The Applications of Automata in Game Theory

Chapter · May 2013

DOI: 10.4018/978-1-4666-4038-2.ch011

3 authors, including:

Khaled Suwais
Arab Open University - Saudi Arabia
29 PUBLICATIONS **26** CITATIONS
SEE PROFILE

Muhammad Rafie Mohd Arshad
Universiti Sains Malaysia
49 PUBLICATIONS **119** CITATIONS
SEE PROFILE

Intelligent Technologies and Techniques for Pervasive Computing

Kostas Kolomvatsos
University of Athens, Greece

Christos Anagnostopoulos
Ionian University, Greece

Stathes Hadjiefthymiades
University of Athens, Greece

A volume in the Advances in Computational Intelligence and Robotics (ACIR) Book Series

Information Science REFERENCE

Managing Director: Lindsay Johnston
Editorial Director: Joel Gamon
Book Production Manager: Jennifer Yoder
Publishing Systems Analyst: Adrienne Freeland
Development Editor: Austin DeMarco
Assistant Acquisitions Editor: Kayla Wolfe
Typesetter: Christina Henning
Cover Design: Jason Mull

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Intelligent technologies and techniques for pervasive computing / Kostas Kolomvatsos, Christos Anagnostopoulos and Stathes Hadjiefthymiades, editors.
pages cm
Summary: “This book provides an extensive discussion of such technologies, theories and practices in an attempt to shed light on current trends and issues in the adaption of pervasive system”-- Provided by publisher.
Includes bibliographical references and index.
ISBN 978-1-4666-4038-2 (hardcover) -- ISBN 978-1-4666-4039-9 (ebook) -- ISBN 978-1-4666-4040-5 (print & perpetual access) 1. Ubiquitous computing. 2. Multiagent systems. 3. Computational intelligence. I. Kolomvatsos, Kostas, 1973- II. Anagnostopoulos, Christos, 1978- III. Hadjiefthymiades, Stathes, 1971-
QA76.5915.I545 2013
006.3--dc23
2013001109

This book is published in the IGI Global book series Advances in Computational Intelligence and Robotics (ACIR) Book Series (ISSN: 2327-0411; eISSN: 2327-042X)

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 11
The Applications of Automata in Game Theory

Sally Almanasra
Universiti Sains Malaysia, Malaysia

Khaled Suwais
Arab Open University, Saudi Arabia

Muhammad Rafie
Universiti Sains Malaysia, Malaysia

ABSTRACT

In game theory, presenting players with strategies directly affects the performance of the players. Utilizing the power of automata is one way for presenting players with strategies. In this chapter, the authors studied different types of automata and their applications in game theory. They found that finite automata, adaptive automata, and cellular automata are widely adopted in game theory. The applications of finite automata are found to be limited to present simple strategies. In contrast, adaptive automata and cellular automata are intensively applied in complex environment, where the number of interacted players (human, computer applications, etc.) is high, and therefore, complex strategies are needed.

INTRODUCTION

Any problem with interacted participants and actions can be treated as a game. When car drivers put a plan to drive in heavy traffic, they are actually playing a driving game. When users bid on bidding-based Websites, they are actually playing an auctioning game. In election, choosing the platform is a political game. The owner of a factory deciding the price of his product is an economic game. Obviously, game theory can be presented in wide range of applications.

Game theory is a mathematical tool that can analyze the interactions between individuals strategically. The interactions between agents, who may be individuals, groups, firms are interdependent. These interdependent interactions are controlled by the available strategies and their corresponding

DOI: 10.4018/978-1-4666-4038-2.ch011

payoffs to participants. However, game theory studies the rational behavior in situations involving interdependency (McMillan, 1992). Therefore, game theory will only work when people play games rationally and it will not work on games with cooperational behavior.

Generally, the game consists of the following entities:

- **Players:** Where one side of the game tries to maximize the gain (*payoff*), while the other side tries to minimize the opponent's score. However, these players can be humans, computer applications or any other entities.
- **Environment:** This includes board position and the possible moves for the players.
- **Successor Function:** The successor function includes actions and returns a list of (move, state) pairs, where each pair indicates a legal move and the resulting state.
- **Terminal Test:** The terminal test specifies when the game is over and the terminal state is reached.
- **Utility Function:** The utility function is the numeric value for the terminal states.

The scientists of inter-disciplinary community believe that the time has come to extend game theory beyond the boundaries of full rationality, common-knowledge of rationality, consistently aligned beliefs, static equilibrium, and long-term convergence (Izquierdo, 2007). These concerns have led various researchers to develop formal models of social interactions within the framework of game theory.

