Social Simulation Using a Multi-agent Model Based on Classifier Systems: The Emergence of Vacillating Behaviour in the "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 reward distribution technique. When using selfish reward distribution a vacillating agent emerges which, although obtaining no reward itself, enables 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.
Analysis of the values of the rules' performance show that it is the amount of reward received by each XCS combined with its reinforcement mechanism which cause the emergent behaviour.
MAXCS has proved to be a good modelling tool for social simulation, both because of its performance and providing the explanation for the actions.
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) [15,31] that learns using XCS [38] – is used for social modelling on the"El Farol" Bar problem [2]. The aim of this paper is to analyse the usefulness of a LCS as the learning agent for a MAS in social modelling.

P.L. Lanzi, W. Stolzmann, and S.W. Wilson (Eds.): IWLCS 2001, LNAI 2321, pp. 88–111, 2002.

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 [35]. 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 parameter 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.

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 (EFBP)

A good benchmark for multi-agent systems is the "El Farol" Bar Problem (EFBP), invented by B. Arthur[2]. 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

[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[16].

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 EFBP was stated, many approaches have followed – most of them used for economics - due to its formalisation, as an evolutionary game: the Minority Game [12,11,13]. 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). The agents are initialised with a random number of rules and they only update a value that estimates how good the rule is. The GA works periodically every several rounds(100). The fittest agent is cloned and its rules mutated and the measure initialised to 0 to replace the worst performing agent in the system.

The results reported for EFBP and the MG are oscillating lines around the comfort threshold, with an average close or equal to the threshold selected. However, there are no reports of what the strategies used by each agent (for the MG), nor reports for different thresholds.

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 [30,40]. The most interesting approach used genetic programming [26] to solve the problem by letting 16 agents communicate with each other [14]. The fact that communication is allowed destroys the very nature of the problem itself.

2.1 Why Is this Problem Difficult to Learn?

Unlike other games, the EFBP has no Nash equilibrium, i.e. there is no ideal situation or answer that the agents can choose, the solution must arise from the interaction between the agents.

The EFBP 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 difficult 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 to solve the problem partially.

EFBP is a single-step problem, because the answers of the agents are taken, the attendance is computed and the reward is distributed immediately. The problem is non-Markovian, i.e. the environmental perception of the agent is not

sufficient to solve the problem. It is also non-stationary, because it is a very dynamic environment. The complexity of the problem lays in accomplishing the correct number of agents in the bar, that depends in the action taken by each of the agents. For this problem the same state perceived by the agents does not have the same correct answer.

In the minority game, the abstraction is extreme: the agents cannot see the attendance figure, but can only see one bit 0 or 1; the former if it was good and the latter not good 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) [22,17], invented by Wilson in 1995 [38]. 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 [36], real intervals, fuzzy rules [3,5,4] and even S-expressions [1](LISP expressions which may resemble genetic programming[26]).

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.

[2] There is a LCS that does not use a performance indicator, based on Holland's ECHO systems [22], questioning whether there should be a performance indicator or not [6].

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 reinforcement learning algorithm is used to update the three parameters mentioned above (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 have the same condition and action, there will be only one with numerosity n.

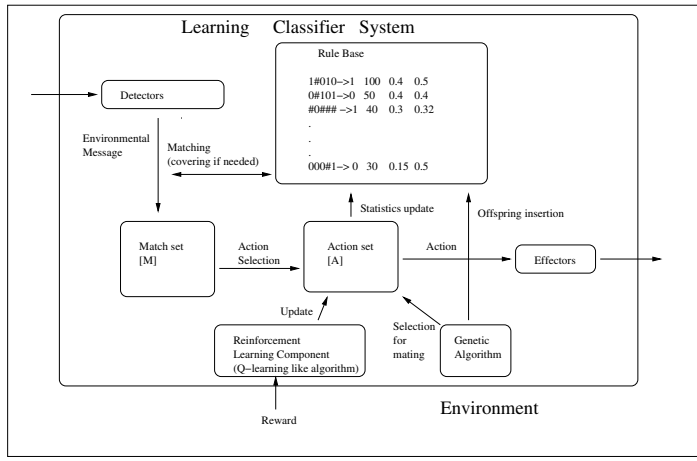

Fig. 1. Functional diagram of XCS

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 [39] 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, for a single-step problem, gives the reward to XCS. As explained before, XCS uses a reinforcement learning algorithm [33] to update the p, ϵ and F values for all the classifiers in [A] in single-step problems and $[A]_{-1}$ in 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.

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

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	$F_j \leftarrow F_j^{'} + \beta(\kappa_j^{'} - F_j^{'})$	

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 [M] is null, then 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 [19]). 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], instead of the whole population [7], so it imposes a selection pressure as well as an implicit niching technique [18,23]. 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. When there is only one classifier in [A], cloning and mutation take place. The children are inserted in the population. If the maximum population size is reached, some classifiers are deleted based on their experience and fitness (see [24] for deletion techniques).

3.3 Motivation

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] [24,28].

XCS has been tried in several test problems: the multiplexer[17], "woods" environments [37], some mazes [27], the monk's problems [32] and lately it has been used for data mining as a real world problem [36], 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 due to the lack of an internal memory, XCS cannot cope with aliasing sates. The bit register memory introduced by Lanzi [29] is not a feasible solution for this particular problem, as it is discussed in the results section.

A non-Markovian, non-stationary problem as a multi-agent system, where the answer to the problem does not depend uniquely on the actions by the agent itself, but by all the agents, is a great challenge for LCS and XCS in particular.

