An Architecture Based on M5P Algorithm for Multiagent Systems

Alex Machado[1], David Carvalho[1], Esteban Clua[1], Cristiano G. Duarte[2], Marcos V. Montanari[2], Willian M. P. Reis[2]

[1]Instituto de Computação - Universidade Federal Fluminense - Niterói, RJ - Brazil

[2]Departamento de Computação - Instituto Federal do Sudeste de Minas Geral - Rio Pomba, MG - Brazil

alexcataguases@hotmail.com, dbcdavid.clk@gmail.com, esteban@ic.uff.br, crgdcomp@gmail.com, marcosviniciusmontanari@gmail.com, willianmpreis@yahoo.com.br

Abstract—Character-based interactive storytelling, life simulation and game difficulty dynamic balancing are examples of topics that need to deal with autonomous agent evolution. Although the commercial appeal of such kind of feature, the research of new behaviors emergence in virtual societies is restricted to intelligent agents that do not learn. This work proposes a novel architecture of a multiagent system based on the M5P algorithm for generation of emergent behaviors in complex virtual worlds. Based on our obtained results, we describe the advantages of the reinforced learning based on numerical classifiers when compared with traditional interaction adaptations.

1. Introduction

In a previous work [Machado et al 2010], we defined an architecture for implementing strategy games with the possibility of occurrence of emergent behaviors. Through experiments, we demonstrated the advantages of using the M5P algorithm over others qualitative attributes classifiers. In this paper we extend the experiments for a multiagent based environment.

In a similar way, [Bakkes et al 2009] define a game strategy as a configuration of parameters that determine strategic behavior (such as how many aircraft units should be constructed, maximum air group size or maximum number of storage buildings). In this paper, domain knowledge is required in order to adapt the characters to game circumstances automatically by the game AI, and is exploited immediately to evoke effective behavior in a controlled manner in online matches. Experiments in a strategy game shown that proposed case-based adaptive game AI provides a strong basis for effectively adapting game AI. Our approach differs from this, since defines a simplest architecture in order to achieve the same goals and makes an evaluation about the interaction between the multiagents systems characters. Our method also maintains the rapid and reliable adaptation of game AI characteristics.

We maintain the emergent property achieved in [Machado et al 2010], enhancing knowledge properties presented by Crocomo et al. [Crocomo & Simões 2008] and [Hong & Cho 2005], where it is shown that machine learning processes in games that rules NPCs (non player character) behaviors (in this case, evolutionary algorithms) are able to emerge more complex strategies from simple behaviors.

2. Related Work

According to [Manslow 2002], the usage of adaptive learning in games is a trend and must be one of the most important improvements to the field. This topic is important not only for the improvement of AI behaviors, but specially for generating more robust and smart gameplay, according to the player characteristics. Within this field, our work is more focused on reinforcement and supervised learning approaches.

Our approach uses the supervised learning related to the examples input requirements and induction rules. We also use reinforcement learning, since the presence of rewards is a key factor for adapting the system for new cases.

In [Andrade et al 2005] and [Spronck et al 2004] it is shown efficient implementation of reinforcement learning as a mechanism of player adaption. The first work introduces the concept of a challenge function, which estimates the current player´s level and changes on the action selection mechanism for fighting games. The second work, presents rules that can be applied for each opponent type of the game.

This paper proposes the creation of a novel architecture using adaptive intelligent agents in a game, with dynamic parameters configuration based on the M5P algorithm.

AI-based storytelling is the mechanisms for automatic story generation, which can be based on autonomous artificial actors' or on explicit plot representations [Cavazza et al 2002]. In the first the story diversity emerges from dynamic interaction between characters.

Although we present in this paper a novel adaptive game balancing, we also discuss the possibility of the usage of the technique for a narrative mediation approach in order to achieve more freedom and coherence for the plot generation guidelines, as presented in [Riedl 2003].

Social AI in games study is important because in a reasonably deep virtual world, there are always "boring" roles, that typical players are not going to feel pleasure to control, such as quest-givers, merchants, henchmen and thugs [Witze el al 2002]. A detailed study about life simulation and interaction can be found in [Pallay 2009]. The virtual world chosen for this study was Second Life [L.Lab 2011].

In [Goertzel et al 2008] the authors proposed a general approach for teaching behaviors to AI-controlled artificial animals embodied in Second Life [L.Lab 2011] using imitative-reinforcement-corrective learning. It is shown the possibility of teaching an agent (animal) to generate new behaviors with gestures.