The first formal study of games was done by Antoine Cournot in 1838. A mathematician, Emile Borel suggested a formal theory of games in 1921, which was extended by another mathematician John von Neumann in 1928. Game theory was established as a field in its own right after the 1944 publication of the monumental volume Theory of Games and Economic Behavior by von Neumann and other economists. In 1950, John Nash proved that finite games have always have an equilibrium point, at which all players choose actions which are best for them given their opponents' choices. This concept has been a basic point of analysis since then. In the 1950s and 1960s, game theory was extended and developed theoretically and applied to problems of war and politics. Since the 1970s, it has created a revolution in economic theory. Moreover, it has found applications in sociology and psychology, and found links with evolution and biology. Game theory received special attention in 1994 with the awarding of the Nobel Prize in economics to Nash (Almanasra, 2007).

The attractive point in studying games is that models used in games are applicable to be used in real-life situations. Because of this, game theory has been broadly used in economics, biology, politics, low, and also in computer sciences. Examples on the use of game theory in computer science include interface design, network routing, load sharing and allocate resources in distributed systems and information and service transactions on Internet (Platkowski & Siwak, 2008).

One of the crucial factors which affect the performance of players in a given game is the behavior (strategy) representation. We found that different techniques are used to represent players' behavior in different games. One of the successful techniques is the use of automata-based model to represent and control participated agents. In the next sections, we will discuss different types of automata and their affect on games.

AUTOMATA TYPES

An automaton (plural: automata or automatons) is a self-operating machine. The output to one automaton is a combination between the consequences of the current input and the history of the machine's previous input. An automaton is designed to automatically follow a predetermined sequence of operations or respond to encoded

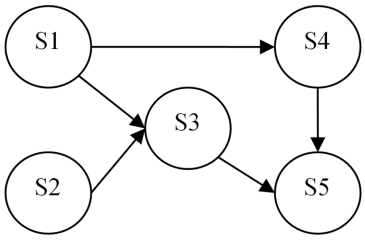
instructions. There are many types of well known automata, such as finite automata, turing machines, push–down automata, adaptive automata as well as other variations such as, random access machines, parallel random access machines and arrays automata. Experimental simulations of automata methods carried out in different researchers have recommended the automaton approach in the solution of many interesting examples in parameter optimization, hypothesis testing, and in game theory (Almanasra & Rafie, 2010).

Learning Automata Theory

Learning is strongly associated with past experiences, where these experiences are meant to permanently change the entity behavior. Learning system is characterized by its ability to improve its behavior with time in some sense tending towards an ultimate goal (Narendra & Wright, 1977). Numerous approaches have been proposed for developing learning systems. One of the approaches which gained considerable attention is based on automata. This approach tends to present a given problem as one of finding the optimal state among a set of possible states. This approach is referred to by learning automaton. Figure 1 illustrates the concept of automaton, where the possible states are presented by S1, S2, S3, S4, and S5.

A learning automaton operates in a random environment. The actions are chosen based on the inputs received from the environment to find the optimal. The learning schemes come in two structures: fixed structure and variable structure. Fixed structure schemes in stationary random environments are described by homogeneous Markov chains. On the other hand, variable structure schemes are described by Markov processes. Much of the effort in these areas has been directed towards achieving expedient, optimal behavior (Narendra & Wright, 1977). However, learning automaton has a finite number of output actions, one of which is selected at each instant. Every selected action will be either rewarded or punished by the environment, which leads to update the probability distribution defined over the actions. Choosing the proper updating algorithms, leads to desirable asymptotic behavior of the learning automaton (Thathachar, 1985).

Figure 1. The concept of automaton

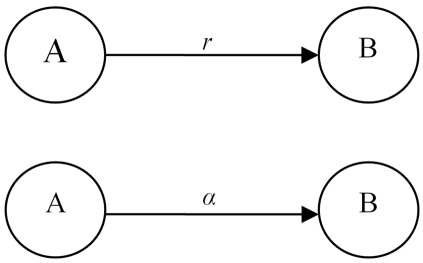


Finite-State Automata

Finite automaton is one of the well-studied computation models in theoretical computer science. Finite automata are composed of fixed finite memory, which takes its decisions using that finite memory, a finite non-empty alphabet, a transition function, an initial state and a set of final states. Transitions map ordered pairs specifying the current state and the current input symbol into a new state (Maenner, 2008). Finite Automata are classified into Deterministic Finite Automaton (DFA) and Non-Deterministic Finite Automaton (NFA). DFA is only permitted to be in one state in any time. On the contrast, NFA can be in two or more states at any given time. Technically, adding restrictions to DFA is sufficient to transform it to NFA. However, there are two types of transitions from state A to state B: *Input transitions* (**A**, r) $\rightarrow$ **B**, with an input symbol r, and *Empty transitions* (**A**, α) $\rightarrow$ **B**, which do not modify the input (Figure 2).

