Multi-Agent Off-line Coordination: Structure and Complexity

Carmel Domshlak and Yefim Dinitz

Department of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, Israel
{dcarmel,dinitz}@cs.bgu.ac.il

Abstract. Coordination between processing entities is one of the most widely studied areas in multi-agent planning research. Recently, efforts have been made to understand the formal computational issues of this important area. In this paper, we make a step toward this direction, and analyze a restricted class of coordination problems for dependent agents with independent goals acting in the same environment. We assume that a state-transition description of each agent is given, and that preconditioning an agent's transitions by the states of other agents is the only considered kind of inter-agent dependence. Off-line coordination between the agents is considered. We analyze some structural properties of these problems, and investigate the relationship between these properties and the complexity of coordination in this domain. We show that our general problem is provably intractable, but some significant subclasses are in NP and even polynomial.

1 Introduction

Coordination between processing entities is one of the most widely studied areas in multi-agent planning research. Recently, efforts have been made to understand the formal computational issues in this important area. In this paper, we make a additional step toward this direction. We describe a restricted class of coordination problems for agents with independent goals acting in the same environment, that is the strategy of each agent should be planned while taking into account strategies of other agents as planning constraints. In these problems, (i) the set of feasible plans for each agent is defined by a state-transition graph, and (ii) preconditioning an agent's *transitions* by the *states* of other agents is the only considered kind of inter-agent dependence. For this problem class, off-line coordination is considered: structural and computational properties are investigated, and both tractable and intractable subclasses are presented. Although the examined class of problems can be seen as significantly restricted, its place in multi-agent systems was already discussed in AI literature, e.g. [6, 14, 22]. We believe that the formal model we suggest for the presented problem class can serve as a basis for further extensions toward a representation of richer multi-agent worlds yet preserving convenience for computational analysis.

In the area of multi-agent coordination, we identify two main research directions: *multi-agent collaboration* and *synthesizing multi-agent plans*. In multi-agent collaboration, goal achievement is a team activity, i.e., multiple agents collaborate towards the achievement of an overarching goal. During the last decade, various theories for multi-agent collaboration, e.g. [9, 18, 20], and several generic approaches for corresponding planning, e.g. [11, 25], were proposed; for an overview of this area see [17]. In addition, a few formal computational results concerning the coalition formation are found for both benevolent [24] and autonomous [12] agents.

Synthesizing multi-agent plans addresses synchronization between agents that work on independent goals in the same environment. In [16], centralized merging of preconstructed individual agent plans is presented, and it is based on introducing synchronization actions. In [15], a heuristic approach that exploits decomposed search space is introduced. Whereas in [15, 16] methods for merging plans at the primitive level are described, in [8] a strategy for plan merge on different levels of hierarchical plans is presented. In [1, 2], a generic approach for distributed, incremental merging of individual plans is suggested: the agents are considered sequentially, and the plan for the currently processed agent is synchronized with the plans chosen for the previously processed agents. Besides, a few more algorithmic approaches to an application-specific plan synchronization are proposed in [7, 23, 26]. Note that all these approaches for inter-agent synchronization assume that the personal plans are already constructed, and that the merging operation is considered only on this fixed set of individual plans (possibly, certain local modifications of the plans may be allowed). This essentially restricts not only the quality, but even the ability to find a coordinated plan.

A wider formal model for multi-agent systems that concerns both determining personal agent plans and their synchronization, is suggested in [22]. With respect to this model, the cooperative goal achievement problem is defined. It can be roughly stated as follows: Given a set of benevolent agents in the same environment, each one with its own, independent goal, does there exist a satisfying system of coordinated personal plans? In [22], this problem is shown to be PSPACE-complete.

In this paper, we concentrate on a particular class of the cooperative goal achievement problem. In this problem class, which we denote to as STS (state-transition support), an action (transition) of an agent can be constrained by the states of other agents. Following [6, 22], each agent is assumed to be representable as a finite automaton. As in [8, 15, 16], the coordination process is assumed to be off-line and centralized. Note that we are not dealing here with many complementary issues of multi-agent systems such as determining agent behavior, collaboration establishing, decision process distributing, etc.

We suggest to represent an STS problem domain using a graph language. The two interrelated graphical structures of our model are as follows:

1. The *strategy* (di)*graph*: the state-transition graph describing behavior of an agent.
2. The *dependence* (di)*graph*: description of the dependence between the agents.

The strategy graph compactly captures all alternative personal plans for the corresponding agent: its strategies correspond to the *paths* from the source node to the target node in its strategy graph. Transition of an agent along an edge is conditioned by the *states* of the agents on which it depends, i.e., their locations at certain nodes in their strategy graphs. The reader can find some relation between our strategy graphs and the interactive automata of [21]. The main differences are that we (i) address preconditioning, not interaction between agents, and (ii) consider off-line planning.

