COORDINATING AGENTS
An Analysis of Coordination in Supply-chain Management Tasks

Chetan Yadati, Cees Witteveen and Yingqian Zhang
Dept of Software Technology, Delft University, Mekelweg 4, 2628 CD Delft, The Netherlands

Keywords: Multi-agent planning, Coordination mechanisms, Approximation algorithms.

Abstract: A multi-agent planning problem consists of a set of activities that need to be planned by several agents. Here, plan coordination methods play an important role, since the independently generated plans by different agents can lead to an infeasible joint solution. We study one particular plan coordination approach, called coordination-by-design, which allows each agent to make its own plan completely independent of the others, while guaranteeing the feasibility of the combined plan of all the agents as a joint solution to the multi-agent planning problem. In this paper, we are interested in a class of multi-agent planning problems that arise in supply-chain management applications. Although the coordination problem in general is Σ_2^p-complete, it turns out for this special class, the complexity of coordination checking is polynomial and deciding a minimum coordination set is NP-complete. We develop a polynomial-time approximation algorithm to compute a sufficient coordination set.

1 INTRODUCTION

Multi-agent planning requires a collection of agents to solve a joint problem together. Typically, each of the agents is capable of solving only a (disjoint) part of the problem and would like to choose its problem solving method, solution or plan *independently* of the others. However, due to possible dependencies between the solutions provided, it is not guaranteed that these individual, partial solutions always can be merged into a feasible overall solution. Therefore, we need to provide a *coordination mechanism*. Let us give a simple transportation example to illustrate the need for such a coordination mechanism.

Example 1.1. *We have to transport two packages p_1 and p_2. Package p_1 at location X has to be delivered first to location Y (task t_1), and then to location Z (task t_3); package p_2 at Z has to be delivered first to Y (task t_4) and then to X (task t_2). There are also two truck agents A_1 and A_2. Agent A_1 can access locations X and Y, while agent A_2 can access only Y and Z. To transport the packages, both agents have to construct a move plan. If they would plan completely independently from each other, agent A_1 could come up with a plan to move from Y to X (transporting p_2) and then from X to Y (transporting p_1). Likewise, A_2 could decide first to move from Y to Z (transporting p_1) and then to move from Z to Y (transporting p_2). Combining those two plans, however, gives an infeasible plan: both agents will have to wait for a package to arrive at Y and therefore can not complete their tasks. In this example, a coordination mechanism could require A_1 to plan transporting p_1 before p_2, thereby ensuring the feasibility of a joint plan.* $\square$

Coordination in multi-agent planning can be viewed as an attempt to achieve a conflict-free joint plan given a set of self-interested multi-agent planners. In the multi-agent system community, quite some effort has been done to attack this problem (Durfee, 1999). The *plan merging* approach (Foulser et al., 1992; von Martial, 1992; Tonino et al., 2002), for example, focuses on the techniques that resolve conflicts *after* independently generated plans have been developed by different agents. Here, agents are required to exchange, negotiate, and revise their (partial) individual plans to arrive at a joint solution. Another approach treats coordination and planning as *intertwined* processes (Durfee and Lesser, 1991; Ephrati and Rosenschein, 1993; Lesser et al., 2004), where agents continuously exchange information during their planning in order to achieve a conflict-free solution. In these two coordination approaches,

Yadati C., Witteveen C. and Zhang Y. (2010).
COORDINATING AGENTS - An Analysis of Coordination in Supply-chain Management Tasks.
In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence - Agents*, pages 218-223
DOI: 10.5220/0002734002180223

agents have to be cooperative: they should be willing to exchange their private information with others, and to revise their plans if necessary.