We proposed use machine learn techniques for a social environment instead for just one autonomous agent.

3. Advantages of M5P Learning

M5P [Quinlan 1992] is a reconstruction of Quinlan's M5 algorithm for inducing trees of regression models. M5P combines a conventional decision tree with the possibility of linear regression functions at the nodes.

In previous work [Machado et al 2010], we experimentally demonstrate the advantages of using that algorithm (M5P) over decision tree learner (C4.5) and probabilistic learn method (Nayve Bayes). This behavior is due to its numerical classification, which guarantees the least loss of information (so accurately) for the next generations (stages). While its competitors just map the set of attribute values to a categorical target value.

When we build an architecture so that the M5P algorithm stays responsible to on-line generate the evaluation function, we create an agent capable of adapting itself to dynamic and inaccessible ambient. In this case, this possible architecture takes advantage in comparison with traditional methods like the genetic algorithm, considering that although the latter does not require knowledge derived from the problem, it requires a pre-defined evaluation method of the result [Goldberg 1989].

According to [Etemad-Shahidi & Mahjoobi 2009] and [Bhattacharya & Solomatine 2005], neural networks (NNs) are not as transparent as semi-empirical regression-based models. For this reason, the main advantage of model trees is that, compared to NNs, they represent understandable rules. In addition, NNs approach needs to find network parameters such as number of hidden layers and neurons by trial and error, which is time consuming.

In [Bhattacharya & Solomatine 2005] the predictive accuracy of the M5P model was observed to be very high and with similar results than the NN model built with the same data.

4. Development Framework

4.1. Approach

Since the environment where the character is inserted is complex (inaccessible, stochastic, sequential and dynamic), this work discusses the intelligent agent architecture with individual learning. This not only happens in situations where each agent pursues its own learning objectives, but also when it's learning can be affected by other agents.

Altogether these agents form a coordinated multi-agent system [Malone & Crowston 2004], where the coordination between the agents are defined by the dependency relations of shared resources. A multi-agent system is composed by two or more agents, in society form, with potential for self-sufficiency and emergency behavior.

4.2. System Architecture Abstraction

M5P-Reinforcement Learn Architecture (Figure 1) uses the M5P algorithm, which is capable of generating rules dynamically, with reinforcement learning, to update the database instances. Therefore this architecture has the characteristics of evolutionary classifier systems [Booker, Goldberg and Holland 1989] that are designed to absorb new information and devise continuously competing sets of hypotheses (expressed as rules) without disturbing already significantly capabilities acquired. Its steps are:

1. Test of random cases at the environment: as happens in an on-line learning system, the agent starts with no previous knowledge, so at the risk of making mistakes. It explores the environment several times, with its attributes and/or actions

defined in a varied form. The interaction mechanisms with the environment and the other agents are sensors and actuators.

2. Incorporation to the instances base: in this step each case previously lived is added to the instances base, which functions as a "memory" of the agent that stores its experiences with their results. These results serve as reward of the architecture, characteristic of the reinforcement learning.

3. Induction of learning rule: using the M5P algorithm, the trees and rules are induced from the base of instances.

4. Generation of candidate instances, classification and selection of the best: this is a module diversifier so that the generated solution does not get stuck in local optima. Uses the elitist criterion to define the best solution of a new set randomly generated and ranked (classified) by the learning rule.

5. Test of this case in the environment: this new generated instance is released in the environment for testing and evaluation. At this moment, the agent, with the same interaction mechanisms, explores the environment creating a new set of experiences for a later restart of the evolution system process.

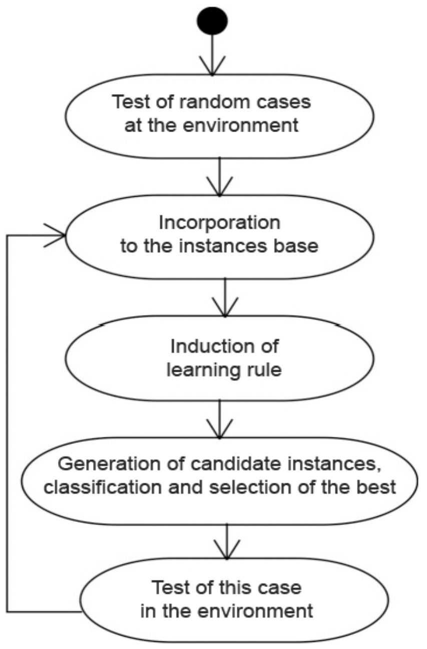
Test of random cases at the environment
Incorporation to the instances base
Induction of learning rule
Generation of candidate instances, classification and selection of the best
Test of this case in the environment

Figure 1 - M5P-Reinforcement Learn Architecture

Although this work evaluates a multi-agent system, this architecture specifies the learning process of only one agent. In the next session, this architecture is redesigned, applied and evaluated in a multi-agent system.