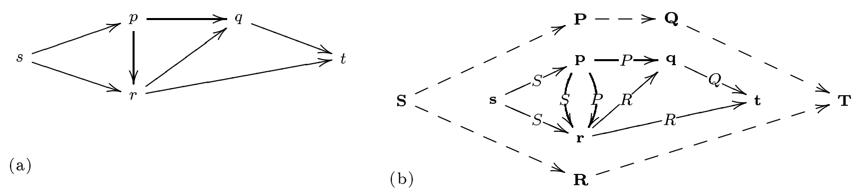

Fig. 1. Examples of strategy graphs for: (a) single agent, (b) two-agent group: transitions of X depend on states of Y.

The nodes of the dependence graph correspond to the agents. An edge (p, q) appears in the dependence graph if some action of the agent q depends on the state of the agent p. We define subclasses of STS, mainly with respect to structural properties of the dependence graph, and analyze their complexity. The complexity is measured in terms of the total size of the strategy graphs and the size of the dependence graph (which is actually the number of agents). We show that our general problem is provably intractable, but some significant subclasses are in NP and even polynomial. As far as we know, such an analysis of the relation between the structural and the computational properties of a multi-agent coordination problem was not done in previous research. Note that this paper presents only first results of our ongoing research, thus, in particular, we do not discuss relative significance of the presented problem classes, and we do not provide wide final conclusions.

The rest of the paper is organized as follows: Section 2 presents and discusses our model. Section 3 is devoted to the basic case of two agents. Section 4 introduces a classification of STS's subclasses, and presents additional complexity results. For the proofs and some of the detailed algorithms we refer to the technical report [13]. Concluding remarks are given in Section 5, together with some discussion of related issues and future work.

2 Graphical Model for STS

In this section we define two graphical structures of the STS problem domain. The *strategy graph* G^A captures the alternative personal strategies of an agent A. The agent's states are represented by the nodes of G^A. Each (directed) edge represents a possible transition between the two corresponding states. Multiple transitions between two states are possible. If a transition is conditioned, then the corresponding edge has a *label* which describes the condition; when a transition can be done under several alternative conditions, this is modeled by multiple edges, each labeled by a single condition. There are two unique nodes in G^A, the source node and the target node, which represent the initial and goal states of A, respectively. The possible agent's strategies are represented by the paths from the source node to the target node. For an illustration see Fig. 1(a), where the possible strategies are: $spqt$, $sprt$, $sprqt$, srt, $srqt$.

The size of this structure is linear in the length of the problem description. Observe that in general, the number of different, potentially applicable, strategies may be exponential in the size of the problem even for an acyclic structure; for a general structure, this number can, formally, be infinite.

Consider a pair of agents X and Y so that X depends on Y in the following sense: each transition of X is preconditioned by a certain state of Y. In graph terms, each edge in G^X is labeled by the name of a node in G^Y. An example is given in Fig. 1(b), where the inner graph models X and the outer one models Y. Note that there are two transitions of X from the state p to the state r which are preconditioned by different states of Y. The following pair of strategies for X and Y is coordinated: The agent X begins from state s and Y begins from state S. Subsequently, (i) X moves to p, (ii) Y moves to P, while X is in p, (iii) X moves to q, (iv) Y moves to Q, (v) X moves to t, and finally, (iv) Y moves to T. We denote this coordinated multi-agent strategy by the sequence $\langle(s,p)(S,P)(p,q)(P,Q)(q,t)(Q,T)\rangle$. The other possible coordinated plans for X and Y are $\langle(s,p)(p,r)(S,R)(r,t)(R,T)\rangle$ and $\langle(s,r)(S,R)(r,t)(R,T)\rangle$. No other coordinated pair of strategies exists. For example, if the agents begin with $\langle(s,p)(S,P)(p,r)\rangle$, then X cannot leave r, since it is impossible for Y to reach R after being in P.

In general, there are several agents $A_1, A_2, \ldots, A_n$ with strategy graphs $G^i = G^{A_i}$, $1 \leq i \leq n$. We say that an agent A_i depends on an agent A_j if transitions of A_i have states of A_j as preconditions. Note that a few agents may depend on the same agent, and an agent may depend on a few other agents. Hence, a transition of A_i is preconditioned by states of agents that A_i depends on; in general, a transition can depend on a part of them.