We would like to address the problem from the *coordination-by-design* perspective, that is, how to provide just sufficient coordination constraints on agents *before* the agents start to solve their own sub-problem, such that *(i)* a joint solution is always coordinated, yet *(ii)* each agent can construct its plan independently from others. One advantage of such a pre-planning coordination approach is that it can be applied without relying on the assumption of cooperativeness of the agents. The general results obtained in this approach can be summarized as follows (Steenhuisen et al., 2008): Checking whether or not agents can plan independently without coordinating them is a co-NP-complete problem. Secondly, finding a *minimum* number of constraints such that coordination is achieved is Σ_2^p-complete. Later, the authors presented a polynomial algorithm, called the Depth-Partitioning Algorithm, which can be used to obtain a sufficient (but not necessarily minimum) set of coordination constraints. However, it only has been shown that this algorithm provides rather good results for logistic planning problems (as used in the AIPS-planning competition). Furthermore (ter Mors, 2004) identified two special cases where the coordination problem becomes computationally easier: the case where an agent has at most *one* single task, and the cases where all tasks with incoming or outgoing inter-agent arcs are *totally ordered* amongst themselves. However, these two cases are very restricted and hardly represent any real-world problems.

Built upon the previous work on the coordination-by-design approach, our main contributions in this paper are as follows:

Firstly, we identify a special class of interesting multi-agent planning problems that is well-motivated by *supply chain management* applications and we demonstrate that these problems form a class of coordination instances with a particular structure, called *intra-free* instances (Section 3).
Secondly, we show (in Section 4) that for intra-free coordination instances can be easily decided whether they are coordinated or not, using a so-called *agent dependency graph* and it can be showed that the coordination problem becomes NP-complete, which means the complexity can be lowered one step in the polynomial hierarchy.
Thirdly, we propose a new polynomial approximation algorithm for obtaining sufficient, but not necessarily minimum, coordination sets for the intra-free instances (Section 4.1).

Before presenting our main contributions, in the next section we introduce a formal model of the multi-agent coordination problem.

2 PRELIMINARIES

To formalize the multi-agent coordination problem, we assume that there is complex task $\mathcal{T}$ that has to be solved by a set $A = \{A_1, \ldots, A_n\}$ of agents. Such a complex task is a partial order $\mathcal{T} =< T, \prec >$ consisting of a finite set $T = \{t_1, \ldots, t_m\}$ of primitive tasks t_i and a precedence relation $\prec$ on T, where $t_i \prec t_j$ indicates that t_i must be completed before t_j can start[1]. Each task $t \in T$ is assumed to be performed by a single agent and we also assume that each agent A_i has already been assigned to a disjoint subset T_i of T. Hence, the set T can be represented by a partitioning $\{T_i\}_{i=1}^n$ of T. Note that each block T_i is partially ordered by the relation $\prec_i$ obtained by restricting $\prec$ to T_i.

We will often represent a coordination instance $\mathcal{T} = \langle \{T_i\}_{i=1}^n, \prec \rangle$ by an acyclic task graph $G_T = \langle V_T, E_\prec \rangle$, having T as it set of nodes and $(t, t') \in E_\prec$ iff $t \prec t'$. For every T_i, the associated subgraph $G_{T_i} = \langle V_{T_i}, E_{\prec_i} \rangle$ of G_T is defined analogously.

Example 2.1. *Consider Example 1.1 discussed in the previous section. The set of tasks $T = \{t_1, t_2, t_3, t_4\}$ is (partially) ordered by $t_1 \prec t_3$;[2] similarly, $t_4 \prec t_2$. As the agents A_1 and A_2 can only access a part of the infrastructure, the task allocation is as follows: $T_1 = \{t_1, t_2\}$ is assigned to agent A_1, $T_2 = \{t_3, t_4\}$ assigned to agent A_2 and $T = \{T_i\}_{i=1}^2$. Notice that there are no* intra-agent *precedence constraints, i.e., $\prec_1 = \prec_2 = \emptyset$.* □

