COOPERATING ADAPTIVE DEVICES APPLIED IN GENERAL GAME PLAYING

Jose Maria Novaes dos Santos and Joao Jose Neto

Escola Politecnica – Universidade de Sao Paulo (USP), Sao Paulo, Brazil

ABSTRACT

There are formalisms in literature where behavior can be described by a finite set of rules that maps the current device's state into a new one like the finite state machines, statecharts and petri nets. Those formalisms are named as rule-driven devices. A formal device is said to be adaptive if its behavior changes dynamically in response just to its input stimuli and its current state without any external help. Adaptive rule-driven devices can be used for modeling complex problems in artificial intelligence, natural language, reactive systems, synchronous system and others applications. General game playing (GGP) is a research subfield of Artificial Intelligence which aims at developing systems able to play a variety of games, knowing and understanding the rules only in execution time. Most of the proposed GGP systems are based on statistic methods. This paper aims at a new approach in GGP systems, proposing a solution based on adaptive technology modeling. In brief, a GGP system has been developed in a way that it changes its behavior dynamically in response to rules information and its history of games played. This proposal is presented here as well as results of some games played comparing their strategies approach.

KEYWORDS

Modelling, Reactive System, Cooperating Adaptive Devices, Rule-Driven Formalisms, Self-Modifying Machine, Adaptive Automata, General Game Playing, Adaptive Device

1. INTRODUCTION

Many traditional formalisms can be described as a finite set of rules which maps its current state into a new one in response to some stimulus or event. Some states are defined as final states and indicate successfully finished operations. Such formalisms are named rule-driven devices.

A formal device is said to be adaptive if its behavior changes dynamically without any external help. The behavior changes in response to stimuli occurred in the environment. Such device property is called self-modification or adaptivity and has been formally described in [1].

In some situations, modeling of the problem requires simultaneous usage of devices of different nature types.

General Game Playing (GGP) research is in the field of Artificial Intelligence and its purpose is to design and implement systems with the ability to understand the rules of new games and be able to play any game described in a specific language called Game Description Language (GDL). So, the most meaningful characteristic of the GGP is that players do not know the games rules before beginning to play. Hence, this field encourages the development of learning

David C. Wyld et al. (Eds) : SAI, CDKP, ICAITA, NeCoM, SEAS, CMCA, ASUC, Signal - 2014
pp. 291–303, 2014. © CS & IT-CSCP 2014 DOI : 10.5121/csit.2014.41127

algorithms and search mechanisms which could be applied in a wide range of games, similar to human learning in playing games.

Concerning the feature of the adaptive technology and the general game playing, this work presents a GGP system where the behavior is modeled based on adaptive technology.

In this work is presented an approach based in adaptive technology and learning process which uses the history of games, thereby as more games are played, the system will provide more and more options of new moves.

In section 2, we briefly describe the static rule-driven devices. Afterwards, we introduce the concept of adaptivity in section 3. In section 4, we present the concept of Cooperating Adaptive Devices. In section 5, we introduce the main ideas of general game playing. In section 6, we present our system proposal and the results of the tests. Section 7 brings the conclusion and some remarks.

2. RULE-DRIVEN DEVICES

Many formalisms change their state in response to an environment stimulus according to their set of rules. The state of the device comprehends the contents of the whole set of elements that hold information along with the current status of the device.

Some of these traditional formalisms include finite state machines, statecharts, natural language grammars and parsers, ontologies, decision trees, decision tables and petri nets. These devices have been used to model and represent complex problems in many fields such as Artificial Intelligence and Natural Language Processing. One of the most popular among these formulations is the finite state machine, which is widely used for describing the behavior of real time systems, communication protocols, software design and language parsing.

A rule-driven device is characterized by a set of rules that determines its reaction to a given set of conditions. Each rule represents a change in its state in response to a set of conditions and events. The device starts its operation at some known initial state and modifies its state by applying the best suited of its rules.

The device is said to be deterministic if and only if, for any given state and for any input stimuli, there is just a single next state defined by its set of rules. Otherwise, the device is said to be non-deterministic. In other words, a non-deterministic device has a set of rules that maps at least one possible state into more than one next state. We usually achieve better efficiency from deterministic devices than from non-deterministic equivalent ones. Unfortunately, for some problems, it may be very difficult or even impossible to obtain solutions based on deterministic devices.

3. ADAPTIVE RULE-DRIVEN DEVICES

A formal rule-driven device is said to be adaptive if its behavior may change dynamically. The concept of adaptivity applies to any device that is able to change its own behavior. In particular, when the behavior is determined by a set of rules, adaptivity is easily achieved by changing the set of rules that define the device's behavior.