5. Case Study

5.1. Virtual environment plot

The developed application, called “New World” (Figure 2), created for testing purposes of M5P-Reinforcement Architecture, fits in the context of life simulation. Its 3D environment reconstructs aspects of a real city, with a resting place and options of work and leisure. Each NPC is considered foreigner, initially. As such, it must experience/explore the city to be able to define the routine that best suits it later.

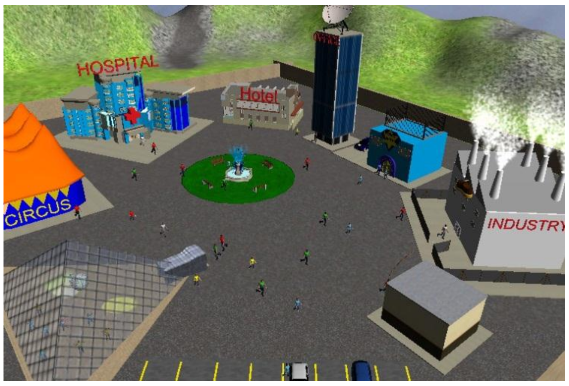

Figure 2 – "New World", the life simulation experiment.

The agents have two types of intelligent behavior: simple reactive and with learning aspects. The first refers to its instincts, such as moving, mapping a route between origin and destination, avoiding obstacles, stopping to chat when it meets with another agent (with no exchange of information). The second relates to decide, based on experience, which is the best routine for its day.

The options the city offers to them, here called "waypoints", may be one of the types: work (with payment), leisure (that, although has financial costs, gives satisfaction) or inn/hotel (to spend the night with no expenses and no satisfaction). A routine consists of the waypoints that the agent decided to visit during the working time of the day.

The day has 24 hours. Around 23:00 the agents must go rest until 7:00. So they have an average of 16 working hours to assemble an agenda. Thus, given the need for money (which can be obtained at the waypoints that offer work) to achieve satisfaction (at the waypoints that provide entertainment), the NPC needs to explore the environment to learn how to optimize its routine during the day.

Among the complications of this virtual environment, in which the programmer or a player's character can influence, we can cite:

- Assuming that the agent is employed, if another company discloses the increased value of the base salary, the agent can end by pleading for this job.
- Assuming that a new leisure option appears in the city, the agent may want to experience.
- In a particular route can be placed a "thief", in this case its diary routine may be compromised, so the agent will need to define a new one.

5.2. Developed application

The application New World was developed on the M5P-Reinforcement Architecture, the suggested technology in [Machado et al 2010] and the exchange of information made through threads for the multi-agent system. The experiments in this environment were realized through alterations in the waypoints parameters.

As [Machado et al 2010], to develop an intelligent agent architecture with learning applied to the generation of new strategies for a game, it was used a framework integrating Unity 3D (game engine), Microsoft Visual Studio (server development IDE), C#, Weka and IKVM.

Despite all the advantages of using Mono, the C# compiler version included in the Unity3D package, it had incompatibilities with the Weka DLL interpreted by IKVM in the experiments we ran. More specifically, the problem was related to the garbage collector. To circumvent this problem, we opted for integrating Unity3D with a server module developed using Visual Studio and making the connections through sockets.

This client-server system can be seen as implementing a kind of reinforcement learning strategy because it generates an output that has never been presented to it, which is rewarded with a score (which represents the knowledge generated by the classifiers).

A multithreading system was developed with a communication protocol between the client and the server to guarantee the generation of multiple agents in the environment.

