Social simulation using a Multi-Agent Model based on Classifier Systems: The Emergence of Vacillating Behaviour in "El Farol" Bar Problem

Luis Miramontes Hercog and Terence C. Fogarty

School of Computing, South Bank University
103 Borough Rd.
London SE1 0AA, U.K.
Telephone:+44-20-78157008
Fax:+44-20-78157499
{miramol, fogarttc}@sbu.ac.uk

Abstract. In this paper, MAXCS – a Multi-Agent system that learns using XCS – is used for social modelling on the "El Farol" Bar problem. A cooperative reward distribution technique is used and compared with the original "selfish" "El Farol" Bar problem distribution technique. When using selfish reward distribution a vacillating agent emerges which, although obtaining no reward itself, enables all the other agents to benefit in the best way possible from the system.
Experiments with 10 agents and different parameter settings for the problem show that MAXCS is always able to solve it. Furthermore, emergent behaviour can be observed by analysing the actions of the agents and explained by analysing the rules utilised by the agents. The use of a learning classifier system has been essential for the detailed analysis of each agent's decision, as well as for the detection of the emergent behaviour in the system. The results are divided into three categories: those obtained using cooperative reward, those obtained using selfish reward and those which show emergent behaviour.
Keywords: Multi-Agent Systems, emergent behaviour, XCS, Learning Classifier Systems, "El Farol" Bar problem.

1 Introduction

In this paper, MAXCS – a Multi-Agent system (MAS) [16][31] that learns using XCS [39] – is used for social modelling on the"El Farol" Bar problem [3].
A Multi-agent System is a set of individual entities that can partially perceive their environment and interact autonomously with it. The agents learn from the rewards obtained from the environment. The resources and skills available to each agent, its perception and representation of the environment and, in some cases, communication with other agents, direct the behaviour of the agent toward satisfying its objectives either individually or in association with others. Multi-agent simulation is based on the idea of a computerised representation of entities' behaviour in the world. This computerised representation gives the possibility of representing a phenomenon, which emerges from the interactions of a set of agents with their own operational autonomy [15]. The operational autonomy, in this particular case of study, is provided by a learning classifier system: XCS. The agents base their decisions on evolving rule sets, which are improved against the performance of the agent on the desired problem.
Experiments with 10 agents and different parameters settings for the problem show that MAXCS is always able to solve it. Furthermore, emergent behaviour[1] can be observed by analysing the actions of the agents and explained by analysing the rules utilised by the agents. The use of a learning classifier system has been essential for the detailed analysis of each agent's decision, as well as for the detection of the emergent behaviour in the system.

[1] Emergent behaviour is considered as any computation that achieves predictable global effects, formally or stochastically, by communicating directly with only a bounded number of neighbours and without the use of global visibility[17].

The "El Farol" Bar problem is explained in the following section. There is then a brief description of XCS. After that, the use of XCS as the learning engine for the Multi-agent system MAXCS is explained and the results are presented and discussed. The results are divided into three categories: those obtained using cooperative reward, those obtained using selfish reward and those which show emergent behaviour.

2 The "El Farol" Bar Problem

A good benchmark for multi-agent systems is the "El Farol" Bar problem invented by B. Arthur. The problem is based on a bar in Santa Fe which offers entertainment each week. The place is small and uncomfortable if overcrowded but boring if not full.

A comfort **threshold** (Λ_{cm}) is defined then, as the number of people which makes the place an enjoyable one. The original problem assumes 100 agents and 60 seats in the bar. The agents have to decide, based on their strategies, whether or not to go the bar each week. The agents are rewarded in two different ways: if the agent decides to go to the bar and the comfort threshold is not exceeded then the agent is rewarded, if the agent decides to stay at home and the threshold is exceeded then it gets rewarded, otherwise they are not rewarded. In other words, if they make the correct choice they are rewarded.

Each agent takes one of a number of fixed strategies to predict the number of agents who will go, based on past data, e.g. the same as two weeks ago, the average of the last 3 weeks, etc. Each week, each agent evaluates its strategies against past data and chooses the strategy with the best predictor. The predictor is updated according to its reliability. The agent predicts the number of agents that will attend the bar and then decides to go if this prediction is below the value of Λ_{cm}.

After the "El Farol" Bar problem was stated, many approaches have followed – most of them used for economics - due to its formalisation, as an evolutionary game: the Minority Game [11][10][12]. The Minority Game(MG) is an abstraction, it considers Λ_{cm}=0.5 and the agents are rewarded if they are on the side with less agents. The agents base their decisions in rules that state whether it was good to attend (1), or to stay at home (0). After several rounds, the fittest agent is cloned and its rules mutated to replace the worst performing agent in the system.

There have also been some MAS approaches but, unfortunately, little can be taken from them, since they over-simplify the problem by allowing the agents to choose a particular night out of seven to go to the bar [30][42]. The most interesting approach used genetic programming [25] to solve the problem by letting 16 agents communicate with each other [14].

2.1 Why is this problem difficult to learn?

The "El Farol" Bar problem can be considered a coordinated collaboration task which the agents have insufficient resources and skills to achieve. Coordinated collaboration is the most complex of cooperation situations, since it combines task allocation problems with aspects of coordination shaped by limited resources. It is a very hard problem to learn, since the agents are not endowed with any notion of their previous actions – they must base their current action on the overall statistics of attendance at the bar in previous weeks. All of them have access to the same information, therefore this problem goes beyond deductive rationality. The agents can "think" that "if the attendance was 80 last week and 40 the previous week then it's good to go this week"; if all the agents think this they will overcrowd the bar. This is the reason that the problem is called a bounded rationality problem; rationality takes the agents some way but not far enough to solve the problem.

"El Farol" bar problem is a single-step problem, because the answers of the agents are taken, the attendance is computed and the reward is distributed immediately.

In the case of the minority game, the abstraction is extreme, since the agents cannot see the attendance figure, but can only see whether it was good (1) or not good (0) to go to the bar in previous weeks. The abstraction level has been kept for this research.

2.2 Reward distribution schemes

In this paper two reward distribution schemes are used. The traditional "El Farol" Bar reward, called from now on the **selfish** reward scheme, and a **cooperative** reward scheme. The cooperative reward scheme was introduced in [20]: if the bar is overcrowded none of the agents are rewarded, otherwise every agent is rewarded with a scalar value proportional to the attendance with a maximum of 1000.