4 MAXCS: A Multi-agent System that Learns Using XCS

The idea of representing an agent as a LCS [9,8,41] – 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[15].

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 cognitive and 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

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

Table 2. Comparison of the agent and XCS characteristics

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

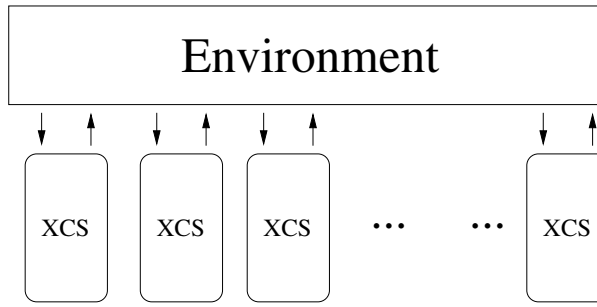

Fig. 2. Diagram of MAXCS

the cognitive agent definition. It can memorise the situations in a very simple way. Using the accuracy measure XCS can foresee the 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 [4,9].

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 at the same time. Interestingly, setting the explore or exploit at the same time and at different time for each agent yielded the same results.

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 leftmost bit and what was good to do 5 weeks ago with the rightmost bit.

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 if 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, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 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 is used only at GA level. The effector covering is enabled(see section 3.2).

Unlike in the EFBP and MG, all the agents are preserved throughout the experiment, no elimination nor reproduction of agents happens in the experiments here reported. The agents start with random generated strategies, that they evolve, according to their needs.

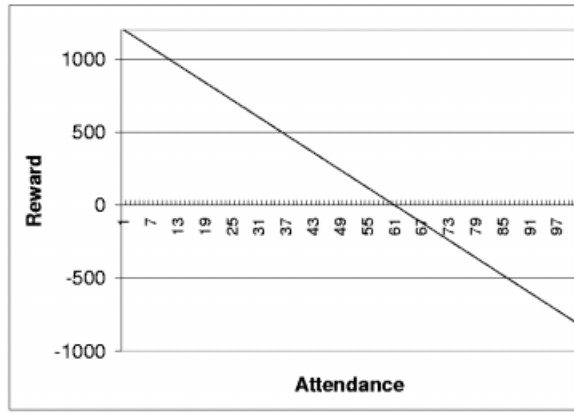

Fig. 3. The Minority Game reward function for Λ_{cm}=0.6

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 func-

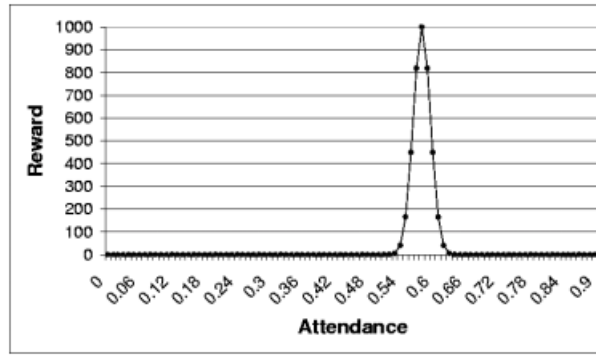

Fig. 4. The "peak" reward function for Λ_{cm}=0.6

tion, referred as "peak function" from now on, was inspired by the NN back-propagation error update function:

$R = 1000/(e^{(attendance-\Lambda_{cm})2} * 1000)^2$; plotting this function (see Fig. 4) it can be seen that it has only one maximum, corresponding to the comfort threshold. The NN inspiration refers to squaring and escalating the error, represented in this equation by $attendance - \Lambda_{cm}$.

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 EFBP, in the sense that it will keep the attendance to an enjoyable level [2].

An important remark that has to be made is, that when using 10 agents, the peak function would give only 1000 or 0 as rewards. In order not to lose the idea of the layered payoff of the peak function, a 679 reward is introduced for $attendance = \Lambda_{cm} + 1$ and $attendance = \Lambda_{cm} - 1$.

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 and 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 exact 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}=[0.1,0.9]).

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.

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(this leaves the LCS without any rule discovery means). 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. 7 - 10, the right graphic is filtered from the left one, to show only the exploitation trials.

Tables 3 and 5 show the sum of the individual attendance every 100 steps only for exploitation trials, that is, each row corresponds to 100 possible exploitation trials; e.g. if the number in the table is 2, that the agent decided to go 2 times out of 100 possible, if the number is 0, it means it never went. A sample of the correlation between the individual answer for each agent and the environmental state they perceive is presented in tables 4 and 6.

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.

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. Therefore MAXCS solves the problem.

It can be seen from the figures 7-8, 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 (exploration and exploitation) 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.

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 3) and 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. 5, after 1000 weeks, the system has almost converged to the desired Λ_{cm}=0.3.

