Learning Sequences of Rules using Classifier Systems with Tags

A. Sanchis*, J. M. Molina*, P.Isasi* and J. Segovia**

* Departamento de Informática, Universidad Carlos III de Madrid
C/Butarque 15, 28911- Leganés (Madrid)
** Departamento de Lenguajes y Sistemas, Facultad de Informática, UPM
Campus de Montegancedo, Boadilla del Monte (Madrid)

ABSTRACT The objective of this paper was to obtain an encoding structure that would allow the genetic evolution of rules in such a manner that the number of rules and relationship in a Classifier System (CS) would be learnt in the evolution process. For this purpose, an area that allows the definition of rule groups has been entered into the condition and message part of the encoded rules. This area will be named Internal Tags. This term was coined as the system has some similarities with natural processes that take place in certain animal species, where the existence of tags allows them to communicate and recognize each other. Such CS has been named Tags Classifier System, TCS. The TCS has been tested in the game of draughts and compared with the classical CS. The results show an improving of the CS performance.

1. INTRODUCTION

A Classifier System [1, 2, 3, 4, 5, 6, 7, 8] is composed of three main components, which can be considered as activity levels. The first level (Classifier System) is responsible for giving responses (satisfactory or otherwise) to solve the problem proposed. At this level, there are system rules, encoded by means of restricted alphabet character strings. When this level is executed, a response is given to a particular situation. The fitness of the response to the problem that is to be solved is measured by means of the reward received by the above rule from the environment. The second level (Credit Assignment) evaluates the results obtained at the lower level, distributing the rewards received by the rules that provide the output among all those that contributed to activating each of the latter rules. As this is a reinforced learning method, this evaluation can be adjusted by applying a reward or payment by the environment, whose value will be high if the solution is satisfactory and low if it is not. Reassignment can be carried out by means of different algorithms [4, 9], of which the Bucket Brigade [3] is the most commonly used and the one employed in this paper. At this level, it is not possible to modify system behaviour by changing its rules; however, it is possible to adjust their values and establish some sort of hierarchy of good and bad rules. The mission of the third level (Discovery) is to find new means for the system to discover new solutions, for which purpose a Genetic Algorithm (GA) is used.

Rules can be activated in parallel at the CS action level, whereas they are activated in series in traditional production systems. During each recognition cycle, a traditional system activates a single rule. This rule-by-rule procedure is a bottleneck for productivity growth; moreover, many of the differences between production systems architectures are related to the selection of the best strategy for activating the rule in question. CS's overcome this bottleneck by allowing the parallel activation of rules during a particular recognition cycle, or internal cycle. So, different activities can be coordinated in parallel in a CS. When a choice has to be made between mutually exclusive environmental actions or when the size of a rule has to be pruned to adapt its length to that of the listed messages, these decisions are left until the last possible moment when they are selected competitively, for example. So, the sequence of operations of a traditional CS can be outlined as shown in Table 1.

Table 1: Sequence of operations at the action level of a traditional CS.

Step	Operation
1	A *k*-length encoded message from the environment comes in through the input interface.
2	Clear messages list.
3	The environmental message is placed on the messages list.
4	All the classifiers whose condition coincides with the message will be activated. One message can activate several classifiers.
5	The activated classifiers will send their messages to the list.
6	Steps 4 and 5 will be repeated for *n* internal cycles.
7	Finally, a message will be chosen to produce the output through the respective interface.

Therefore, traditional CS operation is based on three fundamental concepts:

1. The solution of the global problem is a set of rules (a subset of rules is a solution to a particular situation, and a single rule may even be a solution for a very specific situation, although this is unusual).
2. Each rule's payment is distributed among the rules that activated it in the internal cycles.
3. The Genetic Algorithm allows rules to be generated from the best rules, which leads, theoretically, to an improvement in overall system operation.

The manner in which Classifier Systems operate has some drawbacks, of which the following deserve a special mention:

1. With regard to the system's ability to learn chains of rules which, moreover, do not break from one learning instant to another; the loss of a rule from the chain can lead to a loss of all the knowledge due to the interrelations between rules.

0-7803-5731-0/99$10.00 ©1999 IEEE

The rules make sense not individually but only as groups which are unknown *a priori*.