DFA can be viewed in two schemes: Graph-based (*transition diagram*) and Table-based transition-listing schemes. Graph-based scheme is considered efficient for visually represent player's behavior, while Table-based transition-listing scheme is efficient for computer simulation (Sipser, 1997). However, the classifications pre-

Figure 2. Input and empty transitions from A to B



sented in (Ghneamat, 2005) showed that automata can be weighted, and therefore these types of automata are referred to by weighted automata (*Transducers*). Transducers are automata which generate new output based on a given input and/or a state using action. Transducers come in two types as follows:

- **Moore Machines:** The output depends only on a state (e.g. entry state, random state, etc.). Moore machine is efficient in simplifying a given behavior.
- **Meally Machines:** The output depends on specific input and a state. Meally machine is efficient in reducing the number of states.

Mathematically (Guerberoff, Queiroz, & Sichman, 2010), finite automata are 4-tuple structures described by R as follows:

$$\mathbf{R}=(S, I, F, S_0) \quad (1)$$

where:

- S is a set of states
- I is an input symbol
- F: $S \times I \mapsto S$ is a transition function
- $S_0 \in S$ is an initial state

This definition induces the concept of current state for a given input sequence. That is, given a sequence of inputs $i^* = (i_1, i_2, \ldots, i_t) \in I^t$, the current state S_t is defined recursively for $t > 0$ as:

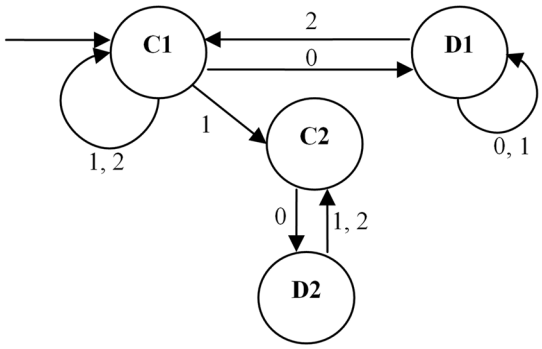
$S_t = t(S_{t-1}, i_t)$ together with the initial state S_0.

Finite automata are widely used in game theory. For instance, in the iterated *n*-player prisoners' dilemma (INPPD), finite automata can be used to represent players' strategies. The automata states represent the players' action and the input is the strategy profile observed from the other players (e.g. the number of cooperated agents). Therefore, if *n* represent the number of players in the game, then the set I is defined as $I = \{0, 1, \ldots, n\text{-}1\}$. The set of states S can be divided two states C and D, such that the player cooperate or defect if $s_t \in C$ or $s_t \in D$, respectively. Figure 3 shows an example of a strategy for the NPPD with 3 participants.

The automaton presented in Figure 3 has four states: C1, C2, D1 and D2. The arcs between the states represent the transitions. The four states can only be triggered by the numbers associated with the arcs, resulting in a transition between the states. In this example, F(C1, 0) = D1 and F(D2, 2) = C2, and so on. The initial state is the one with an empty arrow in its direction (S_0 = C1). The following sequence of plays illustrates the role which finite automata can play in representing strategies:

- First, the player selects "C" as its initial state.
- If in this round none of the other two players played "C", the total number of cooperator will be 0 (excluding the player itself).

Figure 3. Example on representing INPPD strategy



- The transition "0" will change the current state from C1 to D1.
- In the following round, the agent will play "D", since the current state is D1.
- Assume that one player is cooperated; therefore the total number of cooperator will be 1.
- The transition "1" change the state from D1 to D1 itself.
- In the next round, the player will play "D" again.

However, using finite automata as a modeling language has its limitations. Finite automata are only capable to recognize regular languages (Guerberoff, Queiroz, & Sichman, 2010). In other words, it is possible to guarantee arrival at a specific state only for a determined class of input sequences.

Pushdown Automata

The pushdown automata are finite automata equipped with stack. Several operations can be applied on the stack including read, push and pop operations. Mathematically, pushdown automata can be defined as a septuple **R** as follows:

$$\mathbf{R}= (S; \alpha; \Gamma; \delta; I_0; Z_0; F) \quad (2)$$

- **State Tuples:** This category includes tuples S, I_0 and F, such that:
  - **S** is a finite set of states.
  - $\mathbf{I_0}$ is the initial state of the pushdown automata.
  - **F** is the final (acceptance) state.
- **Input Tuples:** This category includes tuples α and Z_0, such that:
  - **α** is a finite set of input symbols.
  - $\mathbf{Z_0}$ is the initial input symbol to pushdown automata.
- **Stack Tuples:** This category includes tuple Γ, such that:
  - **Γ** is finite stack alphabets that are allowed to be pushed onto the stack.
- **Transition Tuples:** This category includes tuples δ, such that:
  - **δ** is the transition function, where **δ** is designed to control the behavior of the automaton.
  - The function **δ** takes three arguments (*m*, *n*, *r*), where *m* is a member in **S**, *n* is a member in **α** and *r* is a member in **Γ.**

The output of this function is a finite set of pairs (*p*, *q*), where *p* is the new generated state and *q* is the string of the stack that replaces *r* from the top of the stack.