The general formulation for rule driven adaptive devices, can be thought as an adaptive layer placed around the original subjacent non-adaptive device (standard rule-driven device).

By building such devices, one can conceptually identify two major components: an equivalent underlying device similar to those described in the previous section and an adaptive mechanism responsible for the adaptivity. One notation elaborated for representing adaptive formalisms in a way as similar as possible to its original non-adaptive underlying formulation was presented in detail in [1].

Historically, adaptive devices emerged from automata theory and most of the early applications were in the fields of formal languages, and later, in computer languages. Some of those early works are described in [2], [3] and [4].

In [5], we have the formal formulation of the Adaptive Automaton. Hereafter, some works have been developed applying the adaptive technology using classical formalisms like Finite Automaton [6], Statechart [7], Markov Chain [8], Grammar [9] and Decision Table [10]. In [11], we have the conceptual proof that Adaptive Automaton and the Turing Machine have the same power of expression.

In [12], the use of adaptive technology is presented as an alternative approach for modeling biological species while in [13] we have the adaptive version of the GARP (Genetic Algorithm for Rule-set Production) algorithm for mapping the environmental distribution of the “Penonapis” and “Cucurbita”.

The adaptive technology has been applied also in other fields like robotic [14], programming language [15], [16] and [17] and reactive system modeling [18], [19] and [20].

4. COOPERATING ADAPTIVE DEVICES

We have briefly mentioned the adaptive rule-driven device´s main idea in the previous section. In this section, we present the Cooperating Adaptive Devices. Concisely, this formalism is formed by a group of adaptive devices with the feature of communication between them, increasing the number of real complex problems that can be modelled. One device can send messages to any other devices, that in response, can modify its state. Based on this behavior, one device can cooperate with another one, sending messages to communicate a condition in the environment that may cause a state changing in the second device.

Cooperating Adaptive Devices (CAD) are composed of a finite group of rule-driven devices of potentially heterogeneous types with a common communication mechanism (CCM). A further device or mechanism is employed for managing and coordinating message communication (MCM), resulting from interactions between a pair of distinct devices. The main consequence of such a decision is that devices have now the capability of sending a message that may cause the behavior changing of other devices.

The concept of the CAD can be summarized as the property in which, based on its behavior or on the basis of its current situation and input stimuli, any member of the group of heterogeneous devices can send communication instructions resulting in changing device rules.

Any action for changing rules belonging to any other device is initiated by a message communication from a source device to a target device, using a common communication mechanism as an intermediary agent for such interactions. We call this intermediary agent the common communicate mechanism (CCM).

Figure 1 illustrates an adaptive interaction among devices. Device 4 sends a message to device 1 through the common communicate mechanism (CCM). Such interaction between devices is represented by the red arrow in figure 1.

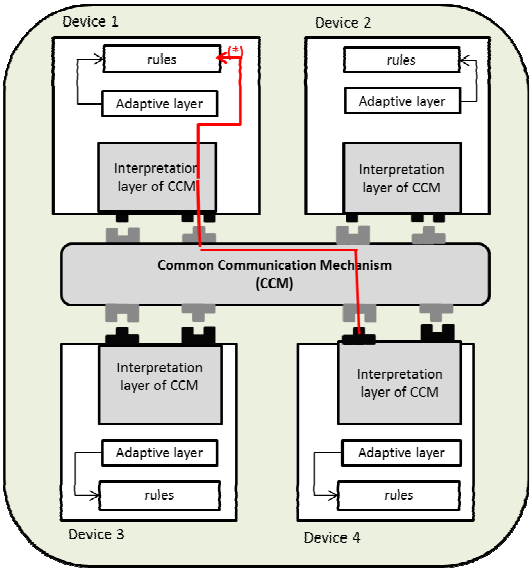

Figure 1. Illustration of cooperating adaptive devices

Technically, we can say that each adaptive device itself changes its own rules, however, any device belonging to the group of CAD has the capability to send messages to any other device of the group, which may cause rules modifications. This capability is represented in figure 1 by the layer of communication and interpretation.

Communications between devices, managed and controlled by CCM mechanism is performed using a communication protocol, which is named CP protocol. The communication between devices is formed of a sequence of standard messages from CP, like “insert rule”, “delete rule”, “trigger event”, etc.

Definition

Similar to the behavior of adaptive devices, a built-in counter T2 is defined for cooperating adaptive devices with initial value 0 and automatic increments by 1 whenever a non-null communication message is executed. Thus, each name of a device during a step tk ($tk \geq 0$) is identified for each value assumed by T2.