3 Extended Classifier System

The extended classifier system (XCS) is a Michigan style learning classifier system (LCS) [22][18], invented by Wilson in 1995 [39]. XCS has a simple architecture for the study of LCSs. The knowledge in XCS is encoded in rules called classifiers. The rules are generally formed by a **condition** and an **action** part, represented as *condition→action*. The rule syntax is very simple, representing very fine-grained knowledge. This simple representation allows the LCS to use the evolutionary algorithm for rule discovery. Each rule has a performance indicator (fitness) associated. The fitness is used for conflict resolution by the LCS when many rules are active. The reinforcement mechanism updates the fitness value generating a competitive scheme. This scheme ensures that good rules survive and bad rules die off when applying the genetic algorithm, improving the population performance [2].
The rule representation depends on the problem being solved: ternary (most common) {0,1,#} (the # is known as the don't care symbol), integers, integer intervals [37], real intervals, fuzzy rules [5][1][6] and even S-expressions [2](LISP expressions which may resemble genetic programming[25]).
XCS incorporates accuracy based fitness, considered to be a breakthrough in the LCS field. XCS learns to maximise the reward obtained from the environment, this reward is what drives the search and the self-improvement of the system.
One of the innovations in XCS is that each classifier has three parameters to measure its fitness:

- Prediction (p): Is the value for the reward expected if this rule was selected
- Prediction error (ϵ): Estimates the error of the prediction
- Fitness (F): Evaluates the quality of the prediction, based on the error

Each of these three parameters is updated every time that the reward is obtained by the system. A Q-learning like algorithm [13] [35] is used for the update (see Table 1 for the formulae).
XCS also adds a concept called macro-classifiers. A macro-classifier is the fusion of identical classifiers into a single one. This is done for practical reasons, instead of having n classifiers that are the same, there will be only one with numerosity n.

3.1 XCS functionality

XCS (see Fig.1) perceives the environment through its sensors and encodes this into an environmental message. The classifiers in the rule base that satisfy the environmental message form the match set [M]. If there are no classifiers that match the environmental message, a new one is generated using covering.
The classifiers that have the action with the best prediction in [M] form the action set [A], when exploiting. If XCS is exploring, the action is chosen randomly.
XCS selects the action [41] by computing $\frac{\Sigma F_j p_j}{\Sigma F_j}$ for all the classifiers in the match set, where j are the classifiers with the same action, this is done for each different action in [M].
Then, the selected action is put into the effectors and posted to the environment.
The environment evaluates the action and, in the case of a single-step problem, gives the reward to XCS. As explained before, XCS uses a Q-learning-like algorithm [33] to update the p, ϵ and F values for all the classifiers in [A] in the case of single-step problems and $[A]_{-1}$ in the case of multiple-step problems, $[A]_{-1}$ is the action set at the previous time step.
The reinforcement mechanism updates are calculated as shown in Table 1 for each $classifier_j$ in [A], first p_j and ϵ_j are updated; then the accuracy (κ) for each classifier is computed. A relative accuracy ($\kappa_j^{'}$) is calculated for each classifier, and finally the fitness is updated.

[2] There is a LCS that does not use a performance indicator, based on Holland's ECHO systems [7], questioning whether there should be a performance indicator or not

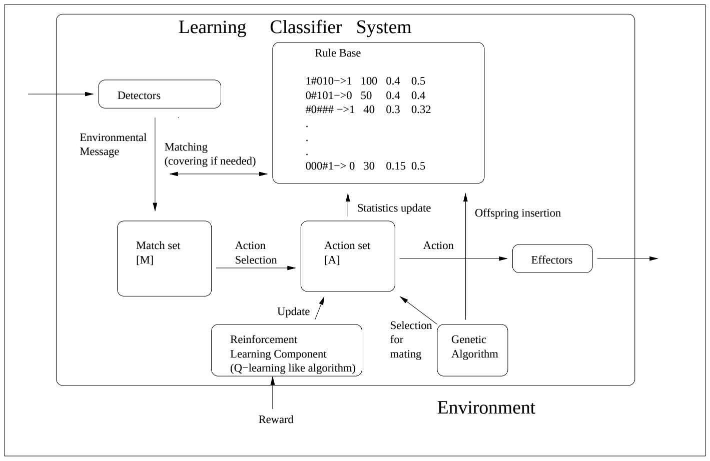

Fig. 1. Functional diagram of XCS

Prediction update	$p_j \leftarrow p_j^{'} + \beta(R - p_j^{'})$	
Prediction error update	$\epsilon_j \leftarrow \epsilon_j^{'} + \beta(\lvert R - p_j\rvert - \epsilon_j^{'})$	
Accuracy	$\kappa_j = \exp(\ln(\alpha)\frac{\epsilon_j - \epsilon_0}{\epsilon_0})$	or
	$\kappa_j = \alpha(\frac{\epsilon_j}{\epsilon_0})^{-n}; n = 5$	if $\epsilon_j > \epsilon_0$;
	$\kappa_j = 1.0$	otherwise
Relative accuracy	$\kappa_j^{'} = \frac{\kappa_j}{\sum \kappa_j}$	
Fitness	$F_j \leftarrow F_j^{'} + \beta(\kappa_j^{'} - F_j^{'})$	

Table 1. XCS Formulae for classifier update used by the Q-learning like algorithm; where $p_j^{'}, \epsilon_j^{'}, F_j^{'}$ are the current values and $F_j, \epsilon_j, p_j$are the new values.

Therefore the fitness is based on how accurate the prediction of the reward is. After the updates are done, a new environmental message is encoded and the process continues.

3.2 Discovery mechanisms

There are two discovery mechanisms in XCS, one, the genetic algorithm(GA), is evolutionary and the other, covering, is not.
Covering happens when there are no classifiers in the rule base that match the environmental message, a new classifier is created: the condition is produced by taking the environmental message and performing a mutation to introduce don't care(#) symbols and the action is generated randomly (a similar mechanism is used in [29]). A new effector covering operator has been introduced, it inserts a new classifier for every possible action that is not covered by the classifiers in the match set.
The GA is applied only in [A] [8], instead of the whole population, so it imposes a selection pressure as well as a niching technique [19] [23]. Each classifier records when was the last time that the GA was applied, and if the average of this value in the classifiers in [A] is greater than θ_{GA}, then the GA is applied.
The GA selects 2 classifiers (parents) using a roulette wheel algorithm based on the fitness. The parents are cloned producing children. The children are recombined, with a χ probability using a two-point crossover. Then, mutation takes place with a μ probability. In case there is only one classifier in [A], cloning and mutation takes place. The children are inserted in the

population. If the maximum population size is reached, some classifiers are deleted based on their fitness (see [24] for deletion techniques).
Experiments have shown that XCS evolves accurate, complete, and minimal representations for Boolean functions [24]. The GA provides accurate generalisation and helped by a deletion technique, its application will get rid of the overgeneral classifiers [3] [39] [40] [24] [27].
XCS has been tried in several test problems: the multiplexer[18], "woods" environments [38], some mazes [26], the monk's problems [32] and lately it has been used for data mining as a real world problem [37], though more research has to be done in complex environments. Markovian and non-Markovian test problems have been tried and XCS has proved to be quite effective, though it is very vulnerable to environments where just a partial number of states can be observed [26].