Each individual agent A_i receives a partially ordered set of tasks $\langle T_i, \prec_i \rangle$ and has to come up with a *plan* to complete its set of tasks T_i respecting the set of constraints $\prec_i$. We don't need to know the exact details of such plans, only the consequences w.r.t. the *ordering* of the tasks in T_i are important. Therefore, we assume a plan of agent A_i to be represented by an arbitrary partial order $\prec_i^*$ *refining* $\prec_i$, i.e., a relation that extends $\prec_i$. A *joint plan* for $\mathcal{T}$ then is a structure $\langle \{T_i\}_{i=1}^n, \prec \cup \prec_1^* \cup \ldots \cup \prec_n^* \rangle$. Clearly, such a joint plan is infeasible if the relation $\prec^* = \prec \cup \prec_1^*$

[1] To avoid trivial instances, we do assume that every task t participates in at least some precedence constraint. If a task set T would contain some isolated tasks, they can be simply neglected.

[2] Since package p_1 has to be first delivered from X to location Y, and then it can be delivered to the final location Z.

$\cup \ldots \cup \prec_n^*$ is not acyclic.[3]

Example 2.2. *Continuing Example 2.1, the agents A_1 and A_2 receive the subtask $\langle T_1 = \{t_1, t_2\}, \prec_1 = \emptyset \rangle$ and $\langle T_2 = \{t_3, t_4\}, \prec_2 = \emptyset \rangle$, respectively. Suppose they construct the following plans independently from each other: A_1 comes up with a refinement $\prec_1^* = \{t_2 \prec t_1\}$, while A_2's refinement is $\prec_2^* = \{t_3 \prec t_4\}$. These two refinements $\prec_1^*$ and $\prec_2^*$, together with the precedence relation $\prec = \{t_1 \prec t_3, t_4 \prec t_2\}$, create a cycle: $t_3 \prec^* t_4 \prec^* t_2 \prec^* t_1 \prec^* t_3$. Therefore, the joint plan is infeasible.* □

In general, a joint plan can become infeasible since the individual constraints $\prec_i$ on the tasks T_i given to the agents do not take into account the *inter-agent* constraints $t \prec t'$ where t and t' belong to different agents. Therefore, we have to come up with *additional constraints* to ensure that whatever partially ordered extensions of the individual precedence relations $\prec_i$ are chosen, their combination together with the inter-agent constraints always constitutes a partial order, i.e., a feasible plan for the complex task $\mathcal{T}$.

Example 2.3. *Continuing Example 2.2, let us give an additional constraint $t_1 \prec t_2$ to agent A_1. Notice that now the subproblems of the two agents become: $\langle T_1 = \{t_1, t_2\}, \prec_1 = \{t_1 \prec t_2\} \rangle$ and $\langle T_2 = \{t_3, t_4\}, \prec_2 = \emptyset \rangle$. In this case, any feasible refinement of agent A_2 on $\prec_2$ will always constitute a partial order (feasible solution) for T.* □

So, we state the *coordination checking* problem as follows: Given a coordination instance $\langle \{T_i\}_{i=1}^n, \prec \rangle$ check whether *every possible* combination of partially ordered refinements $\langle T_i, \prec_i^* \rangle$ results in a joint plan that is feasible, i.e., whether $\prec^* = \prec \cup \prec_1^* \cup \ldots \cup \prec_n^*$ is acyclic. More in general, the *coordination* problem is to find a *minimum* set of additional precedence constraints such that feasibility of the joint plan is always guaranteed. As mentioned before, these easy-to-state pre-planning problems, however, turn out to be computationally very difficult: Checking whether a coordination instance $\mathcal{T}$ is already coordinated is co-NP-complete and the problem to find a minimum extension of $\prec$ that guarantees a coordinated solution is Σ_2^p-complete.

These results, however, do hold for the general case. While special cases like constraining the number of agents or the number of tasks already have been investigated (ter Mors, 2004), we would like to investigate *structurally restricted* sub cases. Hence, in the next section, we look into some special applications of the coordination problem, such as *supply chain management*, showing that in these applications it is often sufficient to consider instances of the coordination problem where no precedence relations exist between tasks given to the same agent. We call such instances *intra-free coordination* instances.

