J. EXPT. THEOR. ARTIF. INTELL. 8(1996)43–62.

An action-oriented perspective of learning in classifier systems

GERHARD WEIß

Institut für Informatik (H2), Technische Universität München, D-80290 München, Germany
email: weissg@informatik.tu-muenchen.de

Abstract. Classifier systems constitute a general model of low-level rule-based systems that are capable of environmental interaction and learning. A central characteristic and drawback of the traditional approaches to learning in such systems is that they exclusively work on the rule level, without taking into consideration that the individual rules possess a very complex activity behaviour. This article investigates an alternative, action-oriented perspective of learning in classifier systems which does not suffer from this drawback. According to this perspective learning is realized on the finer action level instead of the coarser rule level. Comparative theoretical and experimental results are presented that show the advantages of the action-oriented over the traditional perspective.

Keywords: classifier systems, bucket brigade, profit sharing plan, action-oriented learning

Received 30 March 1994; accepted 13 June 1994

1. Introduction

1.1. *Classifier systems*

The foundations for classifier systems (CSs) were laid by Holland (1975) within the frame of his theoretical study of genetic algorithms, and the first CS was introduced by Holland and Reitman (1978). Since then many different types of CSs have been described in the literature (for an overview see Goldberg 1989, Booker *et al.* 1989, Wilson and Goldberg 1989). Generally, CSs are parallel, message-passing, rule-based systems that are capable of environmental interaction and of learning through credit assignment and rule discovery. They are made up of four *structural parts*:

- An input interface which consists of at least one detector providing information about the environment in the form messages.
- An output interface which consists of at least one effector enabling the system to interact with the environment.
- A classifier list $\mathscr{C}$ which consists of a number of rules called classifiers. Each classifier C_i is of the form $Cond_{i1}, \ldots, Cond_{in}/Act_i$ where the $Cond_{ij}$ and the Act_i are strings of fixed length L over $\{0, 1, \#\}$ called condition string and action string, respectively. Associated with each classifier is a specific value called its strength.
- A message list $\mathscr{M}$ which contains the messages sent by the detectors and the classifiers, where each message m is a string of fixed length L over $\{0, 1\}$.

A classifier C_i is said to be satisfied, if there is tuple $T = (m_1, \ldots, m_n) \in \mathcal{M}^n$ of messages such that for each $j \in \{1, \ldots, n\}$ $Cond_{ij}$ is matched by m_j, where matching is done position by position and '#' acts as a 'don't care' symbol. T implies a specific action a in form of a new message according to the following simple mechanism. Let $\alpha_k \in \{0, 1, \#\}$ be the symbol on position k of the action string Act_i, and let $\mu_k \in \{0, 1\}$ be the symbol on position k of the message that matches $Cond_{i1}$ (the so-called *pass-through condition*). Then the action a implied by T has on position k the symbol α_k, if $\alpha_k = 1$ or $\alpha_k = 0$, and the symbol μ_k, if $\alpha_k = \#(k = 1, \ldots, L)$. The generation of new messages is illustrated in Figure 1. As this figure shows, different message tuples can imply different messages. With that, a classifier is capable of a highly differentiated activity behaviour.

CSs are active on three *functional levels*: the performance level, including activities like environmental interaction and message processing; the credit-assignment level, including the activity of learning by strength modification; and the discovery level, including the activity of learning by discovering new classifiers. The activities at these three levels are concerted in the following *major execution cycle*:

1. Activation of the input interface: the detector messages are added to the message list.
2. Activation of the classifier list: a competition runs between the satisfied classifiers and only the winners are allowed to produce new messages.
3. Activation of the output interface: the system interacts with its environment in dependence on the new messages.
4. Credit assignment: strength-update rules are applied to adjust the classifier strengths such that they reflect the classifiers' relevance to goal attainment.
5. Rule discovery: a genetic algorithm is applied in order to create new (more useful) and to replace old (useless) classifiers.
6. Message list updating: the contents of the message list is replaced by the new messages.

The repeated execution of this cycle makes up the overall activity of a CS.

1.2. *The traditional perspective of learning*

There are two different learning schemes for credit assignment in CSs that have been proposed in the literature: the *bucket brigade* (BB for short; e.g. Booker 1982, Goldberg 1983, Holland 1985, Riolo 1988) and the *profit-sharing plan* (PSP for short; Holland and Reitman 1978, Grefenstette 1988). The fundamental difference between these two schemes is the following: the BB is an *incremental* learning algorithm

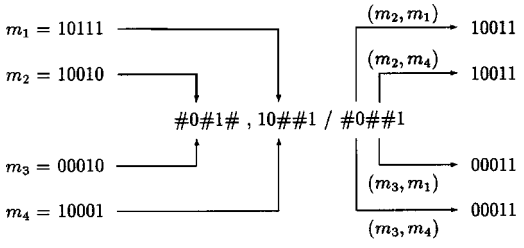

Figure 1. Generation of new messages.

according to which the strengths are updated each cycle; against that, the PSP is an *episodical* learning algorithm according to which the strengths are only updated at the end of each episode (epoche, trial), where an episode is defined as the time interval between the receipts of two successive external rewards.[1]

The BB in its standard form works as follows.[2] If a competition runs, each satisfied classifier makes a single, action-unspecific bid Bid_i,

$$Bid_i = k \cdot Str_i \cdot Spec_i \quad (\text{'BB without support'}) \tag{1}$$

or

$$Bid_i = k \cdot Str_i \cdot Spec_i \cdot Sup_i \quad (\text{'BB with support'}) \tag{2}$$

where b is a small constant called risk factor, Str_i is C_i's strength, $Spec_i$ is C_i's specifity (defined as the number of 0s and 1s in the condition part of C_i divided by the length thereof), and

$$Sup_i = \sum_{m:\exists T \in \mathscr{T}_i (m \in T)} Int(m) \tag{3}$$

is the *support* for C_i, where $\mathscr{T}_i$ is the set of all message tuples that satisfy C_i and $Int(m)$, the *intensity* of the message m, is a value equal to the bid according to which m was sent.

The probability that a bidding classifier wins the competition is given by

$$\frac{Bid_i}{\sum_{C_k \in \mathscr{S}} Bid_k} \tag{4}$$

where $\mathscr{S}$ is the set of all satisfied classifiers. A winning classifier produces the messages implied by the message tuples that satisfy it, and pays its bid to its predecessors, that is, to the classifiers that sent the messages contained in these tuples. Formally, this leads to the following strength modifications:

$$Str_i = Str_i - Bid_i \tag{5}$$

and

$$Str_k = Str_k + \frac{Bid_i}{|\mathscr{P}_i|} \qquad \forall C_k \in \mathscr{P}_i \tag{6}$$

where

$$\mathscr{P}_i = \{C_k \in \mathscr{C} : \exists T \in \mathscr{T}_i \exists m \in T (C_k \text{ sent } m)\} \tag{7}$$

is the set of the predecessors of C_i. Additionally, if an external reward is received from the environment, then it is equally distributed among the classifiers that sent the effector-activating messages.

The standard PSP works as follows.[3] A satisfied classifier C_i competes for its right to fire by making the bid

$$Bid_i = b \cdot Str_i \tag{8}$$

where b is the risk factor and Str_i is the strength of C_i. Analogous to the BB, a winning classifier sends all messages that are implied by the message tuples satisfying it. At the end of each episode (i.e. whenever an external reward Ext is received), the strength Str_i of each classifier C_i that was active at least one times during the episode is modified according to

$$Str_i = Str_i - Bid_i + b \cdot Ext \tag{9}$$

Approaches to a synthesis of BB-type and PSP-type learning are Grefenstette's (1988) system called RUDI and Weiß's (1994) hierarchical chunking algorithm.