4 MAXCS: A multi-agent system that learns using XCS

The idea of representing an agent as a LCS [28][9][43] – such as XCS – is based on the following reasoning: if the characteristics of an agent and XCS are put together, XCS covers the features needed by an agent[16].

XCS	**Agent**
Detectors	Perception of the environment
Classifier database	Partial representation of the environment
Accuracy measure	Skill (to evaluate its own performance)
Reinforcement component	Behaviour toward satisfying its objectives
Environmental interaction	Autonomy, not directed by commands
Action posting	Capacity to alter its environment

Table 2. Comparison of the agent and XCS characteristics

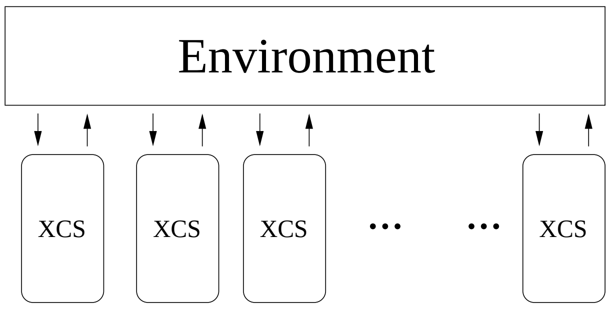

Fig. 2. Diagram of MAXCS

Each agent in the system is represented as an XCS (see Figure 2).
By this analysis it can be seen that the classifier system engine brings a category where a cognitive and a reactive agent types [21] are interlaced and complementary: XCS gives a reliable internal representation (encoded individually in each classifier) and an inference mechanism (GA and covering)– complying with the cognitive agent definition. It can memorise the situations in a very simple way. Using the accuracy measure XCS can foresee the

[3] An overgeneral classifier is a classifier which has many # symbols and it will interact in too many action sets, but is not accurate.

possible reaction (taken as the reward from the environment) to its actions, therefore it can decide which action to take, and why.
On the other hand, the classifier system can be seen also as a reactive agent. XCS shows a specific behaviour depending on the environmental state sensed, selecting the rules and triggering an action.
XCS evolves readable sets of rules which help in understanding how the system is adapting to a specific problem or to a variable environment [6] [36] [28].

5 Experiments

Ten agents have been used for all the experiments. This number of agents eases the task of keeping track of both the populations and the decisions taken by each agent. To simplify the control of the system as a whole, all the agents explore or exploit all together.
For the experiments reported here, each XCS can only see whether it was good (1) or not (0) to attend the bar the previous 5 weeks (M=5). The system can only see the correct answer for each of those weeks, i.e. a 0 if the bar was overcrowded and a 1 if it was good to attend. The rule conditions represent what was good to do last week with the most significant bit (leftmost) and what was good to do 5 weeks ago with the least significant bit (rightmost).
The maximum population size for all the agents is 64 classifiers. This value is set to allow the system to keep all the possible states visible for both actions in the extreme case that no generalisation would happen at all.
The system has been extensively tested using both reward schemes –selfish and cooperative (see subsection 2.2)– with the whole range of possible comfort thresholds for 10 agents (Λ_{cm}=0, Λ_{cm}=0.1, Λ_{cm}=0.2, Λ_{cm}=0.3, Λ_{cm}=0.4, Λ_{cm}=0.5, Λ_{cm}=0.6, Λ_{cm}=0.7, Λ_{cm}=0.8, Λ_{cm}=0.9, Λ_{cm}=1).
The following parameters have been used for all the experiments: # probability = 0.333, explore/exploit rate = 0.5, crossover rate(χ)= 0.8, mutation rate(μ) = 0.02, θ_{GA} = 25, minimum error(ϵ_0)=0.01 and learning rate (β) =0.2.
Subsumption deletion was tried in the beginning, but it yielded populations with just a general classifier. The subsumption deletion is not used for the results here reported, since it has proved to deprive the system of rule diversity [4].
The effector covering is enabled: when a match set is formed and all the possible actions are not present, a new classifier is inserted to cover the action missing.
Unlike the minority game, all the agents are preserved throughout the experiment, no elimination nor reproduction of agents happens in the experiments here reported. Furthermore, the agents do not have predefined strategies, but they choose and evolve their own, according to their needs.

