ELSEVIER

Robotics and Autonomous Systems 15 (1995) 135–142

Robotics and Autonomous Systems

Distributed reinforcement learning

Gerhard Weiß

Institut für Informatik, Technische Universität München, D-80290 München, Germany

Abstract

In multi-agent systems two forms of learning can be distinguished: centralized learning, that is, learning done by a single agent independent of the other agents; and distributed learning, that is, learning that becomes possible only because several agents are present. Whereas centralized learning has been intensively studied in the field of artificial intelligence, distributed learning has been completely neglected until a few years ago.

This paper summarizes work done on distributed reinforcement learning. The problem addressed is how multiple agents can learn to coordinate their actions such that they collectively solve a given environmental task. Two learning algorithms called ACE and DFG are described that provide answers to the following two questions:

- How can multiple agents learn which actions have to be carried out concurrently?
- How can multiple agents learn which sets of concurrent actions have to be carried out sequentially?

Initial experimental results are provided which illustrate the learning abilities of these algorithms.

Keywords: Multi-agent systems; Distributed reinforcement learning; Activity coordination; ACE algorithm; DFG algorithm

1. Motivation

Multi-agent systems establish a central research area in distributed artificial intelligence (see, e.g., [2,15,7,3]). The interest in these systems bases on the insight that many real-world problems are better modelled using a set of interacting agents instead of a single agent [8]. In particular, multi-agent modelling allows to cope with natural constraints like the limited processing power of a single agent or the geographical distribution of data and to profit from inherent properties of distributed systems like robustness, parallelism and scalability. Various multi-agent systems have been described in the literature. According to the standard or "prototypical" point of view a multi-agent system consists of a number of agents being able to interact and differing from each other in their skills and their knowledge about the environment. Each agent is assumed to be composed of a sensor component, a motor component, a knowledge base and a learning component. An agent typically is restricted in its activity for three reasons:

- because of limitations imposed on the sensor component, it knows only a part of the environment (i.e., it is not "omniscient"),
- because of limitations imposed on the motor component, it is specialized in carrying out a specific action (i.e., it is not "omnipotent"), and
- its action can be incompatible with actions carried out by other agents (i.e., different actions may prevent each other from being executed).

This prototypical point of view also underlies the work described in this paper.

Two forms of learning can be distinguished in a multi-agent system (see also [16]). First, isolated or centralized learning, that is, learning that is done by a

SSDI 0921-8890(95)00018-6

single agent and that does not require the presence of other agents (e.g., learning by creating new knowledge structures). Second, collective or distributed learning, that is, learning that is done by several agents and that becomes possible only because several agents are present (e.g., learning by exchanging knowledge). Whereas centralized learning has been intensively studied since the early days of artificial intelligence, distributed learning has been neglected until a few years ago. This is in contrast to the common agreement that there are two important reasons for studying this subject:

- to be able to endow artificial multi-agent systems, which typically are very complex and hard to specify, with the ability to improve their behavior on their own; and
- to get a better understanding of the learning processes in natural (human and animal) multi-agent systems.

Despite some initial research efforts (e.g., [4,16–19]), however, there is a great number of open problems and questions on distributed learning.

This paper summarizes work that focused on the relation between learning and action coordination in multi-agent systems and that has been part of a broader research project aiming at a more unified perspective of learning in parallel and distributed systems [23]. The central problem addressed is how several agents can learn to coordinate their actions such that they collectively solve environmental tasks, even if the learning feedback is minimal and consists only of a scalar reinforcement value. Two algorithms called the ACE algorithm and the DFG algorithm are described which implement distributed reinforcement learning and endow multiple agents with the ability to generate appropriate sequences of sets of compatible actions.

2. The ACE algorithm

2.1. Overview

The ACE algorithm (ACE stands for "ACtion Estimation") is designed to solve the problem of learning appropriate sequences of action sets in multi-agent systems (e.g., [21]) [1]. The working method of this algorithm can be overviewed as follows. Each agent estimates the usefulness of its action in different environmental states. Based on these estimates, in each environmental state the agents compete for the right to become active. Only the winning agents are allowed to perform their actions and, by the way, to transform the actual into the next environmental state. The agents learn by collectively adjusting and improving, over time, the estimates of their actions. This adjustment, which is also known as credit assignment or apportionment of credit, is done according to the action-oriented variant [20] of a reinforcement learning model called bucket brigade [14] which originally comes from the field of classifier systems. All together, according to the ACE algorithm the overall behavior of the multi-agent system results from the repeated execution of the competition and the credit assignment activities. The next subsections give a detailed description of these two activities.