1.3. *An action-oriented perspective*

A central characteristic and a basic drawback of the traditional learning approaches is that their bidding, firing and strength adjustment processes exclusively work on the *classifier level*, without taking into consideration that the individual classifiers possess a very complex activity behaviour.[4] In order to cope with this problem, in previous work it has been proposed to consider learning in CSs from an alternative, action-oriented perspective (Weiß 1991, 1992). This perspective is well known in the field of artificial neural networks and recently has been also applied to multi-agent systems (Weiß 1993a, 1993b). Here we present the details of the action-oriented perspective of learning in CSs. Three variants of the standard BB and the standard PSP are described which strictly operate on the *action level* in the sense that their bidding, firing and strength adjustment processes are guided by the actions of the classifiers (Section 2). These variants—the *action-oriented bucket brigade 1* (ABB1), the *action-oriented bucket brigade 2* (ABB2), and the *action-oriented profit-sharing plan* (APSP)—explicitly take into consideration that a classifier is capable of carrying out different actions and that different actions carried out by a classifier can have different relevance to goal attainment. In other words, the ABB1, the ABB2 and the APSP aim at learning the goal relevance of the actions instead of the goal relevance of the classifiers. Since the goal relevance learnt is also used for guiding the genetics-based discovery of new rules, rule discovery itself works on the principle of action orientation. A theoretical and experimental comparison of the traditional and the action-oriented perspective is provided (Section 3). The article closes with a summary and with suggestions on future research directions (Section 4).

2. Action-oriented learning

2.1. *The algorithm ABB1*

The central quantity underlying the action-oriented perspective of learning in CSs is that of the goal relevance of an action. According to the ABB1, this quantity is estimated as follows. Let $\mathscr{T}_i$ be the set of all actual message tuples T that satisfy the classifier C_i. First, C_i determines the set $\mathscr{A}_i^{\text{poss}} = \{a_i^1, \ldots, a_i^{k_i}\}$ ($k_i \in \mathbf{N}$) of all its possible actions (i.e. of all actions that could be carried out by it). This set is specified by

$$\mathscr{A}_i^{\text{poss}} = \{a : \exists\, T \in \mathscr{T}_i (T \text{ implies } a)\}. \tag{10}$$

Then C_i determines for each of its possible actions $a_i^j \in \mathscr{A}_i^{\text{poss}}$ the set $\mathscr{M}_i^j$ of *action-relevant messages* (i.e. of all messages that make a_i^j possible). $\mathscr{M}_i^j$ is given by

$$\mathscr{M}_i^j = \{m : \exists\, T \in \mathscr{T}_i (T \text{ implies } a_i^j \wedge m \in T)\} \tag{11}$$

(Each $m \in \mathscr{M}_i^j$ corresponds to a specific action that was carried out by some classifier during the previous cycle or to a detector message.) Finally, C_i calculates for each $a_i^j \in \mathscr{A}_i^{\text{poss}}$ the *estimated goal relevance* Est_i^j by

$$Est_i^j = \sum_{m \in \mathscr{M}_i^j} Int(m) \tag{12}$$

where $Int(m)$ is the intensity of the message m.

Table 1

Classifier list $\mathscr{C}$					
$C_1 =$	#1#1#	,	#1011	/	01#01
$C_2 =$	#111#	,	010#0	/	#0011
$C_3 =$	0#0##	,	0#111	/	0##0#
$C_4 =$	0##11	,	#1#00	/	110#1
...					

If the total number of possible actions exceeds the message-list size, then a competition runs between all the possible actions as follows. C_i makes for each $a_i^j \in \mathscr{A}_i^{poss}$ a bid Bid_i^j according to

$$Bid_i^j = b \cdot Str_i \cdot Spec_i \cdot Est_i^j \tag{13}$$

where b is a constant called risk factor (bid coefficient), Str_i is the C_i's strength, and $Spec_i$ is C_i's specificity. The winning actions are selected with probability proportional to the bids, and a classifier C_i is allowed to carry out its action $a_i^j \in \mathscr{A}_i^{poss}$ is given by

$$\frac{Bid_i^j}{\sum_{C_k \in \mathscr{S}} \sum_{a_k^l \in \mathscr{A}_k^{poss}} Bid_k^l} \tag{14}$$

where $\mathscr{S}$ refers to the set of all satisfied classifiers. Each action a classifier C_i is allowed to perform is called an *actual action*, and the set of all actual actions of C_i is denoted by $\mathscr{A}_i^{act}$ ($\mathscr{A}_i^{act} \subseteq \mathscr{A}_i^{poss}$).

Each classifier C_i pays for each of its actual action $a_i^j \in \mathscr{A}_i^{act}$ the corresponding bid Bid_i^j in equal shares to the *action-relevant predecessors*, that means, to all classifiers that sent the action-relevant messages contained in $\mathscr{M}_i^j$. Formally, this leads to the following strength modifications:

$$Str_i = Str_i - Bid_i^j \tag{15}$$

and

$$Str_k = Str_k + \frac{Bid_i^j}{|\mathscr{P}_i^j|} \qquad \forall C_k \in \mathscr{P}_i^j \tag{16}$$

where

$$\mathscr{P}_i^j = \{C_k \in \mathscr{C} : \exists m \in \mathscr{M}_i^j (C_k \text{ sent } m)\} \tag{17}$$

is the set of action-relevant predecessors with respect to a_i^j. For an illustration of the estimated goal relevance of a classifier's action, consider the classifier and the message list depicted by the Tables 1 and 2, respectively. Taking a closer look on classifier C_1, for instance, one finds that the set of matching tuples satisfying it is given by $\mathscr{T}_1 = \{(m_1, m_1), (m_2, m_1), (m_3, m_1), (m_4, m_1)\}$. C_1 has two possible actions, $a_1^1 = 01001$ and $a_1^2 = 01101$. The set of action-relevant message tuples with respect to a_1^1 and a_1^2 is $\{(m_1, m_1)\}$ and $\{(m_2, m_1), (m_3, m_1), (m_4, m_1)\}$, respectively, and the set of action-relevant messages with respect to a_1^1 and a_1^2 is $\mathscr{M}_1^1 = \{m_1\}$ and $\mathscr{M}_1^2 = \{m_1, m_2, m_3, m_4\}$, respectively. Consequently, C_1 estimates the goal relevance of a_1^1 and a_1^2 as $Est_1^1 = Int(m_1) = 90$ and $Est_1^2 = \sum_{i=1}^4 Int(m_i) = 360$, respectively. (The possible actions of the classifiers C_1 to C_4 and their estimated goal relevances are summarized in Table 3. Note that $a_1^1 = a_3^1$, but $Est_1^1 \neq Est_3^1$; this shows that under the ABB1 the same action can be differently judged by different classifiers.)

Table 2

Message list $\mathcal{M}$	
message	intensity
$m_1 = 01011$	90
$m_2 = 11110$	130
$m_3 = 01111$	60
$m_4 = 11111$	80
$m_5 = 01001$	150
$m_6 = 00000$	40
$m_7 = 01000$	50
$m_8 = 00011$	110

2.2. *The algorithm ABB2*

According to the ABB2, which may be considered as a refinement of the ABB1, a classifier does no longer have a single, scalar-valued strength, but an *array* of strength values each indicating the estimated goal relevance of a specific action. In other words, now a classifier C_i has for each of its possible actions $a_i^j \in \mathcal{A}_i^{poss}$ one strength Est_i^j called *estimated goal relevance of* a_i^j. The ABB2 retains the basic working method of the ABB1, but realizes bidding, firing, and strength modification in a refined action-oriented way as follows. If a competition runs, then each satisfied classier C_i makes for each of its possible actions $a_i^j \in \mathcal{A}_i^{poss}$ an action-specific bid Bid_i^j,

$$Bid_i^j = b \cdot Est_i^j \tag{18}$$

where b is the risk factor and Est_i^j is C_i's estimation of the goal relevance of a_i^j.