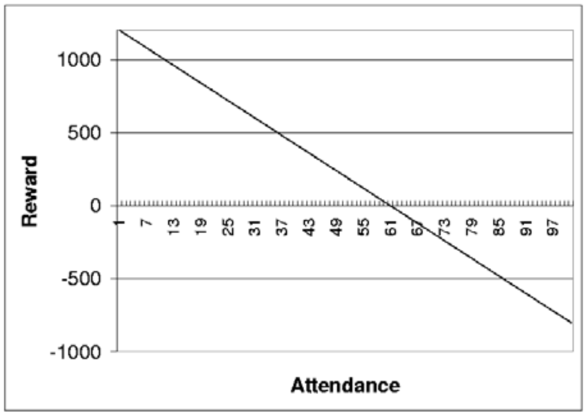

Fig. 3. The Minority Game reward function for Λ_{cm}=0.6

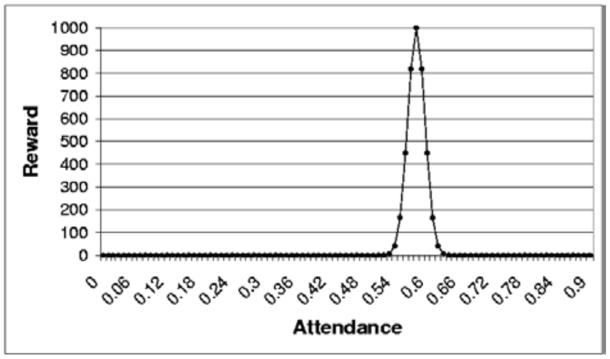

Fig. 4. The "peak" reward function for Λ_{cm}=0.6

The original minority game reward function $R = -a_i * (\sum a_i - A_0)$ e.g. A_0=20 for Λ_{cm}=0.6, A_0=0 for Λ_{cm}=0.5 ; a_i is the action of $classifier_i$ (actions are considered 1 ,-1 instead of 1,0 for this function), plotted in Fig. 3. This function based on the attendance rate has been found misleading for MAXCS, producing random behaviour. This might be explained because the MG function does not give any reward to the system when the attendance equals Λ_{cm}. Then, based on the assumption that a LCS works similarly to a neural network(NN) [34], a different reward function has been used. The new reward function, referred as "peak function" from now on, was inspired by the NN back-propagation error update function: $R = (e^{(attendance - \Lambda_{cm})2} * 1000)^{-1}$; plotting this function (see Fig. 4) it can be seen that it has only one maximum, corresponding to the comfort threshold. The peak function lets the system converge faster when it approaches the desired Λ_{cm} value. The function has been biased intentionally and if the attendance is very low or very high the system will get no reward. This bias resembles the original "El Farol" Bar problem, in the sense that it will keep the attendance to an enjoyable level [3].

6 Results and discussion

Each set of results is presented as a single run of 5000 time steps. The graphics are not plotted as lines, since the values are scattered, discontinuous. Each figure presented shows results for one run and the scattered points represent the attendance as a percentage of the total number of agents.
It has been found that the behaviour of the agents is very extreme when the Λ_{cm} values are close to 0 and 1. Due to the threshold value, they either go or not go. A more interesting behaviour of the population as a whole can be observed for the thresholds in the central interval(Λ_{cm}=[0.3,0.7]).
All the possible Λ_{cm} values were tested in 10 different experiments for each type of reward. MAXCS solved all the problems correctly. After analysing all the results, those obtained for the Λ_{cm}=0.0, 0.3, 0.6, 1.0 values have been chosen. The behaviour observed for these Λ_{cm} values are representative of the system's performance.
The populations of the agents at week 2000 for these experiments, reflect interesting rules, which can explain the smooth behaviour of the system.
For better comprehension the results have been divided in three: 1) those using cooperative reward, 2) those using selfish reward and 3) the emergence of vacillating "altruistic" behaviour in one of the agents in the system using selfish reward.
In order to analyse what happens once the agents have explored and exploited their rules for 2500 time steps, the exploration and the GA was switched off. The Λ_{cm}=0.0,0.3,0.6,1.0 values were tested in 10 different experiments for each type of reward.
The results are displayed throughout Figs. 5 - 12, the right graphic is filtered from the left one, to show only the exploitation trials. The GA and exploration are switched off at step 2500, after that the rules that the agents have learnt are tested.

Tables show the sum of the individual attendance every 100 exploitation trials, that is, each row corresponds to 100 exploitation trials per XCS. The individual answer for each agent (Tables 4,7) can be matched with the environmental state (Tables 3,6) that each agent perceives at a certain time step.

6.1 Cooperative reward

The results show that the optimal percentage of the population= Λ_{cm} learns to go and the rest refrain from attending, in the case of the cooperative reward.
It can be seen from the figures 5-8, when the cooperative reward is used, the system performs an exploitation trial, the performance converges very fast. Random behaviour can be observed when all the trials are taken, i.e. the left side graphics. The reason for this behaviour is that the system is exploring, i.e. all the agents are taking random decisions at the same time.
For all the experiments certain agents go more than others. It has been observed after the experiments with Λ_{cm}=0.0, 0.3, 0.6 and 1.0 using the cooperative reward scheme, that some agents tend to go more than others (see Table 5). There are agents which definitely do not attend the bar.
MAXCS adapts properly for all the values of Λ_{cm} tested. Certain Λ_{cm} values are difficult for the system to learn. It takes longer for the system, to adapt to Λ_{cm}=[0.3, 0.7] values, though. Looking at Fig. 6, after 1000 weeks, the system has almost converged to the desired Λ_{cm}=0.3. Surprisingly Λ_{cm}=0.6 and Λ_{cm}=0.5 have been particularly hard for MAXCS to learn.
One would expect that Λ_{cm}=0.3 would be harder to learn than Λ_{cm}=0.6, since the threshold allows less agents to attend for the former and more for the latter value. As it can be seen from Figs. 6 and 7 when MAXCS exploits, it adapts faster for Λ_{cm}=0.3 than Λ_{cm}=0.6.
The analysis of the results for Λ_{cm}=0.3 have been chosen, based in the performance of the system for both reward schemes. The system showed, for this especific value very interesting behaviour using the selfish reward. The results obtained using the cooperative reward are shown to establish a comparison.
For $\Lambda_{cm} > 0.5$ the system shows a more static behaviour, only the percentage that satisfies Λ_{cm} attends the bar and the rest stay at home. This can be explained by analysing the reward function. The reward function for any Λ_{cm} value, the cooperative reward will give to all the agents: 818 for attendance = Λ_{cm}-0.1, 1000 for attendance = Λ_{cm} and 0 otherwise. That is, all the agents are rewarded only when the attendance equals Λ_{cm} or $\Lambda_{cm} - 0.1$.
The peak function and the cooperative reward stimulate convergence toward static behaviour. Let us suppose that the exact percentage of the population attends the bar several times, during the exploration trials. Then, the classifiers of all the agents, reponsible for that behaviour, will receive 1000 and after that, all the agents will tend to follow those classifiers most of the time. This can be seen in Table 3: the correct answer for the previous weeks (in the column "State") is always 1. Co-relating the rows in Table 4 for these states, it can be seen that it is only agents 1, 4 and 5 that are going to the bar every week. This phenomenon is even more clear in Table 5, where the sum of the attendance is always 100 for these three agents. At row 12, when exploration is turned off, it is clear that there are only 3 agents going and the rest stay at home. It is always only the exact number of agents that attend the bar for every experiment, though the agents are different; for experiment 1 the agents that go are 1, 4 and 5; for experiment 2 the agents that go are 3, 8 and 9, etc. This is due to the independent learning. The system converges very fast when exploiting, as it can be observed in the right graphic of Fig. 6, after step 1000 the system has reached optimum performance, being the last time the bar is overcrowded at step 700.
From a population extract(not shown due to space reasons), agents seem to pay attention to what happened 3 weeks ago. Most of the agents have a rule with the ##1## or ##0## condition, this means that the agents are learning to go every third week. Another point to note is that two rules with condition ##### (one for each action) appear in all the populations, despite having high numerosity, they don't take over the population in the end. One of the fully general classifiers –depending on what the agent is doing– has a higher prediction and fitness. Evaluating the states and the action that XCS would choose confirm that these very general rules, by their numerosity and fitness balance the agent's decision, but they do not guide it: the more specific rules balance the system.

