Achieving Coordination through Combining Joint Planning and Joint Learning

Gerhard Weiß

Abstract. There are two major approaches to activity coordination in multiagent systems. First, by endowing the agents with the capability to jointly plan, that is, to jointly generate hypothetical activity sequences. Second, by endowing the agents with the capability to jointly learn, that is, to jointly choose the actions to be executed on the basis of what they know from experience about the interdependencies of their actions. This paper describes a new algorithm called JPJL (“Joint Planning and Joint Learning”) that combines both approaches. The primary motivation behind this algorithm is to bring together the advantages of joint planning and joint learning while avoiding their disadvantages. Experimental results are provided that illustrate the potential benefits and shortcomings of the JPJL algorithm.

1 Motivation

Multiagent Systems (MAS)—systems in which several interacting, intelligent and autonomous entities called agents pursue some set of goals or perform some set of tasks—have received steadily growing interest in both research and application in the past years (e.g., [4, 14, 28]). A key issue to be addressed when dealing with MAS is that of activity coordination: How can several agents, each capable of executing specific actions, decide together what activity sequence they should carry out in order to accomplish a common task? One possible answer is that the agents should jointly generate hypothetical activity sequences and do some kind of lookahead in order to determine the most promising actions, that is, they should jointly plan. A potential advantage of this approach is that the probability of carrying out unsuccessful and perhaps expensive or irreversible activity sequences is kept low. An inherent difficulty with this approach is, however, that it is limited by the agents’ knowledge about how relevant their individual actions are for goal attainment in different states and how to determine which of several possible next states is most appropriate for reaching the goal state. Another possible answer is that the agents should jointly choose the actions to be executed on the basis of what they already know from experience about the interdependencies among and effects of their actions, that is, they should jointly learn. What makes this approach appealing is that the agents themselves find out which paths of activity are likely to be successful and which are not, and that the amount of a priori knowledge with which the agents have to be equipped by the system designer is kept low. An inherent difficulty with this approach is, however, that the required number of learning trials tend to grow rapidly with the number of possible actions.

The work described in this paper aims at integrating joint planning and joint learning within a single algorithm that brings together the advantages of both approaches while avoiding their disadvantages. The basic idea behind this work is that the agents (i) jointly learn the information they need to know in order to evaluate the hypothetical activity paths generated during planning and (ii) jointly plan in order to reduce the number of uninformed and thus inefficient learning trials. The paper is structured as follows. Section 2 describes a new algorithm called JPJL (“Joint Planning and Joint Learning”) for integrated joint planning and joint learning. Section 3 presents initial experimental results that indicate the performance features of this algorithm. Finally, Section 4 briefly summarizes the paper, provides pointers to related work, and critically discusses limitations of the JPJL algorithm.

2 The JPJL Algorithm

The basic working cycle of the JPJL algorithm is conceptually described in the Figure 1. As the figure shows, the overall activity results from the repeated execution of three major activities, namely, planning, action selection, and learning. During planning, the agents jointly search through the space of possible future environmental states. During action selection, the agents jointly decide on the next action to be carried out based on their planning results. After having chosen and executed the selected action, the agents jointly learn by updating the estimated usefulness (goal relevance) of their actions. Below the three activities are described in detail. The description uses the following simple notation and is based on the following elementary assumptions. There is a finite set of agents \( A \), each capable of carrying out some actions \( a \). \( A_0 \) refers to the set of all agents, and \( A_0 \) refers to the set of actions that can be carried out by \( A \). The environment in which the agents act can be described as a feature-based state space, where the set of environmental features that can be sensed (i.e., identified as being either true or false) by the agents is denoted by \( F = \{ f, g, \ldots \} \). \( F^k \subseteq F \) (for \( k \in N \)) denotes a real or hypothetical environmental state, i.e., the set of environmental features that are known to be true (in the case of a real state) or assumed to be true by the agents (in the case of a hypothetical state). Following the traditional STRIPS approach [5], an agent \( A_i \) associates three lists with each of its actions \( a_j \): a set \( F^i_{pre} \subseteq F \) of preconditions that contains the environmental features that need to be fulfilled before this action can be carried out (“precondition set”); a set \( F^i_{del} \subseteq F \) of environmental features that become false through the execution of this action (“delete set”); and a set \( F^i_{add} \subseteq F \) of environmental features that become true by executing this action (“add set”). An agent is assumed to be able to determine, at each time, which of its actions could be carried out in the current (real or hypo-
1. Initialization:
   - States = \{current real state\}
   - Actual Planning Depth APD = 0