2.2. Competition

In each environmental state S_j a competition runs between the agents. Each agent A_i makes a bid B_i^j for the right to carry out its action and announces this bid to the other agents. This bid is calculated by

$$B_i^j = \begin{cases} (\alpha + \beta) \cdot E_i^j & \text{if } E_i^j > \Theta, \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where α is a small constant called risk factor, β is a small random number called noise term, Θ is a constant called estimate minimum, and E_i^j is A_i's estimate of the usefulness of its action dependent on what it knows about S_j. (E_i^j is initialized with a predefined value E^{init}.) The α indicates the fraction of E_i^j the agent A_i is willing to risk for being allowed to become active. The β introduces noise into the competition process in order to avoid getting stuck into local learning minima. (In the literature on classifier systems various methods of introducing noise into the bidding process have been described; see, for instance, [10].) The Θ helps to prevent executing useless (low-estimated) actions. In the following, $\alpha \cdot E_i^j$ is called the deterministic part and $\beta \cdot E_i^j$ is called the stochastic part of B_i^j.

After the agents have announced their bids, they select the actions that are carried out concurrently. The

[1] There a slightly different notation was used.

agent having made the highest bid is allowed to execute its action, and each potentially active agent whose action is incompatible to the selected one withdraw its bid; this is repeated until no further action associated with a non-zero bid can be selected. Formally, action selection is described by

- $\mathcal{A}^{\text{pot}}[S_j] =_{\text{def}}$ set of agents that could become active in S_j,
  $\mathcal{A}^{\text{act}}[S_j] =_{\text{def}} \emptyset$;
- until $\mathcal{A}^{\text{pot}}[S_j] = \emptyset$ do
  - select $A_i \in \mathcal{A}^{\text{pot}}[S_i]$ with $B_i^j > B_k^j$ for all $A_k \in \mathcal{A}^{\text{pot}}[S_j]$
  - $\mathcal{A}^{\text{act}}[S_j] = \mathcal{A}^{\text{act}}[S_j] \cup \{A_i\}$
  - $\mathcal{A}^{\text{pot}}[S_j] = \mathcal{A}^{\text{pot}}[S_j] \setminus (A_i \cup \{A_k \in \mathcal{A}^{\text{pot}}[S_j] : A_k \text{ and } A_i \text{ are incompatible}\})$;
- only the agents in $\mathcal{A}^{\text{act}}[S_j]$ become active.

(Note that this kind of competition requires a rational or non-egoistic behavior of the agents in the sense that none of the agents insists the execution of a low-bid or an incompatible action.)

2.3. Credit assignment

The agents assign credit to each other by adjusting the estimates of the usefulness of their actions. Informally, this is done as follows. The agents that won the actual competition reduce the estimates of their actions (the actual winners pay for their privilege to carry out their actions) by the amount of the deterministic part of their bids, and hand the sum of all reductions back to the agents that won the previous competition (the previous winners are rewarded for appropriately setting up the environment). The previous winners, in turn, add the received sum to the estimates of their own actions. Additionally, if there is an external reward from the environment, then it is distributed among the actual winners. This adjustment of the estimates is formally described as follows. Let S_j and S_l be the actual and the previous environmental state, respectively. The estimate E_i^j of each actual winner A_i is modified according to

$$E_i^j = E_i^j - \alpha \cdot E_i^j + R^{\text{ext}}/|\mathcal{A}^{\text{act}}[S_j]|, \tag{2}$$

where R^{ext} is the external reward (if there is any). The estimate E_k^l of each previous winner A_k is increased according to

$$E_k^l = E_k^l + B/|\mathcal{A}^{\text{act}}[S_l]|. \tag{3}$$

where $B = \sum_{A_i \in \mathcal{A}^{\text{act}}[S_j]} \alpha \cdot E_i^j$ is the sum of all reductions made by the actual winners.

