already coordinated:

FIXED COORDINATION VERIFICATION (FIXCV) Given a fixed task instance $([T_i]_{i=1}^n, \prec)$, is it true that, for every $i = 1, \ldots, n$, if the extensions $\pi_i \subseteq (T_i \times T_i)$ of $\prec \cap (T_i \times T_i))$ are acyclic, then the relation $\prec \cup \bigcup_{i=1}^n \pi_i$ is acyclic as well?

This problem is co-NP complete: it is in co-NP because we can polynomially verify a counter example consisting of a set of agent plans that create a cycle in the joint plan. Determining that no such counter example can be found — the instance is coordinated — is at

11 Elsewhere ([18]) we have presented a distributed (approximation) algorithm to approximate such a minimum set Δ of additional constraints.

least as hard as the NP-complete PATH WITH FORBIDDEN PAIRS (PWFP, see [8]) problem, however.[12]

Going from fixed task instances to free task instances implies the addition of task assignment problems in the coordination verification problem. These task assignment problems constitute an independent factor of complexity as the total complexity goes up one step in the polynomial hierarchy: the following FREECV-problem turns out to be Σ_2^p-complete:

FREE COORDINATION VERIFICATION (FREECV) Given a free task instance $(T, \rho, \prec, A, \vec{c}(A), \vec{c}(T))$ does there exist a single-agent task assignment such that the resulting fixed task instance $([T_i]_{i=1}^n, \prec)$ is coordinated?

By first guessing a task assignment and then using a FIXCV-oracle for the resulting fixed task instance we could verify a yes-instance in polynomial time. Hence, the problem is in Σ_2^p. Hardness for this class is shown by reducing a Σ_2^p-complete quantified version of the PWFP-problem to it.

It is interesting to note that the problem of finding a suitable single-agent assignment for a free task instance is NP-hard for consumable capabilities[13] and polynomially solvable for non-consumable capabilities. These differences in complexity, however, disappear when these assignment problems interact with the coordination problem: the FREECV-problem turns out to be Σ_2^p-hard for both assignment conditions.

Both verification problems ask whether task instances are coordinated. More complicated coordination problems ask for the existence of bounded sets of precedence constraints (coordination sets) that, when added to a task instance, render it coordinated:

FIXED COORDINATION ($\exists$FIXC) Given a fixed task instance $([T_i]_{i=1}^n, \prec)$ and a positive integer[14] $K > 0$, does there exist a set $\Delta \subseteq \bigcup_{i=1}^n (T_i \times T_i)$ 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 FIXCV-oracle whether the instance $([T_i]_{i=1}^n, \prec \cup \Delta)$ is coordinated. Since FIXCV$\in$ co-NPC, it follows that FIXC$\in \Sigma_2^p$. Elsewhere, we have shown this $\exists$FixC problem to be Σ_2^p-complete.

It would be reasonable to assume that one source of complexity of the coordination problem can be attributed to the *number of tasks* each agent receives and — indirectly — to the complexity of the single-agent planning problems. This, however, turns out *not* to be the case: even if the single-agent planning problems are trivial, this coordination problem remains intractable. For example, if each agent receives only two tasks, the $\exists$FIXC-problem is already co-NP-complete and Σ_2^p-completeness can be proven already for agents having 8 tasks or more.[15]

Since fixed and free variants differ in complexity with respect to the coordination-verification problem, one would expect the same complexity differences to occur between the fixed and free variants of the coordination problem. Consider the following free-variant:

FREE COORDINATION ($\exists$FREEC) Given a free task instance $(T, \rho, \prec, A, \vec{c}(A), \vec{c}(T))$ and a positive integer $K > 0$, does there exist an assignment of tasks to agents and a coordination set Δ 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 FIXCV-oracle that the given instance is a yes-instance. Therefore, the problem is no harder than the$\exists$ FIXC-problem. Hence, the additional task of producing an assignment does not increase the complexity of the problem in an essential way.

It turns out the most difficult coordination problems have to do with guaranteeing that *every* assignment of agents to tasks results in a coordinated task instance if we are allowed to add at most K-constraints:

FREE FOR ALL COORDINATION ($\forall$FREEC) Given a free task instance $(T, \rho, \prec, A, \vec{c}(A), \vec{c}(T))$ and a positive integer $K > 0$, is it true that for every feasible assignment of tasks to agents, there exists a coordination set $\Delta \subseteq \bigcup_{i=1}^n (T_i \times T_i)$ with $|\Delta| \leq K$ such that the 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 $\exists$FIXC -problem, we can verify a counter-example in polynomial time. Hence, the problem is in Π_3^p and also turns out to be complete for this class, too.

