Multi-Agent Coordination in Dynamic Task Environments[1]

Jeroen Valk [ab] Peet van Tooren [a]

[a] Almende B.V., Rotterdam, {jeroen,peet}@almende.com
[b] Delft University of Technology, J.M.Valk@ewi.tudelft.nl

Abstract

This paper considers a dynamic variant of a coordination problem that was studied in, e.g., [9, 7, 8]. The original problem is concerned with a static task environment where all composite tasks consisting of elementary tasks and their interdependencies are given in advance. We introduce a dynamic variant of this problem where multiple composite tasks arrive dynamically in the system. We present a framework for dynamic task environments and explain the coordination problem that arises if multiple autonomous agents have to work together on parts of a composite task. From this explanation of the coordination problem we derive a formal definition. As an approach to solve the coordination problem, we suggest the idea of task partitioning. We conclude the paper with a simple solution method: the partitioning algorithm. This algorithm was originally designed to work in static environments, but turned out to be applicable in a dynamic context as well.

1 Introduction

We consider problems where multiple *autonomous agents* have a number of *interrelated elementary tasks* to be done. Sometimes, elementary tasks can be performed by the agent that owns them, but it may also happen that tasks must be delegated to other agents. If an agent delegates its tasks to different other agents then no longer a single agent is responsible for the inter-task dependencies. Coordination among the group of agents carrying out the elementary tasks is required in order to respect the so-called *inter-agent dependencies*, which are interdependencies between tasks that are carried out by different agents.

Coordination among agents to cope with inter-agent dependencies is an important problem that is addressed in the field of multi-agent systems (MAS): a subfield of distributed artificial intelligence (DAI). In existing solutions to coordinate inter-agent dependencies planning- and coordination processes are often intertwined, e.g., [2, 4]. On the other hand, coordination of inter-agent dependencies is somewhat neglected in approaches that do have a clear separation of planning and coordination, e.g., [6, 3, 5, 1]. For example, in [5] inter-agent dependencies are dealt with by assigning interrelated tasks to the same agent.

[1]This project is a cooperation between the TU Delft, Faculty EEMCS and Almende B.V., Rotterdam. The work is supported by the Dutch Ministry of Economic affairs under the SENTER TSIT program Cybernetic Incident Management (TSIT2021).

In this paper, we take a different approach. Separating planning and coordination, allows us the emphasize on the *planning autonomy* of the agents. We propose solutions that cast *minimal* restrictions on individual planning activities that are just sufficient to guarantee coordination regardless how each agent plans to fulfill its part.

2 Dynamic Task Environments

We consider coordination problems specified by a set $\mathcal{A} = \{A_1, \ldots, A_n\}$ of *agents*, a (possibly infinite) set $\mathcal{T}$ of so-called *elementary tasks*, and a set $\{\Gamma_1, \ldots, \Gamma_n\}$ of subsets $\Gamma_i \subseteq \mathcal{T}$ representing the *agent skills*. Elementary tasks arrive in a dynamic way as disjoint finite partially ordered subsets $T_{i,j} \subseteq \mathcal{T}$. The partial orders on the $T_{i,j}$'s, denoted $\prec_{i,j}$, specify precedences that must be respected during task-execution, i.e., $t_1 \prec_{i,j} t_2$ means that task t_1 must have been completed before execution of t_2 is allowed to be started. A pair $(T_{i,j}, \prec_{i,j})$ is called a *composite task*.

In particular, composite tasks $(T_{i,j}, \prec_{i,j})$ arrive dynamically as a result of an incentive of agent A_i to achieve completion of all tasks in $T_{i,j}$ in such a way that all precedences in $\prec_{i,j}$ are respected. This explains why we use two indices: i is the number of the agent interested in the composite task's completion and j is used to distinguish different task-incentives made by the same agent. At each point in time, the complete *task history*, i.e., union of all composite tasks that have arrived, is denoted by a directed acyclic graph $G = (T, A)$ where $T = \bigcup_{i,j} T_{i,j}$ and A represents the precedences in the $\prec_{i,j}$'s.