Step	State	Attendance	Correct Answer
4700	11111	0.3	1
4701	11111	0.3	1
4702	11111	0.3	1
4703	11111	0.3	1
4704	11111	0.3	1
4705	11111	0.3	1
4706	11111	0.3	1

Table 3. Attendance and states from step 4700 to step 4706 cooperative reward and Λ_{cm}=0.3

Step	Ag. 1	Ag. 2	Ag. 3	Ag. 4	Ag. 5	Ag. 6	Ag. 7	Ag. 8	Ag. 9	Ag. 10	Attendance
4700	1	0	0	1	1	0	0	0	0	0	3
4701	1	0	0	1	1	0	0	0	0	0	3
4702	1	0	0	1	1	0	0	0	0	0	3
4703	1	0	0	1	1	0	0	0	0	0	3
4704	1	0	0	1	1	0	0	0	0	0	3
4705	1	0	0	1	1	0	0	0	0	0	3

Table 4. Agent's individual decisions from step 4700 to step 4705 cooperative reward and Λ_{cm}=0.3

Row	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8	Agent 9	Agent 10
1	20	32	22	51	73	29	27	39	21	38
2	26	24	19	64	82	26	18	57	20	22
3	35	35	23	73	79	26	26	49	27	30
4	20	48	20	80	78	29	23	40	21	17
5	21	77	17	77	86	25	15	20	21	25
3	0	69	26	71	70	29	19	23	25	21
4	17	75	22	77	73	19	21	23	27	16
5	23	76	25	75	72	17	21	24	26	21
6	24	75	27	77	78	27	25	25	24	29
7	20	79	18	75	75	23	17	17	18	28
8	27	54	22	78	70	24	26	21	24	30
9	79	21	24	79	66	20	22	23	18	18
10	86	16	13	87	87	15	15	14	16	14
11	100	0	0	100	100	0	0	0	0	0
12	100	0	0	100	100	0	0	0	0	0
13	100	0	0	100	100	0	0	0	0	0
14	100	0	0	100	100	0	0	0	0	0
15	100	0	0	100	100	0	0	0	0	0
16	100	0	0	100	100	0	0	0	0	0
17	100	0	0	100	100	0	0	0	0	0
18	100	0	0	100	100	0	0	0	0	0
19	100	0	0	100	100	0	0	0	0	0
20	100	0	0	100	100	0	0	0	0	0
21	100	0	0	100	100	0	0	0	0	0
22	100	0	0	100	100	0	0	0	0	0
23	100	0	0	100	100	0	0	0	0	0
24	100	0	0	100	100	0	0	0	0	0
25	100	0	0	100	100	0	0	0	0	0

Table 5. Individual attendance sum for exploitation trials(2500), cooperative reward and Λ_{cm}=0.3

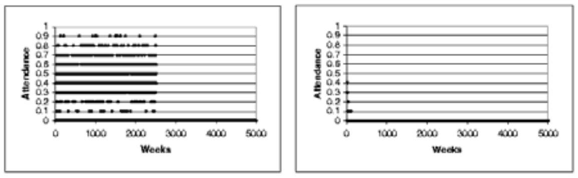

Fig. 5. All (left) and exploitation (right) trials for cooperative reward Λ_{cm}=0.0

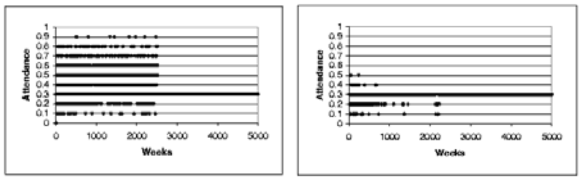

Fig. 6. All (left) and exploitation (right) trials for cooperative reward Λ_{cm}=0.3

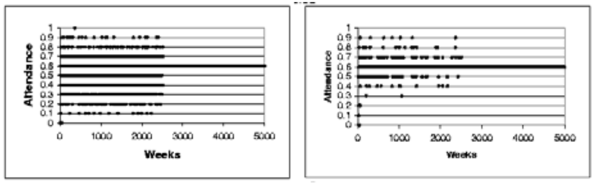

Fig. 7. All (left) and exploitation (right) trials for cooperative reward Λ_{cm}=0.6

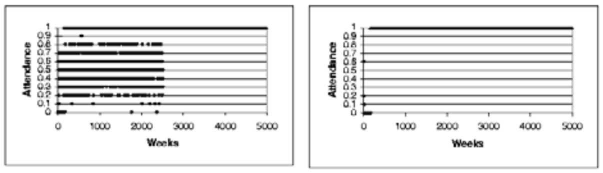

Fig. 8. All (left) and exploitation (right) trials for cooperative reward Λ_{cm}=1.0

6.2 Selfish reward

As stated before, the results analysed are representative of the behaviour observed for all the Λ_{cm} values tested.
In the case of the selfish reward the system adapts very well to the Λ_{cm}=0.0 and Λ_{cm}=1.0 (see Figs. 9 and 12), though the behaviour is not so straight forward for other Λ_{cm} values. It can be seen in Fig. 10, the attendance oscillates between 0.3 and 0.4, when Λ_{cm}=0.3.
Similar oscillations have been found when analysing Λ_{cm}=0.6 (see Fig. 11), the interesting point is that MAXCS oscillates 10% above the Λ_{cm} value.
This behaviour is not really understandable until the reward function is analysed. The peak function for any Λ_{cm} value and using the selfish reward, will give to the goers: 818 for attendance = Λ_{cm}-0.1, 1000 for an attendance of Λ_{cm} and 0 for an attendance greater than Λ_{cm}; on the other hand it will give to the non-goers: 818 for attendance = Λ_{cm}+0.1 and 0 otherwise. In this case, the system as a whole –not individually– is obtaining, the maximum reward possible.
The peak function used for reward has a double effect depending on the reward scheme used: stimulating static behaviour using the cooperative reward, and stimulating attendance and dynamic behaviour using the selfish reward. Let us suppose the same example used for the

cooperative reward. In this case the exact percentage that attend the bar would be getting 1000, while the rest get 0 as a reward. This may lead to a big disaster as all the agents that are not going will tend to go seeking reward, overcrowding the bar. This is the manner that dynamism is induced by the selfish reward.

It must be emphasised that all the agents are kept in the system throughout the whole experiment and there is no reproduction, i.e. if there is an agent that is performing badly it is not disposed from the system.