[3]A cycle in a plan would indicate that the plan can't be executed: completion of task t would need the completion of another task t' *and vice-versa*.

3 SUPPLY CHAIN NETWORKS

Supply chain management is the management of material and information flows both in and between enterprises (Kazemi et al., 2009). Figure 1 depicts a simple supply chain network, where five different enterprises are involved: a product manufacturer, a cross dock, a raw material supplier and two retailers. The cross dock is an enterprise which does not produce anything, but is simply involved in distribution of products and raw materials. The flow of goods between the enterprises is indicated by the directed arrows. Coordination in a supply chain network focusses on coordinating the activities and plans of the individual enterprises involved in the supply chain. Note here that we focus only on the topological task structure and ignore all other factors such as costs, time points, inventory levels.

The supply chain management problem has been studied extensively in operations research (see (Chen and Decker, 2004)), and some in the agent community (Caridi and Sianesi, 2000; Kazemi et al., 2009). Most research in this field, however, assumes that each of the enterprises is either cooperative or owned by a single enterprise. Hence a *centralised* optimization solution can be used to solve the problem. However, situations commonly exist where enterprises are selfish and non-cooperative. In such cases, our pre-planning coordination approach provides an attractive alternative for managing and coordinating the plans of selfish enterprises in a supply chain network. The entire set of activities in the supply chain network can be represented as a partial order $(T, \prec)$, where T represents the set of tasks (the nodes) and $\prec$ represents the set of precedence relations (represented by directed arcs). The enterprises are represented by agents each having a set of tasks. The coordination problem now is to find a minimum set of precedence constraints such that each enterprise can plan its own activities independently from the others, while the joint plan is guaranteed to be feasible. As an example, a solution to the particular instance in Figure 1 would be adding a set of two coordination constraints: *Send R_1 before Send P_1, P_2*; and *Send R_2 before Send P_1, P_2*. We can easily check that these two coordination constraints ensure that no cycles or deadlocks can be formed by any plans generated individually by each enterprise.

Notice that there is an interesting feature in the supply chain management problem as a coordination problem: usually, an enterprise has *no internal restrictions* on its own activities or tasks.[4] To the enterprise, one advantage of having no internal (intra-agent) precedence constraints is that the enterprise itself can enjoy the maximal freedom when planning its own activities. As mentioned before, we call such an instance *intra-free*. Notice that in intra-free instances, each subgraph G_{T_i} associated with the set of tasks allocated to agent A_i is the *empty graph on* V_{T_i}, i.e., there are no precedence relations between tasks assigned to the same agent.

We show that for intra-free instances the coordination problems are computationally easier.

4 ANALYZING INTRA-FREE COORDINATION INSTANCES

As we remarked before, in general, the coordination (checking) problem is a computationally intractable problem. Supply-chain like coordination problems, however, can be modelled by intra-free coordination instances and might be easier to solve. To start our analysis, we define a special subclass of intra-free coordination instances. We call it *strictly intra-free*, which will be of special importance in proving properties of intra-free coordination instances:

Definition 4.1. *An intra-free coordination instance* $\langle\{T_i\}_{i=1}^n, \prec\rangle$ *is called* strictly intra-free *if, for every i, the subset of (isolated) nodes* V_{T_i} *of the graph* G_T *can be partitioned in two disjoint subsets* $In(V_{T_i})$ *and* $Out(V_{T_i})$ *such that* $In(V_{T_i})$ *consists of nodes v with outdegree 0 (*$out(v) = 0$*), also called sink nodes, and* $Out(V_{T_i})$ *consists of nodes v with indegree 0 (*$in(v) = 0$*), also called source nodes.*

That is, in strictly intra-free coordination instances, we *exclude* tasks t where both $out(t) > 0$ as well as $in(t) > 0$. First, we will prove some results for strictly intra-free coordination instances. Then we will show that intra-free instances in general can be easily reduced to strictly intra-free coordination instances thereby proving that the properties holding for strictly infra-free instances also hold for this larger class of instances.