An agent A_i is only capable to perform an elementary task $t \in \mathcal{T}$ if the task belongs to its skills, i.e., $t \in \Gamma_i$. An agent could be interested in the realization of some elementary tasks that are beyond its skills, i.e., $T_{i,j}$'s need not be a subset of Γ_i. In this case, or when an agent simply wants to delegate (some of) its tasks to others, a coordination problem occurs. This coordination problem comprises the distribution of responsibilities over the agents to accomplish parts of the composite tasks.

Two issues are distinguished. First, there is the issue of how to allocate elementary tasks to agents. Task allocation has received much attention in literature on multi-agent coordination, e.g., [5, 10, 1]. The second issue emerges when interrelated tasks are allowed to be assigned to different agents. Assignment of interrelated tasks to different agents results in so-called *inter-agent* precedences. In this paper, we focus on the coordination of inter-agent precedences given that existing methods are used to solve the task allocation problem.

We do not assume task allocations are established instantaneously. Rather, we view task allocation as a continuous process which is intertwined with coordination of inter-agent precedences. The collective state of this task allocation process can continuously be captured by a cover $\mathbf{T} = \{T_1, \ldots, T_n\}$. At each point in time, all tasks $T = \bigcup_{i,j} T_{i,j}$ in which the agents are collectively interested are covered, i.e., $\bigcup \mathbf{T} = T$. Each set T_i in the cover contains the tasks that are considered to be assigned to agent A_i. Clearly, it only makes sense to consider tasks for allocation

that belong to the skills of the agent. Therefore, we assume $T_i \subseteq \Gamma_i \cap T$. A newly arrived task would typically be considered for allocation to any agent with the proper skills to carry out the task. Next, tasks could gradually be removed from the T_i's in the cover until eventually a strict partitioning is the result.

It depends on the cover $\mathbf{T}$ whether a precedence is an intra- or inter-agent precedence. A precedence, say $t_1 \prec t_2$, is called an intra-agent precedence if t_1 and t_2 belong to a single T_i. If t_1 and t_2 belong to different T_i's then $t_1 \prec t_2$ is called an inter-agent precedence. Since $\mathbf{T}$ is a cover, a precedence can be intra- and inter-agent precedence at the same time. But during the task allocation process the cover transforms into a partition which leads to a clear separation of intra- and inter-agent precedences.

Intra-agent precedences can be satisfied directly by the agents. However, due to these inter-agent precedences, the activities of one agents may depend on the completion of tasks by others. In this setup, a coordination problem emerges if each agent wants to plan and schedule its own activities independently of others.

3 The Coordination Problem

In our task-oriented setup, a coordination problem arises if multiple self-interested autonomous agents A_i are allocated subsets of a complete set of interrelated elementary tasks. Typically, these agents want to plan and schedule the execution of their own elementary tasks. During planning and scheduling agents don't want to be bothered with the details about other agents' activities. But the inter-agent dependencies make that agents have to rely on each other. Agents are dependent upon others on when they will be allowed to execute their tasks. Thus, to plan/schedule independently information from others must collected in order to determine starting times of tasks.

Autonomous agents, however, need not give this information in advance and an agent may only be willing to make announcements that do not interfere negatively with the possible outcomes of its planning/scheduling process. However, even in very simple situations these negative interferences do occur with a deadlock in information exchange as a result. Roughly speaking, the coordination problem is to resolve this deadlock in an adequate way without putting unacceptable restrictions on individual planning/scheduling activities. To see what we mean by this coordination problem consider[2] the following simple example in which it occurs.

Example 3.1 *We imagine a static world with only four tasks that can be executed. To keep things simple, we assume that these are the only tasks that exist, i.e.,* $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$. *These four task comprise the transportation of two pickup/delivery orders in a multimodal transportation infrastructure as shown in Figure 1. The first order comprises the transportation of a package from* l_1 *to* l_4, *and the second package is to be picked up at location* l_2 *and delivered at* l_3. *For administrative reasons all transportation is supposed to be carried out via the depot d. Therefore, the two orders are decomposed into four elementary tasks*

[2] A formal definition of the coordination problem will be presented later on.

