Achieving Coordination through Combining Joint Planning and Joint Learning

Gerhard Weiß[1]

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 unsucessful 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.

[1] Institut für Informatik, Technische Universität München, D-80290 München, Germany, weissg@in.tum.de

ECAI 2000. *14th European Conference on Artificial Intelligence*
Edited by Werner Horn
Published in 2000 by IOS Press

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_i, each capable of carrying out some actions a_j. $\mathcal{A}g$ refers to the set of all agents, and $\mathcal{A}c_i$ refers to the set of actions that can be carried out by A_i. 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 $\mathcal{F} = \{f, g, \ldots\}$. $\mathcal{F}^k \subseteq \mathcal{F}$ (for $k \in \mathsf{N}$) 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 $\mathcal{F}_j^{pre} \subseteq \mathcal{F}$ of preconditions that contains the environmental features that need to be fulfilled before this action can be carried out ("precondition set"); a set $\mathcal{F}_j^{del} \subseteq \mathcal{F}$ of environmental features that become false through the execution of this action ("delete set"); and a set $\mathcal{F}_j^{add} \subseteq \mathcal{F}$ 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 $\mathcal{F} \in States$ with $\mathcal{F} \neq$ goal state do
     - the agents determine their applicable actions and the corresponding hypothetical successor states that would result from applying these actions
     - $States = States \cup \{$hypothetical successor states$\} \setminus \{\mathcal{F}\}$
     - 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.

thetical) environmental state. This implies that in any given environmental state an agent A_i at least knows which of the features f in the set $\mathcal{F}_i^{aware} = \bigcup_{a_j \in \mathcal{A}c_i} \mathcal{F}_j^{pre}$ are true. Finally, it is assumed that an agent maintains for each of its actions a_j a set $\mathcal{F}_j^{true}$ that contains the environmental features $f \in \mathcal{F}_i^{aware}$ that were true at execution time ($\mathcal{F}_j^{pre} \subseteq \mathcal{F}_j^{true}$ for all a_j).

Planning. The basic idea behind the JPJL algorithm is that an agent A_i maintains an estimate $[f]_j$ for each $f \in \mathcal{F}_i^{aware}$ and each $a_j \in \mathcal{A}c_i$. These estimates are adjusted by the agents (as described below) such that they indicate what features should be true before certain actions are executed. An agent A_i interprets an estimate $[f]_j$ as follows: the higher (lower) it is, the more (less) likely it is that a_j should be only executed if f (and perhaps other features) is true. The values $[f]$ thus indicate under what environmental conditions the actions should be carried out. This approach reflects that different features can be of different relevance for different actions, no matter what agents could carry out these actions. Let $\mathcal{F}^*$ be a real or hypothetical state that is currently considered by the agents (i.e., $\mathcal{F}^* \in States$ as denoted in the Figure 1).[2] Each agent announces the actions it could carry out to the other agents (assuming a blackboard communication structure). After the potential actions are announced, each agent checks the influence of the announced actions w.r.t. its own actions and informs the announcing agents. More specifically, assume that A_i announced a_j together with the corresponding lists $\mathcal{F}_j^{add}$ and $\mathcal{F}_j^{del}$ (which allows for determining the potential successor state). Then each A_k calculates the usefulness U_j^l of this action w.r.t. each $a_l \in \mathcal{A}c_k$ as follows:

$$U_j^l(\mathcal{F}^*) \stackrel{def}{=} \sum_{f \in \mathcal{F}_j^{add} \cap \mathcal{F}_k^{aware}} [f]_l \; - \sum_{f \in \mathcal{F}_j^{del} \cap \mathcal{F}_k^{aware}} [f]_l \quad . \qquad (1)$$

[2] The order in which the agents consider the states in $States$ may be arbitrary (as in the current version of the JPJL algorithm), adaptive, or predefined by the system designer.