2. Planning:
   until APD = Maximal Planning Depth do
   - for each \( F \in \text{States} \) with \( F \neq \text{goal state} \) do
     - the agents determine their applicable actions and the corresponding hypothetical successor states that would result from applying these actions
     - \( States = States \cup \{\text{hypothetical successor states}\} \setminus \{F\} \)
     - the agents jointly estimate the usefulness of these actions
   - \( APD = APD + 1 \)

3. Action Selection:
   - the agents determine the overall usefulness of the action sequences generated during planning based on the individual actions’ estimated usefulness
   - the agents select the most promising sequence with highest probability
   - the first action of the selected sequence is carried out

4. Learning:
   - the agents update the estimated usefulness of their actions based on the observable effects of action execution

5. Goto 1

Figure 1. Conceptual description of the basic working cycle of the JPJL algorithm. Each of the three basic activities—planning, action selection, and learning—is jointly realized by the involved agents.

Figure 2. Illustration of the hypothetical search space of the JPJL algorithm (planning depth = 2).

- \( U_j(F)^\ast \) is called \( A_k \)'s evaluation function w.r.t. \( a_j \) and \( a_k \). After having calculated the usefulness values, \( A_k \) informs \( A_i \) about these values. \( A_i \), in turn, adds all usefulness values about which it was informed by other agents, resulting in an estimated overall usefulness \( U_j \) of \( a_j \) in state \( F^\ast \):

\[
U_j(F)^\ast = \max \left\{ 0, \frac{1}{r} \sum_{i} U_j(F)^{i} \right\}
\]

where \( r \) is the number of agents that responded to \( A_k \) and \( l \) ranges over these agents. \( U_j \) can be interpreted as a joint evaluation function that is represented and calculated in a distributed way by several agents. The result of starting with the current real state (see “Initialization” in Figure 1) and expanding this state up to a certain planning depth can be viewed as a jointly generated tree of potential future states in which the arcs represent potential actions together with their estimated overall usefulness. The Figure 2 illustrates this interpretation.

Action Selection. Let \( \mathcal{F}^0 \) denote the current real state, and assume that

\[
(\mathcal{F}^0, j) \defeq \text{true}
\]
For the purpose of a careful experimental analysis we used a series of synthetic scenarios that capture the characteristics of multiagent learning and planning and allow to efficiently obtain indicative results. This section presents the results for the scenarios summarized in the Tables 1 and 2.4 In the case of scenario 1 the environment consists of 20 features. The task to be solved by the agents is to transform an environmental start state into a goal state. There are four agents capable of carrying out different actions. Agent 1 can carry out just one action, agents 2 and 3 can each carry out two actions, and agent 4 can carry out three actions. What makes the task additionally complicated is that an agent can execute each of its actions in several contexts, differing in their precondition lists as well as their effects (i.e., their add and delete lists). In particular, executing an action under different preconditions results in different effects. For instance, consider action 2 of agent 2. The execution of this action always requires that the features \( f_{10} \) and \( f_{20} \) are true; additionally, one of the features \( f_0 \) (context 1), \( f_2 \) (context 2), or \( f_3 \) (context 3) has to be true. Through the execution of this action the feature \( f_6 \) always becomes true and the feature \( f_{30} \) always becomes false. Additionally, if \( f_0 \) (\( f_1 \), \( f_3 \)) is true at the time of execution, then \( f_3 \) (\( f_1 \), \( f_3 \)) becomes true and \( f_0 \) (\( f_1 \), \( f_3 \)) becomes false. Things are analogously in the scenario 2.

