Action Selection and Learning in Multi–Agent Environments

Gerhard Weiß
Institut für Informatik, Technische Universität München
Arcisstr. 21, 8000 München 2, Germany
weissg@informatik.tu-muenchen.de

Abstract. This paper focusses on reactive multi–agent systems in which *(i)* each agent only knows a specific part of the environment, *(ii)* each agent is specialized in a specific action, and *(iii)* actions of different agents can be incompatible. The central problem addressed is how several agents can collectively adapt to their environment by learning to generate a sequence of action sets that solves an environmental task.

The contents and organization of the paper are as follows. Section 1 briefly motivates the topic of action selection and learning in multi–agent systems. Section 2 introduces a new algorithm called DFG (for "Dissolution and Formation of Groups") for the reinforcement learning of appropriate sequences of groups of concurrently active agents. Section 3 provides theoretical and experimental results on the the DFG algorithm. Section 4 concludes with a brief summary and an outlook on future work.

Keywords: Multi–agent systems, action selection, learning, group development

1. Motivation

The last several years have witnessed a rapidly growing interest in multi–agent systems, that is, in systems that are composed of a number of agents being able to interact and differing from each other in their skills and their knowledge about the environment. Nowadays these systems establish a major research subject in Distributed Artificial Intelligence (e.g. Bond & Gasser, 1988; Huhns, 1987; Gasser & Huhns, 1989; Brauer & Hernandez, 1991). The growing interest is largely founded on the insight that many real–world problems are best modelled using a set of interacting agents instead of a single agent. In particular, multi–agent modelling allows to cope with natural constraints like the limited processing power of a single agent and to profit from inherent properties of distributed systems like robustness, parallelism and scalability.

A great part of the research on multi–agent systems has focussed on the issues of cooperation and communication in the context of distributed problem solving (see e.g. Decker, 1987; Durfee, Lesser & Corkill, 1989). Against that, only little work has been done on learning and adaptation; see (Weiß, 1992b) for an overview of actual literature. This is in contradiction to the common agreement that there are two important reasons for studying learning and adaptation processes in multi–agent systems:

- to be able to endow artificial multi–agent systems (e.g. a system of interacting robots) with the ability to learn and to improve their performance, and
- to get a better understanding of the learning and adaptation processes in natural multi–agent systems (e.g. human groups or societies).

There is a great variety in the multi–agent systems studied in the field of Distributed Artificial Intelligence; see (Huhns, 1987, foreword) for a classifying overview. This paper deals with reactive multi–agent systems in which

- each agent has only local information about the environment (i.e. no agent is "omniscient"),
- each agent can only carry out a specific action (i.e. no agent is "omnipotent"), and
- the actions of different agents can be incompatible.

("Reactive" means that the behavior and the environment of the system are strongly coupled, i.e. that there is a continuous interaction between the system and its environment.) The central problem addressed is how the agents in such a multi–agent system can collectively adapt to their environment by learning to generate a sequence of action sets that solves an environmental task.

2. The DFG Algorithm

2.1. The Basic Working Method

The DFG algorithm (DFG stands for "Dissolution and Formation of Groups") is designed to solve the problem of learning appropriate sequences of action sets in reactive multi–agent systems.[1] Its basic working method can be described as follows. The DFG algorithm distinguishes between single agents and groups of compatible agents as the acting units in a multi–agent system. Collective learning and adaptation encompasses two

[1]The DFG algorithm as it is described in this paper requires that each agent is specialized in only one action; however, it can be easily extended to systems in which each agent can carry out several actions.

interrelated processes. First, *credit assignment*, that is, the process of estimating or approximating the goal relevance of the agents and the groups in different environmental states. Second, *group development*, that is, the process of dissolving and forming groups in dependence on their goal relevance. In each environmental state *action selection* is realized by arranging a competition between the agents and groups on the basis of their goal relevance. Only the winner of this competition is allowed to become active and to change the environment. All together, an appropriate sequence of action sets is learnt by the repeated execution of the following *working cycle* of the DFG algorithm:

1. [Check for activity] The agents and groups check whether they could become active in the actual environmental state.
2. [Group development] Existing groups that do not contribute to the goal attainment dissolve, and agents and groups that are willing to cooperate form new groups.
3. [Action selection] Based on their goal relevance in the actual state, the agents and groups compete with each other for the right to carry out their actions. The winner becomes active and, in this way, transforms the actual into a new environmental state.
4. [Credit assignment] The agents and groups adjust the estimates of their goal relevances by assigning credit or blame to each other.