[4]Note here that based on the policy of the manufacturer, it could either choose to manufacture first or send products first based on his inventory levels and its inventory policy.

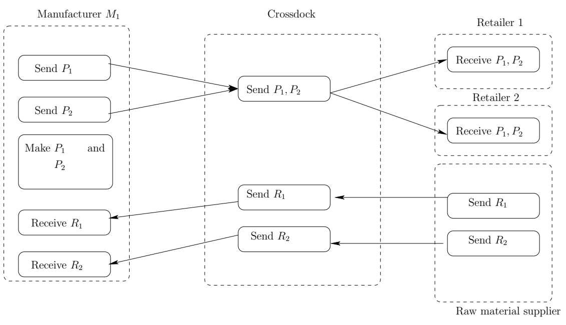
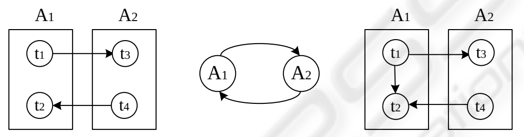


Figure 1: Example supply chain management scenario.



(a) Task graph (b) Agent dependency graph (c) A coordinated instance

Figure 2: The task graphs and their corresponding agent dependency graph.

Given a partitioning $\{V_{T_i}\}_{i=1}^n$ of the set of nodes V_T corresponding to the partitioning $\{T_i\}_{i=1}^n$, the *coordination checking* problem comes down to deciding whether for all possible *acyclic* extensions $G^*_{T_i} = \langle V_{T_i}, E_{\prec^*_i}\rangle$ of the subgraphs $G_{T_i} = \langle V_{T_i}, \emptyset\rangle$ the resulting extension $G^*_T = \langle V_T, E_\prec \cup E_{\prec^*_1} \cup \ldots \cup E_{\prec^*_n}\rangle$ of G_T is still acyclic. While the problem to decide whether a given graph is acyclic is in P, the coordination problem is co-NP complete for general coordination instances.

The rather surprising result we prove now is that for (strictly) intra-free coordination instances, this coordination checking problem can be reduced to checking the acyclicity of an even (smaller) graph and hence, this problem is in P as well.

Here, we need to introduce the *agent dependency* graph G_A derived from G_T and the partitioning $\{T_i\}_{i=1}^n$ of T:

Definition 4.2 (Agent Dependency Graph)**.** *Given a task graph* G_T *and a partitioning* $\{T_i\}_{i=1}^n$*, the agent dependency graph derived from* G_T *is a graph* $G_A = (V_A, E_A)$*, where* $V_A = \{v_{A_i} : A_i \in A\}$ *is the set of nodes corresponding to agents, and* $E_A = \{(v_{A_i}, v_{A_j}) : \exists t \in T_i, \exists t' \in T_j, [t \prec t']\}$ *is the dependency relation between them.*

Example 4.1. *Consider the package delivery example (Example 2.1) discussed in Section 2, where four delivery tasks are allocated to two agents. Its task graph is illustrated in Figure 2(a). Figure 2(b) shows its corresponding agent dependency graph.* □

There is a simple connection between coordination checking for $\mathcal{T}$ and *acyclicity* testing of G_A:[5]

Proposition 4.1. *Let $\mathcal{T} = \langle \{T_i\}_{i=1}^n, \prec \rangle$ be a coordination instance and G_A its associated agent dependency graph. Then G_A is acyclic implies that $\mathcal{T}$ is coordinated, i.e., $\mathcal{T}$ is a yes-instance of the coordination checking problem.*

In general, the converse of this proposition is not true: even if G_A is cyclic, we might have a yes-instance of $\mathcal{T}$.

