Task Coordination for Non-cooperative Planning Agents

Pieter Buzing[1], Adriaan ter Mors[1], Jeroen Valk[1] and Cees Witteveen[1,2]

[1] Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands, {p.buzing, a.w.termors, j.m.valk, c.witteveen}@ewi.tudelft.nl

[2] Centre for Mathematics and Computer Science, P.O. Box 94079, NL-1090 GB Amsterdam, The Netherlands, C.Witteveen@cwi.nl

Abstract. We consider task planning problems where a number of non-cooperative agents have to work on a joint problem. Such a problem consists in completing a set of interdependent, hierarchically ordered tasks. Each agent is assigned a subset of tasks to perform for which it has to construct a plan. Since the agents are non-cooperative, they insist on planning independently and do not want to revise their individual plans when the joint plan has to be assemled from the individual plans. We present a formal framework to study some computational aspects of this *non-cooperative coordination* problem.

1 Introduction

Many real-life task planning problems such as e.g. manufacturing, logistic planning and air traffic control can be easily modelled as multi-agent planning problems: using their individual planning tools and capabilities, a set of planning agents must come up with a joint solution to a problem consisting of a set of interdependent tasks. Typically, none of the agents is capable to solve all tasks and each agent is assigned a disjoint subset of tasks to perform. To complete its part of the job, each agent has to come up with a plan to perform the tasks assigned to it. Since interdependent tasks may be assigned to different agents, in general, the agents are not completely free to construct their own plan; therefore some form of *coordination* is needed between them to come up with a joint plan to complete the planning problem.

In the literature one can distinguish two main approaches to coordination in multi-agent planning problems. In the first approach (cf [1, 7, 12]) coordination between the agents is established *after* the completion of the individual planning processes. It is assumed that agents independently work on their own part of the planning problem and achieve a solution for it. Then, in an after-planning coordination phase, possible conflicts between independently generated individual plans are resolved and positive interactions between them are exploited by exchanging and revising parts of the individual plans.

The second approach (cf. [2–4, 6]) treats coordination and planning as *intertwined* processes where the agents continuously exchange planning information to arrive at a joint solution. For example, in the (G)PGP framework ([2, 3]), planning and coordination are regarded as part of an iterative process, with plans of various levels of abstraction being exchanged (and modified) between agents to achieve a feasible, efficient, and coordinated plan. From a coordination perspective, the main difference with the first approach is that positive (negative) interactions between *partial* individual plans are exploited (resolved) *before* each of the agents comes up with a completely developed plan. This approach therefore should be considered as a coordination *during* planning approach where the agents are assumed to be fully cooperative, not only with respect to exchanging information, but also with respect to willingness to revise already constructed (partial) plans.

In both these *coordination by synergy* (cf. [1]) approaches, it is taken for granted that the individual agents are willing to share information and, in case of conflicts, to revise their individual plans. Hence, these approaches are less appropriate if the agents have a *non-cooperative relationship* to each other: in such cases the participating agents will require *full planning autonomy*, that is they *(i)* insist to plan their part independently from the others; *(ii)* are not prepared to revise their plan if a joint plan is composed.

This implies that the only possibility for the agents to coordinate is to do so *before* they start to plan. In this paper we concentrate on such a coordination *before* planning approach for non-cooperative planning agents. Elsewhere (see [15, 11]), we have discussed some approximation algorithms for achieving such pre-planning coordination. Here, our main goal is to present a formal framework to discuss pre-planning coordination problems and to point out the complexity of the resulting coordination problems.

Organization In Section 2, we present a framework to specify the elements of our coordination approach. We introduce complex tasks to specify the joint planning problems we are interested in and we distinguish between precedence and refinement relations between tasks. In Section 3 we introduce some coordination problems and in Section 4 we analyze their complexity in relation to the assignment problems distinguished. Finally, in Section 5 we point out some relations of this coordination problem to the general revision-by-minimal-change idea and we indicate the significance of this approach for reusing existing single agent planning methods to solve multi-agent planning problems.