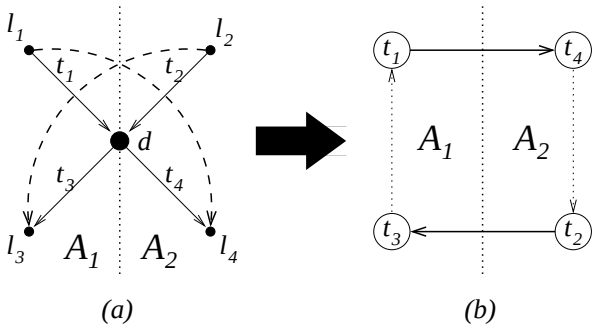

Figure 1: (a) simple pickup/delivery problem and (b) its task-oriented representation

$\mathcal{T} = \{t_1, t_2, t_3, t_4\}$ corresponding to the pickup/delivery pairs (l_1, d), (l_2, d), (d, l_3) and (d, l_4), respectively.

Clearly, the decomposition results in two precedences $t_1 \prec t_4$ and $t_2 \prec t_3$, because post-transport can only be started if pre-transport has already been completed. Let the first task $(T_{1,1}, \prec_{1,1}) = (\{t_1, t_4\}, \{(t_1, t_4)\})$ be announced by agent A_1 and the second task $(T_{2,1}, \prec_{2,1}) = (\{t_2, t_3\}, \{(t_2, t_3)\})$ by agent A_2. We assume that both agents have to start and finish in the depot, and that agent A_1 can only handle odd-numbered locations and agent A_2 only even-numbered locations. Since these are non-overlapping skills, the only feasible cover of $\mathcal{T}$ is: $\mathbf{T} = \{\{t_1, t_3\}, \{t_2, t_4\}\}$. Thus, agent A_1 might offer to carry out task t_3 of agent A_2 and it might want to delegate task t_4 to agent A_2.

Given the re-assignment $\mathbf{T}$ of tasks, each agent aims at a plan/schedule to complete its tasks with the lowest possible costs. The lowest costs correspond to the shortest route to carry out its two tasks. Clearly, agent A_1 would aim for the visiting sequence $d - l_3 - l_1 - d$ and A_2 would aim for the visiting sequence $d - l_4 - l_2 - d$. This means that agent A_1 wants to know when it can start task t_3 before it can announce the ending time of task t_1. Similarly, agent A_2 wants to know when it can start task t_4 before it can announce the ending time of task t_2. The starting time of t_3 and t_4, however, depend on the completion time of t_2 and t_1, respectively. But, for the sake of sticking to their cheapest plan, each agent refuses to announce the required ending times unless the other agent does: deadlock!.

Our general idea is to coordinate by casting constraints on individual agents in such a way that some collective goal can be realized. These intra-agent constraints must be robust in the sense that coordinated solutions are achieved no matter how the individual parties plan/schedule in order to fulfill their part. In our setup, the problem is to cast constraints on the order of task announcements such that potential cycles in information exchange are avoided.

Constraints ϕ_i are interpreted on so-called *announcement plans*. An announcement plan specify the order in which an agent wants to announce ending times of

its tasks. To represent this order we use directed acyclic graphs (dag) $G_i = (T_i, A_i)$ with $T_i \subseteq \mathcal{T}$. For example, agent A_1 in Example 3.1 prefers the announcement plan $G_1 = (\{t_1, t_3\}, \{(t_3, t_1)\})$; agent A_2 prefers the announcement plan $G_2 = (\{t_2, t_4\}, \{(t_4, t_2)\})$. Collectively, the arcs in these two announcement plans consist of the dashed arrows in Figure 1.

Since we use dags to represent announcement plans, agents effectively specify a *partial order* on tasks. Note that this leaves some pairs of tasks unordered. If two task $t_1, t_2 \in T_i$ are unordered in an announcement plan $G_i = (T_i, A_i)$, i.e., there is no path from t_1 to t_2 and no path from t_2 to t_1 in G_i, then we assume that the agent wants to be completely free with respect to the announcement order of t_1 and t_2.