The *dependence graph* $\mathcal{G}$ is a digraph whose nodes are A_i, $1 \leq i \leq n$, and there is an edge from A_j to A_i if agent A_i depends on agent A_j. In what following, we identify nodes of $\mathcal{G}$ with the corresponding agents. Likewise, we identify the nodes and edges of G^i with the states and transitions of A_i, respectively. An edge of G^i has a compound label, each component of which is a state of an immediate predecessor of A_i in $\mathcal{G}$ (i.e., a node in the strategy graph of this predecessor).

Both the strategy graph and the dependence graph can be constructed straightforwardly, given an STS problem. Likewise, the structural properties of the dependence graph, which are investigated in this paper, can be verified in low polynomial time in the size of the graph.

In this paper, *each* transition of an agent is preconditioned by the states of *all* agents that this agent depends on, if the opposite is not declared explicitly. We consider only connected dependence graphs. The reason is that otherwise the set of agents can be divided into disjoint subsets that can be considered independently.

A natural motivation for a pair of agents X and Y as above is that X fulfills some mission, while Y supports it. In practice, the mission of X may be a movement towards a target, while Y either watches X from certain observation points, or feeds it from certain warehouses, or protects it from some fortified positions. Such a support is usual in military and is wide-spread in various other activities. For example, a worker X assembles some compound product, while a mobile crane Y moves the (half-assembled) product from one working place of X to another. In general, when there are several agents, each agent is supported by a few agents, and in turn, supports a few other agents.

Having read this far the reader may argue that it is possible that we never know an agent's behavior in complete detail, and even if we do, the number of states that represent an agent's behavior may be very large. In this case, exploiting and even constructing strategy graphs seems to be infeasible.

Indeed, it is not realistic to present detailed models of many real-life systems as finite state machines. However, as it is argued in [22], the representation an agent uses need not present the world in sufficient detail to require more expressive description language. Intuitively, it is possible to distinguish between the physical state of agents and their computational state. It is the computational state that we have difficulty in modeling by finite-state machines. However, the computational state of an agent is not accessible to the other agents. On the other hand, a finite-state description can serve well as a representation of the physical state. A wider discussion of these and other issues of finite-automata-based knowledge representation and reasoning can be found in [22]. Likewise, number of finite-automata-based architectures are discussed in the AI literature, e.g., see [6].

In case that the number of parameters describing the physical state (and thus the size of the state space) is still large, it is often possible to take advantage of the structure of an agent in order to derive a concise description of its state. In particular, an agent may be viewed as being composed of a number of largely independent components, each with its own state. Each such component (such as robot's hand, motor, wheel, etc.) can be represented by an agent, whose physical state is relatively simple, since it describes a particular feature of a complex system. In this case, the strategy graphs of the agents are expected to be of reasonable size.

3 The Basic Case of Two Agents

In the rest of the paper, we study the complexity of various multi-agent coordination problems based on the model suggested above. In this section we demonstrate our techniques on solving the basic case of agent X depending on agent Y. Given the strategy graphs of X and Y, we build a new graph describing X restricted by Y. As a result, our problem is reduced to the known problem of finding a source-to-target path in this new graph. Such a reduction becomes possible because of the locality of the original restrictions.

We describe a solution of our problem as an interleaved sequence of transitions of agents X and Y, as was shown in Section 2. Let us analyze the structure of a general coordinated plan Π for these agents. As mentioned above, private strategies for X and Y are source-to-target paths in G^X and G^Y respectively, and we denote them by σ^X and σ^Y. Let us divide σ^X into intervals σ_1^X, σ_2^X, ..., σ_m^X of constancy of labels: all edges (transitions) in σ_i^X are labeled (preconditioned) by the same node (state) N_i of Y. The Y's path σ^Y is also divided by the nodes N_i into some parts. The considered plan is as follows: (1) If $N_1 \neq s^Y$, then agent Y moves from s^Y to state N_1 along the initial part of σ^Y; (2) Agent X moves along σ_1^X to its end; (3) Y moves from N_1 to N_2 along the next part of σ^Y; (4) X moves along σ_2^X, etc. Finally, if $N_m \neq t^Y$, then agent Y moves from N_m to t^Y along the final part of σ^Y. For illustration see Figure 2.

Let see what "degrees of freedom" Π has. For simplicity of notation, we denote s^Y by N_0 and t^Y by N_{m+1}. Observe that the replacement of part of σ^Y between N_i to N_{i+1}, $0 \leq i \leq m$, by any other path from N_i to N_{i+1} retains the plan feasible. That is,

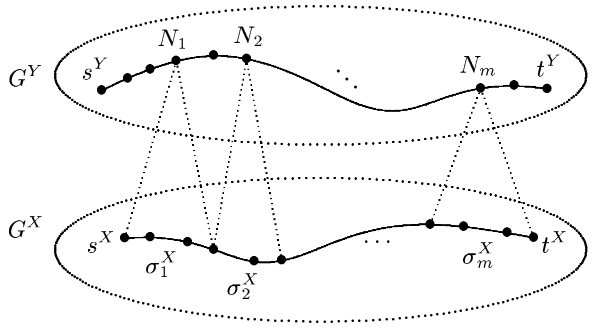

Fig. 2. Relation between personal strategies of X and Y

the only requirement for feasibility of σ^X is the *existence* of such a path for each label change along σ^X. Moreover, replacement of any σ_i^X by another path between its ends such that all its edges are labeled by N_i retains the plan feasible.