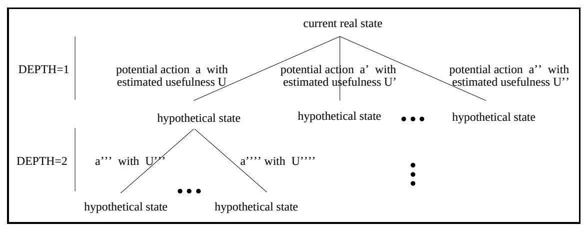

Figure 2. Illustration of the hypothetical search space of the JPJL algorithm (planning depth = 2).

U_j^l is called A_k's evaluation function w.r.t. a_j and a_l. 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 $\mathcal{F}^*$:

$$U_j(\mathcal{F}^*) \stackrel{def}{=} \max\{\ 0\ ,\ \frac{1}{r} \sum_l U_j^l(\mathcal{F}^*)\ \} \quad , \qquad (2)$$

where r is the number of agents that responded to A_i 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

$$\langle \mathcal{F}^0, j \rangle \stackrel{def}{=}$$

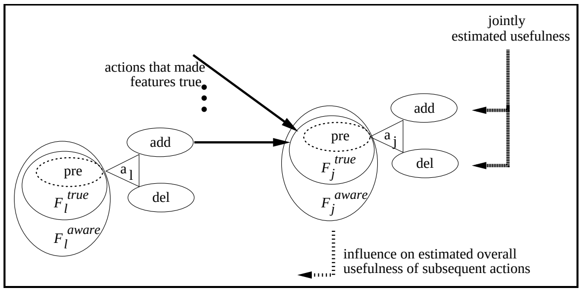

Figure 3. Illustration of the JPJL update rule.

$$\mathcal{F}^0 \stackrel{a_{j_1} \,/\, U_{j_1}(\mathcal{F}^0)}{\longrightarrow} \mathcal{F}^1 \stackrel{a_{j_2} \,/\, U_{j_2}(\mathcal{F}^1)}{\longrightarrow} \mathcal{F}^2 \ \ldots$$
$$\ldots\ \mathcal{F}^{m-1} \stackrel{a_{j_m} \,/\, U_{j_m}(\mathcal{F}^{m-1})}{\longrightarrow} \mathcal{F}^m$$

is one of the jointly generated planning paths (i.e., one path from the root to a leaf in the search tree), where a_{j_k} is an agent's potential action that transfers $\mathcal{F}^{k-1}$ into the successor state $\mathcal{F}^k$, $U_{j_k}(\mathcal{F}^{k-1})$ is the estimated overall usefulness of a_{j_k} applied in state $\mathcal{F}^{k-1}$, and m is the maximal planning depth. Then the estimated usefulness of this path is defined as the sum of the usefulness values of the individual actions along this path:

$$U_{\langle \mathcal{F}^0, j \rangle} \stackrel{def}{=} \sum_{k=1}^{m} U_{j_k}(\mathcal{F}^{k-1}) \quad . \tag{3}$$

Among all potential paths, path $\langle \mathcal{F}^0, j \rangle$ is selected with the probability

$$\frac{\mathrm{e}^{\left(U_{\langle \mathcal{F}^0, j \rangle}\right)}}{\mathrm{e}^{\left(\sum_k U_{\langle \mathcal{F}^0, k \rangle}\right)}} \quad , \tag{4}$$

where k ranges over all potential paths generated during planning. This means that a path's probability of being selected increases with its estimated usefulness. Once a path $\langle \mathcal{F}^0, j \rangle$ is selected, the first action (i.e., a_{j_1}) of this path is executed.[3]

Learning. Learning is realized by jointly adjusting the action-specific estimates of the environmental features. The adjustment is done in a distributed manner by the agents that carried out actions. More specifically, assume that a_j proposed by A_i has been selected for execution in the real state $\mathcal{F}^0$. The A_i updates its estimates $[f]_j$ for all $f \in \mathcal{F}_j^{true}$ as follows:

$$[f]_j = [f]_j + \alpha \cdot \left(\beta \cdot U_j(\mathcal{F}^0) - [f]_j + R\right) \quad , \tag{5}$$

where α and β are constants called learning rates and R is the actual external reward that A_i received after the execution of a_j. (In the case of delayed rewards, R may be equal to zero.) This update rule, which is in the spirit of Q-learning [24, 25] and temporal difference learning [23], aims at increasing (decreasing) A_i's chance to carry out a_j in the future, if the usefulness of this action is jointly estimated as being high (low) and/or if this action results (does not result) in an external reward. The Figure 3 illustrates the update rule. Note that increasing the usefulness of the features that were true at the time of executing a_j (including the preconditions of a_j) increases the execution probability of the actions that made these features true; this in turn increases the execution probability of a_j.