Example 4.2. *Consider the task graph shown in Figure 2(c). Clearly, this instance is coordinated. The agent dependency graph, as shown in 2(b), however, contains a cycle: each of the agents is dependent upon the other as t_3 is dependent upon t_1 while t_2 is dependent upon t_4.* □

However, if $\mathcal{T} = \langle \{T_i\}_{i=1}^n, \prec \rangle$ is a strictly intra-free coordination instance we can actually show that the converse holds, too:

Theorem 4.2. *Let $\mathcal{T} = \langle \{T_i\}_{i=1}^n, \prec \rangle$ be a strictly intra-free coordination instance and G_A its agent dependency graph. Then G_A is acyclic iff $\mathcal{T}$ is a yes-instance of the coordination checking problem.*

Note that these results hold for strictly intra-free coordination instances. It is, however, very easy to generalize them to intra-free coordination instances by polynomially reducing intra-free instances to strictly intra-free instances: we apply the following *task-splitting* procedure to tasks that violate the strict intra-free property.

Given an arbitrary intra-free coordination instance $\mathcal{T} = \langle \{T_i\}_{i=1}^n, \prec \rangle$, for every i and every $t \in T_i$ such that $in(t), out(t) > 0$, do the following:

1. split t in two tasks t_1 and t_2
2. add all precedence constraints $t' \prec t_1$ such that $t' \prec t$ and add $t_2 \prec t'$ whenever $t \prec t'$;
3. remove t and all precedence constraints it is mentioned in.

Clearly, the result is an equivalent strictly intra-free coordination instance.

Example 4.3. *A simple example in Figure 3 illustrates how to convert an intra-free instance to a strictly intra-free instance. The task t of agent A_2 has been split into two tasks t_1 and t_2. We can simply verify that when the (converted) strictly intra-free instance is coordinated, the (original) intra-free instance must be also coordinated.* □

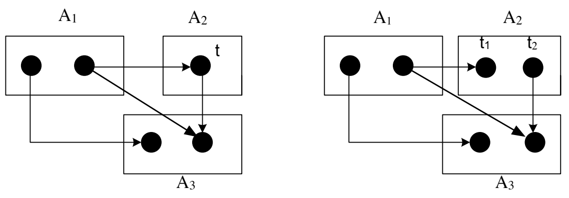


Figure 3: An intra-free coordination instance (first graph), converted to a *strictly* intra-free instance (second graph).

Therefore, we obtain the following observation.

Observation 4.3. *An intra-free coordination instance $\mathcal{T}$ is coordinated whenever its strictly intra-free variant $\mathcal{T}'$ is coordinated.*

Based on these results we can now perform coordination checking in polynomial time: All we need to do is to find out if the agent dependency graph of the given intra-free instance is acyclic. This problem is known to be solvable in polynomial time (Cormen et al., 1990).

Given the fact that coordination checking for intra-free instances is easy, one might hope that finding a *minimum coordination set* is also tractable in the case of intra-free coordination instances. This turns out to be wrong. Finding such a minimum set of coordination constraints is NP-hard as we have the following result:

Proposition 4.4. *The NP-complete* DIRECTED FEEDBACK VERTEX SET *problem is polynomially reducible to the decision version of the minimum coordination problem for intra-free coordination instances.*

Remark. In fact, it can be shown that the problem whether K given coordination arcs are sufficient to make an intra-free coordination instance coordinated can be decided in polynomial time. This shows that the problem is in NP, hence we can conclude that the coordination problem for intra-free instances is NP-complete.

4.1 An Approximation Algorithm

We can use the agent dependency graph and the results obtained in the previous sections to obtain an approximation algorithm for constructing coordination sets in an easy way. The idea is, given an intra-free coordination instance, first to find an (approximate) minimum feedback vertex set V' for the associated

[5]As space is limited, we have decided to omit all proofs of the results stated in the paper. A simple email to one of the authors is sufficient to obtain an extended version of the paper containing all the proofs.

agent dependency graph and then to use this feedback vertex set V' to create a sufficient set of coordination arcs by adequately blocking connections in the corresponding task sets of agents occurring in V'. This blocking can be realized by adding arcs from all source to all sink tasks occurring in the task sets.