The next subsections describe the steps of this working cycle in detail.

2.2. Agents, Groups, and Activity

The DFG algorithm distinguishes between two types of acting units in a multi–agent system: single agents and groups of compatible agents. An **agent** is composed of a *sensor component*, a *motor component*, a *knowledge base*, and a *learning element*. The sensor and the motor component enable the agent to interact with other agents and its environment (e.g. by communicating with other agents, by receiving visual information about the current environmental state, or by carrying out an action that changes the environment). The knowledge base contains the agent's knowledge (e.g. other agents or its environment). The learning element is responsible for the modifications in the knowledge base that improve the agent's interaction abilities (e.g. by refining its motor skills or by increasing the efficiency of its cooperation with other agents). Figure 1 illustrates this view of a single agent. An agent is restricted in a twofold manner. First, because of limitations imposed on its sensor abilities, an agent knows only a part of the environment; and second, because of limitations imposed on its motor component, an agent is specialized in a specific action. As a consequence of these restrictions, different agents may know different aspects of the environment, and they may be specialized in different actions.

A **group** consists 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 another group. This recursive definition is rather general and covers both low and high structured groups; see figure 2 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 autonomous 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.

There are three concepts that are elementary to the learning and adaptation processes induced by the DFG algorithm: the **(potential) activity**, the **activity context**, and the **autonomy** of a unit. An agent is said to be active (potentially active) simply if it carries out (could carry out) its action. The activity of a group results from the concerted activity of the group members, and a group is said to be active (potentially active) if all its members

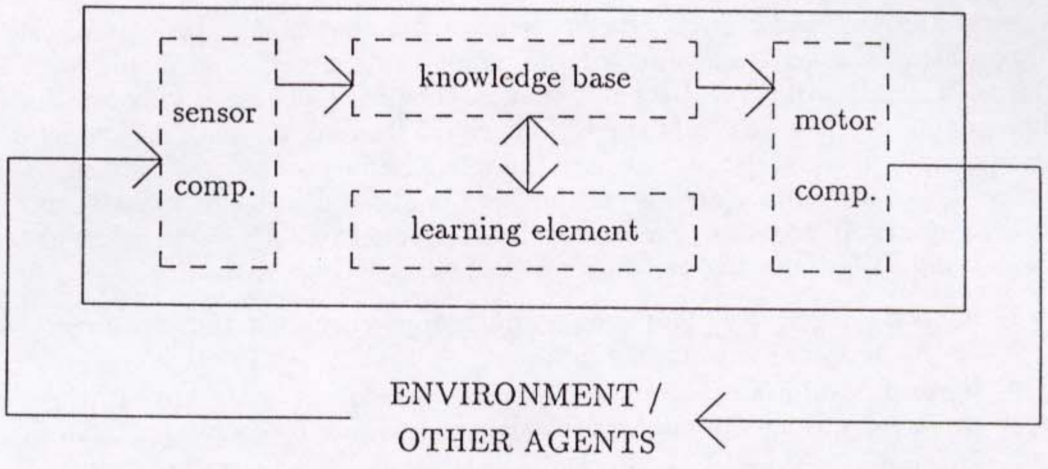


Figure 1: An agent and its components. (See text for details.)