Lets now consider building such a coordinated plan Π. We take as a central issue finding an appropriate path σ^X. As mentioned above, the only critical points in its building are the moments of label change along σ^X: if the label changes from N_i to N_{i+1}, then a "connecting" path from N_i to N_{i+1} in G^Y must exist. In order to include into the same framework the cases in which the label of the first edge is not s^Y and the label of the last edge is not t^Y, we extend G^X as follows. We add to G^X a dummy source $\tilde{s}^X$, with an edge $(\tilde{s}^X, s^X)$ labeled s^Y, and a dummy target $\tilde{t}^X$, with an edge $(t^X, \tilde{t}^X)$ labeled t^Y, and consider path $\tilde{\sigma}^X$ from $\tilde{s}^X$ to $\tilde{t}^X$ in this extended graph $\tilde{G}^X$. It is easy to see that existence of connecting paths in G^Y for all critical points in $\tilde{\sigma}^X$ is necessary and sufficient for the feasibility σ^X. Indeed, the twin path σ^Y is the concatenation of all connecting paths, for $0 \leq i \leq m$. Therefore, the only information we need in order to build an appropriate path in $\tilde{G}^X$ is whether two edges may be consequent on a path $\tilde{\sigma}^X$ in the above sense, for all pairs of adjacent edges in $\tilde{G}^X$.

Information on existence of a path between any pair of nodes N' and N'' of G^Y (*reachability* of N'' from N') can be obtained by a preprocessing of G^Y. Appropriate algorithms are widely known and are very fast; the paths themselves, if any, are provided by these algorithms as well, e.g. see [10]. Hence, we can form the following database for all pairs of consequent edges in $\tilde{G}^X$: If an edge e' labeled N' enters some node and an edge e'' labeled N'' leaves the same node, then the pair (e', e'') is marked "permitted" if either $N' = N''$ or N'' is reachable from N' in G^Y. For example see Figure 3, where dotted lines show the permitted edge pairs.

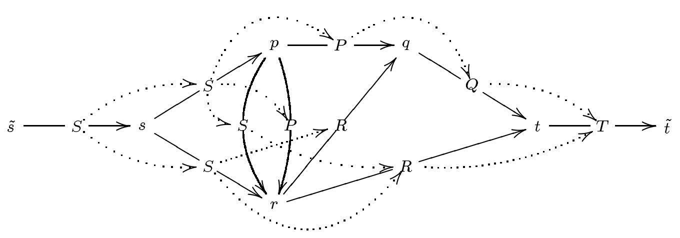

Fig. 3. Illustration of $\tilde{G}^X$ with the corresponding $G^{X|Y}$.

In fact, the suggested construction provides a reduction of the problem of finding a path $\tilde{\sigma}^X$ as defined above to a known problem of finding a path in a graph. Let the new graph $G^{X|Y}$ have the *edges* of $\tilde{G}^X$ as nodes, and let its edges be defined by the permitted pairs of edges in $\tilde{G}^X$.[1] Now, all we need is a path from the source node $(\tilde{s}^X, s^X)$ to the target node $(t^X, \tilde{t}^X)$ in $G^{X|Y}$. Clearly, such a path defines the corresponding path in $\tilde{G}^X$ immediately.

Figure 4 summarizes the algorithm for finding a coordinated plan for agent X depending on ("supported by") agent Y. This algorithm is, evidently, polynomial. The reader can "execute" it by himself on the example given in Figure 3. This example is the same as the one in Section 2, and any of the three feasible coordinated plans listed there can be obtained as the result of such an execution. Thus we have achieved our first result, and in what following, we use the presented approach for analyzing other variants of STS.

1. Extending strategy graph G^X to $\tilde{G}^X$.
2. Preprocessing of strategy graph G^Y: finding a path from any node to any other node (in fact, it is sufficient to do this for the node pairs needed in the next phase only).
3. Constructing permitted-edge graph $G^{X|Y}$.
4. Finding a source-to-target path in $G^{X|Y}$, if any, or reporting on non-existence of a coordinated plan, otherwise.
5. Reconstructing from the path found in phase 4 a path $\tilde{\sigma}^X$ in $\tilde{G}^X$ and its abridged variant σ^X in G^X.
6. Building a path σ^Y in G^Y as the concatenation of paths found at phase 2, for all label changes along $\tilde{\sigma}^X$.
7. Outputting properly interleaved strategies σ^X and σ^Y.