Obviously, agents are not completely free to come up with any announcement plan they like. An announcement plan, say $G_i = (T_i, A_i)$, must respect the intra-agent precedences. This means that for every pair $t_1, t_2 \in T_i$, if $(t_1, t_2) \in A^+$ then there must be a path from t_1 to t_2 in G_i. Effectively, this requirement implies an agent A_i may only add arcs $\Delta_i \subseteq T_i \times T_i$ to a base plan $G_i = (T_i, A_i)$. Collectively, we write $\Delta = \Delta_1 \cup \cdots \cup \Delta_m$ to denote all arcs that are added by the agents. It is important to note that, while adding arcs, we must make sure that each $(T_i, A_i \cup \Delta_i)$ remains acyclic or we are not dealing with an announcement plan anymore.

If the agents are completely free to come-up with any announcement plan they like then a coordination problem can easily occur. To resolve this problem, our idea is to cast constraints ϕ_i on the announcement plans. We write $G_i \models \phi_i$ to denote that a directed acyclic graph G_i satisfies the constraint ϕ_i. The problem is now to find an appropriate vector $\Phi = (\phi_1, \ldots, \phi_m)$ of constraints that guarantee coordination no matter which announcement plans the agents come up with. This gives rise to the following coordination verification problem.

The coordination verification problem (CVP) is: given a triple $(G, \mathbf{T}, \check{T})$ with

(i) a task history, specified as a directed acyclic graph[3] $G = (T, A)$, which may be subject to change due to tasks that arrive in the future,

(ii) a cover $\mathbf{T} = \{T_1, \ldots, T_n\}$ of T which still allows some freedom about the exact allocation of tasks to the agents, and

(iii) a set $\check{T} \subseteq T$ of tasks whose execution must be coordinated.

verify whether no coordination problem can occur due to a deadlock in announcement plans. CVP is a difficult problem: to prove that we are dealing with a no-instance of CVP we must seek a no-certificate which shows the existence of a task allocation (i.e., a partition of tasks) and a vector of corresponding announcement plans that cause deadlock. We structure such certificates as triples $(\hat{T}, \hat{\mathbf{T}}, \Delta)$ with:

(i) $\hat{T}$ a set of tasks that are already in T or that may arrive in the future, i.e., $T \subseteq \hat{T} \subseteq \mathcal{T}$,

[3]The task history $G = (T, A)$ represents all task incentives that have arrived as a graph representation: $T = \bigcup_{i,j} T_{i,j}$ and $A^+ = \bigcup_{i,j} \prec_{i,j}$.

(ii) $\hat{\mathbf{T}} = \{\hat{T}_1, \ldots, \hat{T}_n\}$ a partition of $\hat{T}$ where $\hat{T}_i \subseteq T_i \cup \Gamma_i \setminus T$ for $i = 1, \ldots, n$, and

(iii) $\Delta = \Delta_1 \cup \cdots \cup \Delta_n$ with $\Delta_i \subseteq \hat{T}_i \times \hat{T}_i$ for $i = 1, \ldots, n$.

A triple $(\hat{T}, \hat{\mathbf{T}}, \Delta)$ is a no-certificate of a CVP instance $(G, \mathbf{T}, \check{T})$ if the following two conditions hold:

(i) $(\hat{T}_i, \hat{A}_i \cup \Delta_i)$ where $\hat{A}_i = A^+ \cap T_i \times T_i$ a directed acyclic graph, and

(ii) $(\hat{T}, A \cup \Delta)$ contains a cycle intersecting $\check{T} \times \check{T}$.

The coordination verification problem is now defined as follows.