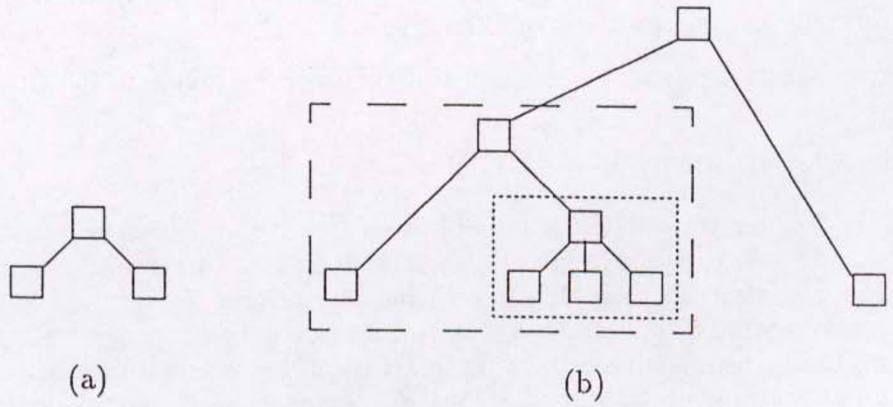


Figure 2: Structural organization of groups. A group is defined as a set of agents (□) that is structured by leader–member relations (/ , \). *(a)* shows the most simple group consisting of a leader and two members each being a single agent. *(b)* shows a a more complicated group having two members, one being a group (dashed box) and the other being a single agent. The "dashed group" again has a group (dotted box) and a single agent as its members, where the "dotted group" has three members each being a single agent.

are so. The activity context of an agent in an environmental state is given by the agent's knowledge about this state, and the activity context of a group is given by the activity contexts of all group members. Finally, an agent as well as a group is said to be autonomous, if it is not a member of a potentially active group. The next sections describe the roles of these concepts in detail.

The following **notation** is used throughout the rest of this paper. S_i refers to an environmental state. U_i refers to a unit of the multi–agent system, where a unit is either an agent A_j or a group G_j. If U_i is a member of G_j then this is symbolized by $U_i \in G_j$. $[S_i]_j$ refers to the part of S_i that is known to the agent A_j. $[U_i, S_j]$ denotes the activity context of the unit U_i in the environmental state S_j; this context can be formally described by

$$[U_i, S_j] = \begin{cases} [S_j]_k & \text{if} \quad U_i = A_k \\ \bigcup_{U_l \in G_k} [U_l, S_j] & \text{if} \quad U_i = G_k \end{cases} .$$

(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.) Finally, $\overline{U_i}$ is defined as

$$\overline{U_i} = \begin{cases} A_j & \text{if} \quad U_i = A_j \\ \text{the leader of } G_j & \text{if} \quad U_i = G_j \end{cases} .$$

This is only a "technical definition" that allows to give an efficient description of the DFG algorithm.

2.3. Action Selection and Credit Assignment

Action selection and credit assignment base on the action–oriented variant (Weiß, 1991, 1992a) of Holland's (1985, 1986) *bucket brigade model* of learning in classifier systems. Formally, the DFG algorithm realizes action selection and credit assignment as follows. Let S_j be the actual environmental state. For each unit U_i that is potentially active and autonomous in S_j, $\overline{U_i}$ makes a bid B_i^j for U_i's right to become active. (The autonomy condition ensures that there is no competition between a group and its members; see below.) This bid is calculated by

$$B_i^j = \alpha \cdot E_i^j + \beta \cdot E_i^j \quad , \tag{1}$$

where α is a small constant called *risk factor*, β is a small random number called *noise factor*, and E_i^j is a scalar value called *estimate*. The $\alpha \cdot E_i^j$ is called the *deterministic part* and the $\beta \cdot E_i^j$ is called the *stochastic part* of B_i^j. The deterministic part of B_i^j is the fraction of E_i^j that $\overline{U_i}$ is willing to risk for U_i's right to become active in S_j. The stochastic part of B_i^j introduces noise into the bidding process in order to avoid getting stuck into local learning minima.[2] The E_i^j is $\overline{U_i}$'s estimate of U_i's goal relevance in the activity context $[U_i, S_j]$. (The first time a unit U_i is potentially active in a state S_j, $\overline{U_i}$ initializes E_i^j with a predefined value E^{init}; E^{init} is called the *initialization value* of the estimates.) On the basis of their bids, a competition runs between the potentially active units and the unit U_i^j with

$$B_i^j = \max_k \{B_k^j\} \tag{2}$$

is allowed to become active. This selection of the winning unit corresponds to the selection of the set of actions that are carried out in the actual environmental state.