The analysis of the results for Λ_{cm}=0.6 (Fig. 6)have been chosen, based in the performance of the system for both reward schemes. The system showed, for this specific 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: 679 for attendance =

Table 3. Individual attendance sum only for exploitation trials, cooperative reward and Λ_{cm}=0.6

Row	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8	Agent 9	Agent 10
1	15	22	25	11	37	18	17	16	30	37
2	13	36	36	26	44	15	35	15	7	49
3	28	15	16	39	12	34	45	16	30	45
4	39	10	14	20	31	41	16	9	33	41
5	32	6	11	34	11	33	30	34	15	38
6	30	25	13	19	29	33	33	32	21	45
7	21	40	10	9	24	35	20	32	18	39
8	53	57	19	7	57	54	4	21	2	56
9	9	8	44	8	47	35	46	13	28	43
10	40	0	54	2	56	57	57	20	53	2
11	0	1	45	0	46	46	46	46	46	1
12	0	0	55	0	55	55	55	55	55	0
13	0	0	53	0	53	53	53	53	53	0
14	0	2	53	29	53	21	53	53	53	1
15	19	32	36	36	34	6	19	49	45	4
16	35	50	50	35	22	20	2	35	52	21
17	2	47	50	8	51	49	5	2	47	51
18	25	2	17	48	1	2	32	49	46	48
19	1	53	50	3	1	54	2	50	54	55
20	5	51	51	4	5	48	4	49	52	51
21	24	5	43	37	44	40	26	27	25	3
22	33	47	0	6	46	38	53	42	2	54
23	51	26	0	24	51	25	51	51	1	27
24	49	0	1	49	49	49	49	29	20	0
25	59	2	33	26	4	59	59	1	59	55
26	100	0	100	0	0	100	100	0	100	100
27	100	0	100	0	0	100	100	0	100	100
28	100	0	100	0	0	100	100	0	100	100
29	100	0	100	0	0	100	100	0	100	100
30	100	0	100	0	0	100	100	0	100	100
31	100	0	100	0	0	100	100	0	100	100
32	100	0	100	0	0	100	100	0	100	100
33	100	0	100	0	0	100	100	0	100	100
34	100	0	100	0	0	100	100	0	100	100
35	100	0	100	0	0	100	100	0	100	100
36	100	0	100	0	0	100	100	0	100	100
37	100	0	100	0	0	100	100	0	100	100
38	100	0	100	0	0	100	100	0	100	100
39-49	100	0	100	0	0	100	100	0	100	100

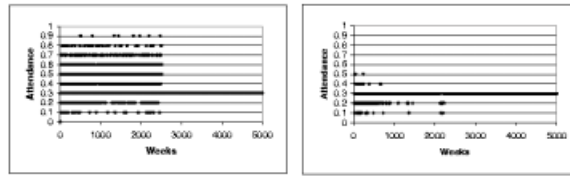

Fig. 5. All (left) and exploitation (right) trials for cooperative reward Λ_{cm}=0.3

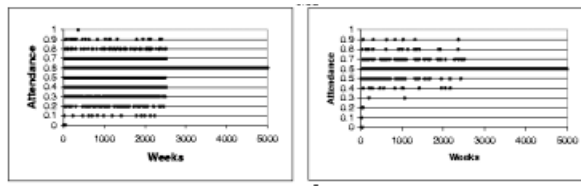

Fig. 6. All (left) and exploitation (right) trials for cooperative reward Λ_{cm}=0.6

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 $\Lambda_{cm} - 0.1$.

Table 4. Agent's individual decisions from step 2900 to step 2930 cooperative reward and Λ_{cm}=0.6

Step	Ag. 1	Ag. 2	Ag. 3	Ag. 4	Ag. 5	Ag. 6	Ag. 7	Ag. 8	Ag. 9	Ag. 10	State	Reward	Att.
2900	1	0	1	0	0	1	1	0	1	1	11111	1000.0	0.6
2901	1	0	1	0	0	1	1	0	1	1	11111	1000.0	0.6
2902	1	0	1	0	0	1	1	0	1	1	11111	1000.0	0.6
2903	1	0	1	0	0	1	1	0	1	1	11111	1000.0	0.6
2904	1	0	1	0	0	1	1	0	1	1	11111	1000.0	0.6
2905	1	0	1	0	0	1	1	0	1	1	11111	1000.0	0.6
2906	1	0	1	0	0	1	1	0	1	1	11111	1000.0	0.6
2907	1	0	1	0	0	1	1	0	1	1	11111	1000.0	0.6
2908	1	0	1	0	0	1	1	0	1	1	11111	1000.0	0.6
2909	1	0	1	0	0	1	1	0	1	1	11111	1000.0	0.6
2910	1	0	1	0	0	1	1	0	1	1	11111	1000.0	0.6
2911	1	0	1	0	0	1	1	0	1	1	11111	1000.0	0.6
...	...	...	...	...	...	...	...	...	...	...	...	1000.0	0.6
2930	1	0	1	0	0	1	1	0	1	1	11111	1000.0	0.6

Once the exact number of agents attend the bar several times, their rules are reinforced strongly and their behaviour remains static.

The agents that tend to go all the time can be seen clearer in Table 3, where the sum of the attendance is always 100 for these six agents. At row 25, when exploration is turned off, it is clear that there are only 6 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, 3, 4, 5, 7 and 8; for experiment 2 the agents that go are 3, 4, 6, 7, 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. 5, after step 1000 the system has reached optimum performance, being the last time the bar is overcrowded at step 700.

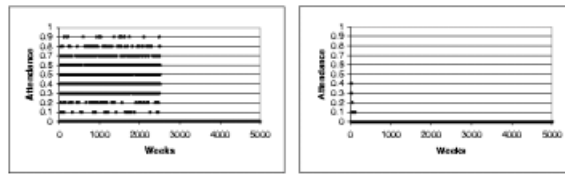

Fig. 7. All (left) and exploitation (right) trials for cooperative reward Λ_{cm}=0.0

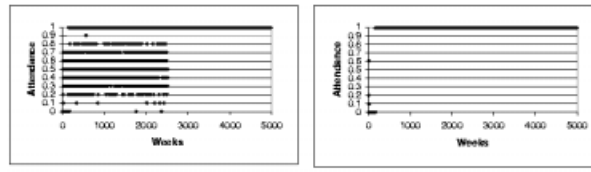

Fig. 8. All (left) and exploitation (right) trials for cooperative reward Λ_{cm}=1.0

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 10), though the behaviour is not so straight forward for other Λ_{cm} values. It can be seen in Fig. 11, the attendance oscillates between 0.6 and 0.7, when Λ_{cm}=0.6.