The effects of this bucket-brigade-type credit assignment are as follows (see [13]). An agent's estimate of its action increases (decreases), if the agent pays less (more) than it receives. As a consequence, the estimates of actions that are involved in successful sequences of action sets (i.e., sequences that lead to external reward) increase over time and stabilize this sequence; and conversely, the estimates of actions that are involved in unsuccessful sequences decrease over time and destabilize this sequence.

3. The DFG algorithm

3.1. Overview

Like the ACE algorithm, the DFG algorithm (DFG is an abbreviation for "Dissolution and Formation of Groups") implements distributed reinforcement learning of action sequences (e.g., [22]). In contrast to the ACE algorithm, however, the DFG algorithm explicitly distinguishes between single agents and groups of compatible agents as the acting units in a multi-agent system. Now agents as well as groups estimate the usefulness of their activities in different environmental states, and agents as well as groups compete for the right to become active. In particular, now learning encompasses two interrelated processes: first, bucket-brigade-type credit assignment; and second, group development, that is, the process of dissolving existing (useless) and forming new (useful) groups. The following subsections describe the DFG algorithm in detail.

3.2. Groups as acting units

As it is known from organization theory and management science, single agents typically serve as building blocks for more complex structured and autonomously acting units (see, e.g., [5,6]). The DFG algorithm adopts this point of view, and distinguishes between single agents and groups as the acting units in a multi-agent system. A group is considered to be composed of a group leader and several compatible group members, where a group leader is a single agent and a group member is either a single agent or an-

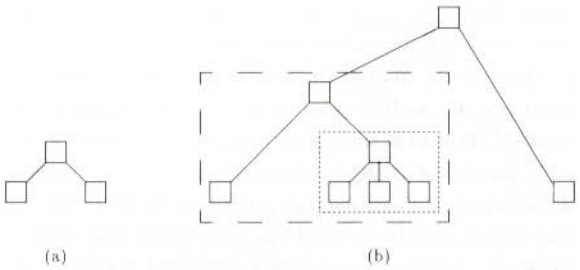


Fig. 1. Examples of group structures. (a) shows the most simple group which consists of a leader and two members each being a single agent. (b) shows a more complicated group which has two members, one being a group (dashed box) and the other being a single agent. The "dashed group" also has a group (dotted box) and a single agent as its members, where the "dotted group" has three members each being a single agent. (Legend: □ single agent, /\ leader–member relations.)

other group. This recursive definition is rather general and covers both low and high structured groups; see Fig. 1 for an illustration. The task of a group leader is to represent the group's interests; this includes, for instance, to decide whether the group should persist as an autonomously acting unit, cooperate with another acting unit, or dissolve. The group members have to be compatible in the sense that the activity of no member leads to environmental changes that prevent the activity of another member.

The following simple notation is used in the following subsections. U_i refers to an acting unit, that is, either to a single agent or to a group. If U_i refers to a group, then $\overline{U_i}$ denotes the leader of this group; otherwise, if U_i refers to a single agent, then $\overline{U_i}$ simply denotes this agent, too. Finally, $[U_i, S_j]$ denotes the knowledge that the agents contained in U_i have about S_j; $[U_i, S_j]$ is called the knowledge context of U_i in S_j. (Note that $[U_i, S_j] \cap [U_k, S_j]$ may but need not be empty, and that $\bigcup_i [U_i, S_j]$ is not necessarily equal to S_j. Similarly, $[U_i, S_j] \cap [U_i, S_k]$ may but need not be empty; in particular, it may be the case that $[U_i, S_j] = [U_i, S_k]$, which means that a unit may be unable to distinguish between different environmental states.)

3.3. Competition and credit assignment

Competition and credit assignment is done analogously to the ACE algorithm. In each environmental state S_j the acting units compete for the right to become active. Each unit U_i calculates a bid B_i^j,

$$B_i^j = (\alpha + \beta) \cdot E_i^j, \tag{4}$$

where α is the risk factor, β is the noise factor, and E_i^j is $\overline{U_i}$'s estimate of U_i's usefulness dependent on the knowledge context $[U_i, S_j]$. Only the unit making the highest bid is allowed to become active and, in this way, transforms the actual into a new environmental state. This selection of a single winning unit corresponds to the selection of the action set that is carried out in the actual state.