2 Framework

We assume that a set of independent non-cooperative agents $A = \{A_1, A_2, \ldots, A_n\}$ has to complete some joint *complex task* $\mathcal{T}$. Such a complex task[1] is a structure $\mathcal{T} = (T, \rho, \prec)$ where $T = \{t_1, t_2, \ldots t_k\}$ is a set of tasks and both ρ and $\prec$ are asymmetric and irreflexive relations between tasks $t \in T$. The relation $\prec$ spec-

[1] Complex tasks can be seen to extend *task trees* (cf. [16]).

ifies a *precedence*[2] 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'. Note that we do not require $\prec$ to be transitively closed[3].
[A few notes on terminology: (*i*) we will use σ^+ to stand for the *transitive closure* of a relation σ and σ^- to denote its *transitive reduction*; (*ii*) two relations σ and τ are called *equivalent*, denoted as $\tau \equiv \sigma$, iff $\tau^+ = \sigma^+$; (*iii*) a relation τ is said to *extend* σ, denoted by $\sigma \ll \tau$, iff $\sigma^+ \subseteq \tau^+$.]
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 [5]). 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. 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 $\rho_\vee$, or completely in $\rho_\wedge$. Intuitively, if $\rho(t) \subseteq \rho_\wedge$, task t can be completed by performing t or by completing every subtask $t' \in \rho(t)$; if $\rho(t) \subseteq \rho_\vee$, t can be completed by performing t 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.

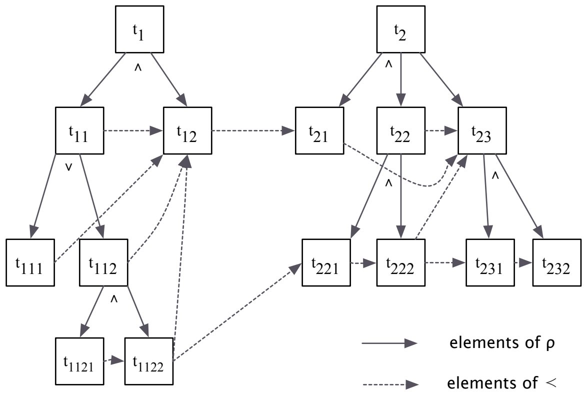

Fig. 1. A complex task with refinement (ρ) and precedence ($\prec$) relations between tasks. $T_0 = \{t_1, t_2\}$ is the set of initial tasks. Other tasks are refinements of these tasks. Note that $\{t_{21}, t_{22}, t_{23}\} = \rho_\wedge(t_2)$, while $\rho_\vee(t_{11}) = \{t_{111}, t_{112}\}$.

The relations ρ and $\prec$ are related as follows: First of all, ρ and $\prec$ are orthogonal relations, i.e., $\rho^+ \cap \prec^+ = \emptyset$: precedence relations only exist between tasks

[2] Note that such a precedence relation can be induced by various other dependency relations like resource dependencies, organizational regulations, etc.
[3] We make a distinction between the relation as specified and the relation induced by its transitive closure.

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

Definition 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 2 (Task network completion). *A task network (T, ρ) is said to be completed if every task t in the set of initial tasks $T_0 = \{t \mid \rho^{-1}(t) = \emptyset\}$ 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 $\rho(t) = \emptyset$).

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 $\boldsymbol{c}(A_i) = (c_1(A_i), \ldots, c_m(A_i)) \in (\mathbb{N} \cup \{\infty\})^m$, where $c_j(A_i)$ specifies *how much* A_i has of capability c_j (we will assume integral quantities). Similarly, the vector $\boldsymbol{c}(t_j) = (c_1(t_j), \ldots, c_m(t_j)) \in \mathbb{N}^m$ specifies how much of each capability it takes to execute task $t_j \in T$. An agent A_i is said to be able to perform a subset of tasks $T_i \subseteq T$ iff $\boldsymbol{c}(A_i) \geq \Sigma_{t \in T_i} \boldsymbol{c}(t)$ where $\boldsymbol{x} \geq \boldsymbol{y}$ iff for all $i = 1, \ldots, m$, $x_i \geq y_i$. 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 capability (i.e., knowledge or a skill). In the sequel, we will abbreviate the set of agent capability vectors and the set of task capability vectors by $\boldsymbol{c}(A)$ and $\boldsymbol{c}(T)$, respectively. A typical instance of our coordination problem is specified as the *free* coordination instance $(T, \rho, \prec, A, \boldsymbol{c}(A), \boldsymbol{c}(T))$. 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 $T' \subseteq T$ we call a *candidate assignment set* and is defined as follows:

Definition 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 ihas to be made to complete a task by OR-subtasks);*
3. *(T, ρ) is completed by performing the tasks in T' (cf. Definition 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[4]. An *assignment set* is a candidate assignment set T' where every task $t \in T'$ can be assigned to an agent capable to perform it:

Definition 4 (assignment set). *$T' \subseteq T$ is an* assignment set *for a free coordination instance $(T, \rho, \prec, A, \boldsymbol{c}(A), \boldsymbol{c}(T))$ if (i) T' is a candidate assignment set and (ii) there exists a partitioning[5] $[T'] = [T_1, T_2, \ldots, T_n]$ of T' such that agent A_i is able to perform T_i, i.e., $\boldsymbol{c}(A_i) \geq \Sigma_{t \in T_i} \boldsymbol{c}(t)$.*

After an assignment set has been found[6] in a free coordination instance $(T, \rho, \prec, A, \boldsymbol{c}(A), \boldsymbol{c}(T))$, we have obtained a *fixed* coordination instance $([T_i]_{i=1}^n, \prec, A, \boldsymbol{c}(A), \boldsymbol{c}(T))$. Since the capabilities are no longer needed and agents are characterized by the partition blocks of a ρ-independent set $T' \subseteq T$, we often abbreviate fixed coordination instances by the tuple $([T_i]_{i=1}^n, \prec)$. Without loss of generality we assume every block T_i to be non-empty.

3 Coordination problems

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

[4] 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: $T' = T$

[5] Since the agents are planning independently, we only consider *single-agent task assignments* (cf. [10]).

[6] *How* tasks should be assigned to agents is a separate problem, which we do not discuss here. We assume that a suitable task assignment mechanism is selected (cf. [13]).

planning tools used by the agents, or about the particular planning problem the agent has to solve. Whatever plan/schedule representation the agents (internally) employ, we assume that such a plan A_i develops for T_i can be represented as a structure $P_i = (T_i, \pi_i)$ extending[7] the structure $(T_i, \prec_i)$, i.e., π_i^+ is a partial order such that $\prec_i \ll \pi_i$.

Remark 1. When making a plan for a set of tasks T_i, the simplest case is when A_i exactly knows how to perform a task t and the actions to perform it bear no relation to the actions performed for another task t'. In that case, a plan for T_i can be easily composed from the set of actions needed to perform each individual task. Sometimes, however, there is no such compositionality: the plan to perform two tasks differs from a simple ordering of the actions performed for each of the tasks separately. For example, if A_i has to perform the set of tasks $T_i = \{\text{buy milk}, \text{buy cookies}\}$ then his plan might be something along the lines of: go to supermarket ; get milk ; get cookies ; pay ; go home — the agent will not first go to the supermarket for the milk, and later return to the supermarket for the cookies. Thus, in the latter case, the agent has to solve a real planning problem to complete T_i. ■

The central coordination problem now can be stated as follows: 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 choice of an individual plan will be allowable in order to construct a joint plan. How then should we coordinate the agents *before planning* in order to guarantee both independent planning and a revision-free combination of plans?

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

Definition 5. *A plan P is a joint plan for the coordination instance $([T_i]_{i=1}^n, \prec)$ if $P = (T', \pi)$, where $T' = \bigcup_{i=1}^n T_i$, π^+ is a partial order and π extends $\prec$, i.e., $\prec \ll \pi$.*

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

Definition 6. *A joint plan $P = (T', \pi)$ respects [exactly respects] the individual plan $P_i = (T_i, \pi_i)$ of agent A_i if $T_i \subseteq T'$ and $\pi_i \ll (\pi \cap (T_i \times T_i))$ [$\pi_i \equiv (\pi \cap (T_i \times T_i))$].*

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 7 (Simple joining). *Given a fixed coordination 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 $J = (\bigcup_{i=1}^n T_i, \rho_J)$, where $\rho_J \equiv (\prec_{inter} \cup (\bigcup_{i=1}^n \pi_i))$.*

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

Now it is not difficult to see that the simple joining exhibits a tell-tale property w.r.t. respecting individual plans:

Proposition 1. *Given a fixed coordination instance* $([T_i]_{i=1}^n, \prec)$*, there exists a joint plan* P *for it (exactly) respecting the individual plans* $P_i = (T_i, \pi_i)$ *of the agents iff the simple joining* $J = (\bigcup_{i=1}^n T_i, \rho_J)$ *of them induces a partial ordering of* $T' = \bigcup_{i=1}^n T_i$*, i.e., if* ρ_J^+ *is acyclic.*

Proof. ($\rightarrow$). Every joint plan $P = (T', \pi)$ (exactly) respecting the collection of individual plans $P_i = (T_i, \pi_i)$ has to satisfy (at least) the inter-agent constraints $\prec_{inter}$ and each of the individual planning constraints π_i. Therefore, $\rho_J \ll \pi$ and since π is acyclic, ρ_J must be acyclic as well. ($\leftarrow$) Note that by definition J is a joint plan satisfying $(T', \prec)$. □

If, for a given fixed coordination 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 whenever a joint plan has to be constructed. Unfortunately, not every fixed coordination instance is coordinated (cf. Example 1) and the question arises how to induce this property for every coordination instance.