The array-valued strengths are modified in a bucket-brigade style as follows. First, a classifier C_i that wins the competition with respect to a_i^j (the probability for that is also given by equation (14)) decreases the estimated goal relevance Est_i^j by the amount of the bid Bid_i^j and pays for each action-relevant message $m \in \mathcal{M}_i^j$ a fraction of this bid to the classifier C_k that sent m. Second, assuming that the message m corresponds to the action a_k^l (performed during the previous cycle), the classifier C_k increases Est_k^l by the amount received from C_i. Formally:

$$Est_i^j = Est_i^j - Bid_i^j \tag{19}$$

and

$$Est_k^l = Est_k^l + \frac{Bid_i^j}{|\mathcal{M}_i^j|} \qquad \forall\, a_k^l \in \mathcal{M}_i^j \tag{20}$$

where $\mathcal{M}_i^j$ is the set of all action-relevant messages as specified by equation (11).

It has to be mentioned that there are two other approaches to learning in CSs that employ multiple classifier strengths. First, Huang's (1989a, 1989b) *context-array bucket brigade*; roughly, the idea underlying this algorithm is to associate with each classifier a strength array of variable length which is used for estimating the goal relevance (usefulness) of a classifier in dependence on the specific contexts under which it could fire. And second, Riolo's (1990) *look-ahead bucket brigade*; here the idea is to associate with each classifier three strength values which are used for building up an

Table 3

Estimated goal relevance versus support			
classifier	action	estimation	support
C_1	$a_1^1 = 01001$	$Est_1^1 = 90$	$Sup_1 = 360$
	$a_1^2 = 01101$	$Est_1^2 = 360$	
C_2	$a_2^1 = 10011$	$Est_2^1 = 300$	$Sup_2 = 360$
	$a_2^2 = 00011$	$Est_2^2 = 150$	
C_3	$a_3^1 = 01001$	$Est_3^1 = 210$	$Sup_3 = 410$
	$a_3^2 = 00000$	$Est_3^2 = 100$	
	$a_3^3 = 01000$	$Est_3^3 = 110$	
	$a_3^4 = 00001$	$Est_3^4 = 170$	
C_4	$a_4^1 = 11011$	$Est_4^1 = 310$	$Sup_4 = 310$

internal model of the CS's environment. Although both the context-array BB, the look-ahead BB and the ABB2 aim at an improved rating of the role each classifier plays in solving a given problem, they do so in completely different ways; in particular, they entirely differ in their use of the multiple strengths and in their bidding, firing and strength adjustment mechanisms.

2.3. *The algorithm APSP*

The APSP retains the basic working method of the PSP, but employs array-valued strengths in the same way as the ABB2 does. If a competition runs, then the satisfied classifiers make action-specific bids according to equation (18) and the winning classifiers perform their actions. At the end of each episode, the estimated goal relevance Est_i^j of each action a_i^j that was performed during any cycle of the episode is modified as follows:

$$Est_i^j = Est_i^j - b \cdot Est_i^j + b \cdot Ext \tag{21}$$

where b is the risk factor and Ext is the external reward obtained at the end of the episode.

A comparison of the PSP and the APSP, as well as of the BB and the ABB1/ABB2, shows that the standard and the action-oriented algorithms significantly differ from each other in their bidding, firing and strength modification mechanisms:

- Bidding. According to the standard algorithms, each satisfied classier C_i makes a *single action-unspecific* bid. Interestingly, this does even hold for the BB with support, which employs the concept of message intensities in a way similar than the ABB1 does (see equations (3) and (12)). The point is that the message intensities are used for completely different purposes: in order to calculate a quantity which is taken as a 'vote for a specific *action* of a classifier' in the case of the ABB1; and in order to calculate a quantity which is taken as a 'vote for the *activation* of a classifier' in the case of the BB with support. This difference is illustrated by Table 3, which bases on the data of Tables 1 and 2. The support of a classifier does not provide any information about the classifier's possible

actions. Against this, a classifier's estimate of the goal relevance of its actions is very differentiated. For instance, C_1 estimates that the goal relevance of a_1^2 is four times as high as the goal relevance of a_1^1, C_2's estimation of the goal relevance of a_2^1 is twice as high as its estimation of the goal relevance of a_2^2, and C_3 makes four different estimations ranging from 100 to 210. Furthermore, whereas the support for different classifiers is equal whenever they are matched by the same messages (as is the case for C_1 and C_2 which both are matched by the same messages (as is the case for C_1 and C_2 which both are matched by m_1, m_2, m_3 and m_2), these classifiers' estimations of their possible actions may significantly differ from each other. (There is only one special case where the support is as expressive as the estimated goal relevance: if a classifier can perform only one action. See classifier C_4 for an example of this trivial case.)

- Firing. According to the standard approach, there is *no* distinction between possible and actual actions, and there is *no* quantity indicating which action should be performed and which not. Hence, after the bidding process a winning classifier simply performs all actions, namely, one for each message tuple that satisfies it. (This 'all-or-nothing' firing may significantly impact the learning quality (Riolo 1989, Weiß 1991) and therefore is often reduced to a 'one-or-nothing' firing.)
- The standard algorithms make *no* distinction between action-relevant and action-irrelevant predecessors, and a winning classifier simply distributes its bid among *all* its predecessors, that is, among all classifiers whose messages (sent at the previous time step) match at least one of its condition strings. Compared to the action-oriented variants, strength distribution is done in a 'brute force manner', regardless of which actions are advocated by the predecessors.

The next section provides an analysis of the effects of these differences.

3. Analysis

3.1. *Theoretical considerations*

3.1.1. *Results, I: Statistical measures*

Given a classifier list $\mathscr{C}$, a probabilistic measure which shows the importance of an action-oriented perspective of learning is the expected number $E_{\mathscr{C}}$ of different actions that a classifier theoretically can perform, i.e. that a classifier can perform if the appropriate messages are available. This measure is an approximation of the number of different actions that a classifier $C_i \in \mathscr{C}$ performs during a run of the CS. (This number is not known a priori but depends on the contents of the message list during the run.) In order to be able to calculate $E_{\mathscr{C}}$, it is useful to introduce a special schema (Holland 1975) called *action schema*. Assume that a classifier C_i has the symbol $\beta_j \in \{0, 1, \#\}$ on the position j of its pass-through condition and the symbol $\alpha_j \in \{0, 1, \#\}$ on the position j of its action part ($j = 1, \ldots, L$). Then the action schema of C_i is defined as

$$\langle \gamma_1 \ldots \gamma_L \rangle = \{s_1 \ldots s_L \in \{0,1\}^L : (\gamma_j \in \{0,1\} \rightarrow s_j = \gamma_j) \qquad \forall j = 1, \ldots, L\} \quad (22)$$

where

$$\gamma_j = \alpha_j \quad \text{if} \quad \alpha_j \in \{0, 1\}$$

and

$$\gamma_j = \beta_j \quad \text{if} \quad \alpha_j = \#$$

Table 4. $E_{\mathcal{C}}$ for different qs and Ls

q	$L = 6$	$L = 8$	$L = 10$
0•0	1•00	1•00	1•00
0•1	1•06	1•08	1•10
0•2	1•26	1•36	1•48
0•3	1•67	1•99	2•36
0•4	2•43	3•37	4•41
0•5	3•81	5•96	9•31
0•6	6•32	11•70	21•64
0•7	10•94	24•29	53•93
0•8	19•45	52•33	140•74
0•9	35•16	115•19	377•38
1•0	64•00	256•00	1024•00