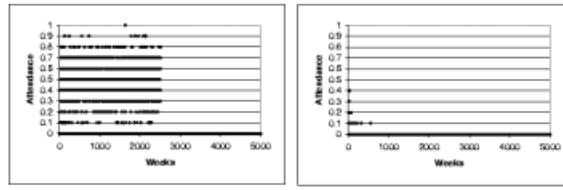

Fig. 9. All (left) and exploitation (right) trials for selfish reward Λ_{cm}=0.0

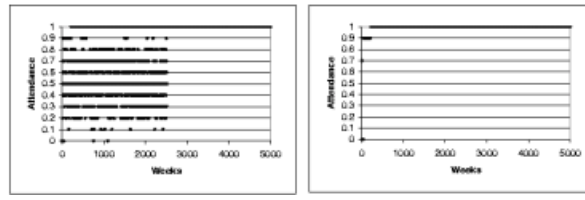

Fig. 10. All (left) and exploitation (right) trials for selfish reward Λ_{cm}=1.0

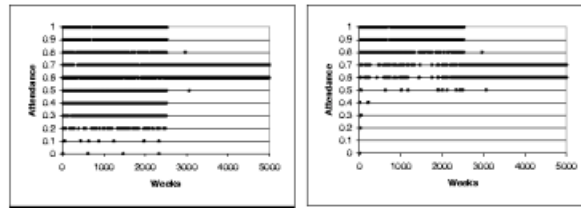

Fig. 11. All (left) and exploitation (right) trials for selfish reward Λ_{cm}=0.6

Similar oscillations have been found when analysing Λ_{cm}=0.3 (see Fig. 12), 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

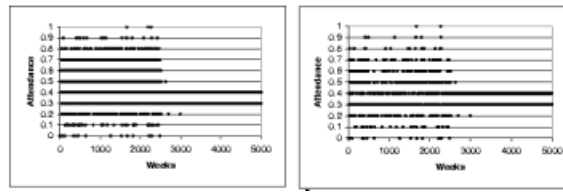

Fig. 12. All (left) and exploitation (right) trials for selfish reward Λ_{cm}=0.3

to the attendees: 679 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 those staying at homes: 679 for attendance=Λ_{cm}+0.1 and 0 otherwise. 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. The reward = 0 could lead all the agents that are not going to go seeking reward, overcrowding the bar.

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 3 and 5, 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 (column "state") and cannot be compared to the monotonous states that the cooperative reward generates (table 4). Thus, is this richness what makes more difficult 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 is representative of the system's performance.

The dynamics of the system using the selfish reward (see table 5) are interesting: a particular kind of alternation has been observed only for Λ_{cm}=[0.1,0.9]. The exact number of agents attend the bar (agents 2, 3, 5, 7, 9, 10 in Table 5), the other agents stay at home except agent 4, which starts going and not going. After some steps agent 4 keeps going to the bar and "alters" one of the previous agents that was not going (agent 2). Agent 2 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.

It seems that once that MAXCS has found a combination that solves the problem, it keeps repeating it, generating loops in the states perceived. This behaviour is also favoured by the way the reinforcement is given.

As it can be seen from Figs. 12 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 679– or the exact number of agents attend, so the agents attending obtain 1000.

Comparing the cooperative and selfish rewards performance (tables 3 and 5) –just after exploration is switched off (row 25)– 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 six agents that are going while the rest stay at home.

Detalied analysis of the individual answers for the different experiments are discussed for Λ_{cm}=0.6 in the following subsections.

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 agents 1 and 4 in Tables 6 and 5)– 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 6 can be seen clearly that agents 2,3,5,7,9 and 10 are going every week; while 1, 6 and 8 are not. This phenomenon would not arise if agent 4 was not overcrowding the bar from time to time. This behaviour can be observed also for $0 < \Lambda_{cm} < 1$ (see Fig. 11 for $\Lambda_{cm} = 0.6$). This behaviour will be referred as the "vacillating" agent(VA).

Analysis of the VA's rules revealed similarity with other agent's populations (Tables 9 8 7). Furthermore, no particular discernible structure 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 its behaviour arises.

The result of the existence of the VA is that the whole system is balanced. If the agent would not be vacillating, all the agents staying at home all the time would have to go to the bar to be rewarded. Moreover, 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 (Fig. 11, Table 5), that the system converges in the late stages toward the vacillating behaviour. In the case of Λ_{cm}=0.6 the agents that

Table 5. Individual attendance sum for exploitation trials, selfish reward and Λ_{cm}=0.6, exploration finishes at row 25