Fig. 4. Algorithm for the basic STS problem.

Theorem 1. *There exists a polynomial time algorithm that, given a pair of agents X, Y, in which X depends on Y, determines existence of a coordinated plan and finds such a plan, if exists.*

This result can be easily generalized to finding an *optimal* coordinated plan. Assume that the cost function is the sum or the product of weights of states and/or transitions used in the plan. Since in step 4 of the algorithm, an arbitrary path can be chosen, let us choose an optimal path; this suffices for global optimality. Any algorithm for finding a cheapest path in a graph can be used, e.g. see [10].

Now let us require, in addition, *simplicity* of Y's strategy, i.e., let us forbid Y to visit the same state twice. A motivation for this can be that each state of Y relates to some consumable resources. It turns out that this seemingly non-essential change in the problem definition, change the problem to be NP-complete. Informally, the reason for this complexity worsening is that the locality of precondition is thus broken. This result is mainly interesting from the theoretical point of view, since it points on a computational sensitivity of our original problem.

[1] Such a construction is a variant of the so called "edge graph" known in graph theory; the addition in our case is the exclusion of non-permitted edges from it.

4 Results for Some Other Subclasses of STS

In this section we discuss various additional complexity results for STS. Problem classes are defined mainly by the structure of their dependence graph. Moreover, both results and methods depend crucially on the number and size of possible agent strategies: if both of them are polynomial in the size of problem domain, then we distinguish such a class as *bounded*. Figure 5 summarizes the achieved results, and each box in the figure denotes an STS's subclass.

The notation is as follows: First, each STS subclass is denoted with respect to the form of the dependence graph: $\mathbb{C}$ stands for *directed chain*, $\mathbb{F}^{\wedge}$ for *directed fork* (graph with exactly one node with a non-zero outdegree), $\mathbb{F}^{\vee}$ for *directed inverse fork* (graph with exactly one node with a non-zero indegree), $\mathbb{P}$ for *polytree* (digraph whose underlying undirected graph is a tree), and $\mathbb{S}$ for *singly-connected DAG* (directed acyclic graph with at most one directed path between any pair of nodes). Second, the subscript denotes the number of agents: k stands for a constant bound, and n indicates no bounds.

By default, we assume that each transition of an agent is preconditioned by the states of *all* agents that this agent depends on. Alternatively, π in a superscript denotes a possibility of *partial* dependence. σ in a superscript denotes the requirement that all strategies chosen are *simple*. Note that cases $\mathbb{C}_2$ and $\mathbb{C}_2^{\sigma}$ have been discussed in Section 3.

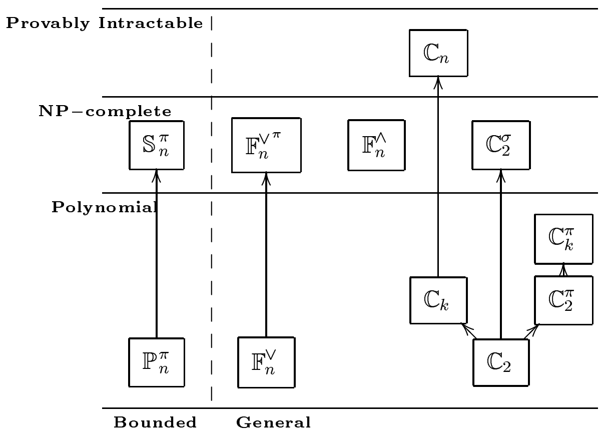

Fig. 5. Complexity results for STS

Coordinating General Agents A natural generalization $\mathbb{C}_n$ of the class $\mathbb{C}_2$ (as in Section 3) occurs when there is a dependence chain: agents A_i, $1 \le i \le n$, with A_i dependent on A_{i-1}, $i \ge 2$. For simplicity of notation, let us abridge A_i to i in superscripts (e.g., G^i instead of G^{A_i}).