Credit assignment again is done in a bucket brigade style. Let U_i be the winning unit in the actual state S_j, and let U_k be the winning unit in the preceding state S_l. $\overline{U_i}$ reduces its estimate E_i^j by the amount of the deterministic part of its bid B_i^j, and hands this amount back to $\overline{U_k}$. $\overline{U_k}$, in turn, adds the received amount to its estimate E_k^l. Additionally, if the activity of U_i leads to an external reinforcement R^{ext}, then $\overline{U_i}$ adds this reinforcement to its estimate E_i^j. All together, credit assignment involves the following adjustments:

$$E_i^j = E_i^j - \alpha \cdot E_i^j + R^{\text{ext}}, \tag{5}$$

$$E_k^l = E_k^l + \alpha \cdot E_i^j. \tag{6}$$

Based on Grefenstette's [11] convergence result it has been shown that under the DFG algorithm the estimates of successively active units tend to converge to an equilibrium level [22]. Moreover, it can be shown that every solution path learnt under the DFG algorithm is cycle-free in the sense that no environmental state is involved in this path more than one time.

3.4. Group development

The DFG algorithm distinguishes two contrary processes of group development, namely, group formation and group dissolution. Both processes largely depend on the past usefulness of the agents' and the groups' activities. In order to be able to decide about the formation of new groups and the dissolution of existing ones, each $\overline{U_i}$ calculates the mean values of its estimates over the previous episodes, where an episode is defined as the time interval between the receipts of two successive environmental reinforcements. More exactly, during each episode $\tau + 1$, $\overline{U_i}$ calculates the gliding mean value $M_i^j[\tau + 1]$ of its estimate E_i^j as

$$M_i^j[\tau+1] = \frac{1}{\nu} \cdot \sum_{T=\tau-\nu+1}^{\tau} E_i^j[T], \quad (7)$$

where ν is a constant called window size and $E_i^j[T]$ is E_i^j at the end of episode T. Both group formation and group dissolution proceed in dependence on the gliding mean values of the estimates.

In the following, let S_j be the actual environmental state and let $\tau+1$ be the actual episode. For each unit U_i that is potentially active in S_j, $\overline{U_i}$ decides that U_i is ready to cooperate and to form a new group with other units in the context $[U_i, S_j]$ if

$$M_i^j[\tau+1] \leqslant \sigma \cdot E_i^j[\tau-\nu], \quad (8)$$

where σ is a constant called cooperation factor which influences the units' readiness to form new groups. In words, according to this decision criterion a unit intends to cooperate in a specific context, if the estimate of its usefulness tends to increase too slowly, to stagnate or to decrease. The units that are ready to cooperate in their activity contexts form new groups according to the simple principles "stronger units choose their cooperation partners first" and "stronger units are preferred as cooperation partners". Formally, group formation is done as follows. Let $\mathcal{U}$ be the set of all units that are ready to cooperate:

until $\mathcal{U} = \emptyset$ do

- Let $U_i \in \mathcal{U}$ be the unit with $E_i^j = \max_l\{E_l^j : U_l \in \mathcal{U}\}$. Then $\overline{U_i}$ announces a "cooperation offer" to the other units.
- For each unit $U_l \in \mathcal{U}$ that is compatible with U_i, $\overline{U_l}$ sends a "cooperation response" to $\overline{U_i}$.
- Let $\mathcal{U}^{\text{resp}} \subseteq \mathcal{U}$ be the set of all responding units. Then $\overline{U_i}$ chooses the unit $U_k \in \mathcal{U}^{\text{resp}}$ with $E_k^j = \max_l\{E_l^j : U_l \in \mathcal{U}^{\text{resp}}\}$ as the cooperation partner of U_i. $\overline{U_i}$ and $\overline{U_k}$ form a new group G that has U_i and U_k as its members and either $\overline{U_i}$ or $\overline{U_k}$ as its leader. (The leader initializes the estimate of the new group's usefulness with a predefined value E^{init}.)
- $\mathcal{U} = \mathcal{U} \setminus \{U_i, U_k\}$.