2. With regard to the need to apply the discovery algorithm to generate increasingly better classifiers and, finally,
3. With regard to the sequencing of the cases put to the system in order to guide learning towards an improvement in overall system behaviour.

The problem addressed in this paper is in particular how to combat the problem of the loss of rules and the need to "maintain acquired knowledge". Both problems are due to CS discovery level action, which leads the mechanisms of the CS to fail when forming and maintaining associations between rules.

The discovery level acts on the set of classifiers that have just been executed in such a manner that the new rules are generated from the best rules prior to discovery level action. This operation can lead to the loss of rules that are necessary for solving certain points of the problem and which appeared at the start of the learning period but failed to do so later on. This means that rules which were very good at the start of the execution can be considered by the GA as less valuable, because other rules are stronger.

"Internal Tags" (IT), proposed by Holland [6] and others for application to Genetic Algorithms, were introduced for this purpose, giving rise to a new class of CS, Classifier Systems with Tags (TCS). Apart from preventing the loss of rules, different rules must be made to coexist at all times, thus stopping the rules becoming uniform, leading to a loss of variety in the rule population.

2. CLASSIFIER HIERARCHIES

Ad-hoc internal CS hierarchies

The problems of rule loss have been addressed from various viewpoints in the literature with a view, in all cases, to improving CS's. Shu et al. [10] c Consider introducing hierarchies into CS's, that is, groups of rules that have to be maintained throughout the learning process. The rule groups are formed *a priori* and are given by the expert problem-solver. This is an attempt to solve the problem which DeJong [11] solved by means of *crowding* in the field of Genetic Algorithms. So, on the one hand, they establish rule groups (families) and, on the other, they propose genetic operators that act intrafamily and interfamily. The payment system is also modified, and when a rule from one group wins, all the other rules in its group also partake of that prize.

Basically, the problem with discovery level action is that all the rules are considered to be equal. This idea, which is logical in other Evolutionary Computing techniques, where each individual is a solution to the problem, and they, therefore, all have to compete with each other, is not directly extendible to CS. This is because no one rule is capable of solving the problem on its own in many cases, which means that not all the rules are equal. A rule that is fired in a particular situation and whose action solves the problem is not the same as a set of rules that must be fired in order so as to address a different situation. Here, the strength of the first rule is likely to grow much more than the strength of all the rules chained in the second case. In order to solve this problem, Shu proposes dividing the CS rule set into subsets, each of which has rules specialized in a particular point of the problem, in such a manner as to make the members of the same family of rules compete.

Furthermore, the distribution of the payment among members of one family means that the knowledge acquired earlier is not so quickly forgotten, as a rule that attains a given strength value continues to receive strength as a result of the execution of rules belonging to its family. The loss of rules is especially critical when the problem that is to be solved requires complex rule chainings, as the loss of a rule in the chain at the discovery level can mean that all the chaining is overlooked and the chain is entirely forgotten, which will mean that it will have to be learnt again later.

Hierarchically organized independent CS

In 1995. Dorigo [12] presented the results of solutions designed to make Classifier Systems learn faster. The tools he used are: parallelism, a distributed architecture and training. With respect to parallelism and the parallel architecture, he proposes a parallel version of ICS [13], and designed a parallel Classifier System. called *Alecsys*, applied to what is termed the "animal problem" [14]. This problem is addressed from the viewpoint of dividing the problem into smaller parts, based on a hierarchical architecture in which a series of ICS's learn to cooperate in solving the learning problem. The different ICS levels are executed in parallel on different machines, and, moreover, different ICS's, responsible for different tasks, are also executed in parallel. The author [12] takes up Brooks's idea of "reactivity" [15], that is, the existence of a set of behaviours, each of which is implemented by means of an ICS and which are independent of each other and produce an output for each input. The whole system is composed of three systems: an ICS to overcome obstacles, another to attain a goal and, finally, a system that decides which of the two possible outputs is the output of the combined system.

The author proposes that internal conditions be included to achieve rule chaining (which is equivalent to behaviour chaining in this case). This allows messages from the environment to be distinguished from messages from earlier cycles. Dorigo's study centres on the usefulness of the internal conditions without clearly explaining how they are used internally by the CS. The results of this part of the paper show that the size of these internal conditions, as applied in this case, is not very relevant for learning.