5.3. Definition of the Instances Base

For the proposed application, the instance base consists of the waypoints that the agent can visit. It is defined dynamically and its attributes are numerical as specified in the M5P algorithm. They are named as follows:

```
@relation RotinaAgente        (1)
@attribute wp1 numeric
@attribute wp2 numeric
        …
@attribute wpN numeric
@attribute satisfaction numeric
```

In this first version, the implemented waypoints are (Table 1):

Table 1. Waypoints and its characteristics

Attr.	Waypoint	Salary	Cost	Satisfaction	Time
wp1	Office	8	0	0	5
wp2	Police station	5	0	0	4
wp3	Hospital	3	0	0	3
wp4	Industry	2	0	0	2
wp5	Pizza Store	0	4	5	4
wp6	Boite	0	3	3	3
wp7	Circus	0	2	2	2
wp8	Square	0	0	1	3

The value of the salary represents the remuneration per day that the waypoint provides. In this first version, the money does not accumulate from one day to another. The cost is the value that the waypoint requires for the agent to use it. The satisfaction is the classifier attribute of the system. It is the sum of the accumulated values of satisfaction on a day that defines whether a routine is good or not. The time represents the hours that each waypoint takes to be used by the agent. Like said before, the sum of the times cannot be higher than16. Analyzing two simple instances we have [Table 2]:

Table 2. Examples of instances

wp1	wp2	wp3	wp4	wp5	wp6	wp7	wp8	satisfaction
2	0	0	0	0	1	1	0	5
0	1	0	0	1	0	0	2	7

Where in the first case the agent worked two shifts at the Office, then went to the boite, and finally to the circus. He raised a total of 16 and spent 5. In the second case it worked a shift at the Police station, went to the pizzeria and spent the rest of the time in the square, having a total compensation of 5 and a total expense of 4. The better of these two instances was the second one, due to its satisfaction value.

If the agent does not have enough money to pay for the leisure waypoint that is in its instance, it will be standing in the front of the waypoint until its end time. The agent will not receive satisfaction.

If the agent decides to go to a waypoint that gives remuneration and the same is closed, destroyed or out of work, the agent will be standing in front of the waypoint until its end of time. The agent will not receive remuneration.

5.4. System Architecture

To avoid the elaboration of more complex schemes, we decided to join, in an activity diagram, the server and the client modules to illustrate the applied architecture (as in Figure 3).

The client initializes after the server, instantiates a predetermined number of agents and warns the latter. The server, in turn, creates a thread for each agent. Each process created uses the learning module with the M5P algorithm. As described in [Machado et al 2010], in this moment the Weka's DLL starts to be interpreted by IKVM.

The client, in turn, starts the simulation. Each agent thread has the client-server communication protocol. As can be seen in the Agent Thread Module (Figure 3), at this moment the M5P-Reinforcement Learn is implemented.

Considering the agent in an individual manner we have it beginning by generating an instance (set of waypoints that forms a routine) randomly. After, it leaves the hostel and visits the waypoints until runs out of valid period of the day. We consider as trial period, this 10 initial days constant that the agents will use to collect information from the environment. Once the visits are over, the routine information and the satisfaction gained each day are sent for the server that induces the tree and linear equations. Then generates a set of 40 valid instances (which sum of hours of their times are less or equal to 16), rank them through the learning rule, elects the better and sends to the client.

This ranking is done by the comparison of the attributes classified by the satisfaction field of the instances base. As can be seen, best instance definition is not based on simple summation of the satisfaction of each waypoint, at the moment. This fact is justified since not necessarily an instance with only entertainment is good, since the agent must work to have money to spend on them. Hence the necessity of learning rule.