Credit assignment happens by means of a collective adjustment of the estimates E_i^j. 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 (i.e. U_k transforms S_l to S_j). Then $\overline{U_i}$ reduces its estimate E_i^j by the amount of the deterministic part of its bid B_i^j, i.e.

$$E_i^j = E_i^j - \alpha \cdot E_i^j \quad , \tag{3}$$

[2]In the literature on classifier systems different methods for introducing noise in the bidding process have been proposed; see e.g. (Goldberg, 1989; Riolo, 1989).

and hands this amount back to $\overline{U_k}$. $\overline{U_k}$, in turn, adds the received amount to its estimate E_k^l of U_k's goal relevance in the context $[U_k, S_l]$, i.e.

$$E_k^l = E_k^l + \alpha \cdot E_i^j \quad . \tag{4}$$

(The current winner pays for the previlege of being active and rewards the preceding winner for appropriately setting up the environment.) Additionally, if the activity of U_i leads to a new state in which a reinforcement is received from the environment, then $\overline{U_i}$ adds this reinforcement to its estimate E_i^j.

The effects of this bucket–brigade–type adjustment of the estimates can be informally described as follows. In a sequence of active units, each unit pays a certain amount to its direct predecessor and receives a certain amount from its direct successor or from the environment. This has two major effects. First, the estimate of a unit's goal relevance increases (decreases), if the unit pays less (more) than it receives. Second, a change (i.e. an increase or decrease) in an estimate of an unit's goal relevance is propagated, over time, from this unit backwards through the chain of its predecessors. These effects lead to a *stabilization* of a sequence of active units, if the last unit of the sequence regularly attains payoff, and they result in a *disintegration* of a sequence, if its last unit does not.

2.4. Group Development

According to the DFG algorithm, group development includes two contrary processes: group formation and group dissolution. 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.[3] 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

$$M_i^j[\tau + 1] = \frac{1}{\nu} \cdot \sum_{T=\tau-\nu+1}^{\tau} E_i^j[T] \quad , \tag{5}$$

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.

Group formation. 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 and autonomous 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

$$M_i^j[\tau + 1] \leq \sigma \cdot E_i^j[\tau - \nu] \quad , \tag{6}$$

where σ is a constant called *cooperation factor* which influences the units' readiness to form new groups. (The autonomy condition ensures that a unit does not cooperate if it is already a member of a potentially active group.) In words, according to this decision criterion a unit intends to cooperate in a specific context, if the estimate of the unit's goal relevance tends to increase too slowly, to stagnate or even to decrease. The units that are ready to cooperate in their activity contexts form new groups as follows. Let $\mathcal{U}$ be the set of all units that are ready to cooperate:

[3] Compared with a cycle–based calculation, an episode–based calculation enables a more balanced group development.

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 $\mathcal{U}^{resp} \subseteq \mathcal{U}$ be the set of all responding units. Then $\overline{U_i}$ chooses the unit $U_k \in \mathcal{U}^{resp}$ with $E_k^j = \max_l\{E_l^j : U_l \in \mathcal{U}^{resp}\}$ as the cooperation partner of U_i. $\overline{U_i}$ and $\overline{U_k}$ form a new group that has U_i and U_k as its members and either $\overline{U_i}$ or $\overline{U_k}$ as its leader.[4]
- $\mathcal{U} = \mathcal{U} \setminus \{U_i, U_k\}$.

It has to be stressed that the formation of each new group occurs within the frame of the group members' activity contexts. If $\overline{U_i}$ and $\overline{U_k}$ form a new group G in the state S_j, then they do so because they are ready to cooperate *in their activity contexts* $[U_i, S_j]$ and $[U_k, S_j]$, respectively. The group G is potentially active in every state S_l with $[U_i, S_l] = [U_i, S_j]$ and $[U_k, S_l] = [U_k, S_j]$. (In each such state $\overline{U_i}$ and $\overline{U_k}$ inform the leader of G that they are potentially active, and then the leader for its part declares G as being potentially active.) With that, cooperation and group formation is a highly context–sensitive process, and each group is strictly attached to a specific activity context.