Row	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8	Agent 9	Agent 10
1	57	40	58	31	46	30	53	50	46	23
2	54	34	45	31	50	34	53	53	50	22
3	51	30	49	19	51	27	51	49	51	21
4	53	54	32	54	49	23	31	51	54	53
5	31	46	38	29	44	43	39	42	32	43
6	48	46	45	48	47	48	48	46	42	48
7	30	54	42	30	49	27	51	33	48	55
8	32	40	44	40	41	37	41	37	43	35
9	39	46	40	45	31	39	46	46	46	36
10	47	50	34	49	46	33	47	50	50	31
11	35	50	50	44	50	50	45	50	50	45
12	39	44	39	37	39	44	37	43	41	42
13	51	38	54	54	49	38	34	52	34	51
14	34	39	33	44	51	38	36	49	48	48
15	43	42	42	48	45	47	32	48	30	24
16	47	52	52	46	52	47	40	51	37	41
17	54	55	51	34	52	32	52	47	30	55
18	47	44	48	48	47	44	47	47	48	44
19	47	44	50	50	47	46	50	46	49	50
20	40	49	49	46	48	47	49	38	39	44
21	55	55	55	47	50	55	55	46	53	46
22	46	46	47	37	37	47	46	39	45	36
23	32	46	39	46	42	46	46	36	45	34
24	42	45	45	44	49	30	42	24	34	43
25	43	58	63	65	38	34	65	34	66	65
26	1	100	100	100	65	0	100	0	100	97
27	0	100	100	100	100	0	100	0	100	53
28	0	100	100	92	100	0	100	0	100	65
29	4	100	100	45	100	1	100	0	100	100
30	62	100	100	0	100	0	100	0	100	100
31	67	100	100	0	100	0	100	0	100	100
32	67	100	100	0	100	0	100	0	100	100
33	66	100	100	0	100	0	100	0	100	100
34	67	100	100	0	100	0	100	0	100	100
35	67	100	100	0	100	0	100	0	100	100
36	66	100	100	0	100	0	100	0	100	100
37	67	100	100	0	100	0	100	0	100	100
38	66	100	100	0	100	0	100	0	100	100
39	67	100	100	0	100	0	100	0	100	100
40	67	100	100	0	100	0	100	0	100	100
41	67	100	100	0	100	0	100	0	100	100
42	66	100	100	0	100	0	100	0	100	100
43	67	100	100	0	100	0	100	0	100	100
44	67	100	100	0	100	0	100	0	100	100
45	66	100	100	0	100	0	100	0	100	100
46	67	100	100	0	100	0	100	0	100	100
47	66	100	100	0	100	0	100	0	100	100
48	67	100	100	0	100	0	100	0	100	100
49	67	100	100	0	100	0	100	0	100	100

attend receive 1000 each, while when Λ_{cm}=0.7, the agents that do not go receive 679 and the VA receives 0.

As none of these actions is preset or imposed on the agents, it can be concluded that the VA arises as an emergent property of the system. It has been found that there are three agents that vacillate throughout the experiment. The number of VAs does not vary by stopping the exploration later nor by running the experiment for 10,000 steps.

The patterns and behaviour observed in these experiments are not as stochastic, but more uniform and less complex than in the original work [2]. 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.

6.4 Detailed Analysis of the Vacillating Agent for Λ_{cm}=0.6

The vacillating agents get as reward 0, they are always taking the wrong decision. The decrease in their predictions reflects correctly this fact, and sometimes, it makes the agents change their behaviour.

It can be seen also as an emergent property of the system to balance the resources and the reward that the agents are getting. It can be argued that if the agents were taking turns to go to the bar, it would be even more profitable. This is not possible considering the information perceived by the agents.

Different agents vacillate throughout the experiment. The last agent that vacillates and cannot change its behaviour again is the final vacillating agent. For this example the vacillating agent (1) appears, to be the same until the end at step 2917(see table 6). Agent 4 has been vacillating for several steps 2811-2907 –just the last part of the vacillation appears in the table to analyse the agent 1 behaviour change– until the system arrives to a state where the agents taking action 0 are not rewarded (steps 2908-2916). This has been found as a detonating factor for the agents to change behaviour and become the vacillating agent in all the experiments. Then agent 1 changes its behaviour and it becomes the vacillating agent until the end of the experiment.

Table 6. Agent's individual decisions from step 2900 to step 2930 selfish reward and Λ_{cm}=0.6

Step	Ag. 1	Ag. 2	Ag. 3	Ag. 4	Ag. 5	Ag. 6	Ag. 7	Ag. 8	Ag. 9	Ag. 10	State	Reward	Att.
2900	0	1	1	1	1	0	1	0	1	1	10101	679.0	0.7
2901	0	1	1	0	1	0	1	0	1	1	01010	1000.0	0.6
2902	0	1	1	1	1	0	1	0	1	1	10101	679.0	0.7
2903	0	1	1	0	1	0	1	0	1	1	01010	1000.0	0.6
2904	0	1	1	1	1	0	1	0	1	1	10101	679.0	0.7
2905	0	1	1	0	1	0	1	0	1	1	01010	1000.0	0.6
2906	0	1	1	1	1	0	1	0	1	1	10101	679.0	0.7
2907	0	1	1	0	1	1	1	0	1	1	01010	679.0	0.7
2908	0	1	1	0	1	0	1	0	1	1	00101	1000.0	0.6
2909	0	1	1	0	1	0	1	0	1	1	10010	1000.0	0.6
2910	0	1	1	0	1	0	1	0	1	1	11001	1000.0	0.6
2911	0	1	1	0	1	0	1	0	1	1	11100	1000.0	0.6
2912	0	1	1	0	1	0	1	0	1	1	11110	1000.0	0.6
2913	0	1	1	0	1	0	1	0	1	1	11111	1000.0	0.6
2914	0	1	1	0	1	0	1	0	1	1	11111	1000.0	0.6
2915	0	1	1	0	1	0	1	0	1	1	11111	1000.0	0.6
2916	0	1	1	0	1	0	1	0	1	1	11111	1000.0	0.6
2917	1	1	1	0	1	0	1	0	1	1	11111	679.0	0.7
2918	0	1	1	0	1	0	1	0	1	1	01111	1000.0	0.6
2919	1	1	1	0	1	0	1	0	1	1	10111	679.0	0.7
2920	0	1	1	0	1	0	1	0	1	1	01011	1000.0	0.6
2921	0	1	1	0	1	0	1	0	1	1	10101	1000.0	0.6
2922	1	1	1	0	1	0	1	0	1	1	11010	679.0	0.7
2923	0	1	1	0	1	0	1	0	1	1	01101	1000.0	0.6
2924	1	1	1	0	1	0	1	0	1	1	10110	679.0	0.7
2925	0	1	1	0	1	0	1	0	1	1	01011	1000.0	0.6
2926	0	1	1	0	1	0	1	0	1	1	10101	1000.0	0.6
2927	1	1	1	0	1	0	1	0	1	1	11010	679.0	0.7
2928	0	1	1	0	1	0	1	0	1	1	01101	1000.0	0.6
2929	1	1	1	0	1	0	1	0	1	1	10110	679.0	0.7
2930	1	1	1	0	1	0	1	0	1	1	01011	679.0	0.7