6. Discussion

We have introduced a task-based framework to discuss some computational aspects of a coordination problem for non-cooperative agents. We have analyzed the computational complexity of some variants of this problem and discussed some factors and their interaction contributing to this complexity. Although these problems turn out to be intractable for already simple task instances, elswhere we have shown that reasonable solutions can be obtained by adding (nearly minimal) sets of additional constraints.

To conclude this paper, we would like to point out some broader perspectives on the approach to the coordination problem(s) as we discussed above.

First of all, our pre-planning coordination problem can be viewed both as a *decomposition problem* as well as a *re-*

12 Due to lack of space, all complexity proofs have been omitted. For details consult [17].

13 By reduction from e.g. PARTITION.

14 Note that for $K = 0$ this problem is equivalent to FIXCV.

15 So we have still a complexity gap between 2 and 8 tasks per agent.

vision by minimal change problem: how to ensure that solutions (plans) to independently solved subproblems always can be integrated into an overall solution (a coordinated plan) by minimally changing the original problem, i.e. by adding a minimal set of additional constraints to the original problem? Essentially, our approach to the coordination problem then suggests the following (central or distributed) algorithmic method to solve these and possibly other multi-agent problems: try to minimally change the original problem such that a divide-and-conquer approach can be used to solve the problem by decomposing it into a number of independent subproblems whose solution can be simply joined to compose the overall solution.

Secondly, with respect to planning technology, we note that methods using this approach would be able to seamlessly *integrate* existing single-agent planning tools into a multi-agent environment: decompose a multi-agent problem into a number of independent single-agent planning problems by minimally revising the original problem, let the agents work on them using their own planning technology and then integrate the results into an overall solution just by joining the individually constructed plans.

References

[1] J.S. Cox and E. H. Durfee. Discovering and exploiting synergy between hierarchical planning agents. In *Second International Joint Conference On Autonomous Agents and Multiagent Systems (AAMAS '03)*, 2003.

[2] M.M. de Weerdt, A. Bos, H. Tonino, and C. Witteveen. A resource logic for multi-agent plan merging. *Annals of Mathematics and Artificial Intelligence 37*, 2003.

[3] K. S. Decker and V. R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence (DAI-94)*, pages 65–84, 1994.

[4] M Bernardine Dias and A.Stentz. Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination. Technical Report CMU-RI -TR-03-19, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2003.

[5] E. H. Durfee and V. R. Lesser. Partial global planning: a coordination framework for distributed hypothesis formation. *IEEE Transactions on systems, Man, and Cybernetics*, 21(5):1167–1183, 1991.

[6] E. Ephrati and J. S. Rosenschein. Multi-agent planning as the process of merging distributed sub-plans. In *Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence (DAI-93)*, pages 115–129, 1993.

[7] K. Erol, J. Hendler, and D.S. Nau. HTN planning: Complexity and expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, volume 2, pages 1123–1128, Seattle, Washington, USA, 1994. AAAI Press/MIT Press.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness*. W.H.Freeman, 1979.

[9] B. P. Gerkey and M.J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. In *International Journal of Robotics Research*, pages 23(9):939–954, 2004.

[10] N. R. Jennings. Coordination techniques for. pages 187–210. 1996.

[11] V. Lesser, K. 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/TAEMS Domain-Independent Coordination Framework. *Autonomous Agents and Multi-Agent Systems*, 9(1):87–143, July 2004.

[12] Victor R. Lesser. Cooperative multiagent systems: A personal view of the state of the art. *IEEE Trans. Knowl. Data Eng.*, 11(1):133–142, 1999.

[13] Th. W. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM computing surveys*, 1993.

[14] F. Von Martial. *Coordinating Plans of Autonomous Agents*, volume 610 of *Lecture Notes on Artificial Intelligence*. Springer Verlag, Berlin, 1992.

[15] T. W. Sandholm. Distributed rational decision making. pages 201–258, 1999.

[16] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 1998.

[17] A. W. ter Mors, J. Valk, and C. Witteveen. Complexity of coordinating autonomous planning agents. Technical Report PDS-2004-002, Delft University of Technology, 2004.

[18] J. Valk and C. Witteveen. Multi-agent coordination in planning. In *Seventh Pacific Rim International Conference on Artificial Intelligence*, pages 335–344, Tokyo, Japan, august 2002. Springer Verlag, Berlin.

[19] W. Walsh and Wellman. A market protocol for decentralized task allocation and scheduling with hierarchical dependencies. In *Proc. of the 3rd Int. Conf. on Multi-Agent Systems*, pages 325–332, 1999.

[20] G. Weiß, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, San Francisco, CA, 1999.

[21] R. Zlot and A. Stentz. Market-based multirobot coordiation using task abstraction. In *Market-based Multirobot Coordiation Using Task Abstraction, International Conference on Field and Service Robotics*, 2003.