Thus, the cooperative adaptive device can be described as:

$$CAD_{tk} = (\{ AD^{1}_{k1,tk} , AD^{2}_{k2,tk} , \ldots , AD^{m}_{km,tk} \} , CCM, MCM) \quad (1)$$

In this formulation $AD^{r}_{kr,tk}$ represents an adaptive device. CAD_{tk} is said cooperative adaptive device when for all operations, in each step tk ($tk \geq 0$), any element of the set of devices follows the behavior of the corresponding element until the execution of some non-null message in the common communicate mechanism (CCM), when the current step tk terminates and the next one (tk+1) starts.

Similar to adaptive actions, each step increment is composed of two groups of messages. The first one is performed before the rule execution and the second one after the execution. For a non-null message, at least one of the components must be non-null. A message is formed by elementary messages and its execution may cause multiple additions and/or multiple deletions in some other device´s rules.

The cooperative adaptive device starts its operation at some known initial shape for all m (m > 1) devices of the CAD, (AD^1_0, AD^2_0, …, AD^m_0) from the perspective of the message communications. Remembering that each device AD^r can change its rules through conventional adaptivity, we can express CAD as ($AD^1_{k1,0}$, $AD^2_{k2,0}$, …, $AD^m_{km,0}$) whereas each $AD^r_{kr,0}$ (for r=1,..,m and kr=k1,…,km ; all kr ≥0) means a device at some internal step of the adaptivity and at the initial step of the message communications. So, in this state, no message communication has occurred. However, each device can have changed its own rules through conventional adaptivity.

At step tk (tk≥0), an input stimulus changes the cooperative adaptive device CAD to the next step (tk+1) if, and only if, any non-null message communication is performed. Then, in any combination of step kr for all m devices (kr=k1, k2, …, km) and the step tk, every device can be represented in the form $AD^r_{kr,tk}$. In this formulation, kr indicates the step of its adaptivity while tk indicates the step of the message communications for all devices.

So, we have:

$$CAD = \{(AD^r)_{tk}\} \quad (2)$$

$$(AD^r)_{tk} = AD^r_{kr,\,tk} \quad (3)$$

$$AD^r_{kr,\,tk} = (C^r_{kr,tk}, IAR^r_{kr,tk}, S^r, c^r_{kr,tk}, A^r, NA^r, BA^r, AA^r, IBA, IAA) \quad (4)$$

Where r=1,..,m; kr=k1,…,km and tk ≥ 0.

- In this formulation, we have:
- CAD = ({ AD_0^1 , AD_0^2 , ... , AD_0^m }, CCM, MCM). The cooperative adaptive device consists of an initial set of m adaptive devices, a common communicate mechanism (CCM), managed by a mechanism in order to ensure only one concurrent communication (MCM).
- $(AD^r)_{tk}$, for r = 1, ... , m represents the adaptive state of each device of the cooperative adaptive device, $(AD^r)_0$ is its initial state (tk=0) and it is defined by its set of rules $IAR^r_{kr,0}$ for kl step of the adaptivity (kr ≥ 0).
- $GADM_{tk} = \{AD^1_{k1,tk}, AD^2_{k2,tk}, ..., AD^m_{km,tk}\}$ represents the devices at step tk of message communication, being $AD^r_{kr,0}$ its initial state. Each device $(AD^r_{kl})_{tk}$ is the mirror of adaptive device AD^r at step tk, each one within its own step kl (kr ≥ 0 and m > 1; for kr = k1 ...,km) of adaptivity.
- $C^r_{kr,tk}$ is the set of all possible states for $AD^r_{kr,tk}$ for tk and kr steps (tk ≥ 0 e kr ≥ 0 , for r=1...m).

- IBA e IAA (for r=1,…,m) are sets of message communication, both containing the null action ε ($\varepsilon \in IBA \cap IAA$).

- S^r (for r=1,…,m) is a finite set of all possible events considered valid input stimuli for AD^r, containing the null event ($\varepsilon \in S^r$).

- The input stimulus w^r is:
- $w^r = w_1^r\, w_2^r\, w_3^r\, w_4^r \ldots\ldots w_n^r$ ($w^r \in S^r$) (for r=1,...,m and nr ≥ 0).

- $c^r_{kr,0}$ belongs to C^r and is the initial device configuration ($c^r_{kr,tk} \in C^r_{kr,tk}$), for r = 1 .... , m; kr= k1 ...,km and tk ≥ 0. Before the occurrence of the first adaptive action (kr = 0) and the first communication message (tk = 0), $c^r_{0,0}$ is the initial configuration of the device "r".