### Table 1. Specification of scenario 1. Top: range of features, start and goal state. Bottom: agents and their context-specific actions.

<table>
<thead>
<tr>
<th>Feature Set ( \mathcal{F} )</th>
<th>Start State</th>
<th>Goal State</th>
</tr>
</thead>
<tbody>
<tr>
<td>( {f_1, \ldots, f_9 } )</td>
<td>( {f_1, f_5, f_6, f_{13}, f_{15}, f_{20} } )</td>
<td>( {f_1, f_5, f_6, f_{18}, f_{20} } )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Agent</th>
<th>Action</th>
<th>Context 1</th>
<th>Context 2</th>
<th>Context 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pre del add</td>
<td>( f_1, f_5 )</td>
<td>( f_2, f_3 )</td>
<td>( f_8, f_9 )</td>
</tr>
<tr>
<td>2</td>
<td>pre del add</td>
<td>( f_1, f_5, f_{10} )</td>
<td>( f_2, f_3, f_{10} )</td>
<td>( f_8, f_9, f_{10} )</td>
</tr>
<tr>
<td>3</td>
<td>pre del add</td>
<td>( f_1, f_5, f_6, f_{13} )</td>
<td>( f_2, f_3, f_6, f_{13} )</td>
<td>( f_8, f_9, f_{13} )</td>
</tr>
<tr>
<td>4</td>
<td>pre del add</td>
<td>( f_1, f_5, f_6, f_{15} )</td>
<td>( f_2, f_3, f_6, f_{15} )</td>
<td>( f_8, f_9, f_{15} )</td>
</tr>
</tbody>
</table>

\(^3\) This selection process could be iterated such that not only one but several (compatible) actions are selected for execution within the current cycle.

\(^4\) The results we obtained for other scenarios (differing in the number of environmental features, the number of agents, and the number of actions) are qualitatively identical to those presented here.
as any sequence of at most 10 basic working cycles that transforms the start state into the goal state or any other state. Whenever the goal state is reached, the next trial starts (with the start state as the initial state). Each data point shows the mean reward achieved in the previous 25 cycles, averaged over 5 independent runs. As the curves show, the JPJL algorithm resulted in a clear performance improvement over time. The maximum reward was closely approached (above 95 percent) for different planning depths after about 280 cycles in the case of the scenario 1 and after about 370 cycles in the case of the scenario 2. The results also show that the choice of the planning depth is crucial to the overall system performance. Our major observations concerning the effects of the planning depth, as they are also indicated by the performance curves shown in the Figures 4 and 5, can be summarized as follows:

- Smaller planning depths tend to result in smoother, but slower increasing performance curves.
- Larger planning depths tend to result in performance curves that are less smooth (particularly in early stages), but increase faster.
- There is a risk of choosing a planning depth that is too large, resulting in relatively large and undesirable “performance jumps.”

These observations indicate that the planning depth is a very critical parameter that has to be chosen extremely carefully. According to our experience it is not feasible to try to compensate the negative effects of a badly chosen planning depth through modifying other parameters like the learning rates (α and β)—this just results in considerable experimental efforts that are not guaranteed to eventually succeed.

4 Conclusions

The JPJL algorithm aims at enabling multiple agents to achieve coordinated activity through combining their learning and planning efforts. The primary idea behind this algorithm is to interweave learning and planning within a single algorithm such that (i) learning helps to evaluate the results of planning and (ii) planning helps to reduce the number of required learning trials. Instead of “pure learning” or “pure planning,” the JPJL algorithm realizes a kind of “planning-based learning” or “learning-based planning.” The primary characteristic of this algorithm is that both learning and planning are jointly and distributedly realized by multiple agents.