Our approach for $\mathbb{C}_n$ is to iterate the analysis and the algorithm for $\mathbb{C}_2$ along the dependence chain. A naive scheme is as follows. For processing pair A_3, A_2, we need a database on *coordinated* reachability in G^2. Observe that our algorithm for $\mathbb{C}_2$ applied to pair A_2, A_1 provides such an information for nodes s^2, t^2; clearly, this can be done for any pair of nodes of G^2. Considering pair A_2, A_1, lets check reachability and store paths, if any, for *all* pairs of nodes in G^2, as required. Now, we are as if sure that if we would determine a coordinated plan for pair A_3, A_2 based on the above reachability relation for A_2, a corresponding strategy for A_1 will be obtained easily. Hence, the situation with A_3, A_2 is similar to that with A_2, A_1, and thus we can iterate up to A_n. Personal coordinated strategies can be consequently reconstructed, from A_{n-1} to A_1.

However, this scheme turns to be inconsistent. The point is that, in fact, the coordinated reachability relation in G^2 is on its *edges*, not on its nodes, and it is captured by the edge graph $G^{2|1}$. Therefore, informally, the right approach is to create an edge graph for G^3 basing on $G^{2|1}$, not on G^2. This idea of recursive edge graph creating can be generalized for dependence chain of arbitrary length. Using this method, we never arrive at a deadend as in the naive scheme, and we can obtain any coordinated plan for this problem. For the detailed algorithm for $\mathbb{C}_n$, together with an illustrating example of its execution and a counterexample for the naive approach we refer to the technical report [13].

Theorem 2. **(I)** *There exists an algorithm for $\mathbb{C}_n$ that determines existence of a coordinated plan and finds such a plan, if exists. This algorithm is polynomial in the total size of strategy graphs and exponential in n; thus $\mathbb{C}_k$ is proved to be polynomial.*
(II) *$\mathbb{C}_n$ has instances with exponentially sized minimal solutions.*

The proof of the exponential lower bound is by the following example. Each agent A_i, $2 \leq i \leq n$, has the strategy graph as follows:

$$\begin{tikzcd} s^i \arrow[r, bend left, "t^{i-1}"] & q^i \arrow[l, bend left, "s^{i-1}"] \arrow[r, bend left, "s^{i-1}"] & t^i \arrow[l, bend left, "t^{i-1}"] \end{tikzcd}$$

The strategy graph for A_1 looks the same, except that its edges are unlabeled. This problem instance has a unique minimal coordinated plan of total length $2^{n+1} - 2$ transitions.

By showing that $\mathbb{C}_n$ is provably intractable, theorem 2 emphasizes our motivation for exploiting various structural restrictions on STS in order to find problem classes which are polynomial, e.g., $\mathbb{C}_k$, or at least belong to NP. In particular, the subclass of $\mathbb{C}_n$ strategy graphs of which are acyclic can be easily shown to belong to NP. However, the question of its exact hardness is still an open question. The same question is open for the decision version of $\mathbb{C}_n$.

Theorem 3 summarizes the complexity results for some other classes of STS (for notations see Section 4).

Theorem 3. **(I)** *There exists a polynomial time algorithm for $\mathbb{F}_n^{\vee}$ that determines existence of a coordinated plan and finds such a plan, if exists.* **(II)** *There exists a polynomial time algorithm for $\mathbb{C}_2^{\pi}$ that determines existence of a coordinated plan and finds such a plan, if exists.* **(III)** *$\mathbb{F}_n^{\wedge}$ is* NP*-complete.* **(IV)** *$\mathbb{F}_n^{\vee^{\pi}}$ is* NP*-complete.*

Proof sketch. *(I)* In $\mathbb{F}_n^{\vee}$, a single agent A_1 depends on a group of independent supporting agents $A_2, \ldots, A_n$. The algorithm is similar to that for $\mathbb{C}_2$. The idea is that several preconditions on the same transition can be checked independently. *(II)* $\mathbb{C}_2^{\pi}$ is an extension of $\mathbb{C}_2$ in which only *some* of transitions of the supported agent depend on the state of the supporting agent. The algorithm for $\mathbb{C}_2^{\pi}$ is similar to that for $\mathbb{C}_2$, except that a certain modification of the permitted-edge graph $G^{X|Y}$ is used. *(III),(IV)* In $\mathbb{F}_n^{\wedge}$, a

```
Main
  Repeat Pruning while strategy sets are changing.
  If all strategy sets are empty, then report that no coordinated plan exists.
    Otherwise, perform Construction.
Pruning
  For all edges $(A_j, A_i)$ in $\mathcal{G}$ do:
    For each $\sigma^i \in \mathcal{S}^i$ do:
      If Compatibility-Check($\sigma_i, \sigma_j$) returns false for all $\sigma^j \in \mathcal{S}^j$ then remove $\sigma^i$ from $\mathcal{S}^i$.
    For each $\sigma^j \in \mathcal{S}^j$ do:
      If Compatibility-Check($\sigma_i, \sigma_j$) returns false for all $\sigma^i \in \mathcal{S}^i$ then remove $\sigma^j$ from $\mathcal{S}^j$.
Compatibility-Check $(\sigma_i, \sigma_j)$
  If $l(\sigma^i)[j]$ with neighboring repetitions removed is a subsequence of $\sigma^j$,
    then return true, else return false.
Construction
  Pick any agent $A$ and any strategy for it.
  Traverse $\mathcal{G}$ undirectly from the node $A$. For each visited node $A'$, choose any strategy of $A'$
    that is compatible with the strategy chosen for the node from which we came to $A'$.
```