For instance, the action schema of the one-condition classifier

$$0\#\#1\#\#01/\#1\#00\#1\#$$

is given by

$$\langle 01\#00\#11\rangle = \{01000011, 01000111, 01100011, 01100111\}$$

An action scheme is a formal specification of the set of all actions that a classifier theoretically can perform. The number of elements contained in an action scheme is what can be called the *action potential* (AP for short) of a classifier. Let p denote the probability for the occurrence of a non-# symbol on a position of a condition/action string, $q = 1 - p$ the corresponding probability for a # symbol, L the message length, and $P[\mathrm{AP} = k]$ the probability that the action potential of a classifier is equal to $k \in \mathbf{N}$. Because the action potential of a classifier always is equal to 2^i for some $i \in \{0, \ldots, L\}$ and because γ_i always is equal to # with probability q^2 for each $i \in \{1, \ldots, L\}$, it follows that

$$P[\mathrm{AP} = k] = \begin{cases} \binom{L}{i}(q^2)^i(p^2+2pq)^{L-i} & \text{for} \quad k = 2^i \quad \text{and} \quad i \in \{1, \ldots, L\} \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

Now the approximating measure $E_{\mathcal{C}}$ can be calculated by

$$\begin{aligned} E_{\mathcal{C}} &= \sum_{i=1}^{2^L} iP[\mathrm{AP} = i] = \sum_{i=0}^{L} 2^i P[\mathrm{AP} = 2^i] \\ &= \sum_{i=0}^{L} 2^i \binom{L}{i}(q^2)^i(p^2+2pq)^{L-i} = (1+q^2)^L \end{aligned} \tag{24}$$

Table 4 illustrates the approximating measure $E_{\mathcal{C}}$ for different values of L and q. As this table shows, even for standard configurations $E_{\mathcal{C}}$ is clearly greater than one (e.g. $E_{\mathcal{C}} = 5{\cdot}96$ for $q = 0{\cdot}5$ and $L = 8$) and quickly grows for increasing values of q and L.

Another informative measure in view of the action-oriented perspective of learning in CSs is the probability $P[\mathrm{AP} > 1]$ that a classifier theoretically can perform more

than one action. From the above considerations it follows that this probability is given by

$$P[\mathrm{AP} > 1] = 1 - (1 - q^2)^L \tag{25}$$

where q and L are defined as above. Again, even for standard configurations this probability is considerably high (e.g. $P[\mathrm{AP} > 1] = 0{\cdot}9$ for $q = 0{\cdot}5$ and $L = 8$).

3.1.2. *Results, II: BB versus ABB1*

Despite the fundamental differences between the BB with support and the ABB1, it is possible to relate the learning behaviour of these algorithms:

Proposition 1. *Let* $P^{\mathrm{BB}}[a_i^j \in \mathscr{A}_i^{\mathrm{act}}]$ *denote the probability given by expression* (4) *and* $P^{\mathrm{ABB1}}[a_i^j \in \mathscr{A}_i^{\mathrm{act}}]$ *the probability given by expression* (14).

(i) *If the action potential of each classifier is equal to one (i.e. each classifier* C_i *theoretically can perform exactly one action* a_i^1*) then*

$$P^{\mathrm{ABB1}}[a_i^1 \in \mathscr{A}_i^{\mathrm{act}}] = P^{\mathrm{BB}}[a_i^1 \in \mathscr{A}_i^{\mathrm{act}}] \tag{26}$$

(ii) *If the action potential of a classifier is greater than one (i.e. a classifier* C_i *is capable of performing more than one action) and the sets of action-relevant messages are disjoint (i.e.* $\mathscr{M}_i^p \cap \mathscr{M}_i^q = \emptyset \ \forall p, q$ *with* $p \neq q$*), then*

$$P^{\mathrm{ABB1}}[a_i^j \in \mathscr{A}_i^{\mathrm{act}}] = P^{\mathrm{BB}}[a_i^j \in \mathscr{A}_i^{\mathrm{act}}] \cdot \frac{Est_i^j}{\sum_{a_i^j \in \mathscr{A}_i^{\mathrm{poss}}} Est_i^j} \tag{27}$$

Proof. Part (i) immediately follows from the fact that in this case $Est_i^1 = Sup_i$ does hold. Part (ii) requires more effort. From the disjointness of the sets $\mathscr{M}_i^j$ it follows that

$$\sum_{a_i^j \in \mathscr{A}_i^{\mathrm{poss}}} Est_i^j = \sum_{a_i^j \in \mathscr{A}_i^{\mathrm{poss}}} \sum_{m \in \mathscr{M}_i^j} Int(m) = \sum_{m \in \bigcup_j \mathscr{M}_i^j} Int(m) = Sup_i$$

and therefore

$$\sum_{a_i^j \in \mathscr{A}_i^{\mathrm{poss}}} Bid_i^j = Bid_i$$

Consequently:

$$\begin{aligned} P^{\mathrm{ABB1}}[a_i^j \in \mathscr{A}_i^{\mathrm{act}}] &= \frac{Bid_i^j}{\sum_{C_k \in \mathscr{P}} \sum_{a_k^l \in \mathscr{A}_k^{\mathrm{poss}}} Bid_k^l} \\ &= \frac{Bid_i}{\sum_{C_k \in \mathscr{P}} \sum_{a_k^l \in \mathscr{A}_k^{\mathrm{poss}}} Bid_k^l} \cdot \frac{Bid_i^j}{Bid_i} \\ &= \frac{Bid_i}{\sum_{C_k \in \mathscr{P}} Bid_k} \cdot \frac{Bid_i^j}{Bid_i} \\ &= P^{\mathrm{BB}}[a_i^j \in \mathscr{A}_i^{\mathrm{act}}] \cdot \frac{Est_i^j}{Sup_i} \\ &= P^{\mathrm{BB}}[a_i^j \in \mathscr{A}_i^{\mathrm{act}}] \cdot \frac{Est_i^j}{\sum_{a_i^j \in \mathscr{A}_i^{\mathrm{poss}}} Est_i^j} \end{aligned}$$

□

Part (i) means that in the trivial case of minimal action potentials the ABB1 and the BB with support coincide in their learning behaviour. Part (ii) means that under the

assumption of disjointness the ABB1 shows the same learning behaviour as an extended BB according to which each winning classifier does not perform all its actions but probabilistically selects an actual action in dependence on the actions' estimated goal relevance.

3.1.3. *Results, III: BB versus ABB2*

Making some simplifying assumptions that aid the mathematical analysis, Grefenstette (1988) showed the following convergence property of the BB: if the classifiers C_i and C_j are coupled in the sense that each firing of C_i is immediately followed by the firing of C_j, and the strength of C_j converges to a constant value, then the strength of C_i also converges to that value. This result can be extended to the ABB2, leading to the following

Proposition 2. *Assume that during each cycle at most one action is performed. If the two actions a_i^j and a_k^l are coupled in the sense that each performance of a_i^j is immediately followed by the performance of a_k^l, and the estimated goal relevance Est_k^l converges to a constant value $(Est_k^l)^*$, then the estimated goal relevance Est_i^j also converges to $(Est_k^l)^*$.*

Proof. If a_i^j is performed during the cycle t and a_k^l during the cycle $t+1$, then

$$Est_i^j[t+2] = Est_i^j[t] - b \cdot Est_i^j[t] + b \cdot Est_k^l[t+1]$$

where $Est_i^j[t+2]$ is the estimated goal relevance Est_i^j at the end of cycle $t+1$. From this it follows that

$$Est_i^j[\sigma(n)+2] = (1-b)^n \cdot Est_i^j[0] + \sum_{m=1}^{n} b \cdot (1-b)^{n-m} \cdot Est_k^l[\sigma(m)+1]$$

where $n \in \mathbb{N}$ and $\sigma(n)$ is defined as $\sigma(n) = t$ iff the nth performance of a_i^j occurred during the cycle t. Consequently:

$$\lim_{t \to \infty} Est_i^j[t] = \lim_{n \to \infty} Est_i^j[\sigma(n)+2] = (Est_k^l)^* \qquad \square$$

The significant difference between these convergence results for the BB and the ABB2 lies in their coupling conditions: in practice a coupling at the classifier level is less likely to occur than a coupling at the action level, simply because a classifier may perform several actions that satisfy different classifiers.