Comparing table 5 and 8, it can be seen that the selfish reward stimulates dynamism within the population's individual behaviour. A real need for exploration is simulated by the fact that only the agents that take the correct action are rewarded. The richness of the different states perceived by the system using the selfish reward can be observed in table 6 and cannot be compared to the monotonous states that the cooperative reward generates (table 3). Thus, is this richness what makes harder for MAXCS using the selfish reward.

As stated before, all the possible Λ_{cm} values were tested in 10 different experiments and MAXCS solved all the problems correctly. After analysing all the results, those obtained for the Λ_{cm}=0.0, 0.3, 0.6, 1.0 values have been chosen. The behaviour observed for these Λ_{cm} values are representative of the system's performance.

The dynamics of the system under the selfish reward (see table 8) are interesting: a particular kind of alternation has been observed only for Λ_{cm}=[0.1,0.4]. The exact number of agents attend the bar (agents 2, 6, 9 in Table 8), another agent(10) starts going and ends up "displacing" one of the previous agents that goes (agent 2). Then, another agent(4) balances the system, so the agents that go and those staying at home are always rewarded. This behaviour is analysed in detail in the next subsection.

The alternation for Λ_{cm} in [0.5, 1.0] is more static, there is not as much alternance as for the other values, though the agent that brings the balance appears too.

This means that agents could learn how to alternate their attendance, so all of them could go to the bar on a certain week, without having agents that are not attending. It seems that once that MAXCS has found a combination that solves the problem, it keeps repeating it. This behaviour is also favoured by the way the reinforcement is given.

As it can be seen from Figs. 10 and 11 this need is reflected in the general attendance. Especially in the case of Λ_{cm}=0.6 (Fig. 11) the attendance does not fall below 0.5, but 4 out of 5000 possible trials. In all of the possible Λ_{cm} values tested, the overall behaviour of MAXCS is successful: either the agents overcrowd the bar only by 0.1 – so the agents that do not go are rewarded with 818– or the exact number of agents attend, so the agents attending obtain 1000.

Comparing the cooperative and selfish rewards performance (tables 5 and 8) –just after exploration is switched off (row 12)– a more even distribution can be observed in the selfish reward case, converging slowly toward the emergent behaviour, reflected in the attendance of each agent. In the case of the cooperative reward, it can be clearly seen that is only three agents that are going while the rest stay at home, but as stated before, this has to do with the way the reward scheme works.

Deeper analysis of the individual answers for the different experiments are discussed for Λ_{cm}=0.6 and Λ_{cm}=0.3 in the following subsection.

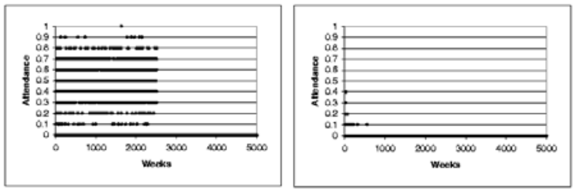

Fig. 9. All (left) and exploitation (right) trials for selfish reward Λ_{cm}=0.0

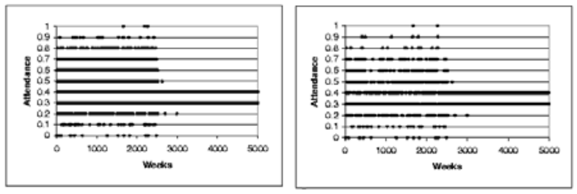

Fig. 10. All (left) and exploitation (right) trials for selfish reward Λ_{cm}=0.3

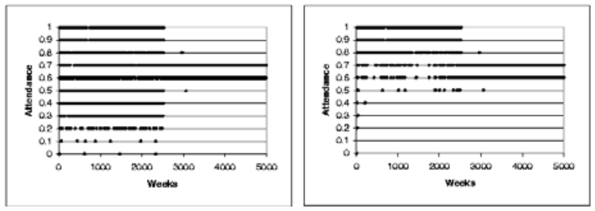

Fig. 11. All (left) and exploitation (right) trials for selfish reward Λ_{cm}=0.6

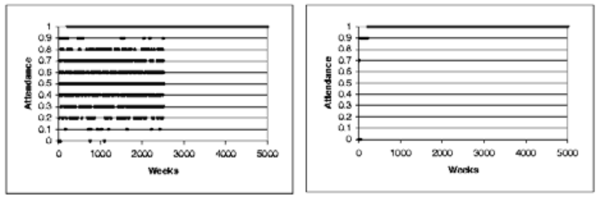

Fig. 12. All (left) and exploitation (right) trials for selfish reward Λ_{cm}=1.0

Step	State	Attendance	Correct Answer
4700	11001	0.3	1
4701	11100	0.4	0
4702	01110	0.3	1
4703	10111	0.4	0
4704	01011	0.3	1
4705	10101	0.3	1
4706	11010	0.4	0

Table 6. Attendance and states from step 4700 to step 4706 selfish reward and Λ_{cm}=0.3

Step	Ag. 1	Ag. 2	Ag. 3	Ag. 4	Ag. 5	Ag. 6	Ag. 7	Ag. 8	Ag. 9	Ag. 10	Attendance
4700	0	0	0	0	0	1	0	0	1	1	3
4701	0	0	0	0	0	1	0	0	1	1	3
4702	0	0	0	1	0	1	0	0	1	1	4
4703	0	0	0	0	0	1	0	0	1	1	3
4704	0	0	0	1	0	1	0	0	1	1	4
4705	0	0	0	0	0	1	0	0	1	1	3

Table 7. Agent's individual decisions from step from step 4700 to step 4705 selfish reward and Λ_{cm}=0.3

Row	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8	Agent 9	Agent 10
1	54	61	33	38	52	39	41	38	52	48
2	37	61	56	51	37	35	42	31	41	36
3	33	57	39	43	50	55	34	44	58	37
4	17	81	31	26	31	84	23	22	79	33
5	54	58	39	32	47	66	56	31	47	46
6	73	51	45	37	36	58	47	28	41	40
7	63	39	60	35	29	36	61	54	32	37
8	43	36	67	44	24	43	64	52	45	42
9	65	63	48	26	32	51	69	32	35	33
10	38	61	63	42	40	51	36	35	39	65
11	28	66	52	64	23	36	36	50	50	56
12	28	44	42	42	33	69	35	52	30	62
13	18	76	28	14	18	84	19	42	12	85
14	0	99	0	0	0	100	0	4	63	85
15	0	98	0	0	0	100	0	0	100	62
16	0	18	0	17	0	100	0	0	100	100
17	0	0	0	34	0	100	0	0	100	100
18	0	0	0	27	0	100	0	0	100	100
19	0	0	0	27	0	100	0	0	100	100
20	0	0	0	39	0	100	0	0	100	100
21	0	0	0	50	0	100	0	0	100	100
22	0	0	0	40	0	100	0	0	100	100
23	0	0	0	49	0	100	0	0	100	100
24	0	0	0	40	0	100	0	0	100	100
25	0	0	0	49	0	100	0	0	100	100