Group dissolution. Again let S_j be the actual environmental state and $\tau+1$ the actual episode. For each group G_i that is potentially active and autonomous in S_j, $\overline{G_i}$ decides that G_i has to dissolve in its members if

$$M_i^j[\tau+1] \leq \rho \cdot E^{init} \quad , \tag{7}$$

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 (see 2.3). (Here the autonomy condition ensures that a group does not dissolve as long as it is a member of another group.) According to this decision criterion a group dissolves, if the estimate of its goal relevance, averaged over the previous episodes, falls below a certain minimum level.

3. Analysis

3.1. Learning Convergence

The DFG algorithm aims at generating appropriate sequences of action sets. Therefore an important question is how this algorithm changes the estimates of the goal relevances of units that are successively active. An answer to this question can be found by extending Grefenstette's (1988) convergence result on the bucket brigade algorithm to the DFG algorithm. This leads to the following

Proposition. Consider a collection of units $U_{i_1}, \ldots, U_{i_n}$ in which

(i) each unit U_{i_k}, $1 \leq k < n$, is coupled with unit $U_{i_{k+1}}$ in the sense that each activity of U_{i_k} in the context $[U_{i_k}, S_{j_k}]$ is followed (in the next cycle) by the activity of $U_{i_{k+1}}$ in the context $[U_{i_{k+1}}, S_{j_{k+1}}]$, and

[4]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.

(ii) the only external reward (if any) is received by U_{i_n}.

If $E_{i_n}^{j_n}$ converges to a constant value E^*, then $E_{i_k}^{j_k}$, $1 \leq k < n$, also converges to E^*.

Proof. If U_{i_k}, $1 \leq k < n$, is active during the cycle t and $U_{i_{k+1}}$ is active during the cycle $t+1$, then

$$E_{i_k}^{j_k}[t+2] = E_{i_k}^{j_k}[t] - \alpha \cdot E_{i_k}^{j_k}[t] + \alpha \cdot E_{i_{k+1}}^{j_{k+1}}[t+1] \quad ,$$

where $E_{i_k}^{j_k}[t+2]$ is $E_{i_k}^{j_k}$ at the beginning of cycle $t+2$. This yields

$$E_{i_k}^{j_k}[\overline{s}+2] = (1-\alpha)^n \cdot E^{init} + \sum_{r=1}^{s} \alpha \cdot (1-\alpha)^{s-r} \cdot E_{i_{k+1}}^{j_{k+1}}[\overline{r}+1] \quad ,$$

where $s \in \mathsf{N}$ and $\overline{s}$ is defined as $\overline{s} = t$ iff the s^{th} activity of U_{i_k} in the context $[U_{i_k}, S_{j_k}]$ occurred during the cycle t. Now suppose that $E_{i_{k+1}}^{j_{k+1}}$ converges to E^*. Then

$$\lim_{t\to\infty} E_{i_k}^{j_k}[t] = lim_{s\to\infty} E_{i_k}^{j_k}[\overline{s}+2] = E^* \quad ,$$

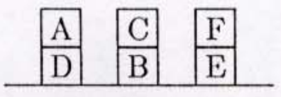
i.e. $E_{i_k}^{j_k}[t]$ also converges to E^*. ■

This result shows that under the DFG algorithm the estimates of the goal relevance of successively active units tend to converge to an equilibrium level. Under equilibrium conditions, a unit pays to its predecessor the same amount than it receives from its successor. With that, the estimates serve to predict the internal rewards.

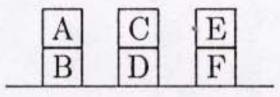
3.2. Experimental Results

The blocks world is chosen as a task domain. This domain is well known in Artificial Intelligence and has been intensively studied in the fields of problem solving and planning, and it is clear enough for 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 figure 3. 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 information constraint, an agent is unable to distinguish between all different environmental states.[5] In particular, an agent may be unable to distinguish between a state in which its action is relevant and a state in which its action is not relevant to goal attainment. For instance, agent A_2 cannot distinguish between a "relevant state" in which block B is placed on the bottom and an "irrelevant state" in which block B is placed on block F; similarly, A_5 cannot distinguish between a state in which block D is placed on the bottom and a state in which D is placed on A. (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. (Ginsberg, 1986).) Two actions are considered to be *incompatible* if their concurrent execution is not possible. Formally, two actions, $put(x, y)$ and $put(u, v)$, are incompatible if $x \in \{u, v\}$ or $u \in \{x, y\}$ or $y = v \neq \perp$. 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*.