3.1.4. *Results, IV: PSP versus APSP*

In comparing the convergence properties of the BB and the PSP, Grefenstette (1988) showed for the PSP that under a constant external reward the strengths of the firing classifiers converge to this constant. Extending this convergence property to the APSP results in the following

Proposition 3. *If the external reward is a constant value, then the estimated goal relevance of the performed actions converge to this constant.*

Proof. From expression (21) it follows that

$$Est_i^j[\sigma(n)+1] = (1-b)^n \cdot Est_i^j[0] + \sum_{m=1}^{n} b \# (1-b)^{n-m} \cdot Ext[\sigma(m)+1]$$

where $Est_i^j[\tau+1]$ is the estimated goal relevance Est_i^j at the end of episode τ, $Ext[\tau+1]$

is the external reward obtained at the end of episode τ, $n \in \mathbb{N}$, and $\sigma(n)$ is defined as $\sigma(n) = \tau$ iff the nth episode during which a_i^j was performed (at least one times) corresponds to the episode τ. If Ext is a constant, Ext^*, then it follows

$$\lim_{\tau \to \infty} Est_i^j[\tau] = \lim_{n \to \infty} Est_i^j[\sigma(n)+1] = Ext^* \qquad \square$$

With that, the APSP shows at the finer level of the actions the same convergence qualities than the PSP shows at the coarser level of the classifiers. As it is easy to see, in the trivial case in which the action potential of each classifier is equal to one the APSP collapses into the PSP.

3.1.5. *Results, V: ABB2, APSP, and classifier lists*

The following proposition shows a direct consequence of using one strength value for each action and modifying the strengths according to the APSP or the ABB2.

Proposition 4. *For each classifier list $\mathscr{C}$ a classifier list $\mathscr{C}'$ with $E_{\mathscr{C}'} = 1$ can be constructed such that the algorithms ABB2 and APSP show for $\mathscr{C}'$ the same learning behaviour than they show for $\mathscr{C}$.*

Proof. Construct $\mathscr{C}'$ from $\mathscr{C}$ as follows.

1. $\mathscr{C}' = \emptyset$
2. $\forall C \in \mathscr{C}$:
   if $C = Cond_i, \ldots, Cond_n / Act$ and $\langle \gamma_1, \ldots, \gamma_L \rangle$ = action schema of C then $\mathscr{C}' = \mathscr{C}' \cup \{Cond_1, \ldots, Cond_n / Act' : act' \in \langle \gamma_1, \ldots, \gamma_L \rangle\}$.

Each classifier in $\mathscr{C}'$ can perform exactly one action, and each action that can be performed by some classifier $C \in \mathscr{C}$ can be also performed by some classifier $C' \in \mathscr{C}'$ under the same conditions. $\square$

From this proposition and the above considerations it follows that the ABB2 and the APSP show for a classifier list $\mathscr{C}$ the same learning behaviour than the BB and the

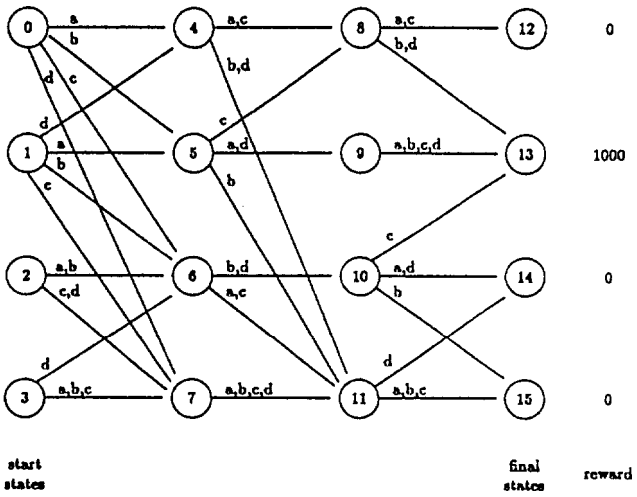

Figure 2. The FSW1 world.

PSP, respectively, only show for an extended classifier list $\mathscr{C}'$ with $E_{\mathscr{C}'} = 1$, where $\mathscr{C}'$ is constructed from $\mathscr{C}$ as described in the proof above. (Under this construction $|\mathscr{C}'| \approx E_{\mathscr{C}} \cdot |\mathscr{C}|$.)

3.2. *Experimental considerations*

3.2.1. *Task description*

For an experimental analysis of the action-oriented algorithms we chose a problem domain for classifier systems first described by Grefenstette (1987). Consider the three finite-state worlds FSW1 (also known as the GREF1 world), FSW2 and FSW3 depicted in Figures 2 to 4, respectively. Each of these worlds consists of several start

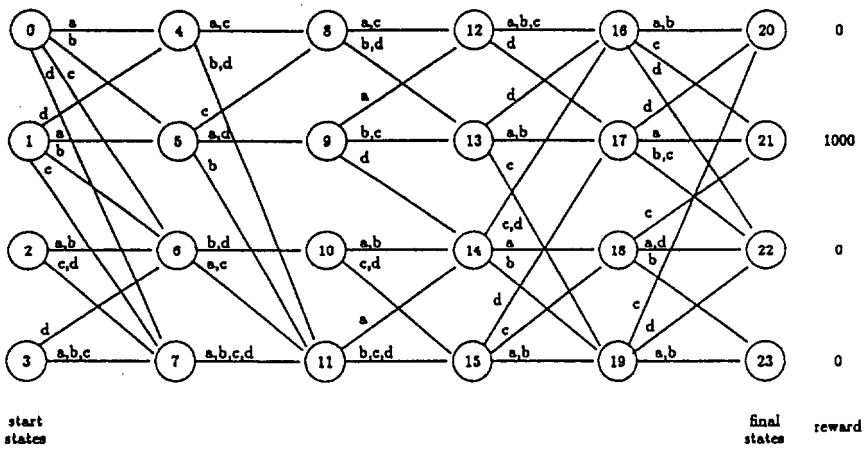

Figure 3. The FSW2 world.

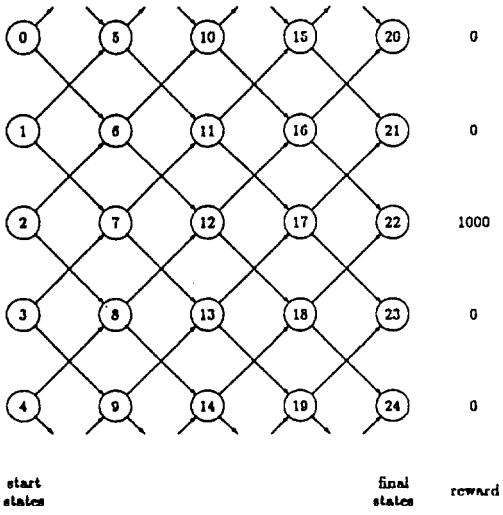

Figure 4. The FSW3 world.

states, intermediate states and final states; only one of the final states (the *goal state*) is associated with a non-zero external reward. For each of these worlds the task is to learn moving from the start states to the single goal state. This task captures the essential features of both the standard and the action-oriented algorithms, and allows a direct comparison of them. As already pointed out by Riolo (1989), the FSW domain is very challenging for several reasons. In particular, learning to solve this task requires the ability to learn to solve several sub-tasks in parallel (because there are several start states from which the goal state has to be reached) as well as the ability of delayed-reinforcement learning (because an external reward is only provided after a final state is reached).