As shown in Table 6 Agent 1 is receiving 679 reward every time that agent 4 takes action 1, because the bar is overcrowded. Step 2908 is crucial, since at this point agent 4 stops going to the bar, taking from then on action 0. From step 2908 until step 2917, the agents taking action 0 receive 0 as reward, since the bar has the exact attendance.

Table 7. Rules of agent 7 at step 2916, prediction avg.=440.3188, fitness avg.= 0.0571

Rule no.	Classifier	P	ϵ	Fitness	Num.	Exp.	[A]Size	timeStamp
1	#####→1	913.2894	266.53976	0.7131391	21	1918	24.81754	2496
2	###0#→1	528.4965	482.1391	0.04860587	2	1039	26.919956	2496
3	###1#→1	921.4973	232.63841	0.082458295	1	849	24.432928	2468
4	#####→0	8.006927	35.460114	0.54073155	24	602	25.974117	2478
5	#01##→1	234.05333	322.5261	0.061474573	2	359	27.372623	2444
6	##000→1	562.779	519.4714	0.044096783	1	199	28.62868	2496
7	##1##→1	793.4511	377.15775	0.24452665	2	504	25.300354	2444
8	#00##→0	157.1317	250.40042	0.04259053	3	66	28.173223	2478
9	###00→1	459.8784	532.9017	0.03268224	1	195	28.762745	2496
10	1#11#→0	0.0	6.9350743	0.7107297	1	15	25.153269	2293
11	##00#→1	781.7744	416.60828	0.083382346	1	199	26.093882	2496
12	###0#→0	7.5382147	35.489113	0.20297545	1	41	26.513472	2478
13	####0→1	850.80255	330.6244	0.20356946	2	331	25.765448	2496
14	0####→1	790.68896	356.83777	0.1907639	1	303	25.732357	2496
15	##0##→0	0.0	14.914063	0.4540896	1	4	27.0	2478

By analysing the rules in Table 8 that match state 11111 before and after the vacillation starts, it can be seen that it is the update of the different values the classifiers have for tracking the performance that determines the change of behaviour. Because there is no covering and no GA applied to the rule set at this point.

Table 8. Rules of agent 1 at step 2916, prediction avg.=99.85586, fitness avg.=0.05727675

Rule no.	Classifier	P	ϵ	Fitness	Num.	Exp.	[A]Size	timeStamp
1	#####→1	79.594185	207.26578	0.46554962	11	1666	22.554201	248
2	#####→0	58.87646	180.9805	0.8118321	22	1177	24.53473	2480
3	1####→1	191.22229	356.85776	0.07090828	3	710	23.259945	2454
4	#1###→1	127.67248	276.75278	0.10058066	2	711	23.053572	2484
5	0####→1	26.972912	102.21316	0.4226791	2	770	21.523392	2484
6	##0##→1	39.979782	133.40735	0.23753978	3	754	21.582243	2468
7	#11##→0	172.57654	281.33066	0.25770864	2	134	24.858635	2480
8	#1#01→0	199.61969	334.89005	0.047840085	1	74	25.507643	2402
9	###1#→1	110.8296	271.1363	0.10387483	4	211	23.069704	2484
10	##1##→1	114.7772	280.54065	0.094019644	3	208	22.80423	2484
11	#11#1→0	219.78224	334.37183	0.046121255	1	36	25.595444	2402
12	###00→1	144.99974	258.79248	0.06815666	1	94	22.374504	2438
13	####0→1	131.8298	267.13098	0.11887836	3	157	23.024149	24841
14	###1→1	335.3514	477.07156	0.013172091	1	64	22.471684	2454
15	###0#→0	320.15088	327.37244	0.20033106	1	262	23.955145	2480
16	0##0#→0	91.64518	230.51915	0.13170789	1	124	24.789629	2480
17	#0###→1	56.2212	176.35359	0.019435128	1	44	21.388403	2468
18	##0##→0	106.08536	201.76761	0.44617447	1	250	23.498508	2429
19	#0##1→1	214.912	425.85266	0.009202877	1	7	21.616	2468

Rules 1,2,3,4,7,9,10,11,14 are the only rules that match the 11111 message. This message is the only message from step 2913 to 2917, therefore the only rules stated previously are the only rules that are updated. The estimate $\frac{\Sigma F_j p_j}{\Sigma F_j}$, i.e. the aggregate for selection for the rules with action 0, for these rules changes from

537.7766 at step 2908 to 91.792 at step 2916. On the other hand, the estimate for action 1 remains constant at 102.7141. This proves that it is the update that causes the vacillation.