In short. Dorigo's paper [12] proposes a sort of hierarchy, since the final CS is composed of three CS's: two basic CS's and another that decides which CS is appropriate for each situation. In each case, rules are evolved independently, in such a manner that each behaviour evolves separately. The problem with this hierarchical approach as compared with Shu's proposal is that it is impossible to perform genetic operations that allow holistic evolution. as each Classifier System is evolved independently and is unrelated to the others. That is, no relationships can evolve between each behaviour such that a rule from one classifier can activate a rule from another. The question is whether the separation of the Classifier System into several Classifier Systems raises system effectiveness in particular situations. In any case, it prevents the generalization of learning.

Automatic category generation within a CS has not been addressed in any paper to date. The idea can perhaps be borrowed from nature: some species use "tags" to limit a "call or warning" to a set of individuals, discriminating a subset among the total set. In the same manner, parts can be included in rules that allow some to be discriminated from others. What we will call *Internal Tags* (IT) can be defined in an *ad hoc* manner by creating a given string of calls [10] or can be defined in such a

manner that the ITs themselves evolve, determining what groups are necessary. In short, each rule can be provided with a field which will evolve genetically and which identifies that the rule in question is a member of a group, similarly to the tags proposed by Holland [6].

3. EVOLUTION OF TAGS IN A CS: THE TCS

As discussed in the preceding section, any solution that seeks to prevent the loss of rules necessarily involves creating subsets within the set of classifiers of which the CS is composed. The two solutions studied are: creation of a hierarchy in the *a priori* population and composition of several independent Classifier Systems. These solutions limit the generality of the CS learning process, as the hierarchies are considered to have been learnt in one case or because of the decomposition into several CS in the other, in such a manner that the learning context is subject to these constraints.

The proposed solution must, therefore, combine the ability to learn without *a priori* knowledge and the capability of generating some kind of internal subdivision within the CS to allow categories of rules to exist. A CS, called TCS, has been designed that allows groups to evolve automatically. For this solution to be implemented, the encoding of the classifiers will have to be modified to include a field that represents the type or group to which each classifier belongs. So, for example, given a CS, such as:

Classifier	Rule	Message
1	0110	0000
2	0001	1110
3	1101	1000
4	0000	0001
5	1110	1101
6	0001	0110

A 1-bit field can be reserved to establish the classes making up the CS, and the resulting CS would be as follows:

Classifier	Rule	Message
1	0110X	0000X
2	0001X	1110X
3	1101X	1000X
4	0000X	0001X
5	1110X	1101X
6	0001X	0110X

This field can be used to subdivide the CS into several groups of classifiers, each of which contains the classifiers that have the same value in the new field. This field can be said to establish the classifier type or group. So, for example, the specimen CS could contain the following classifiers:

Classifier	Rule	Message
1	01101	0000X
2	00011	1110X
3	11011	1000X
4	00000	0001X
5	11100	1101X
6	00010	0110X

According to the definition of the value of the field that establishes the classes, there are 2 classes: one defined by classifiers whose value is 1 (classifiers 1, 2 and 3) and the other by those whose value is 0 (classifiers, 4, 5 and 6). Note that the definition of a class is determined by the value of the above field in the condition part of the rule, that is, rules that must have the same value in the field for activation are members of the same group.

This field, which appears in the encoding, evolves in the same manner as the other fields, which means that the number and size of each class in the CS hierarchy is variable and must be learnt. Wide ranging groups can be established, and all the classifiers could actually have the same value, in which case the system would operate like a classical CS.

Apart from establishing the classifier type according to the value of the condition part, as it is included in the message part which evolves similarly, not only are the rule groups evolving, so is the form of intergroup activation. So, in the preceding example, a set of classifiers could evolve as follows:

Classifier	Rule	Message
1	01101	00000
2	00011	11100
3	11011	10000
4	00000	00011
5	11100	11011
6	00010	01101

In this case, the group 1 classifiers activate group 0 classifiers and vice versa. Obviously, this type of activation must be learnt by the CS and there are range of possible configurations.