In the area of multiagent systems a lot of work is available on both activity coordination through joint learning (e.g., [1, 12, 15],...
explores how planned-based and reactive behavior can be efficiently
rent work concentrates on the “fixed planning-depth limitation” and
possibilities of combining joint learning and joint planning. Our cur-
take the JPJL algorithm as a starting point for further exploring the
tations we think that the encouraging results available so far clearly
distributed modeling and diagnosis (e.g., [7, 9]). Despite these limi-
to cope with this limitation is to extend the JPJL algorithm toward
that, the JPJL algorithm runs into coordination problems in domains
to solve this problem through communicating these effects. Against
agent (not necessarily the one carrying out this action), it is possible
of its actions, that is, that an agent’s world model is perfect. In do-
portionally to the overall performance). Another, even more flexi-
results indicated, it is desirable that this is handled more flexible. One
way to cope with this limitation is to use a time-varying planning
depth (e.g., starting with a low depth which is then increased pro-
overall performance). Another, even more flexible way is that the agents on their own learn to adopt the depth of
their planning activities. The second limitation is that in general it
can not be assumed that an agent is always aware of all the effects
of its actions, that is, that an agent’s world model is perfect. In do-
where every effect of an action can be sensed by at least one agent
(not necessarily the one carrying out this action), it is possible
to solve this problem through communicating these effects. Against
that, the JPJL algorithm runs into coordination problems in domains
in which significant effects of actions are not so easy to detect. A way
to cope with this limitation is to extend the JPJL algorithm toward
distributed modeling and diagnosis (e.g., [7, 9]). Despite these limi-
tions we think that the encouraging results available so far clearly
justify to continue research in the directions indicated above and to
take the JPJL algorithm as a starting point for further exploring the
possibilities of combining joint learning and joint planning. Our cur-
ent work concentrates on the “fixed planning-depth limitation” and
explores how planned-based and reactive behavior can be efficiently
and effectively combined in multiagent settings.

REFERENCES
reinforcement learning agents’, Machine Learning, 33(2(3)), 235–262,
International Journal of Intelligent Cooperative Information Systems,
framework for distributed hypothesis formation’, IEEE Transactions on
cation of theorem proving to problem solving’, Artificial Intelligence,
in Proceedings of the Third National Conference on Artificial Intelli-
[7] B. Horling, V. Lesser, R. Vincent, A. Bazzan, and P. Xuan, ‘Diagnos-
sis as an integral part of multi-agent adaptability’, Technical Report
99-03, Computer Science Department, University of Massachusetts at
Amherst, (1999).
based coordination for multi-agent systems’, in Proceedings of the 2nd
[9] T. Hudlická and V.R. Lesser, ‘Modeling and diagnosing problem-
solving system behavior’, IEEE Transactions on Systems, Man, and
Proceedings of the First International Conference on Multi-Agent Sys-
Notes in Artificial Intelligence, Vol. 610, Springer-Verlag, Berlin et al.,
coordination in cooperative multi-agent systems’, Autonomous Agents
sous environment’, in Distributed Artificial Intelligence Meets Machine
Learning, ed., G. Weiß, Lecture Notes in Artificial Intelligence,
planning’, in Proceedings of the 2nd International Conference on Multi-
[18] S. Sen and M. Sekaran, ‘Multiagent coordination with learning classi-
fier systems’, in Adaptation and Learning in Multiagent Systems, eds., G.
study in robotic soccer’, in Adaptation, Coevolution and Learning in Multi-
Sen, Technical Report SS-96-01, 88–92, AAAI Press, Menlo Park, CA,
(1996).
in Proceedings of the First International Conference on Multi-Agent Systems
in Working Papers of the 12th International Workshop on Distributed
[22] T. Sugawara and V. Lesser, ‘Learning to improve coordinated ac-
tions in cooperative distributed problem-solving environments’, Ma-
[23] R.S. Sutton, ‘Learning to predict by the method of temporal differ-
King’s College, Cambridge University, 1989.
From Animals to Animats 2 – Proceedings of the Second Interna-
tional Conference on Simulation of Adaptive Behavior, pp. 502–510,
(1993).
Proceedings of the 13th International Joint Conference on Artificial In-