Planning and Learning Together

Gerhard Weiß
Institut für Informatik, Technische Universität München
D-80290 München, Germany
weissg@in.tum.de

ABSTRACT

This paper describes a novel algorithm for activity coordination in multiagent systems that combines joint planning and joint learning. The basic idea underlying this algorithm is to combine the advantages of planning and learning, while at the same time avoiding their disadvantages.

1. INTRODUCTION

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.

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.

2. COMBINING JOINT PLANNING AND JOINT LEARNING

This section describes a novel algorithm called JPJL ("Joint Planning and Joint Learning") that aims at achieving multiagent coordination through combined joint planning and joint learning. According to this algorithm the overall multiagent activity results from the repeated execution of three major joint activities: 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 $\mathcal{F} = \{f, g, \dots\}$. $\mathcal{F}^k \subseteq \mathcal{F}$ (for $k \in \mathsf{N}$) denotes a real or hypothetical environmental state. 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 ("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"). In any 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.

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

Agents 2000 Barcelona Spain
Copyright ACM 2000 1-58113-230-1/00/6...$5.00

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. Let $\mathcal{F}^*$ be a real or hypothetical state that is currently considered by the agents. 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 $\mathcal{F}_j^{del}$. 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:

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

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 $\mathcal{F}^*$:

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

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.

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

$$\langle \mathcal{F}^0, j \rangle \overset{def}{=} \mathcal{F}^0 \;\; {}^{a_{j_1}} / \overset{U_{j_1}(\mathcal{F}^0)}{\longrightarrow} \;\; \mathcal{F}^1 \;\; {}^{a_{j_2}} / \overset{U_{j_2}(\mathcal{F}^1)}{\longrightarrow} \;\; \mathcal{F}^2 \ldots$$

$$\ldots \mathcal{F}^{m-1} \;\; {}^{a_{j_m}} / \overset{U_{j_m}(\mathcal{F}^{m-1})}{\longrightarrow} \;\; \mathcal{F}^m$$

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:

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

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

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

where k ranges over all potential paths generated during planning.

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:

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

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 and temporal difference learning, 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.

3. CONCLUDING REMARKS

For the purpose of an experimental analysis we used a series of synthetic scenarios that capture the characteristics of multiagent learning and planning. The results gained so far show that the JPJL algorithm can lead to a clear improvement in the overall system performance.

In the area of multiagent systems a lot of work is available on both activity coordination through joint learning and activity coordination through joint planning. 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, namely the work by Sugawara and Lesser on LODES (e.g., [2]) and the work by Nagendra Prasad and Lesser on COLLAGE (e.g., [1]). 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.

Acknowledgments. Research reported in this paper has been financially supported by Deutsche Forschungsgemeinschaft DFG under contract We1718/6-3.

4. REFERENCES

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

[2] T. Sugawara and V. Lesser. Learning to improve coordinated actions in cooperative distributed problem-solving environments. *Machine Learning*, 33(2/3):129–153, 1998.