Finally, it is important to take into account that the inclusion of a field in the classifiers means that a value must also be entered in the input message in the above position. This value is not determined by the environment; it is defined *a priori* by means of a value encoding the fact that the message in question is the environmental message. In this manner, the CS will have to learn which rule group having the same group definition field value is to be activated in response to the environmental message.

The appearance of hierarchies in the CS is subject to the information about the category to which the rule belongs being maintained in each rule. This information must evolve genetically; obviously, if the information about the category in each rule is capable of representing "n" different categories, the solution to the problem could be composed of m ($m<n$) categories and the remaining categories would be irrelevant. If this information is represented in each rule and it is allowed to evolve, the number of rules associated with a particular category is also variable; in this respect, the genetic evolution of the categories will not only allow the categories required to evolve but also for each one to have the size required to solve the problem.

TCS Operational Schema

The sequence of operations at the action level is the same as for traditional CS (Table 1). However, there are fields in the conditions and messages whose sole mission is to define the category to which the rules which sent their messages to the message list in the preceding cycle belong. The first activation takes place using the environmental message, as shown in Table 1. The environmental message should contain default values that

indicate that it is an environmental message. As of this first message, the activated classifiers will send their messages to the message list, and we will then have a generation of messages of different categories which will have specialized in responding to the environmental message. This information in the messages will have an impact on the next classifiers activated, that is, it will be involved in chaining the rules that are fired. In this manner, rules of the following type will evolve:

Starting instant:

IF The external_signal is <type 1>
THEN message <001...> and group <X>

Subsequent instants:

IF the message is <001...> and the group is <X>
THEN message <101...> and group <Y>

In short, the mechanism of including Internal Tags (IT) in rules is beneficial for evolving complex solutions within a CS. As the TCS is executed in parallel and all the rules are activated at the same time, a range of complex strategies are generated in the messages list by chaining rules from different groups. These strategies are maintained during the internal CS execution cycles and the best are learnt by means of credit reassignment and discovery processes.

Apart from having to differentiate the encoding for different groups, another two levels of the CS will have to be adjusted: the credit reassignment algorithm (BBA) and the discovery algorithm (GA). This is due to the need for each rule group or hierarchy to gradually evolve in parallel. On the one hand, the credit earned by one rule needs to be distributed among all the rules of its group in order for these rules to beat other groups, in such a manner that the strength of each group can be considered as a factor to be taken into account when performing intergroup genetic operations. In this case, it is not only an individual that evolves; evolution is focused on the generation of compact groups, which are widely used and should, therefore, have a better rule set, without overlooking the need for groups whose elements, though perhaps fewer, are essential for developing the final strategy. Note that if the strength of all the rules of a group increases when one of the rules of the group is assessed as positive by the BBA, the strength of those groups of rules that are chained with this group will also be increased, as the percentage strength awarded for activation will be calculated on larger sums.

4. TSC EVALUATION IN THE GAME OF DRAUGHTS

In this paper, we seek to get a measure of the contribution of Internal Tags (IT) to the learning process in a Classifier System. A clear evaluation of the contribution of ITs in the encoding calls for a problem that is solved in a perfectly defined environment. The environment chosen in this case was the learning of draughts end games, that is, draughts matches where only a few pieces remain on the board at an advanced stage of the game.

The objective of applying the TCS to learning the game of draughts is not to obtain a CS that plays draughts; it is to apply Classifier Systems in a clear and defined environment that allows traditional Classifier Systems to be compared with the modification proposed in this paper, including IT. Obviously, there are a lot of systems that play draughts, some very successfully [16]. However, for the purposes of this study and comparison, a player following a random strategy will be used, and measurements will be taken of the games each type of CS (classical/with IT) wins against the random player using different configurations.

Game Rules

There are a lot of variations on the game of draughts. In this paper, a 64-square board is used with black and white squares. The game is played by two players one with white pieces and the other with black pieces, which are either pieces or kings. Initially, the white pieces are placed at the bottom of the board and the black pieces at the top, and there no kings. In this paper, the opening boards are not used, as we work only with end games, where the maximum number of pieces is 5. These can be pieces of any kind and be situated in any valid position on the board. The kings are crowned when a piece reaches opposite end of the board. The edges of the board are the limits of the moves. The edges of the board are not continuous. In this paper, the directions of the moves are considered as absolute. When an opponent's piece is positioned in any of the directions in which a player's piece can be moved, the latter will take the piece that is in its path, by jumping over it onto the next vacant square in that direction. The piece captured will be removed from the board. This process will be repeated as many times as possible before the opponent player can take its turn. When either player has made a move or taken a piece (and cannot capture another piece), it will be the opponent's turn. The game will end when only one player's pieces remain on the board or there is a draw. There is a draw when the player whose turn it is cannot make any move.