Example 1. Consider the following simple case (see Figure 2 (a)): There are two agents A_1 and A_2 and four tasks $T = \{t_1, t_2, t_3, t_4\}$. 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 consists in adding a *minimum* set of additional intra-agent precedence constraints (called a *coordination set*) such that the independence threatening inter-agent constraints are made harmless: Looking back at Example 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, such a solution consists in specifying, 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 2. *Let* $([T_i]_{i=1}^n, \prec)$ *be a fixed coordination 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 coordination instance that is coordinated.*

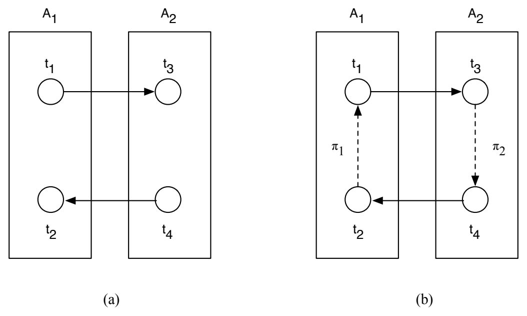

Fig. 2. A set of interdependent tasks $T = \{t_1, t_2, t_3, t_4\}$ and two agents A_1 and A_2 each assigned to a part of T (a). If agent A_1 decides to make a plan where t_2 precedes t_1 and A_2 makes a plan where t_3 precedes t_4 (see b), these plans cannot be combined.

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 coordination 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 2, it follows that there always exists a cardinality-minimal set $\Delta \subseteq \bigcup_{i=1}^n T_i \times T_i$ such that $([T_i]_{i=1}^n, \prec \cup \Delta)$ is a coordination instance that is coordinated. In another paper [15], we have presented a distributed (approximation) algorithm to approximate such a minimum set Δ of additional constraints.
In the following section we will now analyze the computational complexity of some variants of the coordination problem.

4 Coordination problems: complexity results

In the previous section we introduced a simplest version of the coordination problem. More formally, this problem is specified as follows:

Fixed Coordination Verification (FixCV) Given a fixed coordination instance $([T_i]_{i=1}^n, \prec)$, is it coordinated, i.e. is it true that whenever, for every $i = 1, \ldots, n$, the extensions $\pi_i \subseteq (T_i \times T_i)$ of $\prec \cap (T_i \times T_i))$ are acyclic, the relation $\prec \cup \bigcup_{i=1}^n \pi_i$ is acyclic as well?

In a previous paper ([14]) we have shown this problem to be co-NP complete[8]. Intuitively, a counter example for such an instance can be verified in polynomial

[8] Due to lack of space, all complexity proofs have been omitted.

time, but finding it is at least as hard as finding examples to instances of the complement of the NP-complete Path With Forbidden Pairs problem (PWFP) [9].

Also *free coordination instances* can be checked for being coordinated:

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

It should come as no surprise that the FREECV-problem is harder than the FIXCV-problem: by first guessing a task assignment and using a FIXCV-oracle for the resulting fixed coordination 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 (cf. [14]).

Remark 2. It is easy to show that the problem to find a suitable single-agent assignment for a free coordination instance is NP-hard for consumable capabilities[9] 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 problems ask whether coordination instances are coordinated. More complicated coordination problems ask for the existence of bounded sets of precedence constraints that, when added to a coordination instance, make it coordinated:

FIXED COORDINATION ($\exists$FIXC) Given a fixed coordination instance $([T_i]_{i=1}^n, \prec)$ and a positive integer[10] $K > 0$, does there exist a coordination set $\Delta \subseteq \bigcup_{i=1}^n (T_i \times T_i)$ with $|\Delta| \leq K$ such that the fixed coordination 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 FIXC-oracle whether the instance $([T_i]_{i=1}^n, \prec \cup \Delta)$ is coordinated. Since FIXCV$\in$ co-NPC, it follows that FIXC$\in \Sigma_2^p$. In a previous paper we have shown the $\exists$FixC problem to be Σ_2^p-complete [14].