[5]Instead, each agent only distinguishes between the class of states in which its action is applicable and the class of states in which its action is not applicable. A less strong restriction is used in (Weiß, 1992c).



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

Figure 3: Blocks world task. (See text for details.)

An analysis of the search space of this task shows that 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 $\langle\{put(A,\perp), put(C,\perp), put(F,\perp)\}, \{put(A,B), put(C,D), put(E,F)\}\rangle$. There is no solution sequence containing less than 5 actions; consequently, a sequential "one–action–per–cycle" approach would require at least 5 cycles to solve the task. In the case of a random walk through the search space (i.e. in the case of randomly choosing, in each environmental state, an applicable set of compatible actions), 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 at most four action sets transforms the start into the goal constellation is less than 10 percent!

The experimental setting is 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), and $\nu = 4$.

Figure 4 shows the learning results of the DFG algorithm for $\sigma = 1 + 3\alpha$ and $\rho = 1 - \alpha$ (*DFG1*), $\sigma = 1 + \alpha$ and $\rho = 1 - \alpha$ (*DFG2*), and for $\sigma = 1 + \alpha$ and $\rho = 1 - 3\alpha$ (*DFG3*).[6] 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. The learning performance of both the DFG1, the DFG2 and the DFG3 variant is clearly above the random performance level (which is less than 100, see above). The highest performance level of the DFG1, the DFG2 and the DFG3 variant was 950, 800, and 700, respectively. The DFG1 performed better than the DFG2 which performed better than the DFG3 variant. The reason for these performance differences is that the DFG1 uses a less strict criterion for the formation of groups than the DFG2, and the DFG2 uses a less strict criterion for the dissolution of groups than the DFG3 (cf. the σ and ρ values). As a

[6]Similar results have been obtained in the experiments that we carried out with various other agent and block constellations; additionally, in all these experiments the DFG algorithm turned out to be robust over a broad range of parameter settings.

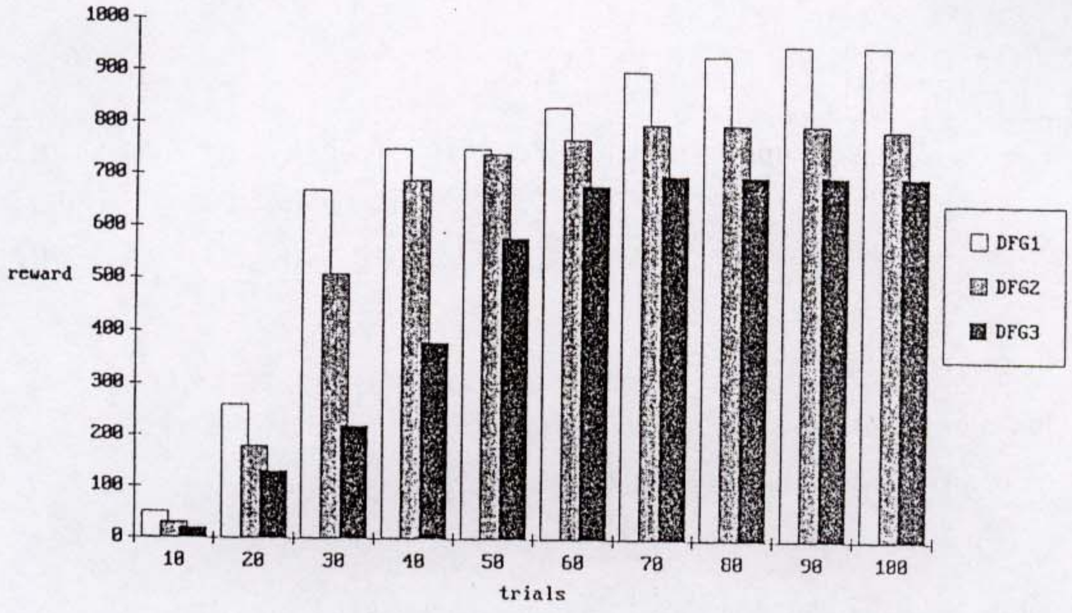


Figure 4: Learning performance of the DFG algorithm. (See text for details.)