A state-space traversal from any start to any final state—such a traversal corresponds to an *episode*—consists of t elementary state-to-state steps, where $t = 3$ for FSW1, $t = 5$ for FSW2, and $t = 4$ for FSW3. After having reached a final state, a new start state is chosen at random with uniform probability. The maximal external reward per episode is 1000 for both FSW1, FSW2 and FSW3; the random reward per episode (i.e. the expected reward for a random space traversal) is 150 for both FSW1 and FSW2 and 200 for FSW3.

3.2.2. *Implementation details*

By taking Riolo's (1988) elaborated software package CFS-C as a model, a CS simulator called CFS-SIM (Hutter 1991) was developed that allows a direct comparison of the standard and the action-oriented learning algorithms. Here some details of CFS-SIM and its implementation are described, in as far as they are relevant to an understanding of the experimental results summarized below.

In order to avoid that a few classifiers impact the learning performance by filling up the entire message list, the BB is modified such that each classifier fires at most one message per cycle. Thereby the message tuple $T \in \mathscr{T}_i$ that determines the new message of a classifier C_i is probabilistically selected from the set $\mathscr{T}_i$ in proportion to the sum of the intensities of the messages contained in T.

At the end of each cycle ineffective messages (i.e. messages that do not contribute to effector activation) as well as hallucinatory messages (i.e. classifier messages that look like effector messages) are deleted from the message list because they are known to enormously decrease the learning performance (e.g. see Riolo 1989, Weiß 1991). Because we are interested in the 'pure' learning abilities of the action-oriented algorithms in combination with a genetic algorithm, additional mechanisms like tax payment (Riolo 1988, Robertson and Riolo 1988) or triggered rule formation (Riolo 1989) are not applied. (The word 'triggered' indicates that rule formation is activated by specific triggering conditions. Against that, the genetic algorithm typically is not triggered, but is activated with a fixed probability each cycle; an unconventional approach to the triggering of the genetic algorithm was presented by Booker 1989.)

The application of a genetic algorithm requires that each classifier is assigned a measure representing its 'reproductive fitness'. In the case of the BB, the ABB1 and the PSP this measure is given by the scalar-valued strength of a classifier. In the case of the ABB2 and the APSP this measure is calculated as the average of the estimated goal relevance of a classifier's actions. With that, in the case of the action-oriented algorithms the rule discovery is guided by the goal relevance of the individual actions.[5]

The output interface of a CS always depends on the problem domain under

consideration. In the case of FSW1 and FSW2, the following four effectors have been used:

######00 / 111111## ≡ go path labelled with 'a'
######01 / 111111## ≡ go path labelled with 'b'
######10 / 111111## ≡ go path labelled with 'c'
######11 / 111111## ≡ go path labelled with 'd'

Similarly, the effectors used for FSW3 world are

#######0 / 1111111# ≡ go left path
#######1 / 1111111# ≡ go right path

CFS-SIM treats the effectors in exactly the same way as the classifiers; this means, the steps two and three of the major cycle are based on the same matching, bidding and activation mechanisms.

3.2.3. *Results, I: BB versus ABB1 and ABB2*

Except where otherwise noted, the following parameters are used in the experiments reported below. The classifier list size is 40, the message list size is 5, and the message length is 8. The genetic algorithm is applied with probability p per cycle, where $p = 0{\cdot}0\overline{3}$ for FSW1, $p = 0{\cdot}02$ for FSW2, and $p = 0{\cdot}025$ for FSW3. When applied, 5% of the classifiers are selected with probability proportional to the inverse of their reproductive fitness and replaced by new classifiers. The new classifiers are created as follows: until no further classifier is required, two classifiers are selected with probability proportional to their reproductive fitness and modified by means of mutation and crossover. The strength values of the new classifiers are set to random initial values. Figures 5 to 8 show the results of eight experiments. Each data point in these figures reflects the mean reward per episode obtained during the previous 1000 episodes, averaged over five runs each starting with a different set of randomly generated classifiers.

Figure 5 shows the performance profiles of the ABB1, the BB with support and the BB without support for FSW1 (top), FSW2 (middle) and FSW3 (bottom). The mean specificity was 0·45. In all three experiments the ABB1 showed the best learning behaviour. Both the ABB1 and the BB with support performed better than the BB without support, and all three learning algorithms clearly exceeded the random performance level. Furthermore, each of the three learning algorithms showed nearly the same learning qualities for FSW1, FSW2 and FSW3, although these finite-state worlds differ in their episode length.

Figure 6 shows the performance profiles of the ABB2, the ABB1 and the BB with support for the finite-state world FSW1, using a mean specificity of 0·3 (top), 0·45 (middle) and 0·6 (bottom). In all three experiments the performance levels of the three algorithms were significantly above the random performance level, and the ABB2 performed clearly better than the BB and the ABB1. Furthermore, the ABB2 performed equally well for the different mean specificities. Against that, the BB and the ABB1 showed a contrary learning behaviour for low and high specificities: the BB performed better for low specificities than it performed for high ones, whereas the ABB1 performed better for high specificities than it performed for low ones. An examination of the development of the classifier strengths during the runs shows the following reason for this contrary behaviour: the BB tends to increase the strengths

too quickly for high specificities, whereas the ABB1 tends to decrease the strengths too quickly for low specificities.

3.2.4. *Results, II: PSP versus APSP*

Figure 7 shows the performance profiles of the PSP and the APSP for the finite-state world FSW1, using a mean specificity of 0•45. The random performance level is clearly outdone by the performance level of the PSP, which in turn is clearly outdone by the performance level of the APSP.