Table 8. Individual attendance sum for exploitation trials(2500), selfish reward and Λ_{cm}=0.3

6.3 Emergent behaviour in MAXCS

For all the experiments with selfish reward and $0 < \Lambda_{cm} < 1$: there is an agent – called the "vacillating" agent (shown as agent 4 in Tables 7 and 8)– that is always taking the wrong decision: it overcrowds the bar or it stays at home when the *attendance* $<= \Lambda_{cm}$. This phenomenon does not happen when the cooperative reward is used, as it can be seen in Table 4.
Considering that the agents are rewarded when they attend the bar and the attendance (att) $att <= \Lambda_{cm}$ or when they stay at home and $att >= \Lambda_{cm}$. From Table 7 can be seen clearly that agents 6, 9 and 10 are going every week; while 1, 2, 3, 5, 7 and 8 are not. This phenomenon would not arise if agent 4 was no overcrowding the bar from time to time. This behaviour can be observed also for $0 < \Lambda_{cm} < 1$ (see Fig. 11 for $\Lambda_{cm} = 0.6$). We will refer to this behaviour as the "vacillating" agent(VA).
After analysing the structure of the rules of the VA, they look quite similar to other agent's rules. There is no particular discernible structure that directs the behaviour of the VA, but its behaviour is directed by reacting to the states in the environment.
There are two features about the VA: One is the result of its existence, the other how and why does it behaviour arise.
The reason for the existence of the VA can be explained from the behaviour of MAXCS as a whole: the vacillating agent balances the whole system. If the agent would not be vacillating, all the agents that decide to stay at home all the time would have to try to go to the bar, otherwise they would not get any reward. Furthermore, the agents that are going all the time would not be rewarded if the vacillating agent was not staying at home, since the bar would always be overcrowded. It can be observed, that the system converges in the late stages toward the vacillating behaviour. In the case of Λ_{cm}=0.3 the agents that attend receive 1000 each, while when Λ_{cm}=0.4, the agents that do not go receive 818 and the VA receives 0.
Analysing thoroughly the rewards obtained by different agents, it can be observed that the VA (agent 4) is getting rewards most of the time for staying at home, during the exploration trials. VA's performance is quite good getting 818 most of the time (equivalent for not going and the bar is overcrowded by 10%).
Just after the exploration and GA are switched off (step 2500), the attendance is spread among the agents, most of them attending the bar and getting rewarded (rows 13-15 in Table 8). It is by learning that the agents arrive to the stationary behaviour. A very interesting feature has been observed: there is only one VA in the system at a time, though it is not the same throughout the whole experiment. Firstly, agent 10 takes this role(row 12 in Table 8), the agents that go at this point are agents 2,6 and 9; after step 2983 (row 14 in Table 8) it is agent 2 which leaves its place to agent 10; agent 2 gradually stops attending and then, after step 3083 (row 16 in Table 8) agent 4 starts going and the balance is achieved. The varied alternance of VA described, has been noticed for $\Lambda_{cm} <= 0.4$. For $\Lambda_{cm} < 0.8$ the behaviour is more rigid, and the alternance does not appear. The reason for the different behaviour may be that $\Lambda_{cm} > 0.5$ is quite hard to satisfy and the system can start oscillating easier than with a smaller value of Λ_{cm}.
The appearance of the "vacillating" agent, and the behaviour of all the rest can be explained by emergent behaviour, because none of these actions have been preset in the system, though they arise as a consequence the interaction of the agents with their environment and, indirectly through the environment, with each other.
In the end, the "vacillating" agent will not get any reward at all, but this does not happen before exploration has stopped. Since the VA alternate their role, all agents can get rewards until the system settles. This behaviour has been observed for all the experiments using the selfish reward and Λ_{cm}=[0.1,0.9].
The patterns and behaviour observed in these experiments are not as stochastic, but more uniform and less complex than in the original work [3]. It can be argued that the reason for this behaviour could be the reward function used, since it is more smooth and encourages the attendance to the bar to the extent of making it enjoyable.
It could be argued that the agents should be encouraged to attend at least once a week, but in real life is not like that, there are people who decide to go to a certain bar and after a bad experience they decide not to go ever again.

7 Conclusions

MAXCS is able to solve the "El Farol" bar problem. Experiments with 10 agents against all the possible values prove that MAXCS adapts properly to the threshold set. Furthermore, emergent behaviour has been detected by analysing the XCS population reports.
The ability to read a snapshot of the "mind" of the agents at a certain time step or moment of the problem has proved to be essential to understand the behaviour of the system. This feature stresses the advantage of using MAXCS for social modelling.
The cooperative reward allows MAXCS to perform better than the selfish reward, but this is due to the nature of the problem: MAXCS has to explore less number of possibilities to adapt. By getting the correct attendance several times, the rules are reinforced enough to keep a correct –though static– behaviour. From the behaviour observed, it is certain that MAXCS can adapt to all the thresholds that it was set to. The behaviour is quite extreme, either there are alcoholics or abstemious, and the vacillating agent, whose attendance is not as regular as the alcoholics'.
These experiments prove, to some extent, the feasibility of using XCS as the learning engine of a MAS adapting to a complex environment. So far, the performance of MAXCS on the "El Farol" bar problem is encouraging.
Further work is planned with MAXCS and "El Farol" problem, experiments with other parameters: a bigger number of agents, different θ_{GA} values, varying Λ_{cm} etc. MAXCS will be tested in other benchmark problems.

Acknowledgements

The first author would like to thank the Consejo Nacional de Ciencia y Tecnología for their financial support.