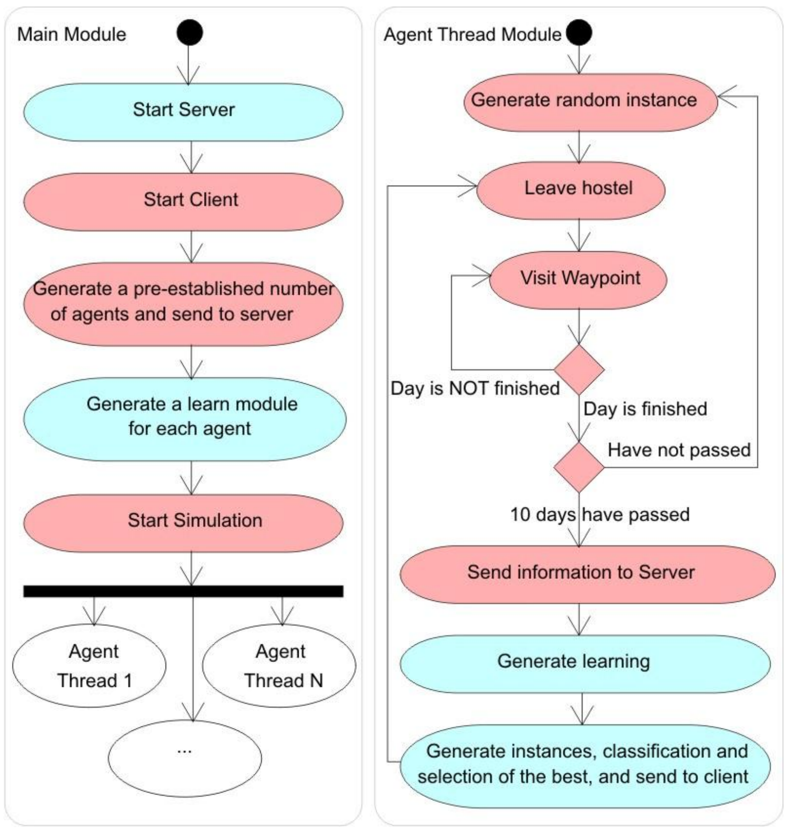

Figure 3 - Example of an application of the M5P-Reinforcement Learn Architecture to a simulation of artificial life. The states represented by blue are executed by the Server and the states in pink are executed by the client (virtual environment).

The simulation, on the client side, restarts from there. If the number of agents and the characteristics of the waypoints do not change, every new day the learned instance tends to be similar to the previous. Otherwise changes may occur. The random component of the stage of diversification of the algorithm ensures that, even if the agent has a belief that a routine is good, perchance it might know another.

Another important feature of the system is the dynamic definition of the attributes of the instances base, which ensures that the agent can receive new information about waypoints that were not programmed by the developer.

5.5. Experiments and preliminary results

We realized two batteries of experiments, one to evaluate the evolution of the characters and another to verify their adaptation to the environment.

The first set of experiments was realized with 4 individuals during 20 days. In the first 9, they gathered data and in the following they defined their routine based on experience to optimize their satisfaction (the higher the better satisfaction). This experiment was repeated 10 times and the average of satisfaction per individual can be seen in the Figure 4.

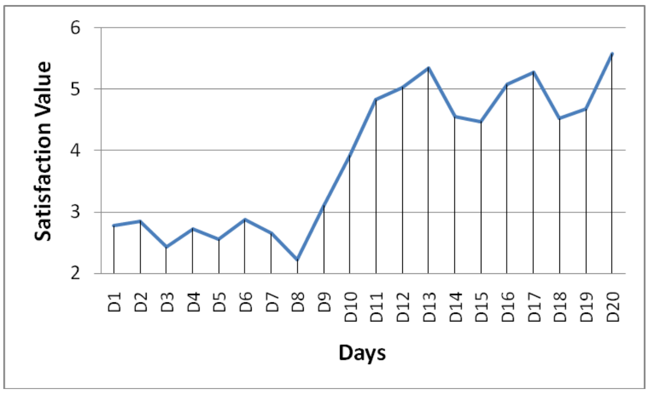

Figure 4: Results of the satisfaction increase based on the learning period

As from the 10th day, satisfaction grew and stabilized close to the value 5, it was shown that the agents are capable to perceive the environment and optimize their results.

To test the adaptation, we compromised the environment right after the agent's learning. Since the 10th day, each agent started to use its learning and since the 15th day we raised the salary of the office, reduced the salary of the industry and took the entire satisfaction of the boite. In this experiment, the instances base for learning was tested in two different ways: with no limit of records and with limit (containing only the 10 and 5 last days). It is observed that 10 tests were performed for each of the three forms described. Results can be seen in the Figure 5:

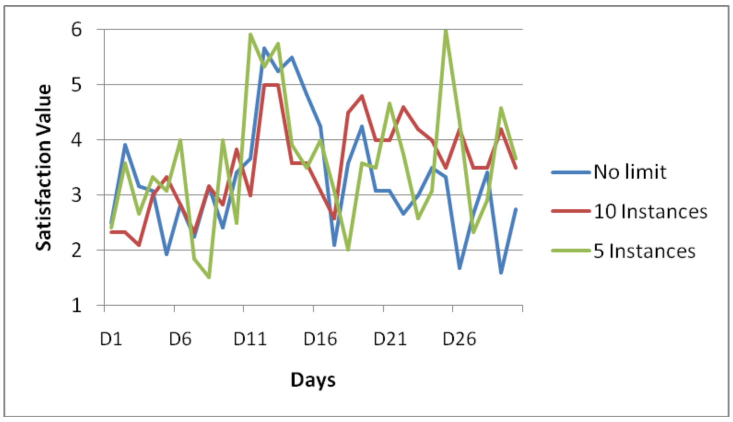

Figure 5: Agent adaption with three experimental bases and different information values.