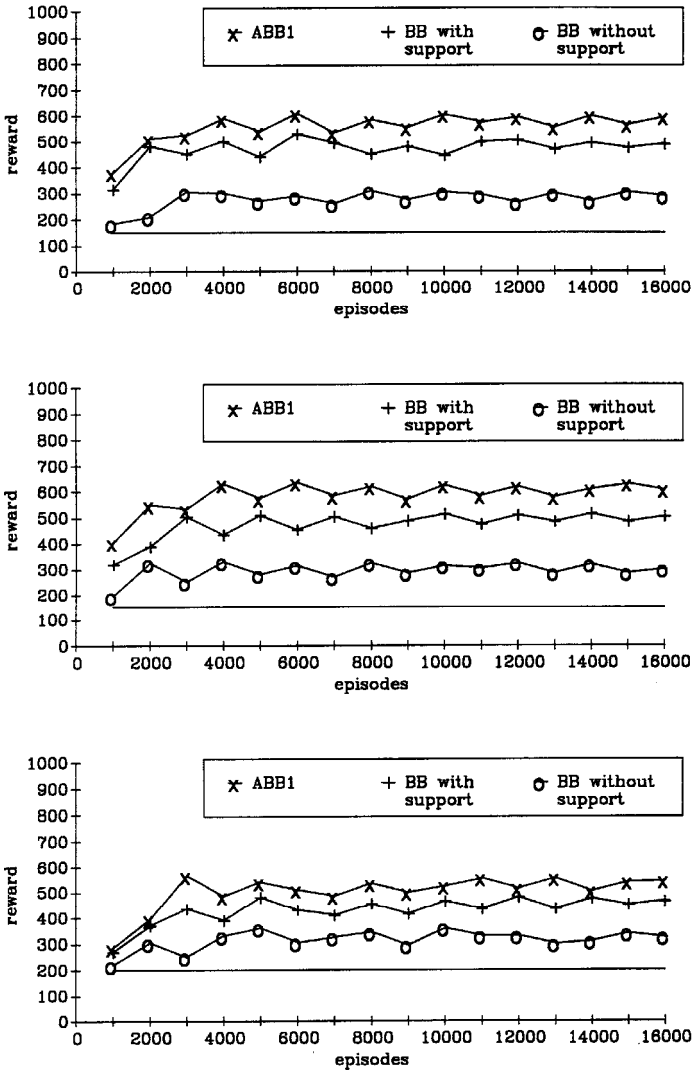

Figure 5. Performance profiles of the ABB1, the BB with support and the BB without support for FSW1 (top), FSW2 (middle) and FSW3 (bottom), using a mean specificity of 0•45.

3.2.5. *Results, III: Multiple strengths versus scalar strengths*

A comparison of the multiple-strengths algorithms and the scalar-strength algorithms shows that the former (the ABB2 and the APSP) obviously performed better than the latter. The reason for this is that a scalar-valued strength necessarily is only a vague measure of the goal relevance of each of the actions performed by a classifier—at best, a scalar-valued strength indicates the average goal relevance of the actions of a classifier. As a consequence, because this vague measure is used for calculating the bids, the bids themselves are vague.

3.2.6. *Results, IV: (A)BB versus (A)PSP*

Finally, in comparing the BB-type and the PSP-type algorithms within the scope of these experiments and under equal conditions, two things can be observed. First, the

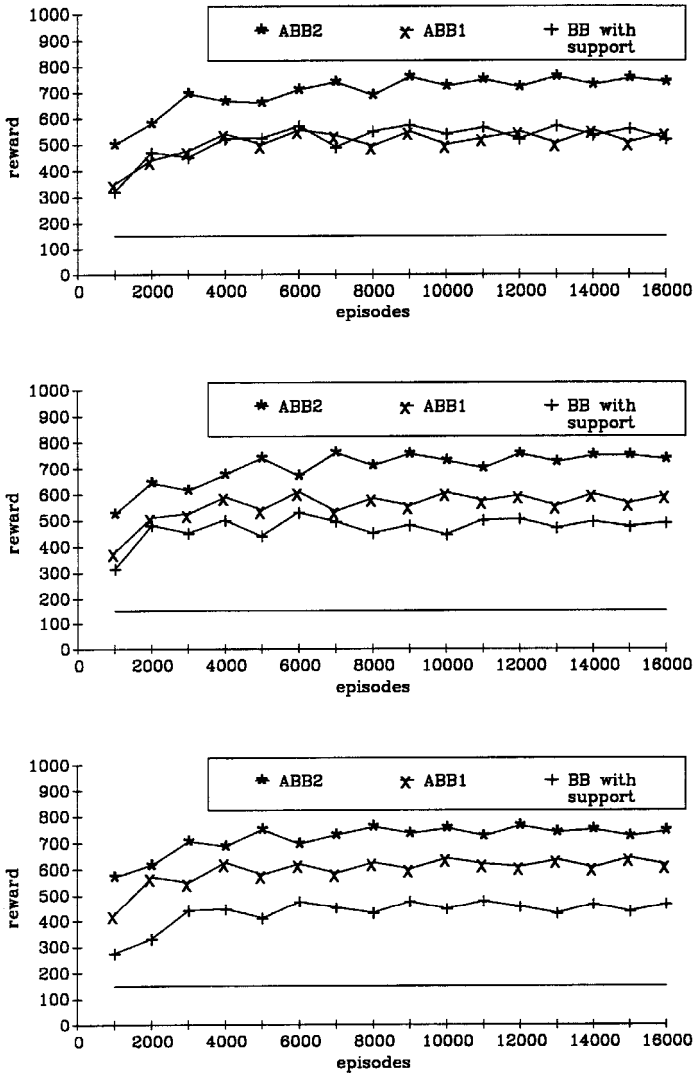

Figure 6. Performance profiles of the ABB2, the ABB1 and the BB with support for FSW1, using a mean specificity of 0•3 (top), 0•45 (middle) and 0•6 (bottom).

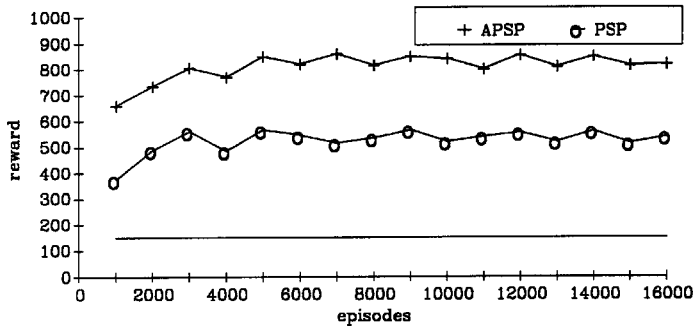

Figure 7. Performance profiles of the PSP and the APSP for FSW1, using a mean specificity of 0·45.

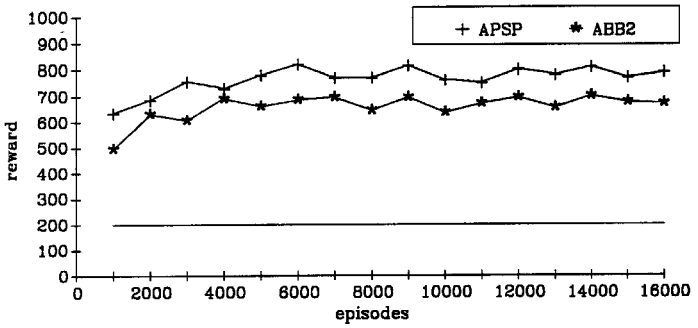

Figure 8. Performance profiles of the ABB2 and the APSP for FSW3, using a mean specificity of 0·45.

performance level of the PSP roughly lies in between the performance levels of the BB with support and the ABB1. Second, as is shown by Figure 8, the APSP clearly performs better than the ABB2. These comparative results are consistent with Grefenstette's (1988) theoretical and experimental comparison of the BB-type and the PSP-type algorithms.

4. Conclusions

This article considered learning in CSs from an action-oriented point of view. Three extensions of the standard BB and PSP—the ABB1, the ABB2, and the APSP—were investigated that strictly work on the principle of action orientation. These action-oriented algorithms realize learning at the finer action level in the same way than the standard ones do at the coarser classifier level. The ABB1, ABB2 and APSP provide several new aspects of learning in CSs, including the notion of estimated goal relevance of an action, the concept of multiple bids per classifier, and the action-oriented mechanisms of bidding, firing and strength adjustment. Their major characteristic is that they correlate the learning processes in a CS with the possible and actual actions of the individual classifiers. The theoretical and experimental considerations showed that the action-oriented perspective leads to a clear improvement over the traditional perspective.

Of course, CSs are very complex learning systems, and there are several issues that have been not addressed in this article. In particular, currently it is not known what effects the action-oriented perspective does have on

- the problem of default hierarchy formation (see Riolo 1987),
- the cooperator/competitor dilemma in CSs (see Wilson and Goldberg 1989), and
- the problem of forming higher-level behavioural modules (see Dorigo and Colombetti 1994, Dorigo and Schnepf 1993).