- A^r is the subset of its accepting states (acceptance) of device r, $A^r \subseteq C^r$ (for r=1,...,m)

- NA^r is a finite set of output symbols of device r (for r = 1, ... , m).

- IAR^r_{tk} is the finite set of all possible CAD rules, given by a relation $IAR^r_{tk} \subseteq IBA \times BA^r \times C^r \times S^r \times C^r \times NA^r \times AA^r \times IAA$. The rules of IAR^r_0 (for r=1,...,m) define the initial performance of CAD devices. Device rule containing message communication changes another device rules set by adding and/or deleting rules. The rules HAR^r_{tk} (r=1, .., m) have the form har^r = (iba, ba^r, c_i^r, s^r, c_j^r, z^r, aa^r, iaa), meaning that, in response to some stimulus $s^r \in S^r$, har^r initially performs the first group of message iba (group before), then, the adaptive rule ar^r = (ba^r, c_i^r, s^r, c_j^r, z^r, aa^r) and finally performs the second group of message iaa (group after). The adaptive rule action, ar^r, is performed as described in [1].

5. GENERAL GAME PLAYING

General game playing is associated to the system's main goal which is to play games without previous knowledge with respect to its rules. The challenge is to construct a system able to understand the games rules, which are described in GDL (General Definition Language) [21], similar to the Prolog language.

Many works have been developed in Artificial Intelligence related to playing specific games like chess. In these systems, the programmer designer specifies the strategies of the programs to play just one game based on a set of known rules. The strategies are specific to a game play and it is difficult to use the same programs to play another game. Such programs have in common that they depend heavily on elaborated game-dependent knowledge provided by the developers.

In general game playing, on the contrary, the aim is to create programs capable of playing a wide range of different games, even those that may have never been played before. Since 2005, the Stanford Logic Group organizes yearly a competition in GGP games, improving studies concerning artificial intelligence, search techniques and knowledge representation, machine learning, knowledge discovery and online optimization.

The GGP games are synchronous and finite and can be modeled using the finite state machine formalism. In this representation, in every step of the game, we must have moves of all players. After the moves of all players, the environment is updated as a response without any external interference according to the set of rules. By definition, every GGP has at least one terminal state that can be reached after a finite number of rounds.

GGP is similar to the traditional game theory, however, there are some exceptions like the movement and modeling. GGP is modeled as a state machine while the traditional games are modeled as a tree. The GGP movements are synchronous.

The challenge in GGP systems is developing general method learning strategies concerns only the definitions rules. For example, a simple game has around many possible states, so identifying all possible moves leading to a final state is not feasible.

Concerning the restriction described, there are two main questions to be addressed in developing GGP systems: search and evaluation. Search implies in the ability of the system to think forward while evaluation intends to provide a mechanism that associates the merits of the current position. The merit value must be as greater as possible, meaning the maximum value is associated to a winning state. And the main challenge is to discover the relevant information that must be considered to build the mechanism during the game.

In GGP, the current state-of-the-art approach to search the game tree is the Upper Confidence Bounds Applied to Tree (UCT), where the goal is to provide balance between exploration and exploitation. In earlier competitions, one of the most applied approach was the Monte-Carlo Tree Search (MCTS) [22].

The basic idea of the MCTS simulation is to play in a randomly way until reaching a terminal state. In this state, a goal value is assigned to all states reached in the path. Each state value assigned is estimated by the average result of all simulations which visited this state. In short, the MCTS approach is comprised of four phases: selection, expansion, simulation and back propagation of results.

The work presented in [23], is a related work to the proposal presented in this paper. See more details concerning general game playing in works [24], [25], [26] and [27].

6. GENERAL GAME PLAYING APPLICATION

A problem related to developing a general game playing is to design a system which plays any game described in GDL language with the capability of learning play strategies dynamically.