Information encoding

This involves analysing how and what information about the board, the pieces, players, turns, moves, etc., can be supplied to the CS as an input message. The encoding chosen for the game of draughts is such that an output from the CS is always interpreted as a move. This means that the CS decisions are interpreted depending on the system status. Obviously, the system must be able to play with both black and white pieces, so an encoding was chosen that does not take into account "the colour" of the piece. Additionally, the directions of the moves have been taken to be absolute as explained above.

Input Message: The information available on the board and that can be entered into the system is: the number pieces on the board, the colour of each piece, coordinates (x,y) of each piece, piece type (piece or king), directions in which it can move or take and how far it can move or take in each direction. The input messages include the status of the board at any one time: total number of pieces, number of pieces belonging to the CS player, colour, who's turn it is, how many kings there are, etc. This information will be encoded in a 57-bit length input message for the traditional CS. The number of bits will be 61 for a classifier with IT, as 4 bits are entered to represent the ITs. The first position of the input message encodes the information about the possibility of taking (with a 1) or only moving (with a 0). The next 4 positions contain information about the total number of pieces there are on the board, and the next 12 on the pieces that belong to the player whose turn it is, how many of these are kings and the number of the opponent's kings, all encoded using 4 bits in each case, considering the percentage represented. Then, the information regarding the position of these pieces is recorded, by transforming this decimal number into a binary number of up to 8 bits. Finally, if the total number of pieces is under 5, the remainder of the message is filled in with "#"

symbols.

Output message : The output message has the same length as the input message, 61 or 57 bits, depending on whether or not the ITs are taken into account. Only the last 16 bits of the entire message sent through the output interface of the CS after having performed the chaining process for several internal cycles are used as an output. With regard to the output messages, the respective positions will be taken and the decoding process will be performed.

5. COMPARISON BETWEEN CS AND TCS

The objective of this section is to compare the traditional CS with the TSC. For this purpose, the above systems will be played against a player who makes random moves, having a variable degree of randomness and starting from different situations. The two systems commence without any previous knowledge, that is, their entire population is randomly generated, which means that their rules and messages are not adapted to any particular case and their moves will, in principle, also be random. The three types of experiments conducted under this point were performed by gradually increasing their difficulty level in order to examine the behaviour of the two systems in face of the above changes. In the first type of experiments, the randomness of the random player is gradually raised. This means that there are different levels of randomness within a random player. This level of randomness is entered in the output message produced by the random player. The output message of the random player has the same make-up as that of the CS; however, it possesses only the sixteen characters required by the decoding process for transformation into a particular move. The output message of the random player for all the games that have been executed in this section is based on a fixed message. Randomness is entered in the output message of the random player depending on the number of characters in the above message that are generated randomly. This generation is regulated proportionally, that is, there are random players whose output message is composed, for example, of 40% random characters. This percentage of randomness in the output message is applied to each move to be made by the random player in each game, which means that a different output message from that created in the previous move is generated in each move.

Three groups of experiments with a different starting situation were performed for the comparison. The experiments were defined in increasing order of complexity, depending on the opening board with which each game that was to be played commenced: first, the opening board will be fixed for all the games, then the positions of the pieces that appear on the board in each game will be altered and, finally, the opening board will be generated at random for each game. In the first experiment, differing degrees of randomness will be applied to the opponent player, starting with 0% randomness and increasing this percentage up to 100% randomness. In the last two experiments, the opponent will 100% random throughout, and the opening boards will be modified incrementally, either by changing the position of the pieces or by generating a new board.The result will show the evolution of the games won and lost by the two types of Classifier Systems. These results correspond to the average of five groups of games. In order to analyse the results obtained in more detail, the percentages of games won at the end of learning for each CS and for each experiment, and the percentage improvement of the TCS as compared with the CS are set out in Figures 1, 2 and 3. Analysing the results, we find that the contribution of ITs to the CS is not relevant in all situations. In problems where the CS has to learn a very simple sequence of operations, because the problem to be solved is less complex, the ITs can turn out to be more of a handicap, as their inclusion means that the system is forced to "learn" how to chain rules, when such chaining may be unnecessary. As the problem becomes more complex, the need for rule chaining increases, and the contribution of the ITs becomes evident, since their existence encourages rule chaining. So, we find that the results of the TCS in the first experiments (Figure 1) only improve on the CS in the last case. On the other hand, an improvement is seen in the results obtained with the TSC as compared with the CS in the subsequent experiments performed (Figure 2 and Figure 3).