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

3 Experimental Results

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_6 (context 1), f_7 (context 2), or f_9 (context 3) has to be true. Through the execution of this action the feature f_8 always becomes true and the feature f_{20} always becomes false. Additionally, if f_6 (f_7, f_9) is true at the time of execution, then f_5 (f_{15}, –) becomes true and f_6 (f_7, f_9) becomes false. Things are analogously in the scenario 2.

Feature Set $\mathcal{F}$	$\{f_1, \ldots, f_{20}\}$
Start State	$\{f_1, f_5, f_9, f_{13}, f_{15}, f_{19}\}$
Goal State	$\{f_1, f_5, f_8, f_{18}, f_{19}\}$

Agent	Action		Context 1	Context 2	Context 3
1	1	pre	f_1, f_5	f_2, f_5	f_3, f_5
		del	f_1	f_2	f_3
		add	f_4, f_{10}	f_4, f_{15}	f_4, f_{20}
2	1	pre	f_2, f_5, f_{20}	f_3, f_5, f_{20}	f_4, f_5, f_{20}
		del	f_2, f_{10}	f_3, f_{10}	f_4, f_{10}
		add	f_1, f_5	f_1, f_{20}	f_1
	2	pre	f_6, f_{10}, f_{20}	f_7, f_{10}, f_{20}	f_9, f_{10}, f_{20}
		del	f_6, f_{20}	f_7, f_{20}	f_9, f_{20}
		add	f_5, f_8	f_8, f_{15}	f_8
3	1	pre	f_5, f_{17}, f_{20}	f_5, f_{18}, f_{20}	f_5, f_{19}, f_{20}
		del	f_5, f_{17}	f_5, f_{18}	f_5, f_{19}
		add	f_{10}, f_{16}	f_{15}, f_{16}	f_{16}
	2	pre	f_{15}, f_{16}, f_{20}	f_{15}, f_{17}, f_{20}	f_{15}, f_{19}, f_{20}
		del	f_{15}, f_{16}	f_{15}, f_{17}	f_{15}, f_{19}
		add	f_5, f_{18}	f_{10}, f_{18}	f_{18}
4	1	pre	f_{11}, f_{15}, f_{20}	f_{12}, f_{15}, f_{20}	f_{14}, f_{15}, f_{20}
		del	f_{11}, f_{20}	f_{12}, f_{20}	f_{14}, f_{20}
		add	f_5, f_{13}	f_{10}, f_{13}	f_{13}
	2	pre	f_{11}, f_{15}	f_{12}, f_{15}	f_{13}, f_{15}
		del	f_{11}	f_{12}	f_{13}
		add	f_5, f_{14}	f_{10}, f_{14}	f_{14}, f_{20}
	3	pre	f_5, f_7, f_{10}	f_5, f_8, f_{10}	f_5, f_9, f_{10}
		del	f_5, f_7	f_5, f_8	f_5, f_9
		add	f_6, f_{15}	f_6, f_{20}	f_6

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

The Figures 4 and 5 show the performance profiles for the scenarios 1 and 2, respectively, for the planning depths 1 (curve "JPJL1"), 2 ("JPJL2"), and 3 ("JPJL3"). For all shown results the experimental setting was as follows. $\alpha = 0.2$, $\beta = 0.9$, and $R = 1000$ *iff* the goal state was reached. The initial values of the estimates $[f]$ were all zero (which means that the initial behavior is random). Learning proceeds by the repeated execution of trials, where a trial is defined

[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.