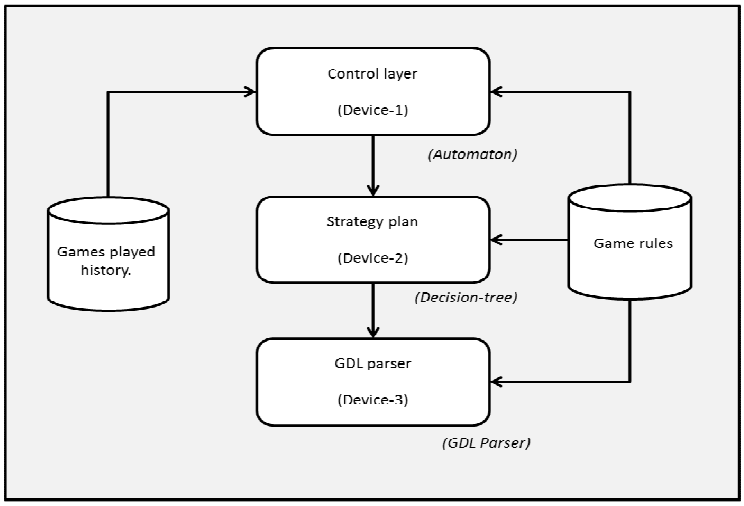

Figure 2. Three devices: an automaton, a decision-tree and a GDL parser

A problem related to developing a general game playing is to design a system which plays any game described in GDL language with the capability of learning play strategies dynamically. This problem can be modeled by the Cooperating Adaptive Devices, which has the feature of representing dynamic and autonomous behavior.

The modeling is based on three distinct type rule-driven devices. The first is an automaton, called "Control layer", which is responsible to analyses the history of the games already played and the game rules to learn new strategies to play the game. Based on the learned strategy information, this device causes the inclusion of new rules in the second device, the decision tree, which is responsible for deciding what move is the best choice to apply in each round play of the game with respect to its rules and the strategy information. The third device, the GDL parser, is responsible to execute the rules, changing the current state of the game. Figure 2 illustrates these Cooperating Adaptive Devices.

6.1. Strategies Approach

We developed a GGP system where strategies are based on learning from historical information about the games played. In our approach, we have four different strategies based on history, one based on random strategy and one based on statistics. The statistic strategy is based on MCTS approach mentioned in section 5. The random strategy was used to compare the efficiency of each different strategy.

Our strategies are based on two simple ideas. The first is to follow some path of a successful game, comparing the state of the current game with the historic games. Based on this idea, we have come up with strategies A and B. The second idea is to follow a final state of a successful game. Based on this idea, we have come up with strategy C. The strategy D is based on a final state, but it considers only relevant information that leads to a final state.

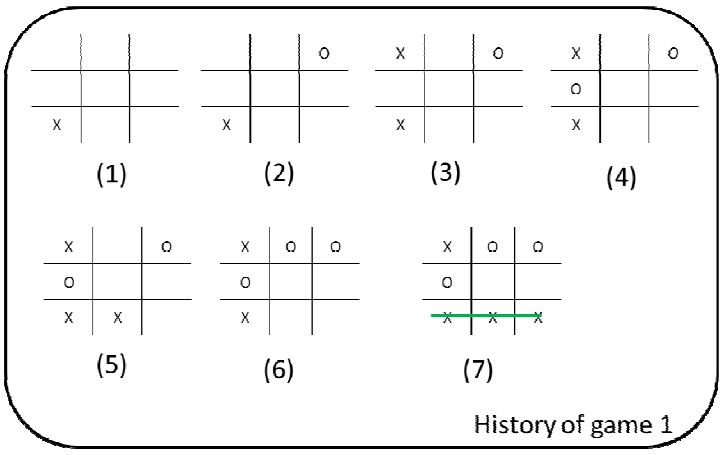

Figure 3. Example of "noughts and crosses" game stored in the historic database

To illustrate our ideas, let's suppose our historic database formed only by one game played as showed in figure 3 (history of game 1). In this situation and in another "noughts and crosses" game, with current configuration as showed in figure 4, a role player "X" has only one option to move based on strategies A or B, the cell(1,1), like the third round of the game 1. On the order hand, based on strategy C, we have three options considering the final state of game. Strategy D considers only two positions, the last row positions considering the game 1. In every round, we consider all possible moves. After, we analyse all next configurations, concerning the strategy.

Concisely, our strategies are:

- Strategy R.
  In every step of the game, there is a random choice of all valid moves.

- Strategy E.
  The statistic strategy verifies the state with the highest average of the visited number and the number of victories.

- Strategy A.
  It looks for a game state that is the same as the current game state. If there is more than one choice, the most recent one is chosen.

- Strategy B.
  It looks for a game state that is most similar to the current game state. If there is more than one choice, the most recent one is chosen.

- Strategy C.
  It looks for a final state of a successful game that is most similar to the current game state.

- Strategy D.
  It looks for a final state of a successful game that is most similar to the current game state, considering only the relevant information in the final state.