Definition 4 (Coordination Verification Problem) *The coordination verification problem (CVP) is: given a coordination instance $I = (G, \mathbf{T}, \check{T})$ and a vector $\Phi = (\phi_1, \ldots, \phi_n)$ of constraints, decide whether there does* not *exist a no-certificate $(\hat{T}, \hat{\mathbf{T}}, \Delta)$ of I such that $(\hat{T}_i, \hat{A}_i \cup \Delta_i) \models \phi_i$ for $i = 1, \ldots, m$.*

5 Task Partitioning

A task-partitioning constraint is an expression of the form $\tau_1, \ldots, \tau_m$ where each τ_i is either a finite set $T \subseteq \mathcal{T}$ of tasks or the complement $\neg T = \mathcal{T} \setminus T$ of some finite set $T \subseteq \mathcal{T}$. We say that an expression of the form $\tau_1, \ldots, \tau_m$ is well formed if the τ_i's form a partitioning. For example, $\{t_1\}, \{t_2\}, \neg\{t_1, t_2\}$ is a well-formed expression, but $\neg\{t_1\}, \neg\{t_2\}, \{t_1, t_2\}$ is not, because $\{\mathcal{T} \setminus \{t_1\}, \mathcal{T} \setminus \{t_2\}, \{t_1, t_2\}\}$ is not a partition. The semantics of partitioning constraints is defined as follows.

Definition 6 *A directed acyclic graph $G = (T, A)$ is a model of a task-partitioning constraint $\phi_i = \tau_{i,1}, \ldots, \tau_{i,k_i}$, denoted $G \models \phi_i$, if there exists a path from t_1 to t_2 in G for every pair of tasks t_1 and t_2 such that, for some $1 \leq i < j \leq k_i$, $t_1 \in \tau_i$ and $t_2 \in \tau_j$.*

It turns out that every coordination instance allows a solution as a vector of task-partitioning constraints.

Proposition 7 *For every coordination instance $I = (G, \mathbf{T}, \check{T})$, a vector $\Phi = (\phi_1, \ldots, \phi_n)$ of task-partitioning constraints exists such that $(G, \mathbf{T}, \check{T}, \Phi)$ is a yes-instance of the coordination verification problem.*

Proof Let $I = (G, \mathbf{T}, \check{T})$ be a coordination instance. To construct a vector of task-partitioning constraints, let $\prec$ be a topological sort of the transitive closure A^+ of A. Note that $\prec$ casts total orders on the sets $T_i \cap \check{T}$ as well, because these sets are subsets of T. Thus, we can rank the members of these sets; we write $t_{i,j}$ to denote the j-th element in $T_i \cap \check{T}$ according to $\prec$. The vector $\Phi = (\phi_1, \ldots, \phi_n)$ such that $\phi_i = \{t_{i,1}\}, \ldots, \{t_{i,k_i}\}, \neg\{t_{i,1}, \ldots, t_{i,k_i}\}$ where $k_i = |T_i \cap \check{T}|$ has the desired properties. Suppose, for the sake of contradiction, that a no-certificate $(\hat{T}, \hat{\mathbf{T}}, \Delta)$ exists. The constraints ϕ_i force the Δ_i's to be compatible with the total order $\prec$ on the sets $T_i \cap \check{T}$. It follows that, in order to satisfy (iv), $(\hat{T}, A \cup \Delta)$ must contain

a cycle that is partly in $\check{T}$ and partly in $\hat{T} \setminus \check{T}$. But this is impossible, because the constraints ϕ_i order tasks in $\mathcal{T} \setminus \check{T}$ to be executed after all tasks in $\check{T}$ have been completed. Hence, it is impossible to find a no-certificate. ■

A distributed algorithm which has been developed for static environment can be applied in a dynamic context with only minor modification. In terms of planning autonomy, this algorithm, called the *partitioning algorithm* generally performs better than the construction used in the proof of Proposition 7 which is based on a topological sort of A^+. The partitioning algorithm is a distributed process where each agent has a separate process which operates in rounds. The constraint ϕ_i which the agent must satisfy is refined in each round by making task selections. That is, if in rounds 1 to k_i the tasks $\tau_{i,1}$ to τ_{i,k_i} has been selected then the agent must to satisfy $\phi_i = \tau_{i,1}, \ldots, \tau_{i,k_i}, \neg\tau_i$ where $\tau_i = \tau_{i,1} \cup \cdots \cup \tau_{i,k_i}$.