References

1. A.Bonarini. *Learning Classifier Systems: From foundations to applications*, chapter An introduction to learning fuzzy classifier systems, pages 83–104. Lecture notes in computer science; Vol. 18113: Lecture notes in artificial intelligence. Springer-Verlag, 2000.
2. M. Ahluwalia and L. Bull. A genetic programming-based classifier system. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 11–18. Morgan Kaufmann: San Francisco, CA, 1999.
3. W. B. Arthur. Complexity in economic theory: Inductive reasoning and bounded rationality. *The American Economic Review*, 84(2):406–411, May 1994.
4. A. Barry. *XCS Performance and Population Structure within Multiple-Step Environments.* PhD thesis, Queens University Belfast, 2000.
5. B.Carse, T.C. Fogarty, and A. Munro. Evolving rule based controllers using genetic algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, 1995.
6. A. Bonarini. Reinforcement distribution for fuzzy classifiers: a methodology to extend crisp algorithms. In *Proceedings of the IEEE Conference 1998.* IEEE, 1998.
7. L. Booker. *Learning Classifier Systems: From foundations to applications*, chapter Do we really need to estimate rule utilities in classifier systems?, pages 125–141. Lecture notes in computer science; Vol. 18113: Lecture notes in artificial intelligence. Springer-Verlag, 2000.
8. L. B. Booker. Improving the performance of genetic algorithms in classifier systems. In D.Schaffer, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 80–92, 1985.
9. L. Bull. On zcs in multi-agent environments. In Springer, editor, *Parallel Problem Solving from Nature V*, 1998.
10. D. Challet, M. Marsili, and R. Zecchina. Statistical mechanics of systems with heterogeneous agents: Minority games. *Phys. Rev. Lett.*, (84), 2000.
11. D. Challet and Y. C. Zhang. Emergence of cooperation and organization in an evolutionary game. *Physica A*, 246:407, 1997.
12. D. Challet and Y. C. Zhang. On the minority game: Analytical and numerical studies. *Physica A*, 256:407, 1998.
13. M. Dorigo and H. Bersini. A comparison of q-learning and classifier systems. In *Proceedings of From Animals to Animats,Third International Conference on Simulation of Adaptive Behaviour*, 1994.
14. B. Edmonds. Gossip, sexual recombination and the el farol bar: modelling the emergence of heterogeneity. *Journal of Artificial Societies and Social Simulation*, 2(3), 1999. http://www.soc.surrey.ac.uk/JASSS/2/3/2.html.
15. T. Eymann, B. Padovan, and D. Schoder. Artificial coordination- simulating organizational change with artificial life agents. Technical report, University of Freiburg, 1998.
16. J. Ferber. *Multi-Agent Systems: An introduction to distributed artificial intelligence.* Adisson Wesley, 1999.
17. D.A. Fisher and H.F. Lipson. Emergent algorithms – a new method for enhancing survivability in unbounded systems. In *Proceedings of the 32th Hawaii International Conference on System Sciences*, 1999.
18. D.E. Goldberg. *Genetic algorithms in Search, optimization and Machine Learning.* Addison Wesley, 1989.
19. D.E. Goldberg, J. Horn, and K. Deb. What makes a problem hard for a classifier system? Technical Report 92007, Illinois Genetic Algorithms Laboratory, May 1992.
20. L. Miramontes Hercog and T. C. Fogarty. Xcs-based inductive intelligent multi-agent system. In *Late Breaking Papers*, pages 125–132. Genetic and Evolutionary Computation Conference, 2000.
21. L. Miramontes Hercog and T. C. Fogarty. *To appear*, chapter Analysing Inductive Intelligence in a XCS-based Multi-Agent System (MAXCS). To appear, 2001.
22. J.H. Holland. *"Adaptation in Natural and Artificial Systems"*. University of Michigan Press, 1975.

23. J. Horn, D.E. Goldberg, and K. Deb. Implicit niching in a learning classifier system: Nature's way. *Evolutionary Computation*, 2(1):37–66, 1994.
24. T. Kovacs. Xcs classifier system reliably evolves accurate, complete, and minimal representations for boolean functions. Technical report, University of Birmingham, 1997.
25. J.R. Koza. *Genetic Programming.* MA: The MIT Press/Bradford Books, 1992.
26. P.L. Lanzi. Generalization in wilson's classifier system. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 501–509, 1997.
27. P.L. Lanzi. Adding memory to xcs. Technical report, Politecnico di Milano, 1998.
28. L.Bull, T.C. Fogarty, and M. Snaith. Evolution in multi-agent systems: Evolving communicating classifier systems for gait in a quadrupedal robot. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, 1995.
29. L. Miramontes Hercog. Hand-eye co-ordination: An evolutionary approach. Master's thesis, University of Edinburgh, 1998.
30. A. Pérez-Uribe and B. Hirsbrunner. The risk of exploration in multi-agent learning systems: A case study. In *Proceedings of the AGENTS-00*, 2000.
31. T. Sandholm and V.R. Lesser. Coalition formation among bounded rational agents. Technical Report 95-71, University of Massachusetts at Amherst, 1995.
32. S. Saxon and A. Barry. *Learning Classifier Systems: From foundations to applications*, chapter XCS and the Monk's problem, pages 272–281. Lecture notes in computer science; Vol. 18113: Lecture notes in artificial intelligence. Springer-Verlag, 2000.
33. W. Shen. *Autonomous learning from the enviroment.* Computer Science Press, 1994.
34. R.E. Smith and H. Brown Cribbs. Is a learning classifier system a type of neural network? *Evolutionary Computation*, 2(1):19–36, 1994.
35. C. Watkins. *Learning from Delayed Rewards.* PhD thesis, Cambridge University, U.K., 1989.
36. R. Weil, A. García-Ortiz, and J. Wootton. Detection of traffic anomalies using fuzzy logic based techniques. In *Proceedings of the IEEE Conference 1998*, 1998.
37. S. W. Wilson. Mining oblique data with xcs. In P.L. Lanzi, W. Stolzmann, and S.W.Wilson, editors, *Proceedings of the 3rd International Workshop on Classifier Systems 2000.* Springer-Verlag, 2000.
38. S.W. Wilson. Zcs: A zeroth level classifier system. *Evolutionary Computation*, 2(1):1–18, 1994.
39. S.W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
40. S.W. Wilson. Generalization in the xcs classifier system. In *From Genetic Programming 1998: Proceedings of the Third Annual Conference.* The Rowland Institute for Science, Morgan Kaufmann, 1998.
41. S.W. Wilson and D.E. Goldberg. A critical review of classifier systems. In *Proceedings of the Third International Conference on Genetic Algorithms*, 1989.
42. D.H. Wolpert, K. R. Wheeler, and K. Tumer. Collective intelligence for control of distributed dynamical systems. Technical report, NASA Ames Research Centre, 1999.
43. S. Zhang, S. Franklin, and D. Dasgupta. Metacognition in software agents using classifier systems. Technical report, The University of Memphis, 1997.