Feature Set $\mathcal{F}$	$\{f_1, f_2, \ldots, f_{30}\}$
Start State	$\{f_5, f_7, f_{14}, f_{18}, f_{23}, f_{28}, f_{30}\}$
Goal State	$\{f_1, f_8, f_{18}, f_{22}, f_{24}, f_{29}\}$

Agent	Action		Context 1	Context 2	Context 3	Context 4
1	1	pre	f_2, f_6, f_{12}	f_3, f_6, f_{12}	f_4, f_6, f_{12}	f_5, f_6, f_{12}
		del	f_2, f_{12}	f_3, f_{12}	f_4, f_{12}	f_5, f_{12}
		add	f_1, f_{18}	f_1, f_{24}	f_1, f_{30}	f_1
	2	pre	f_1, f_6	f_2, f_6	f_3, f_6	f_4, f_6
		del	f_1	f_2	f_3	f_4
		add	f_5, f_{12}	f_5, f_{18}	f_5, f_{24}	f_5, f_{30}
	3	pre	f_7, f_{12}, f_{18}	f_9, f_{12}, f_{18}	f_{10}, f_{12}, f_{18}	f_{11}, f_{12}, f_{18}
		del	f_7, f_{18}	f_9, f_{18}	f_{10}, f_{18}	f_{11}, f_{18}
		add	f_6, f_8	f_8, f_{24}	f_8, f_{30}	f_8
2	1	pre	f_7, f_{12}, f_{24}	f_8, f_{12}, f_{24}	f_{10}, f_{12}, f_{24}	f_{11}, f_{12}, f_{24}
		del	f_7, f_{24}	f_8, f_{24}	f_{10}, f_{24}	f_{11}, f_{24}
		add	f_6, f_9	f_9, f_{18}	f_9, f_{30}	f_9
3	1	pre	f_{13}, f_{18}, f_{24}	f_{14}, f_{18}, f_{24}	f_{16}, f_{18}, f_{24}	f_{17}, f_{18}, f_{24}
		del	f_{13}, f_{24}	f_{14}, f_{24}	f_{16}, f_{24}	f_{17}, f_{24}
		add	f_6, f_{15}	f_{12}, f_{15}	f_{15}, f_{30}	f_{15}
	2	pre	f_{13}, f_{18}	f_{14}, f_{18}	f_{15}, f_{18}	f_{16}, f_{18}
		del	f_{13}	f_{14}	f_{15}	f_{16}
		add	f_6, f_{17}	f_{12}, f_{17}	f_{17}, f_{24}	f_{17}, f_{30}
	3	pre	f_{18}, f_{19}, f_{24}	f_{18}, f_{20}, f_{24}	f_{18}, f_{22}, f_{24}	f_{18}, f_{23}, f_{24}
		del	f_{18}, f_{19}	f_{18}, f_{20}	f_{18}, f_{22}	f_{18}, f_{23}
		add	f_6, f_{21}	f_{12}, f_{21}	f_{21}, f_{30}	f_{21}
	4	pre	f_{19}, f_{24}, f_{30}	f_{20}, f_{24}, f_{30}	f_{21}, f_{24}, f_{30}	f_{23}, f_{24}, f_{30}
		del	f_{10}, f_{30}	f_{20}, f_{30}	f_{21}, f_{30}	f_{23}, f_{30}
		add	f_6, f_{22}	f_{12}, f_{22}	f_{18}, f_{22}	f_{22}
	5	pre	f_{12}, f_{25}, f_{30}	f_{12}, f_{27}, f_{30}	f_{12}, f_{28}, f_{30}	f_{12}, f_{29}, f_{30}
		del	f_{12}, f_{25}	f_{12}, f_{27}	f_{12}, f_{28}	f_{12}, f_{29}
		add	f_6, f_{26}	f_{18}, f_{26}	f_{24}, f_{26}	f_{26}
4	1	pre	f_{18}, f_{25}, f_{30}	f_{18}, f_{26}, f_{30}	f_{18}, f_{28}, f_{30}	f_{18}, f_{29}, f_{30}
		del	f_{18}, f_{25}	f_{18}, f_{26}	f_{18}, f_{28}	f_{18}, f_{29}
		add	f_6, f_{27}	f_{12}, f_{27}	f_{24}, f_{27}	f_{27}
	2	pre	f_{25}, f_{30}	f_{26}, f_{30}	f_{27}, f_{30}	f_{28}, f_{30}
		del	f_{25}	f_{26}	f_{27}	f_{28}
		add	f_6, f_{29}	$f_1 2, f_{29}$	f_{18}, f_{29}	f_{24}, f_{29}

Table 2. Specification of scenario 2.