(Note that each group has exactly two members; this could be easily extended towards multi-member groups by allowing $\overline{U_i}$ to chose several cooperation partners.) There are two things that have to be stressed. First, if two units form a new group, then this occurs within the frame of the units' knowledge contexts. Second, the process of group formation does not require an exchange of environmental information – neither among group members, nor among group members and leaders, nor among leaders of different groups. With that, cooperation and group formation is a highly context-sensitive process, but does not require an exchange of environmental information among the units.

For each group U_i that is potentially active in S_j, $\overline{U_i}$ decides that U_i has to dissolve in its members if

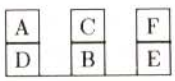
$$M_i^j[\tau+1] \leqslant \rho \cdot E^{\text{init}}, \quad (9)$$

where ρ is a constant called dissolution factor which influences the robustness of the existing groups, and E^{init} is the initialization value of the estimates. According to this criterion a group dissolves, if the estimate of its usefulness, averaged over the previous episodes, falls below a certain minimum.

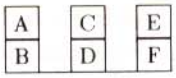
4. Experiments

4.1. Task domain and analysis

As a task domain the blocks world is chosen. This domain has been intensively studied in the fields of problem solving and planning, and is clear enough for doing the initial experimental studies in the unknown field of multi-agent learning. What has to be learnt by a given set of agents is to transform a start constellation of blocks into a goal constellation within a limited time interval. This paper summarizes the results on the task shown in Fig. 2. In this task, each agent is specialized in a specific action; for instance, agent A_1 is able to put block A on the bottom (symbolized by a $\perp$) and agent A_6 is able to put block D on block A. The precondition for applying an action $put(x, y)$ is that no other blocks are placed on x and y, i.e. x and y have to be empty. Each agent is assumed to have only minimal information about the environment: it only knows ("sees") whether the precondition of its action $put(x, y)$ is fulfilled. Because of this constraint, an agent is unable to distinguish between all different environmental states. In particular, an agent may be unable to distinguish between a state in which its action is useful and a state in which its action is not useful. (For instance, agent A_2 cannot distinguish between a "relevant state" in which block B is placed on



Start Constellation



Goal Constellation

Agents: A_1: $put(A, \perp)$ A_2: $put(A, B)$
A_3: $put(B, F)$ A_4: $put(C, \perp)$
A_5: $put(C, D)$ A_6: $put(D, A)$
A_7: $put(E, F)$ A_8: $put(F, \perp)$
A_9: $put(F, E)$

Limited Time Interval: at most 4 cycles

Fig. 2. A basic blocks world task.

the bottom and an "irrelevant state" in which block B is placed on block F. The fact that an action may be relevant in one state but irrelevant in another is sometimes called the Sussman's anomaly; see, e.g., [9].) Two actions are considered to be incompatible if their concurrent execution is not possible. Examples of sets of incompatible actions are $\{put(A, \perp), put(A, B)\}$ (i.e., a block cannot be placed on different positions at the same time), $\{put(B, F), put(E, F)\}$ (i.e., different blocks cannot be put on the same block), and $\{put(C, D), put(D, A)\}$ (i.e., a block cannot be put on a block which, at the same time, is put on another block). The transformation from the start into the goal constellation has to be done in at most four cycles.

There is one solution sequence (i.e., a sequence of action sets that transforms the start into the goal constellation) of length 2, 24 solution sequences of length 3, and 210 solution sequences of length 4. The solution sequence of length 2 is given by $\langle\{put(A, \perp), put(C, \perp), put(F, \perp)\}, \{put(A, B), put(C, D), put(E, F)\}\rangle$. Because every solution sequence contains at least 5 actions, this task cannot be solved by means of a sequential "one-action-per-cycle" approach. In the case of a random walk through the search space, the probability of finding the solution sequence of length 2 is less than 1 percent, the probability of finding a solution sequence of length 3 is less than 4 percent, and the probability of finding a solution sequence of length 4 is less than 5 percent. With that, the probability that a random sequence of maximal length four transforms the start into the goal constellation is less than 10 percent.