Not all tasks are allowed to be selected by the agents. An agent may only select a set of tasks if all its members are so-called *prerequisite-free* tasks. We use a blackboard to administrate which tasks are prerequisite free. When an agent makes a task selection $\tau_{i,j}$, this is registered at the blackboard. In turn, agents query the blackboard to find out which tasks are prerequisite free. Initially, tasks without precedences are labeled prerequisite free by the blackboard. But when announcements come in, tasks are labeled prerequisite free if all its predecessors t have been announced by an agent A_i which has been assigned exclusively to t, i.e., there is a unique T_i in the cover $\mathbf{T}$ such that $t \in T_i$.

The following algorithm specifies the process to be executed by each agent.

Algorithm 7.1
begin

1. let $k_i := 1$ and $S_i := \tau_i := \emptyset$ (S_i contains tasks which the agent knows to be prerequisite free);

2. **repeat**

2.1. select a subset $\tau_{i,k_i} \subseteq S_i \setminus \tau_i$;

2.2. inquire the blackboard for an update S_i' of the prerequisite-free tasks;

2.3. **if** *$S_i = S_i'$ and $\tau_{i,k_i} \neq \emptyset$* **then**

2.3.1. announce τ_{i,k_i} to the blackboard;

2.3.2. let $\tau_i := \tau_i \cup \tau_{i,k_i}$;

2.3.3. let $k_i := k_i + 1$;

2.4. **else**

2.4.1. $S_i := S_i'$;

end

The partitioning algorithm is defined as the distributed algorithm given by the blackboard described above and a process as of Algorithm 7.1 for each agent. It can be shown[4] as an invariant that the partitioning algorithm produces adequate vectors of task-partitioning constraints.

[4]The proof is omitted due to lack of space.

Proposition 8 *During the execution of the partitioning algorithm it holds that* $(G, \mathbf{T}, \check{T}, \Phi)$ *with: (i)* $\check{T} = \tau_1 \cup \cdots \cup \tau_n$*, and (ii)* $\Phi = (\phi_1, \ldots, \phi_n)$ *where each* $\phi_i = \tau_{i,1}, \ldots, \tau_{i,k_i}, \neg\tau_i$*, is a yes-instance of the coordination verification problem.*

9 Conclusions

To cope with dynamic environments, we have presented a generalization of the coordination problem defined in [9, 7, 8]. This dynamic variant is at least as hard to solve as the original problem which was already proven to be a computationally hard problem. This makes it difficult (if not impossible) to come up with solutions providing optimal planning autonomy to the agents. Yet, sacrificing some of the planning autonomy, we have been able to find a fast distributed algorithm to solve the coordination problem. Future work could aim at algorithms with a better performance in terms of planning autonomy. Ideas to distribute the role of the blackboard over the agent deserves further attention as well.

References

[1] Randall Davis and Reid Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, January 1983.

[2] Edmund H. Durfee. Organizations, plans, and schedules: An interdisciplinary perspective on coordinating AI agents. *Journal of Intelligent Systems*, 1991. Special Issue on the Social Context of Intelligent Systems.

[3] Eithan Ephrati, Martha E. Pollack, and S. Ur. Deriving multi-agent coordination through filtering strategies. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 679–687, August 1995.

[4] Eithan Ephrati and Jeffrey 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, May 1993.

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

[6] Yoav Shoham and Moshe Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73(1–2):231–252, 1995.

[7] Adriaan ter Mors, Jeroen Valk, and Cees Witteveen. Coordinating autonomous planners. In *Proceedings of IC-AI'04*, 2004.

[8] Adriaan ter Mors, Cees Witteveen, and Jeroen Valk. Complexity of coordinating autonomous planning agents. Technical Report 2004-002, Delft University of Technology, 2004.

[9] J.M. Valk and C. Witteveen. Multi-agent coordination in planning. In *PRICAI 2002: Trends in Artificial Intelligence: 7th Pacific Rim International Conference on Artificial Intelligence*, Tokyo, Japan, 2002.

[10] William E. Walsh and Michael P. Wellman. Modeling supply chain formation in multiagent systems. In *Agent Mediated Electronic Commerce (IJCAI-99 Workshop)*, pages 94–101, 1999.