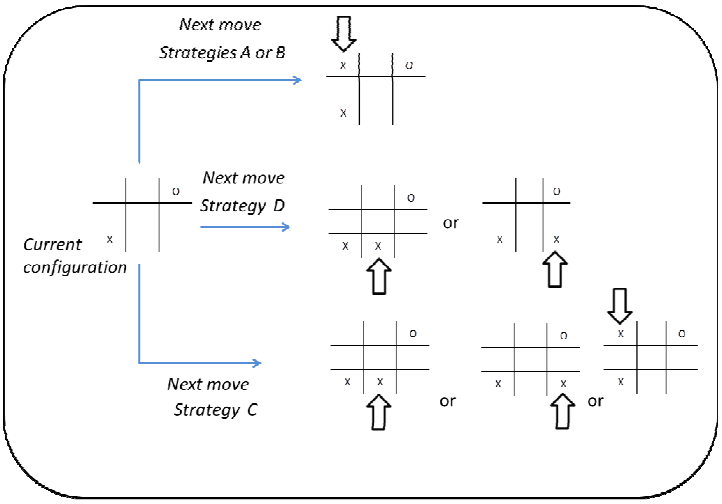

Figure 4. Example of possible moves, according strategy

6.2. Result of Games Played

In our tests, we played the “noughts and crosses” and the “cross-block” games**.** The “noughts and crosses” game is the classical game for two players who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three respective marks in a horizontal, vertical, or diagonal row wins the game. The “cross-block” game is for two players who take turns marking

the spaces in a 4×4 grid. The cross role wins if its marks link the left side to the right side or the upper side to the bottom side. The block role wins if the cross loses.

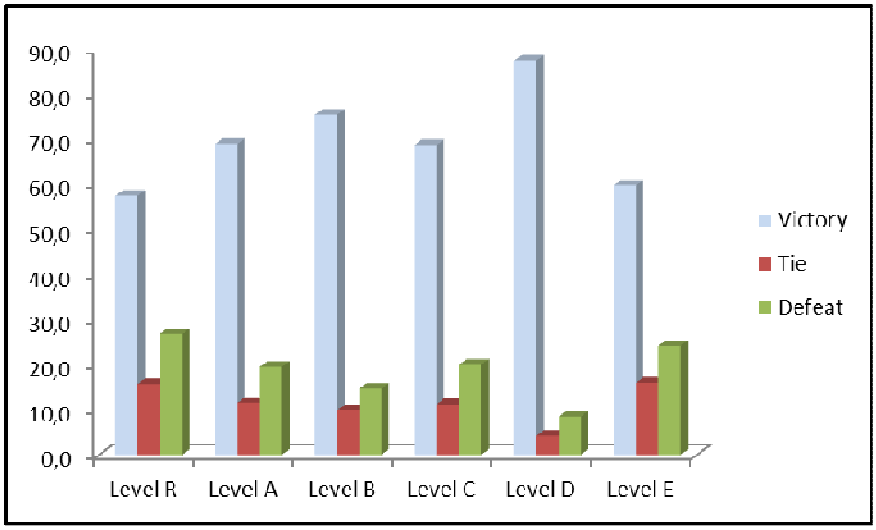

Figure 5. Results of “noughts and crosses” games played

We played several games combining all possible strategies for the “x” role player in the “noughts and crosses” games and maintaining the same strategy for the opponent, the random one. For the “cross-block” game, we combined all possible strategies in the cross role, maintaining the same strategy (random) for the block role. Each combination was played 500 times. In figures 5 and 6, we have the graphs of the results of games played

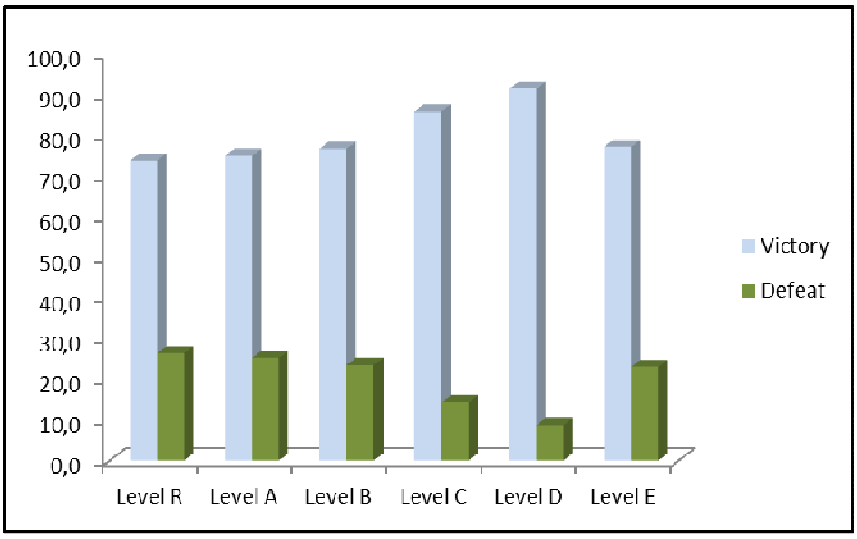

Figure 6. Results of “cross-block” games played.