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.

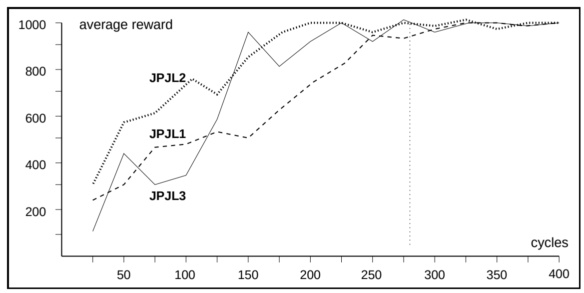

Figure 4. JPJL performance curves for scenario 1.

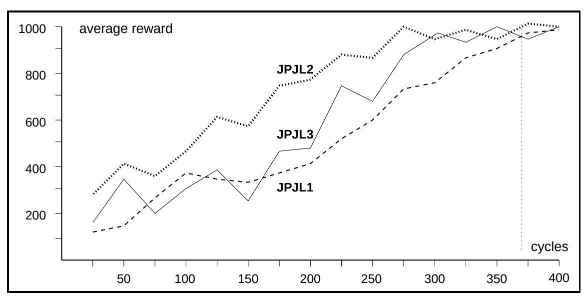

Figure 5. JPJL performance curves for scenario 2.

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 interwine 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, 16, 18, 19, 26, 27]) and activity coordination through joint planning (e.g., [2, 3, 6, 8, 10, 11, 17, 20]). However, there are only very little approaches that combine joint learning and joint planning. There are two exceptions that are related to the JPJL algorithm. The first is the work by Sugawara and Lesser described in e.g. [21, 22]. The basic idea behind this approach is to enable agents to learn situation-specific rules that capture relevant non-local information in order to improve local planning and reasoning. This idea has been investigated within the context of LODES, a distributed diagnosis system for computer communications networks. The second is the approach by Nagendra Prasad and Lesser described in e.g. [13]. Here the central idea is to endow agents with the capability to learn to choose appropriate, situation-specific coordination strategies from a set of available strategies. This idea has been implemented in a system called COLLAGE. The primary difference between the LODES/COLLAGE approaches and the JPJL algorithm is that the former are very knowledge-intensive whereas the latter is not. In particular, in the case of LODES the agents are required to a priori possess deep domain knowledge and in the case of COLLAGE the agents are required to a priori possess sophisticated coordination knowledge in order to be able to appropriately coordinate their activities. Against that, in the case of JPJL coordination "evolves from the scratch," without requiring that particular domain or coordination knowledge is a priori available to the agents.

In its current form the JPJL algorithm is limited in two specific respects. The first limitation is that the JPJL algorithm assumes that the planning depth is fixed and predefined. As the experimental 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 proportionally to the 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 domains 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 limitations 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 current 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

[1] R.H. Crites and A.G. Barto, 'Elevator group control using multiple reinforcement learning agents', *Machine Learning*, **33**(2/3), 235–262, (1998).

[2] K.S. Decker and V.R. Lesser, 'Generalized partial global planning', *International Journal of Intelligent Cooperative Information Systems*, **1**(2), 319–346, (1992).

[3] E.H. Durfee and V.R. Lesser, 'Partial global planning: A coordination framework for distributed hypothesis formation', *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-21**(5), 1167–1183, (1991).

[4] J. Ferber, *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*, John Wiley & Sons Inc., New York, 1999.

[5] R.E. Fikes and N.J. Nilsson, 'STRIPS: A new approach to the application of theorem proving to problem solving', *Artificial Intelligence*, **2(3-4)**, 189–208, (1971).