Even though the previously mentioned rules are not fired at step 2908, since the state is 00101, the comparison is still valid, as it is done only for those rules matching 11111 at step 2908, i.e. if the state perceived by the agents at step 2908 was 11111 the action taken would be 0.

By step 2916, the rules in agent 4 (Table 9) that have action 0 have higher estimates, and it does not go anymore. So the reaction of agent 1 to stop going in the next step is wrong for state 01111, as it does not receive any reward.

Table 9. Rules of agent 4 at step 2916, prediction avg.=8.57109 fitness avg.= 0.04634809

Rule no.	Classifier	P	ϵ	Fitness	Num.	Exp.	[A]Size	timeStamp
1	#####→0	18.226933	59.23972	0.69201475	13	888	19.03696	2480
2	####0→1	0.106381565	1.1257594	0.05204487	1	1035	23.373425	2500
3	####1→1	0.2564734	2.3591378	0.2421	7	873	24.929897	2486
4	#####→1	1.305638E-4	0.0023387873	0.656811	15	1743	24.640448	2500
5	#1###→0	28.535048	78.836464	0.13354418	6	224	19.41842	2480
6	#1###→1	3.246411	21.889786	0.0018795894	6	614	26.667675	2500
7	###00→1	0.5926582	5.0136166	0.07152896	2	365	23.938295	2500
8	#0###→1	0.01489604	0.18779579	0.043282706	1	584	24.49761	2472
9	#1##0→1	9.619884	53.17525	8.4259675E-4	1	236	26.133728	2500
10	1###1→1	1.1823378	9.309471	0.00909294	1	192	25.218025	2486
11	#1#0#→1	27.947147	118.41909	0.007757831	2	176	30.30118	2500
12	0##00→1	12.650696	65.05956	0.007767346	2	105	27.33095	2423
13	#1#0#→0	3.8224301	19.117569	0.022192683	1	45	22.095205	2480
14	###0#→1	0.007780917	0.102105774	0.043223664	1	321	24.92096	2500
15	#11##→1	16.580875	81.37671	0.002010811	1	90	27.13542	2500
16	#00##→0	5.0619984	28.799244	0.19940239	1	27	15.189486	2429
17	###0#→0	0.16811226	1.4424431	0.5014252	2	35	20.041632	2480
18	#0#0#→0	0.0	0.0	0.2793562	1	15	16.544691	2429

It can be argued that the vacillating agent becomes such because of the strategies that it has followed during the period of exploration. Let us analyse the amount of accumulated reward for each agent, this is a direct representation of their performance. In Table 10 can be seen the evolution of the reward accumulated by each agent throughout the experiment.

Table 10. Accumulated reward for the agents throughout the experiment, sorted by performance

step 2500		step 2908		step 3500		step 5000	
agent 6	410885.0	agent 7	569256.0	agent 6	823717.0	agent 6	1502038.0
agent 7	397577.0	agent 3	566503.0	agent 8	796619.0	agent 8	1474940.0
agent 3	394824.0	agent 6	561623.0	agent 7	782256.0	agent 4	1456140.0
agent 5	389170.0	agent 9	559182.0	agent 3	779503.0	agent 2	1257775.0
agent 9	387503.0	agent 2	544775.0	agent 4	777819.0	agent 7	1282256.0
agent 8	387182.0	agent 8	533846.0	agent 9	772182.0	agent 3	1279503.0
agent 4	386404.0	agent 1	525451.0	agent 5	731244.0	agent 9	1272182.0
agent 10	376836.0	agent 4	517083.0	agent 2	757775.0	agent 5	1231244.0
agent 1	374713.0	agent 5	518244.0	agent 10	676836.0	agent 10	1176836.0
agent 2	373096.0	agent 10	463836.0	agent 1	528167.0	agent 1	528167.0

The most interesting aspect of this experiment, is that as an agent (in this case agent 1, which is a XCS) receives as reward 0 for 8 steps it changes so much a "mental" model to twist it in such a way that it stops operating properly.

It is not a very common behaviour from a learning algorithm to be so drastic toward reward, especially XCS which has been proved before to be very reliable (e.g.[29,25]), though lately it has been discovered [10] that XCS cannot remap the environment if it changes drastically. Still, previous research has focused in simple problems, even if they were non-Markovian. To test XCS in a complex, dynamic, non-Markovian environment brings the LCS research a step forward.

These results confirm that the vacillating agent is an emergent phenomenon of the system, since it is not programmed, but arises from the interaction of the different components(the agents). The way the behaviour is obtained then, is a combination of the states perceived by the agent and the reward obtained from the environment. As the agents change their hypothesis, some become stronger, some become weaker, as seen in the example, despite the little difference between the best performing (agent 7, Table 7) and the vacillating agent.