Based on result analysis, we conclude that the best strategy is “D”, because it only considers the information that is essential in the final conditions, having fewer no important moves.

It is important to highlight that all strategies have improvements comparing to the random strategy. These results stimulate new studies in learning algorithm, knowledge representation and adaptive technology applied to GGP.

7. CONCLUSIONS

The GGP system behavior depends which game will be played, represented in the rules of the game. Thus, it is a typical application that can be modeled using adaptive devices.

This work´s main result is the use of the adaptive technology to model and develop a general game playing system. The results have showed that this proposal encourages new approaches concerning techniques in knowledge representation and learning reasoning. We have many GGP application based on statistic approaches and this work presents a new perspective of dealing with unknown environment. The preliminary results have encouraged new challenges and we intend to test the system under even more complex games thus improving the techniques in learning strategies.

We hope this work will assist in the conception and building of Cooperating Adaptive Devices for a new and cleaner perspective in activities involving complex problem solving with adaptive technology.

REFERENCES

[1] Neto, João José, (2002) "Adaptive rule-driven devices-general formulation and case study." Implementation and Application of Automata. Springer Berlin Heidelberg, pp. 234-250.

[2] Cabasino, Simone & Pier S. Paolucci & Gian Marco Todesco, (1992) "Dynamic parsers and evolving grammars." ACM Sigplan notices 27.11, pp. 39-48.

[3] Burshteyn, Boris, (1990) "Generation and recognition of formal languages by modifiable grammars." ACM SIGPLAN Notices 25.12, pp. 45-53.

[4] Rubinstein, Roy S. & John N. Shutt, (1995) "Self-modifying finite automata: An introduction." Information processing letters 56.4, pp. 185-190.

[5] Neto, João José, (1994) "Adaptive automata for context-dependent languages." ACM Sigplan Notices 29.9, pp.115-124.

[6] Pistori, Hemerson & Priscila S. Martins & Amaury Antonio de Castro-Jr., (2005) "Adaptive finite state automata and genetic algorithms: Merging individual adaptation and population evolution". Springer Vienna.

[7] Almeida-Jr, Jorge Rady (1995) "STAD: Uma ferramenta para representação e simulação de sistemas através de statecharts adaptativos". Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo (in portuguese).

[8] Neto, João José & Bruno Arantes Basseto, (1999) "A Stochastic Musical Composer Based on Adaptative Algorithms." Proceedings of the 6th Brazilian Symposium on Computer Music-SBC&M99, Rio de Janeiro.

[9] Iwai, Margarete Keiko, (2000) "Um formalismo gramatical adaptativo para linguagens dependentes de contexto." Departamento de Computação e Sistemas Digitais (PCS)-Escola Politécnica, Tese de Doutorado, Escola Politécnica, Universidade de São Paulo (USP), São Paulo, SP (in portuguese) .

[10] Pedrazzi, Thiago Carvalho & Angela Hum Tchemra & Ricardo L. Azevedo Rocha, (2005) "Adaptive Decision Tables A Case Study of their Application to Decision-Taking Problems". Springer Vienna, pp. 341-344.

[11] Rocha, Ricardo L. Azevedo & João José Neto, (2001) "Autômato Adaptativo, Limites e Complexidade em Comparação com Máquina de Turing". In: Proceedings of the second Congress of Logic Applied to Technology – LAPTEC'2000. São Paulo: Faculdade SENAC de Ciências Exatas e Tecnologia, pp. 33-48 (in portuguese).

[12] Rodrigues, Elisângela Silva da Cunha & Fabricio Augusto Rodrigues & Ricardo Luis Azevedo Rocha, & Pedro Luiz Pizzigatti Correa, (2011) "Adaptive Approach for a Maximum Entropy Algorithm in Ecological Niche Modeling." Latin America Transactions, IEEE (Revista IEEE America Latina) 9.3, pp. 331-338.

[13] Stange, Renata Luiza & Teresa Cristina Giannini & Fabiana Soares Santana & João José Neto & Antonio Mauro Saraiva (2011) "Evaluation of Adaptive Genetic Algorithm to Environmental Modeling of Peponapis and Cucurbita." Latin America Transactions, IEEE (Revista IEEE America Latina) 9.2, pp.171-177.