[6] M. Georgeff, 'Communication and interaction in multi-agent planning', in *Proceedings of the Third National Conference on Artificial Intelligence (AAAI-83)*, pp. 125–129, (1983).

[7] B. Horling, V. Lesser, R. Vincent, A. Bazzan, and P. Xuan, 'Diagnosis as an integral part of multi-agent adaptability', Technical Report 99-03, Computer Science Department, University of Massachussetts at Amherst, (1999).

[8] M.J. Huber and E.H. Durfee, 'An initial assessment of plan-recognition-based coordination for multi-agent systems', in *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS-96)*, pp. 126–133, (1996).

[9] E. Hudlická and V.R. Lesser, 'Modeling and diagnosing problem-solving system behavior', *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-17(3)**, 407–419, (1987).

[10] F. Kabanza, 'Synchronizing multiagent plans using temporal logic', in *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pp. 217–224, (1995).

[11] F. von Martial, *Coordinating plans of autonomous agents*, Lecture Notes in Artificial in Artificial Intelligence, Vol. 610, Springer-Verlag, Berlin et al., 1992.

[12] M. Matarić, 'Reinforcement learning in the multi-robot domain', *Autonomous Robots*, **4**(1), 73–83, (1997).

[13] M.V. Nagendra Prasad and V.R. Lesser, 'Learning situation-specific coordination in cooperative multi-agent systems', *Autonomous Agents and Multi-Agent Systems*, **2**, 173–207, (1999).

[14] G.M.P. O'Hare and N.R. Jennings (eds.), *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons Inc., New York, 1996.

[15] N. Ono and Y. Fukuta, 'Learning coordinated behavior in a continuous environment', in *Distributed Artificial Intelligence Meets Machine Learning*, ed., G. Weiß, Lecture Notes in Artificial in Artificial Intelligence, Vol. 1221, 73–81, Springer-Verlag, Berlin et al., (1997).

[16] L.E. Parker, 'L-alliance: Task-oriented multi-robot learning inbehavior-based systems', *Advanced Robotics*, **11**(4), 305–322, (1997).

[17] A.E.F. Seghrouchni and S. Haddad, 'A recursive model for distributed planning', in *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS-96)*, pp. 307–314, (1996).

[18] S. Sen and M. Sekaran, 'Multiagent coordination with learning classifier systems', in *Adaption and Learning in Multiagent Systems*, eds., G. Weiß and S. Sen, Lecture Notes in Artificial in Artificial Intelligence, Vol. 1042, 218–233, Springer-Verlag, Berlin et al., (1996).

[19] P. Stone and M. Veloso, 'Collaborative and adversarial learning: A case study in robotic soccer', in *Adaptation, Coevolution and Learning in Multiagent Systems. Papers from the 1996 AAAI Symposium*, ed., S. Sen, Technical Report SS-96-01, 88–92, AAAI Press, Menlo Park, CA, (1996).

[20] T. Sugawara, 'Reusing past plans in distributed planning', in *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pp. 360–367, (1995).

[21] T. Sugawara and V. Lesser, 'On-line learning of coordination plans', in *Working Papers of the 12th International Workshop on Distributed Artificial Intelligence*, (1993).

[22] T. Sugawara and V. Lesser, 'Learning to improve coordinated actions in cooperative distributed problem-solving environments', *Machine Learning*, **33**(2/3), 129–153, (1998).

[23] R.S. Sutton, 'Learning to predict by the method of temporal differences', *Machine Learning*, **3**, 9–44, (1988).

[24] C.J.C.H. Watkins, *Learning from Delayed Rewards*, Ph.D. dissertation, King's College, Cambridge University, 1989.

[25] C.J.C.H. Watkins and P. Dayan, 'Q-learning', *Machine Learning*, **8**, 279–292, (1992).

[26] G. Weiss, 'Action selection and learning in multi-agent environments', in *From Animals to Animats 2 – Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pp. 502–510, (1993).

[27] G. Weiss, 'Learning to coordinate actions in multi-agent systems', in *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, pp. 311–316, (1993).

[28] G. Weiss (ed.), *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, Cambridge, MA, 1999.