At first sight the next problem, Free Coordination, might seem to be more difficult than the Fixed version:

FREE COORDINATION ($\exists$FREEC) Given a free coordination instance $(T, \rho, \prec, A, \boldsymbol{c}(A), \boldsymbol{c}(T))$ 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 coordination instance $([T_i]_{i=1}^n, \prec \cup \Delta)$ is coordinated?

Note, however, 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

[9] By reduction from e.g. PARTITION.

[10] Note that for $K = 0$ this problem is equivalent to FIXCV.

is a yes-instance. Therefore, the problem is no harder than the FIXC-problem. So, in this case, the complexity of coordination absorbs the complexity of assignment.

As it turns out the most difficult coordination problems have to do with guaranteeing that every assignment of agents to tasks results in a coordinated or nearly coordinated instance:

FREE FOR ALL COORDINATION ($\forall$FREEC) Given a free coordination instance $(T, \rho, \prec, A, \boldsymbol{c}(A), \boldsymbol{c}(T))$ 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?

Note that by guessing an assignment and using a Σ_2^p-oracle for the resulting $\exists$FIXC -problem, we can verify a counter-example. Hence, the problem is in Π_3^p and turns out to be complete for this class, too.

5 Relations to other problems and Conclusions

Conceptually, the pre-planning coordination problem has all the characteristics of a *revision by minimal change* problem (cf. [8]): given a system S and a property P that S does not but should have, how to minimally change S into a nearly equivalent system S' such that P applies to S'. In our coordination problem, the system S is a set of interdependent tasks assigned to agents and the desired property is to obtain a joint plan by independent planning. Note that the problem to solve here is slightly more complicated because the object S is a construction from a free coordination instance and a task assignment process. Therefore, more complicated revision-like problems can be raised as: does there exist a process (a task assignment) resulting in a system S satisfying P, or: how to ensure that the (task assignment) process results in a system S that minimally differs from a system S' having property P.

From an algorithmic point of view the way we approached the coordination problem is in spirit similar to the well-known *divide-and-conquer* method: how to decompose a given problem in a number of independent subproblems that each can be solved independently whereafter the solutions can be easily assembled into an overall solution. Essentially, our approach to the coordination problem comes down to an attempt to *extend* the divide-and-conquer method by making it applicable even to problems that do not allow decomposition: by allowing minimal revision of the original problem, more problem instances can be solved by the divide-and-conquer approach than without.

Finally, with respect to planning technology, we note that this approach also can be used to *integrate* existing single-agent planning tools into a multi-agent environment: just decompose your favorite multi-agent problem into a number of independent single-agent problems by minimally revising the original problem, let the agents work on them, and then integrate the results into an overall solution just by joining the individually constructed plans. Since we have shown that minimal revision is computationally infeasible, practical applications of this idea should aim at *approximating* such a minimal revision. In [15] we have applied such approximations to solve logistic planning problems.

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. 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.
3. 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.
4. 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.
5. 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.
6. V. Lesser et al. Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. *Autonomous Agents and Multi-Agent Systems*, 9(1):87–143, July 2004.
7. D.E. Foulser, M. Li, and Q. Yang. Theory and algorithms for plan merging. *Artificial Intelligence Journal*, 57(2–3):143–182, 1992.
8. N. Friedman and J. Y. Halpern. Modeling belief in dynamic systems, part II: Revision and update. *Journal of Artificial Intelligence Research*, 10:117–167, 1999.
9. M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness*. W.H.Freeman, 1979.
10. 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.
11. J.M.Valk, M.M. de Weerdt, and C. Witteveen. Algorithms for coordination in multi-agent planning. *I. Vlahavas and D. Vrakas (ed.), Intelligent Techniques for Planning (to appear)*, 2004.
12. F. Von Martial. *Coordinating Plans of Autonomous Agents*, volume 610 of *Lecture Notes on Artificial Intelligence*. Springer Verlag, Berlin, 1992.
13. O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 1998.
14. A. W. ter Mors, J. Valk, and C. Witteveen. Complexity of coordinating autonomous planning agents. Technical Report PDS-2004-002, Delft University, 2004.
15. J. Valk and C. Witteveen. Multi-agent coordination in planning. In *Proceedings PRICAI 2002*, pages 335–344, Tokyo, Japan, august 2002. Springer.
16. R. Zlot and A. Stentz. Market-based multirobot coordination using task abstraction. In *Market-based Multirobot Coordiation Using Task Abstraction, International Conference on Field and Service Robotics*, 2003.