So the FVS-algorithm essentially consists of the following steps:

1. Given a coordination instance $\langle \{T_i\}_{i=1}^n, \prec \rangle$, create the associated agent dependency graph $G_A = \langle V_A, E_A \rangle$;
2. Find an approximation[6] of a minimum directed feedback vertex set $F \subset V_A$ of G_A. ;
3. For every $v_{A_i} \in F$, for every $t \in Out(T_i)$ (source task) and for every $t' \in In(T_i)$ (sink task), add the coordination constraint $t \prec_i^* t'$ thereby blocking any cycle passing through tasks belonging to A_i.

It can be easily shown that in most cases this algorithm will outperform the existing depth partitioning algorithm (Steenhuisen et al., 2008)[7].

5 CONCLUSIONS AND FUTURE WORK

This paper identified a new subclass of coordination problems called *intra-free* coordination instances which have a relevance to model production and distribution planning problems in particular and supply chain planning in general. We have shown that the (decision version of the) plan coordination problem for this subproblem is NP-complete, instead of the Σ_2^p complexity of the general coordination problem. We also proposed a specialized approximation technique to design a 'good' set of coordination constraints in polynomial time.

As part of our future research, first of all we are investigating better heuristics for producing subset minimal coordination sets e.g., as we have discussed in the previous section, and we would like to refine the theoretical performance comparison analysis based upon them.

REFERENCES

Caridi, M. and Sianesi, A. (2000). Multi-agent systems in production planning and control: An application to the scheduling of mixed-model assembly lines. *International Journal of Production Economics*, 68(1):29 – 42.

Chen, W. and Decker, K. (2004). Managing multi-agent coordination, planning, and scheduling. In *Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004. Proceedings of the Third International Joint Conference on*, pages 1360–1361.

Cormen, T. T., Leiserson, C. E., and Rivest, R. L. (1990). *Introduction to algorithms*. MIT Press, Cambridge, MA, USA.

Durfee, E. H. (1999). Distributed problem solving and planning. In Weiß, G., editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 121–164. MIT Press, Cambridge, MA, USA.

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

Ephrati, E. and Rosenschein, J. S. (1993). 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.

Foulser, D., Li, M., and Yang, Q. (1992). Theory and algorithms for plan merging. *Artificial Intelligence Journal*, 57(2–3):143–182.

Kazemi, A., Zarandi, M. F., and Husseini, S. M. (2009). A multi-agent system to solve the productiondistribution planning problem for a supply chain: a genetic algorithm approach. *International Journal of Advanced Manufacturing Technilogy*, 44:180–193.

Lesser, V., Decker, K., Wagner, T., Carver, N., Garvey, A., Horling, B., Neiman, D., Podorozhny, R., NagendraPrasad, M., Raja, A., Vincent, R., Xuan, P., and Zhang, X. (2004). Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. *Autonomous Agents and Multi-Agent Systems*, 9(1):87–143.

Steenhuisen, J. R., Witteveen, C., and Zhang, Y. (2008). Plan-coordination mechanisms and the price of autonomy. In Sadri, F. and Satoh, K., editors, *Computational Logic in Multi-Agent Systems*, volume 5056 of *Lecture Notes in Artificial Intelligence*, pages 1–21. Springer-Verlag.

ter Mors, A. (2004). Coordinating autonomous planning agents. Master's thesis, TU Delft, Delft, The Netherlands.

Tonino, H., Bos, A., de Weerdt, M., and Witteveen, C. (2002). Plan coordination by revision in collective agent based systems. *Artificial Intelligence*, 142:121–145.

von Martial, F. (1992). *Coordinating Plans of Autonomous Agents*, volume 610 of *Lecture Notes on Artificial Intelligence*. Springer Verlag, Berlin.

[6]We can use any known approximation algorithm for this problem.

[7]Due to space limitations, we omit the performance analysis in this paper.