Fig. 6. Algorithm for $\mathbb{P}_n^\pi$.

single agent A_1 supports a group of agents $A_2, \ldots, A_n$. The membership in NP can be easily shown for both $\mathbb{F}_n^\wedge$ and $\mathbb{F}_n^{\vee^\pi}$. In turn, the proofs of hardness for $\mathbb{F}_n^\wedge$ and $\mathbb{F}_n^{\vee^\pi}$ are by polynomial reductions from 3-SAT and PATH WITH FORBIDDEN PAIRS problems, respectively.

Remarks: The algorithm for $\mathbb{C}_n$ can be extended for $\mathbb{C}_n^\pi$, similarly to the extension of the $\mathbb{C}_2$ algorithm for $\mathbb{C}_2^\pi$. Results presented for the cases of chain and inverse fork dependence graphs can be generalized to the case of dependence graph being a tree directed to its root.

Coordinating Bounded Agents In this section we consider the class of problems for which the number and size of strategies for each agent is polynomial in the size of the problem domain. In what follows, such agents are referred as *bounded*. Let us denote by $\mathcal{S}^i$ the set of allowed strategies σ^i of A_i (i.e., source-to-target paths in G^i). We show that if a dependence graph forms a polytree then STS for bounded agents is polynomial. However, its extension to singly connected directed graphs is already NP-complete.

Consider $\mathbb{P}_n^\pi$. Notice that any polytree is an acyclic graph. Denote by $l(\sigma^i)$ the sequence of edge labels along σ^i, and by $l_j(\sigma^i)[j]$ the projection of $l(\sigma^i)$ to the nodes of G^j. Figure 6 presents our algorithm for $\mathbb{P}_n^\pi$.

Theorem 4. (I) *There exists a polynomial time algorithm for $\mathbb{P}_n^\pi$ with bounded agents that determines existence of a coordinated plan and finds such a plan, if exists.* **(II)** $\mathbb{S}_n^\pi$ *for bounded agents is* NP-*complete.*

Proof sketch. (I) According to the analysis in Section 3, the condition of *Compatibility-Check* confirms that the checked pair σ^i, σ^j is coordinated. Therefore, convergence of the pruning process ensures that for each agent A: (i) any one of its strategies has at least one supporting strategy of each predecessor of A in $\mathcal{G}$, and (ii) any one of its strategies supports at least one strategy for any successor of A in $\mathcal{G}$. Hence, if *Construction* has a personal strategy to begin with, then it is surely successful. Polynomiality holds, since each execution of *Pruning* decreases the total number of personal strategies.

(II) Membership in NP for $\mathbb{S}_n^\pi$ is straightforward, and the hardness proof is by a polynomial reduction from 3-SAT.

5 Discussion and Conclusions

In this paper, we concerned coordination for a set of agents that work on independent goals in the same environment. We investigated a particular family of the problems of finding coordinated plans, where actions of an agent depend on the local states of certain other agents. First, we presented a graphical representation model for this family of problems. Then, we classified these problems mainly according to the structural properties of the model. In the main part of the paper, we analyzed the computational properties of various problem classes. We gave polynomial solutions for several classes, while for some other we proved their NP-completeness or even provable intractability. A number of unsolved problems remains for the further research. The suggested approach—to find a sufficiently formal description for some problem classes, and to analyze their computational properties—seems to be prospective in the multi-agent area. Following some additional observations with respect to results presented in this paper.

First, let each agent and its states be considered as a multi-valued variable and its values, respectively, and let the set of agents' transitions be considered as a set of operators over the above variables. In this context, our results concern complexity analysis in the area of classical planning over multi-valued variables [4, 19]. Specifically, our results address the planning problems over multi-valued variables and only unary (= single effect) operators.

Second, in our model, each agent is assumed to be assigned to a personal goal, which is independent of the personal goals of all other agents. However, a supporting agent may have no explicit personal goal, while supporting activities of some other agents may be its only destiny. This particular relaxation can be immediately added into the presented model, with no negative impact on the computational properties of the problems. The coordinated group of dependent agents can be viewed as collaborating towards the achievement of some global goal.