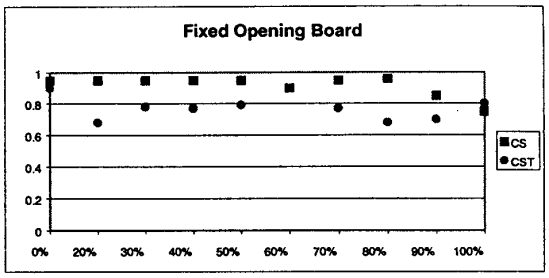

Figure 1: Percentage of games won by the TCS and the classical CS, averaged out over 5 different situations on the same opening board, against an opponent player whose randomness increases from 0% to 100%.

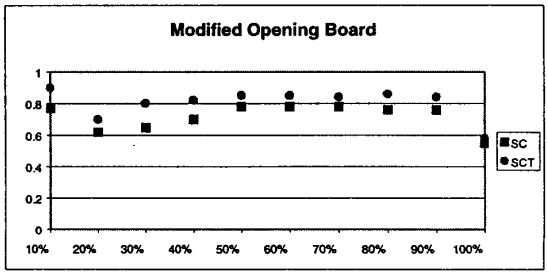

Figure 2: Percentage of games won by the TCS and the classical CS, averaged out over 5 different situations on the an opening board modified by between 10% and 100%, against a 100% random opponent player.

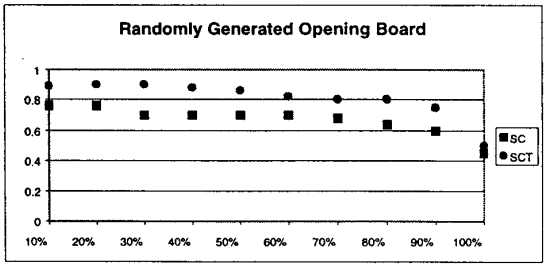

Figure 3: Percentage of games won by the TCS and the classical CS, averaged out over 5 different situations on an opening board of which between 10% and 100% was randomly generated, against a 100% random opponent player.

Figure 1 shows the results of the experiments in which the opening board was unchanged. In this case, the problem appears not to require rule chaining to develop strategies that can be used in unexpected situations, since the opening board is fixed and there are, therefore, only limited possibilities of different moves. So, the CS is faced with a player who, for all intents and purposes, makes a well-defined series of moves whose variability is very restricted. This is why the TCS results are 14% worse on average than those obtained by the CS. Considering that this is the simplest possible case, it appears that is counterproductive to force the CS to employ ITs, as it makes the TCS play worse than the CS. In the last case, where the systems face maximum variability, the results are very similar, and those obtained by the TCS are slightly better, mainly because the need for chained strategies starts to become evident. Figure 2 shows the results obtained when the opening board is modified using an incremental degree of randomness. In this case, the TCS performs 10% better on average than the CS; this is because the system has to start to generate more complex actions to be able to respond to more diverse situations. It is noteworthy in this case that the two systems obtain poor results at the maximum level of randomness, compared to the results that they obtained at lower levels of variability. This is perhaps due to the fact that these are very indeterminate situations where it is difficult for the system to be able to extract knowledge. In Figure 3, the results obtained show that as the degree of uncertainty in opponent player performance is increased, a higher percentage of the results of the TCS are better than those of the CS, in this case 15% on average. Again neither of the two CS are able to obtain results of over 60% of games won with the effect of maximum randomness.