In case of trying to coordinate the agents by using an internal bit memory. Let us think the extreme case of threshold 0.5, where the alternation of 5 and 5 different agents attending in each turn. How can the agents coordinate, if the only perception they have is 11111, in the case that only 5 of them have attended the pub the previous 5 weeks; it can be argued that this can be solved by an internal memory register, so for example agents 1,2,3,4,5 go at time 2n and agents 6,7,8,9,10 go at time 2n+1. This argument can be easily refuted by analysing the case of a very low threshold, as 0.1, now each agent should go once every 10 times, or once every 100 times in the case of 100 agents, the agents would need a memory size of 100! If thought from a global perspective, the vacillating agent is an "intelligent" (emergent) solution to this problem. In the case of having 100 agents, instead of having to coordinate 100 agents to go each time step, one agent "fixes" its behaviour to go all the time, getting 1000 when the bar is not overcrowded, then the vacillating agent appears, so in this way the 98,(or 8 agents in the case of 10 agents) which stay at home receive a reward. In this case the vacillating agent is left with no reward, to the benefit of the majority.

7 Conclusions

MAXCS is able to solve the EFBP. 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. And by analysing the behaviour, an interesting feature about XCS's internal mechanisms has been found. So far, the performance of MAXCS on the EFBP 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.

The original code for XCS (Java version)[4] written by Martin Butz has been modified in several ways to suit the multi-agent learning engine.

The first author would like to thank the fruitful discussions held with the members of the University of the West of England LCS Group, Martin Butz and Tim Kovacs.

References

1. 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.
2. W. B. Arthur. Complexity in economic theory: Inductive reasoning and bounded rationality. *The American Economic Review*, 84(2):406–411, May 1994. `http://www.santafe.edu/~wba/Papers/El_Farol.html`.
3. 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.
4. A. Bonarini. Reinforcement distribution for fuzzy classifiers: a methodology to extend crisp algorithms. In *Proceedings of the IEEE Conference 1998*. IEEE, 1998.
5. 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.

[4] The code is available at:
`ftp://ftp-illigal.ge.uiuc.edu/pub/src/XCSJava/XCSJava1.0.tar.Z`.

6. 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.
7. 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.
8. L. Bull. On zcs in multi-agent environments. In A.E. Eiben, T. Baeck, M. Schoenauer, and H-P. Schwefel, editors, *Parallel Problem Solving from Nature V*. Springer-Verlag, 1998.
9. L. Bull and 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.
10. L. Bull and J. Hurst. Zcs redux. *Evolutionary Computation Journal*, 2002.
11. D. Challet, M. Marsili, and R. Zecchina. Statistical mechanics of systems with heterogeneous agents: Minority games. *Phys. Rev. Lett.*, (84), 2000. Available at `http://www.unifr.ch/econophysics/`.
12. D. Challet and Y. C. Zhang. Emergence of cooperation and organization in an evolutionary game. *Physica A*, 246, 1997. Available at `http://www.unifr.ch/econophysics/`.
13. D. Challet and Y. C. Zhang. On the minority game: Analytical and numerical studies. *Physica A*, 256:407, 1998. Available at `http://www.unifr.ch/econophysics/`.
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. J. Ferber. *Multi-Agent Systems: An introduction to distributed artificial intelligence*. Adisson Wesley, 1999.
16. 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.
17. D.E. Goldberg. *Genetic algorithms in Search, optimization and Machine Learning*. Addison Wesley, 1989.
18. 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.
19. L. Miramontes Hercog. Hand-eye co-ordination: An evolutionary approach. Master's thesis, University of Edinburgh, 1998.
20. L. Miramontes Hercog and T.C. Fogarty. XCS-based inductive intelligent multi-agent systems. In *Late Breaking Papers*, pages 125-132. Genetic and Evolutionary Computation Conference, 2000.
21. L. Miramontes Hercog and T.C. Fogarty. *Proceedings of the Eurodays 2000 meeting dedicated to the memory of B. Mantel*, chapter Analysing Inductive Intelligence in a XCS-based Multi-agent System (MAXCS), Wiley, 2001.
22. J.H.Holland. *Hidden Order: How adaption builds complexity.* Perseus Books, 1996
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. In Roy, Chawdhry, and Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, pages 59–68. Springer-Verlag, London, 1997.

25. T. Kovacs. Strength or accuracy? a comparison of two approaches to fitness calculation in learning classifier systems. In *2nd. International Workshop in Classifier Systems*, 1999.
26. J.R. Koza. *Genetic Programming.* MA: The MIT Press/Bradford Books, 1992.
27. P.L. Lanzi. Generalization in wilson's classifier system. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 501–509, 1997.
28. P.L. Lanzi. Adding memory to xcs. Technical report, Politecnico di Milano, 1998.
29. P.L. Lanzi and S.W. Wilson. Toward optimal classifier system performance in non-markov environments. *Evolutionary Computation Journal*, 8(4):393–418, 2000.
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/ECML-00 Joint Workshop on Learning Agents*, pages 33–37, 2000.
31. T. Sandholm and R. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, (37):147–166, 1995. Special Issue on the Prisoner's Dilemma.
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. T.Eymann, B.Padovan, and D.Schoder. Artificial coordination- simulating organizational change with artificial life agents. Technical report, University of Freiburg, 1998.
36. 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.
37. S.W. Wilson. Zcs: A zeroth level classifier system. *Evolutionary Computation*, 2(1):1–18, 1994.
38. S.W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
39. S.W. Wilson and D.E. Goldberg. A critical review of classifier systems. In *Proceedings of the Third International Conference on Genetic Algorithms*, 1989.
40. D.H. Wolpert, K. R. Wheeler, and K. Tumer. Collective intelligence for control of distributed dynamical systems. Technical report, NASA Ames Research Centre, 1999.
41. S. Zhang, S. Franklin, and D. Dasgupta. Metacognition in software agents using classifier systems. Technical report, The University of Memphis, 1997.