In this experiment, although since day 15 the three configurations had a drastic and expected income fall, what really matters is the evolution from that day on. When we store the good and bad cases and then completely change the values of the classifiers attributes, our instances base passes through a latency of learning that can compromise next results. This is what happened in the configuration without limit, which until the 30th day failed to achieve the performance demonstrated at its peak period (days 10 to 14). However, if we have a small instance base we will not have a rule ready to handle all cases, and although a drastic change can be quickly bypassed, its result does not have constancy, it is what happens with the base with the last 5 instances. The best case of adaptation has been generated in the intermediate case, where with 10 instances the individual can create a consistent learning rule and at the same time short enough to fit the new cases without much learning latency.

From these experiments we can conclude that when we use the M5P-Reinforcement Learn Architecture and we configure the learning with a regular tax of experience storage we can ensure satisfactory evolution and adaptation of the agents.

Greedy methods such as stabilize the behavior keeping the best rule were not evaluated.

6. Emergent Behaviors Generated

As in [Machado et al 2010], through the resource management system for generating strategies present in the application it was able to adapt to the enemy and environment using emergent behaviors, in the experiments done in the New World environment, the agents have adapted, in their own way, to the situation. As can be seen in Table 3 in a same test and environment, and with the same configuration of waypoints, different agents were able to generate completely different efficient rules.

Table 3. Rules Generated by M5P Algorithm of Three Individuals in the Day 20

```
Pizza_Store <= 0.5 : LM1 (15/26.974%)
Pizza_Store >  0.5 : LM2 (14/51.011%)
LM num: 1
satisfaction =     0.6785 * office          +  1.759 * Pizza_Store
                     + 0.9362 * circus      +  1.3564 * park          +  0.3925
LM num: 2
satisfaction =    1.2996 * office   +  1.8197 * Pizza_Store
                      + 0.8736 * boite      +  2.565

LM num: 1
satisfaction =    1.4356 * police   +  0.8362 * Pizza_Store
                  - 1.187 * boite   +  1.3149 * circus          +  2.0125

LM num: 1
satisfaction =    -0.6679 * industry +  2.9391 * Pizza_Store +  2.5821
```

Reviewing the rules of each individual (Table 3) we realize that the first, if doesn't have much chance to go to the pizzeria, prefers to work in the office and have fun at the pizzeria, at the circus and/or at the square, but if it has the chance to go to the pizzeria, prefers more work in the office and also have fun at the pizzeria and/or at the boite. The second decided much more for the work at the police station, and entertainment at the pizzeria and circus, this avoids the most the club. The third is most simple: works anywhere but avoids the industry and tends to have fun ate the pizzeria.

7. Architecture Applications

Considering a multi-agents environment with learning utilizing a M5P Reinforcement Architecture we can highlight the applications:

- Using criteria such as the restriction on learning mediated by a challenge function, each intelligent agent can learn just enough to ensure good gameplay regarding the balancing of the difficulty in a game of strategy.
- The intelligent collective behavior generated by the interaction of the agents with learning can provide an environment even more diversified for character-based interactive historytelling. For such it would only be necessary the incorporation of a module to ensure consistency of the storyline.

- In applications in the style of life simulation, the use of learning ensures that the agents are ready to adapt to new situations for which they were not previously programed. So responding in a human-like manner to the global interactions.

8. Conclusions

In this work, the M5P Reinforcement Architecture was developed, applied to intelligent agents with learning immersed in complex environments in the field of electronic games and simulation. Its implementation and experiments were conducted in a multi-agent environment called "New World", whose technology was first defined in [Machado et al 2010].

This architecture was suggested for any electronic game genre with multiple agents, especially for character-based interactive storytelling, life simulation and game difficulty dynamic balancing.

The M5P Reinforcement Architecture proposes the on-line training of the individual in the environment, feeding the numerical classifier with the system reward and inducing a tree with linear equations easy to interpret, solving many of the challenges cited by [Manslow 2002].