[14] Hirakawa, Andre Riyuiti & Antonio Mauro Saraiva & Carlos Eduardo Cugnasca, (2007) "WTA 2007-III. 1-Adaptive Automata Applied on Automation and Robotics (A4R)." Latin America Transactions, IEEE (Revista IEEE America Latina) 5.7, pp. 539-543.

[15] Pelegrini, Eder José & João José Neto, (2008) "A dynamically variable code execution model, based on adaptive automata." Latin America Transactions, IEEE (Revista IEEE America Latina) 6.5, pp. 424-435.

[16] Silva, Salvador Ramos Bernardino da & João José Neto, (2011) "Proposal of a High-Level Language for Writing Self Modifying Programs." Latin America Transactions, IEEE (Revista IEEE America Latina) 9.2, pp. 192-198.

[17] Sabaliauskas, Jorge Augusto & Ricardo Luis Azevedo da Rocha, (2011) "Project and Implementation for a Programming Language Suitable to Express Adaptive Algorithms." Latin America Transactions, IEEE (Revista IEEE America Latina) 9.6, pp. 969-973.

[18] Almeida-Jr, Jorge Rady & João José Neto, (1999) “Using Adaptive Models for System Description.” In: IASTED International Conference Applied Modelling and Simulation, 1999, Cairns. Applied Modelling and Simulation, p. 452-457.

[19] Neto, João José & Jorge Rady de Almeida-Jr, (1999) "Modeling Adaptive Reactive Systems." International Conference on Applied Modelling and Simulation. Cairns, Australia:[sn].

[20] Neto, João José & Jorge Rady Almeida-Jr & José Maria Novaes dos Santos, (1998) "Synchronized satecharts for reactive systems.” In: Proceedings of the IASTED International Conference on Applied Modelling and Simulation. Honolulu, Hawaii, pp. 246-251.

[21] Genesereth, Michael & Nathaniel Love & Barney Pell, (2005) "General game playing: Overview of the AAAI competition." AI magazine 26.2: 62.

[22] Cazenave, Tristan, (2009) "Nested Monte-Carlo Search." IJCAI. Vol. 9. 2009.

[23] Swiechowski, Maciej & Jacek Mandziuk, (2013) "Self-Adaptation of Playing Strategies in General Game Playing." . IEEE Trans. Comput. Intell. AI Games.

[24] Méhat, Jean & Tristan Cazenave, (2010) "Combining uct and nested monte carlo search for single-player general game playing." Computational Intelligence and AI in Games, IEEE Transactions on 2.4, pp. 271-277.

[25] Bjornsson, Yngvi & Hilmar Finnsson, (2009) "Cadiaplayer: A simulation-based general game player." Computational Intelligence and AI in Games, IEEE Transactions on 1.1, pp. 4-15.

[26] Schiffel, Stephan & Michael Thielscher, (2007) "Fluxplayer: A successful general game player." Proceedings of the National Conference on Artificial Intelligence. Vol. 22. No. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press.

[27] CLUNE, Jame, (2007) “Heuristic evaluation functions for general game playing.” In:AAAI, vol. 7, pp. 1134-1139.

AUTHORS

José Maria Novaes dos Santos, graduated in Applied Mathematic in São Paulo University (USP), Brazil (1989) and MSc in Computer Engineering in the Polytechnical School (EPUSP) of Sao Paulo University (USP), Brazil (1997). He is doctoral student in Computer Engineering in the Polytechnical School (EPUSP) of Sao Paulo University (USP), Brazil. His main interests are adaptive devices, adaptive technology, machine learning, system modeling, computer learning, ontology and natural language processing.

João José Neto graduated in Electrical Engineering (1971), MSc in Electrical Engineering (1975) and doctor in Electrical Engineering (1980), and "livre docente" associate professor (1993) in the Polytechnical School of Sao Paulo (EPUSP) University. Nowadays he is the head of LTA - Adaptive Technology Laboratory at the Department of Computer Engineering and Digital Systems at the EPUSP. His main experience is in the Computer Science area, with emphasis on the foundation of computer engineering and adaptivity. His main activities include adaptive devices, adaptive technology, adaptive automata and their applications to computer engineering and other areas, especially in adaptive decision making systems, natural language processing, compiler construction, robotics, computer education, intelligent system modeling, computer learning, pattern matching, inference and other applications founded on adaptivity and adaptive devices.