Avg. number per run	DFG1	DFG2	DFG3
created groups	27.5	19.1	20.2
dissolved groups	5.7	6.0	3.8

Table 1: Group development. (See text for details.)

consequence, under the DFG1 more groups were created than under the DFG2, and under the DFG2 more groups dissolved than under the DFG3. In other words, the DFG1 turned out to be more flexible than the DFG2, and the DFG2 turned out to be more flexible than the DFG1. This is illustrated by table 1 which shows, for the three variants, the number of created and dissolved groups per run, averaged over the ten runs. (Note that the ratio of created and dissolved groups under the DFG2 is smaller than under the DFG1 and the DFG3.)

These results clearly demonstrate the ability of the DFG algorithm to produce both useful and stable sequences of active units, although the individual agents do only have local information about their environment.

4. Conclusion

This paper addressed the problem of action selection and reinforcement learning in reactive multi-agent systems. A new algorithm called DFG was presented that implements learning and adaptation on the basis of credit assignment and group development. According to this algorithm, several agents collectively adapt to their environment by learning to coordinate their actions and to generate appropriate sequences of action sets. Theoretical and experimental results were provided that show the learning abilities of the DFG algorithm.

Further investigations are required for a complete understanding of the approach described in this paper. Important topics for future research are, for instance, the development of alternative criteria and strategies for group dissolution and group formation, the extension of the concept of a group (e.g. towards multi-member groups with different bindings between the members, or towards groups of successively active agents), and the detailed analysis of the mutual influence of the critical parameters (e.g. window size, cooperation and dissolution factor).

Acknowledgements

The author would like to thank Martin Eldracher for fruitful discussions on learning and group development in multi-agent systems.

References

Bond, A. H., & Gasser, L. (Eds.). (1988). *Readings in distributed artificial intelligence*. San Mateo, CA: Morgan Kaufmann.

Brauer, W., & Hernández, D. (Eds.). (1991). *Verteilte Künstliche Intelligenz und kooperatives Arbeiten*. Springer.

Decker, K. S. (1987). Distributed problem-solving techniques: a survey. *IEEE Trans. on Systems, Man, and Cybernetics, SMC-17(5)*, 729–740.

Durfee, E. H., Lesser, V. R., & Corkill, D. D. (1989). Trends in cooperative distributed problem solving. *IEEE Trans. on Knowledge and Data Engineering, 1(1)*, 63–83.

Gasser, L., & Huhns, M. N. (Eds.). (1989). *Distributed artificial intelligence* (Vol. 2). Pitman.

Ginsberg, M. L. (1986). Possible worlds planning. In M. P. Georgeff, & A. L. Lansley (Eds.), *Reasoning about actions and plans – Proceedings of the 1986 workshop* (pp. 213–243). Timberline, Oregon: Morgan Kaufmann.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Grefenstette, J. J. (1988). Credit assignment in rule discovery systems based on genetic algorithms. In *Machine Learning*, 3, 225–245.

Holland, J. H. (1985). Properties of the bucket brigade algorithm. In J. J. Grefenstette (Ed.), *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (pp. 1–7). Pittsburgh, PA: Lawrence Erlbaum.

Holland, J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2, pp. 593–632). Los Altos, CA: Morgan Kaufmann.

Huhns, M. N. (Ed.). (1987). *Distributed artificial intelligence*. Pitman.

Riolo, R. L. (1989). The emergence of coupled sequences of classifiers. In *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 256–264). Fairfax, VA: Morgan Kaufmann.

Weiß, G. (1991). *The action-oriented bucket brigade*. Technical Report FKI-156-91. Institut für Informatik, Technische Universität München.

Weiß, G. (1992a). Learning the Goal Relevance of Actions in Classifier Systems. In B. Neumann (Ed.), *Proceedings of the 10th European Conference on Artificial Intelligence* (pp. 430–434). Vienna, Austria: Wiley.

Weiß, G. (1992b). *Collective learning and action coordination*. Technical Report FKI-166-92. Institut für Informatik, Technische Universität München.

Weiß, G. (1992c). *Learning to coordinate actions in multi-agent systems*. Internal Working Paper (submitted, available from the author).