To clarify these issues has to be a central goal of future research. The work reported in this article provides a solid basis for achieving this goal.

Acknowledgements
The author would like to thank Rick Riolo for providing a copy of his CFS-C software package. The article greatly benefited from the valuable and stimulating comments of Marco Dorigo and Rick Riolo on our previous related work.

Notes

1. As a consequence of this difference, the BB requires less memory and less peak computation than the PSP, but the PSP typically achieves a better performance level than the BB. Roughly, this is because the BB has difficulties in generating long classifier sequences, whereas the PSP needs to maintain detailed information about the past activities carried out by the CS. See Weiß (1994) and also Smith and Goldberg (1990) for further aspects of this consequence.
2. By 'standard BB' we refer to the BB as described, for instance, by Holland (1986) and Riolo (1988). (The concept of support-based bidding described below is a relatively young, yet well established extension of the original BB; see also Booker 1988.)
3. By 'standard PSP' we refer to the PSP as described by Grefenstette (1988). As mentioned by Grefenstette, there are many variations of this standard form; for instance, instead of expression (9) the bids might be collected for each firing of a classifier, and in this case even support-based bidding and strength adjustment could be realized.
4. Of course, many modifications of the BB and the PSP have been proposed in the literature; for instance, see the message-based BB (Dorigo 1991), the context-array BB (Huang 1989a, 1989b), the look-ahead BB (Riolo 1990), the implicit BB (Wilson 1985), the hierarchical BB (Wilson 1987), and the modifications of the PSP mentioned in Grefenstette (1988). However, none of these modifications solves the problem of lacking differentiation at the action level.
5. As Huang (1989a, 1989b) emphasized, array-valued strengths can provide additional information for rule modification and discovery. For instance, the array-valued strengths could be used for protecting those classifiers from replacement whose estimated goal relevance of at least one action is greater than some pre-defined value. However, within the frame of the experiments described here such additional information is not used. See Huang (1989a, 1989b) for further details of the scalar-versus-multiple-strengths differences.

References

Booker, L. B. (1982) Intelligent Behavior as an Adaptation to the Task Environment. Doctoral Dissertation. Department of Computer and Communication Sciences, University of Michigan, Ann Arbor.

Booker, L. B. (1988) Classifier systems that learn internal world models. *Machine Learning*, **3**, 161–192.

Booker, L. B. (1989) Triggered rule discovery in classifier systems. In *Proceedings of the Third International Conference on Genetic Algorithms*, Fairfax, VA (San Mateo, CA: Morgan Kaufmann), pp. 265–274.

Booker, L. B., Goldberg, D. E. and Holland, J. H. (1989) Classifier systems and genetic algorithms. *Artificial Intelligence*, **40**, 235–282.

Dorigo, M. (1991) New perspectives about default hierarchies formation in learning classifier systems. Report No. 91-002. Dipartimento di Elettronica, Politecnico di Milano.

Dorigo, M. and Schnepf, U. (1993) Genetics-based machine learning and behaviour based robotics: A new synthesis. *IEEE Transactions on Systems, Man, and Cybernetics*, **23**, 141–153.

Dorigo, M. and Colombetti, M. (1994) Robot shaping: developing autonomous agents through learning. *Artificial Intelligence*, **71**, 321–370.

Goldberg, D. E. (1983) Computer-aided Gas Pipeline Operation using Genetic Algorithms and Rule Learning. Doctoral Dissertation. Department of Civil Engineering, University of Michigan, Ann Arbor.

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

Grefenstette, J. J. (1987) Multilevel credit assignment in a genetic learning system. In *Proceedings of the Second International Conference on Genetic Algorithms*, Cambridge, MA (Hillsdale, NJ: Lawrence Erlbaum), pp. 202–209.

Grefenstette, J. J. (1988) Credit assignment in rule discovery systems based on genetic algorithms. *Machine Learning*, **3** (2–3), 225–245.

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems* (Ann Arbor, MI: University of Michigan Press).

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

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, and T. M. Mitchell (eds), *Machine Learning: An Artificial Intelligence Approach*, Volume 1 (Los Altos, CA: Morgan Kaufmann), pp. 593–632.

Holland, J. H. and Reitman, J. S. (1978) Cognitive systems based on adaptive algorithms. In D. A. Waterman and F. Hayes-Roth (eds), *Pattern-directed Inference Systems* (New York: Academic Press), pp. 313–329.

Huang, D. (1989a) Credit-apportionment in Rule-based Systems: Problem Analysis and Algorithm Synthesis. Doctoral Dissertation. Department of Computer and Communication Sciences, University of Michigan.

Huang, D. (1989b) The context-array bucket-brigade algorithm: An enhanced approach to credit-apportionment in classifier systems. In *Proceedings of the Third International Conference on Genetic Algorithms*, Fairfax, VA (San Mateo, CA: Morgan Kaufmann), pp. 311–316.

Hutter, M. (1991) *Implementierung eines Klassifizierungssystems*, Institut für Informatik, TU München.

Riolo, R. L. (1987) Bucket brigade performance: II. Default hierarchies. In *Proceedings of Second International Conference on Genetic Algorithms*, Cambridge, MA (Hillsdale, NJ: Lawrence Erlbaum), pp. 196–201.

Riolo, R. L. (1988) CFS-C: A package of domain independent subroutines for implementing classifier systems in arbitrary, user-defined environments. Technical Report. Division of Computer Science and Engineering, University of Michigan.

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

Riolo, R. L. (1990) Lookahead planning and latent learning in classifier systems. In J.-A. Meyer and S. Wilson (eds), *From Animals to Animats – Proceedings of the First International Conference on the Simulation of Adaptive Behavior* (Cambridge, MA: The MIT Press).

Robertson, G. G. and Riolo, R. L. (1988) A tale of two classifier systems. *Machine Learning*, **3**, 139–159.

Smith, R. E. and Goldberg, D. E. (1990) Reinforcement learning with classifier systems. In B. Zeigler, and J. Rozenblit (eds), *AI, Simulation and Planning in High Autonomy Systems* (Los Alamitos, CA: IEEE Computer Society Press), pp. 184–192.

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

Weiß, G. (1992) Learning the goal relevance of actions in classifier systems. In *Proceedings of the 10th European Conference on Artificial Intelligence,* Vienna, Austria (Chichester: Wiley), pp. 430–434.

Weiß, G. (1993a) Collective learning and action coordination. In *Proceedings of the 13th International Conference on Distributed Computing Systems,* Pittsburgh, PA (Los Alamitos, CA: IEEE Computer Society Press), pp. 203–209.

Weiß, G. (1993b) Learning to coordinate actions in multi-agent systems. In *Proceedings of the 13th International Conference on Artificial Intelligence,* Vol. 1, Chambéry, France (San Mateo, CA: Morgan Kaufmann), pp. 311–316.

Weiß, G. (1994) Hierarchical chunking in classifier systems. In *Proceedings of the 12th National Conference on Artificial Intelligence*, Seattle, WA (Menlo Park, CA: AAAI Press), pp. 1335–1340.

Wilson, S. W. (1985) Knowledge growth in an artificial animal. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Pittsburgh, PA (Hillsdale, NJ: Lawrence Erlbaum), pp. 16–23.

Wilson, S. W. (1987) Hierarchical credit allocation in a classifier system. In L. Davis (ed.), *Genetic Algorithms and Simulated Annealing* (Los Altos, CA: Morgan Kaufmann), pp. 104–115.

Wilson, S. W. and Goldberg, D. E. (1989) A critical review of classifier systems. In *Proceedings of the Third International Conference on Genetic Algorithms*, Fairfax, VA (San Mateo, CA: Morgan Kaufmann), pp. 244–255.