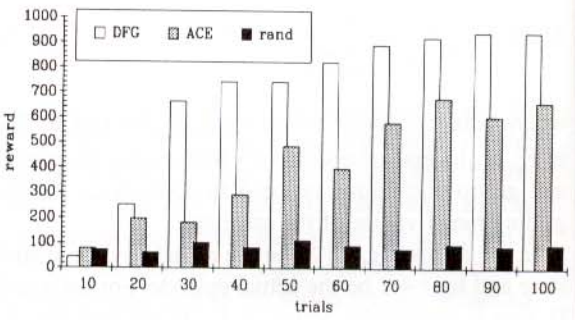


Fig. 3. Performance profiles.

4.2. Results

The experimental setting was as follows. A trial is defined as any sequence of at most four cycles that transforms the start into the goal constellation (successful trial), as well as any other sequence of exactly four cycles that transforms a start into a non-goal constellation. Learning proceeds by the repeated execution of trials. At the end of each trial the start constellation is restored, and the agents again try to solve the task. Additionally, only at the end of each successful trial a non-zero external reward R^{ext} is provided. Parameter setting: $E^{init} = R^{ext} = 1000$, $\alpha = 0.15$, $\beta \in [-\alpha/5 \ldots + \alpha/5]$ (randomly chosen for every bid), $\nu = 4$, $\sigma = 1 + 3\alpha$, and $\rho = 1 - \alpha$.

Fig. 3 shows the performance profiles of the ACE algorithm, the DFG algorithm, and a random-walk algorithm that randomly chooses an applicable set of compatible actions at the beginning of each trial. (Further experimental results are described in [21,22].) Each data value reflects the mean environmental reward per trial obtained during the previous 10 trials, averaged over 10 runs started with different random-number-generator seeds. There are several important observations. First, the learning performance of ACE and the DFG algorithm was significantly above the random performance level which is about 100. Both algorithms reached their highest average performance level after about 80 trials. This shows that under both algorithms the agents are able to learn useful sequences of action sets. Second, the DFG algorithm clearly performed better than the ACE algorithm; additionally, the DFG algorithm shows a more 'smooth" performance than the ACE algorithm. The reason for that is that under the ACE algorithm the usefulness of an action is estimated only in dependence on the envi-

ronmental state, whereas under the DFG algorithm it is estimated in dependence on both the environmental state and other actions carried out concurrently. As a consequence, the DFG algorithm does better cope with the fact that the usefulness of an action set does nothing say about the usefulness of a subset or a superset of this action set. Finally, as a consequence of the minimal-information constraint described above, the average performance of the ACE and the DFG algorithm remained below the maximum reward level (1000). This observation clearly shows the importance of information and information exchange mechanisms in the context of cooperating agents.

5. Critique and future work

This paper summarized work on learning and action coordination in multi-agent systems. Two algorithms, ACE and DFG, were described which enable multiple agents to learn useful action sequences.

The experimental results are very encouraging, but they also showed that the ACE and the DFG algorithm leave room for improvement. A limitation of both algorithms is that their learning success relies on an explicit exploration of a sufficient number of state-action pairs. This leads to an impractical amount of time required for learning, if the state and action spaces are too large and complex. There are two standard methods which can be used in order to cope with this kind of limitation: to endow the agents with the ability to generalize over the search space; and to endow the agents with the ability to built up an internal world model which can be used for look-ahead and planning activities. The ACE and the DFG algorithm are very general learning schemes which both allow an incorporation of these methods. This incorporation has to be a topic of future research.

Other important research topics are, for instance, the investigation of alternative group concepts (e.g., groups with variable bindings between the members) and alternative strategies for the dissolution and the formation of groups. In attacking these topics, it may well be useful to take also into consideration related work from other disciplines such as psychology (e.g., [12]) and economics (e.g., [1]). A long-term goal should be the development of a general framework of organizational adaptation in multi-agent systems.

References

[1] C. Argyris and D.A. Schön, *Organizational Learning* (Addison Wesley, Reading, MA, 1978).