The experiments demonstrated that the numerical M5P classifier, present in this architecture, offers better performance for a partial instance base (not complete). Which guarantees a satisfactory evolution and a better adaptation of the agents (questioning presented in [Rollings and Ernest 2006]).

In our experiments all the agents had the same loading parameters. A future project would be to evaluate the behavior of this multi-agent system with agents with different profiles. For example, an individual with less willingness to work, other with higher yield (works better or moves faster), another with a slower rate of adaptation (which can take more time to identify a good schedule). Evaluation of the system performance in an overgrowth in the generation of such individuals is also considered future work.

References

Andrade, G., Ramalho, G., Santana, H. and Corruble, V., (2005) "Automatic computer game balancing: A reinforcement learning approach" in *Proceedings of the International Conference on Autonomous Agents*, pp.1229-1230.

Bakkes, S., Spronck P., and van den Herik, J. (2009) "Rapid and Reliable Adaptation of Video Game AI." in *IEEE Transactions on Computational Intelligence and AI in Games*. Vol.1, No, 2.

Bhattacharya, B. and Solomatine, D. P. (2005) "Neural networks and M5 model trees in modelling water level-discharge relationship" in *Neurocomputing*, Vol. 63, pp. 381-396.

Booker, L.B., Goldberg, D.E. and Holland, J.H. (1989) "Classifier Systems and Genetic Algorithms", Artificial Intelligence, vol. 40, pp. 235-282.

Cavazza, M., Charles, F. and Mead, S. J., (2002) "Emergent situations in interactive storytelling" in *SAC '02 Proceedings of the ACM symposium on Applied computing,* pp. 1080-1085.

Crocomo, M. K. and Simões, E. V. (2008) "Um Algoritmo Evolutivo para Aprendizado On-line em Jogos Eletrônicos" In: *SBGames.*

Etemad-Shahidi, A. and Mahjoobi, J., (2009) "Comparison between M5′ model tree and neural networks for prediction of significant wave height in Lake Superior " in *Ocean Engineering* Volume 36, Issues 15-16, pp 1175-1181.

Goertzel, B., Pennachin, C., Geisweiller, N., Looks, M., Senna, A., Silva, W., Heljakka, A. and Lopes, C., (2008) "An integrative methodology for teaching embodied non-linguistic agents, applied to virtual animals in second life". In: *Proceedings of the First Artificial General Intelligence Conference.*

Goldberg, D. E., (1989) "Genetic Algorithms in Search, Optimization and Machine Learning ", in *Addison-Wesley Longman Publishing Co..*

Hong, J. and Cho S., (2005) "Evolving Reactive NPCs for the Real-Time Simulation Game" In: *IEEE 2005 Symposium on Computational Intelligence and Games.*

L.Lab, Secondlife.http://secondlife.com/, (2011).

Machado, A. F. V., Clua, E. W. and Zadrozny B. (2010) "A Method for Generating Emergent Behaviors using Machine Learning to Strategy Games". In: *SBGames.*

Malone, T. W. and Crowston, K., (2004) "The interdisciplinary study of coordination" in *ACM Computing Surveys (CSUR)*, Volume 26 Issue 1, pp. 87-119.

Manslow, J., (2002) "Learning and Adaptation" in *AI GAME PROGRAMMING WISDOM,* pp. 557-566.

Pallay, C., Rehm M. and Kurdyukova E., (2009) "Getting acquainted in Second Life: human agent interactions in virtual environments." In: *Proceeding ACE '09 Proceedings of the International Conference on Advances in Computer Enterntainment Technology .*

Quinlan, J. R., (1992) "Learning with Continuous Classes" in *AI' 92*, pp. 343-348.

Riedl, M., Saretto, C. J., Michael Young, R. (2003) "Managing Interaction Between Users and Agents in a Multi-agent Storytelling Environment." In: *AAMAS '03 Proceedings of the second international joint conference on Autonomous agents and multiagent systems.*

Rollings, A., Ernest, A., (2006) "Fundamentals of Game Design", *New Challenges for Character-Based AI for Games. Chapter 20: Artificial Life and Puzzle Games. Prentice Hall,* pp. 573-590.

Spronck, P., Kuyper, S. I. and Postma, E., (2004) "Difficulty scaling of game AI" in *Proceedings of the 5th International Conference on Intelligent Games and Simulation*, pp. 33-37.

Witze, A., Zvesper, J. A. and Kennerly, E., (2008) "Explicit Knowledge Programming for Computer Games" in *AIIDE.*