Third, in this work we address only groups of fully controlled entities. Therefore, in particular, our results do not concern state-transition settings where agent's strategy depends on uncontrolled environment, like that studied in [3, 5].

In the future, we plan to continue with analysis of various classes of STS. In addition, we want to examine other forms of dependence between the agents, and other forms of goal(s) definition for a group of agents. This issues will be examined in the context of their impact on the complexity of coordination. We also plan to address related optimization problems.

References

1. R. Alami, F. Ingrand, and S. Qutub. A Scheme for Coordinating Multi-robot Planning Activities and Plans Execution. In *Proceedings of the 13th European Conference on Artificial Intelligence – ECAI*, Brighton, UK, 1998.

2. R. Alami, F. Robert, F. Ingrand, and S. Suzuki. Multi-robot Cooperation through Incremental Plan-Merging. In *International Conference on Robotics and Automation*, 1995.
3. R.C. Arkin and T. Balch. Cooperative Multiagent Robotic Systems. In *Artificial Intelligence and Mobile Robots*. MIT Press, 1998.
4. C. Bäckström and B. Nebel. Complexity Results for SAS$^+$ Planning. *Computational Intelligence*, 11(4):625–655, 1995.
5. T. Balch, G. Boone, T. Collins, H. Forbes, D. MacKenzie, and J. Santamaria. Io, Ganymede and Callisto - a Multiagent Robot Trash-Collecting Team. *AI Magazine*, 16(2):39–53, 1995.
6. R. A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.
7. Y. Cao, A. S. Fukunaga, and A. B. Kahng. Cooperative Mobile Robotics: Antecedents and Directions. *Autonomous Robots*, 4, 1997.
8. Bradley J. Clement and Edmund H. Durfee. Top-Down Search for Coordinating the Hierarchical Plans of Multiple Agents. In *Proceedings of the Third International Conference on Autonomous Agents*, pages 252–259, 1999.
9. P. Cohen and H. Levesque. Confirmations and Joint Actions. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 951–957, 1991.
10. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press.
11. K. Decker and V. Lesser. Designing a Family of Coordination Algorithms. In M. Huhns and M.Singh, editors, *Readings in Agents*, pages 450–457. Morgan Kaufmann, 1998.
12. M. d'Inverno, M. Luck, and M. Wooldridge. Cooperation Structures. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 600–605, 1997.
13. C. Domshlak and Y. Dinitz. Multi-Agent Off-line Coordination: Structure and Complexity. Technical Report CS-01-04, Department of Computer Science, Ben-Gurion University, 2001.
14. G. Dudek, M.R.M. Jenkin, E. Milios, and D. Wilkes. A Taxonomy for Multi-Agent Robotics. *Autonomous Robots*, 3(4):375–397, 1996.
15. E. Ephrati and J. S. Rosenschein. Divide and Conquer in Multi-Agent Planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 375–380, 1994.
16. M. Georgeff. Communication and Interaction in Multi-Agent Planning. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 125–129, 1983.
17. B. Grosz. Collaborative Systems. *AI Magazine*, 17(2):67–85, 1995.
18. B. Grosz and S. Kraus. Collaborative Plans for Complex Group Action. *Artificial Intelligence*, 86(2):269–357, 1996.
19. P. Jonsson and C. Bäckström. State-variable Planning under Structural Restrictions: Algorithms and Complexity. *Artificial Intelligence*, 100(1–2):125–176, 1998.
20. D. N. Kinny, M. Ljungberg, A. S. Rao, E. S. Sonenberg, G. Tidhar, and E. Werner. Planned Team Activity. In *Artificial Social Systems*, volume 830 of *LNCS*, pages 227–256. 1994.
21. J. McCarthy and P. Hayes. Some Phylosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 6, 1969.
22. Y. Moses and M. Tennenholtz. Multi-entity Models. *Machine Intelligence*, 14:63–88, 1995.
23. C. Le Pape. A Combination of Centralized and Distributed Methods for Multi-agent Planning and Scheduling. In *Proceedings of the IEEE International Conference on Robotics and Automation – ICRA*, pages 488–493, 1990.
24. O. Shehory and S. Kraus. Formation of Overlapping Coalitions for Precedence-ordered Task Execution among Autonomous Agents. In *Proceedings of the Second International Conference on Multi-Agent Systems*, pages 330–337. AAAI Press / MIT Press, 1996.
25. R. C. Smith. The Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solving. *IEEE Transactions on Computers*, 29(12), 1994.
26. S. Yuta and S. Premvuti. Coordinating Autonomous and Centralized Decision Making to Achieve Cooperative Behaviors between Multiple Mobile Robots. In *Proceedings of International Conference on Intelligent Robots and Systems*, pages 1566–1574, 1992.