6. CONCLUSIONS

One of the major problems related to Classifier Systems is the loss of rules, when the learning process presents individual cases and allows the system to learn gradually from these cases. Each learning interval with a set of individual cases can lead the strength to be distributed in favour of a given type of rules that would in turn be favoured by the Genetic Algorithm. If this reasoning is extended to the entire learning process, genetic diversity, which is so necessary for learning, can disappear due to the growth of a given type of rules in the population. Furthermore, when different rule sets are needed to solve part of the problem, these may disappear if part of the problem (corresponding to the rules that can be lost) is not presented in the examples found up to a certain point. However, the above rules can be very necessary.

The objective of this paper was to obtain an encoding structure that would allow the genetic evolution of these groups in such a manner that their number and relationship would also be learnt in the evolution process. For this purpose, an area that allows the definition of rule groups has been entered into the condition and message part of the encoded rules. This area will be named Internal Tags. This term was coined as the system has some similarities with natural processes that take place in certain animal species, where the existence of tags allows them to communicate and recognize each other.

The results presented show how the proposed Classifier Systems are capable of improving on the classical approach of Classifier Systems in cases in which rule chaining is relevant. The importance of this contribution is the discovery of a learning method that allows similar or related knowledge to be grouped. This property of ITs, the automatic grouping of rules that share the same objective, is of special interest, and a study has, therefore, been conducted to analyse what effect they have and what results are obtained in each of the proposed Classifier Systems. In short, we can infer from the results obtained that Classifier Systems are able to learn in games environments and that when the game is complicated, it requires a complex solution which is not satisfactorily provided by classical CS's and thus requires the inclusion of tags.

7. REFERENCES

[1] J. Holland, *"Adaptation in Natural and Artificial Systems"*. University of Michigan Press, Ann Arbor, (1975).

[2] J. Holland, *"Adaptive Algorithms for Discovering and Using General Patterns in Growing Knowledge Bases"*, International Journal of Policy Analysis and Information Systems, vol. 4, 245-268, (1980).

[3] J. Holland, *"Properties of the Bucket Brigade"*. In Proc. of International Conference on Genetic Algorithms and their Applications, vol. 1, 1-7, (1985).

[4] J. Holland, *"A Mathematical Framework for Studying Learning in Classifier Systems"*, Physica D, 22, 307-317, (1986).

[5] J.H. Holland, *"Escaping Brittleness, The Possibilities of General Purpose Learning Algorithms Applied to Rule-Based Systems"* in [MCM86], 593-623, (1986).

[6] J.H. Holland, *"Hidden order: how adaptation builds complexity"*. Reading Massachusetts, Addison-Wesley, (1995)

[7] D.E. Goldberg, *"Genetic Algorithms in Search, Optimization, and Machine Learning"*. Addison Wesley, Reading Massachusetts, (1989).

[8] M. Mitchell, *"An Introduction to Genetic Algorithms"*, MIT Press, Massachusetts, (1996).

[9] G.E. Liepins, M.R. Hilliard, M. Palmer and G. Ranjaran, *"Credit Assignment and Discovery in Classifier Systems"*, International Journal of Intelligent Systems, vol. 6, 55-69, (1991).

[10] L. Shu and J.Schaeffer, *"HCS: Adding Hierarchies to Classifier Systems"*, Proceedings of the 4th International Conference on Genetic Algorithms, 339-345, (1991).

[11] L. Booker, D.E. Goldberg and J.H. Holland, *"Classifier Systems and Genetic Algorithms"*, Artificial Intelligence, 235-282, (1989).

[12] M. Dorigo, *"ALECSYS and the AutonoMouse: Learning to Control a Real Robot by Distributed Classifier Systems"*, Machine Learning, 19, 209-240, (1995).

[13] M. Dorigo and U. Schnepf, *"Genetics-Based Machine Learning and Behavior Based Robotics: A New Synthesis"*, IEEE Trans. on Systems, Man and Cybernetics, 23:1, 141-154, (1993).

[14] S. Wilson, *"Knowledge Growth in an Artificial Animal"*, Proc. of the First International Conference on Genetic Algorithms and their Applications, 16-23, (1985).

[15] R.A. Brooks, *"Intelligence Without Representation"*, Artificial Intelligence, 47, 139-159, (1991).

[16] J. Schaeffer, *"One Jump Ahead"*, Springer-Verlag, (1997).