[2] A.H. Bond and L. Gasser, eds., *Readings in Distributed Artificial Intelligence* (Morgan Kaufmann, San Mateo, CA, 1988).

[3] W. Brauer and D. Hernández, eds., *Verteilte Künstliche Intelligenz und kooperatives Arbeiten* (Springer, Berlin, 1991).

[4] P. Brazdil and S. Muggleton, Learning to relate terms in a multiple agent environment, in: Y. Kodratoff, ed., *Machine learning – EWSL-91* (Springer, Berlin, 1991) 424–439.

[5] M.S. Fox, An organizational view of distributed systems, *IEEE Trans. Sys. Man Cybernet.* SMC-11(1) (1981) 70–80.

[6] J. Galbraith, *Designing Complex Organizations* (Wiley, New York, 1973).

[7] L. Gasser and M.N. Huhns, eds., *Distributed Artificial Intelligence (Vol. 2)* (Pitman, London, 1989).

[8] M.P. Georgeff, Many agents are better than one, SRI International, Menlo Park, CA, Technical Report 417, 1987.

[9] M.L. Ginsberg, Possible worlds planning, in: M.P. Georgeff and A.L. Lansley, eds., *Reasoning About Actions and Plans – Proceedings of the 1986 Workshop* (Morgan Kaufmann, 1986) 213–243.

[10] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley, Reading, MA, 1989).

[11] J.J. Grefenstette, Credit assignment in rule discovery systems based on genetic algorithms, *Mach. Learning* 3 (1988) 225–245.

[12] R.A. Guzzo, *Improving Group Decision Making in Organizations – Approaches from Theory and Research* (Academic Press, New York, 1982).

[13] J.H. Holland, Properties of the bucket brigade algorithm, in: J.J. Grefenstette, ed., *Proc. 1st Internat. Conf. on Genetic Algorithms and Their Applications* (Lawrence Erlbaum, Hillsdale, NJ, 1985) 1–7.

[14] J.H. Holland, Escaping brittleness: The possibilities of general-purpose learning algorithms to parallel rule-based systems, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach (Vol. 2)* (Morgan Kaufmann, 1986) 593–632.

[15] M.N. Huhns, ed., *Distributed Artificial Intelligence* (Pitman, London, 1987).

[16] M.J. Shaw and A.B. Whinston, Learning and adaptatation in distributed artificial intelligence, in [7], 413–429.

[17] S.S. Sian, Adaptation based on cooperative learning in multi-agent systems, in: Y. Demazeau and J.-P. Müller, eds., *Decentralized AI (Vol. 2)* (Elsevier, Amsterdam, 1991).

[18] R. Sikora and M.J. Shaw, A distributed problem-solving approach to inductive learning, College of Commerce and Business Administration, University of Illinois at Urbana-Champaign. Faculty Working Paper 91-0109, 1991.

[19] M. Tan, Multi-agent reinforcement learning: Independent versus cooperative agents, in: *Proc. 10th Internat. Conf. on Machine Learning* (1993) 330–337.

[20] G. Weiß, Learning the goal relevance of actions in classifier systems, *Proc. of 10th European Conf. on Artificial Intelligence* (Wiley, Chichester, UK, 1992) 430–434.

[21] G. Weiß, Learning to coordinate actions in multi-agent systems, in: *Proc. 13th Internat. Joint Conf. on Artificial Intelligence* (Morgan Kaufmann, San Mateo, CA, 1993) 311–316.

[22] G. Weiß, Action selection and learning in multi-agent environments, in: *Proc. 2nd Internat. Conf. on Simulation of Adaptive Behavior* (MIT Press, Cambridge, 1993) 502–510.

[23] G. Weiß, *Distributed Machine Learning* (infix Verlag, Sankt Augustin, 1995).

Gerhard Weiß received his doctoral degree in computer science from the Technische Universität München in 1994. He is currently with the AI/Cognition research group at the computer science department at TUM. His work focuses on learning and adaptation processes in parallel and distributed systems, with an emphasis on neural networks, evolutionary systems, rule-based systems (especially classifier systems), and multi-agent systems. He is particularly interested in intelligent systems that combine the traditional knowledge-oriented and the novel behavior-oriented AI perspectives.