Turing Machine

A Turing machine is an abstract computing device which consists of a finite control, unbounded memory (represented by tapes), and a finite program. The program is a list of instructions that guide the machine to take actions based on the current state of its finite control and the bits in its current 'window' on the information on the tapes. A step normally involves both a change of state in the finite control, a rewrite of the bits in the window, and a move of the window to the left or to the right by one position (van Leeuwen & Wiedermann, 2001).

Turing Machines were first proposed by Alan Turing (Sipser M., 1997). The idea was to model what humans do in order to solve a problem when following instructions through symbolic means. Alan tried to extract the basic entities of the process together with the basic operations that allow a computer to carry out the process of following a set of instructions. Turing machine assumes that at the entire input data is available on the input tape at the beginning of the computation. The rest of the input tape is blank. If the machine approaches the accepting state, the input is said to be accepted. The result of the computation is

given by the contents of the tapes at this time. Once a new set of input data is started, all previous information will be erased. The framework of Turing machines is suitable for studying the power and efficiency of algorithms, considering the achievements of computability and complexity theory(van Leeuwen & Wiedermann, 2001).

Technically, Turing machines are extended versions of pushdown automata. Similar to the transition function (**δ**) in pushdown automata, Turing machine has a central component which can be in one of a finite number of states and an infinite tape used for storage. However, several characteristics are responsible for distinguishing Turing machines from pushdown automata, including (Sipser M., 1997):

- The tape used by a Turing machine is infinite in both directions.
- Turing machine receive its input written on the tape which they use for storage.
- Turing machine control the head position to where reading and writing on the tape is performed.

Mathematically (Thakkar, 2004), pushdown automata can be defined as a septuple **R** as follows:

$$\mathbf{R}= (S; \alpha; \Gamma; \delta; I_0; B; F) \quad (3)$$

where the tuples of R are classified into the following categories:

- **State Tuples:** This category includes tuples S, I_0 and F, such that:
  - **S** is a finite set of states.
  - $\mathbf{I_0}$ is the initial state of the Turing machine.
  - **F** is the final (acceptance) state.
- **Input Tuples:** This category includes tuples α and B, such that:
  - **α** is a finite set of input symbols.
  - **B** is the blank symbol. **B** is a member in **Γ** but not in **α**. The blank appears initially in all but the finite number of initial cells that hold input symbols.
- **Tape Tuples:** This category includes tuple Γ, such that:
  - **Γ** is a complete set of tap symbols.
- **Transition Tuples:** This category includes tuples δ, such that:
  - **δ** is the transition function, where **δ** is the central processing part of Turing machine.
  - The function **δ** takes three arguments (*t*, *u, d*), where *t* is the next state, *u* is a symbol in **Γ** that is written in the cell being scanned, replacing whatever symbol was there, and *d* is the direction (L: left and R:right), indicating the direction of the head tape.

The output of this function is a finite set of pairs (*p*, *q*), where *p* is the new generated state and *q* is the string of the stack that replaces *r* from the top of the stack.

Adaptive Automata

Many studies have been conducted on the field of adaptive and learning systems. The attention on such studies comes from the suitability of those systems in modeling many real world complex systems. In typical Learning Automata (LA) systems, a self-operating machine (Automaton) responds to a sequence of actions to achieve a specific goal. The Automaton, in turn, may responds based on pre-determined rules, or it may adapt to its environmental dynamics. In other words, adaptive actions are attached to the state-transition rules of the adaptive automata and they are activated whenever the transition is applied by removing or inserting new elements to the automaton's transition set.

In Psychology, the term learning refers to the act of acquiring knowledge and modifying one's behavior based on the experience gained. Therefore, in LA, the adaptive automaton aims to learn the best possible actions from a set of possible actions that are offered to it by the environment in which it operates. The Automaton, thus, acts as a decision maker to arrive at the best action

Adaptive automata are internally based on the structured pushdown automata. Structured pushdown automata are equivalent to classical pushdown automata (Neto & Bravo, 2003). A structured pushdown automaton consists of a set of states, a finite non-empty alphabet, initial state, set of final states, pushdown alphabet and a transition function. The transition function is composed of two transitions levels: internal and external. The internal transitions are similar to those in finite-state automata. On the other hand, the external transitions are responsible for the calling and returning scheme. In each transition of an adaptive automaton, the current state and the current input symbol of the automaton determine a set of possible transitions to be applied.

Mathematically (Pistori, Martins, & Jr, 2005), adaptive automata can be defined as a 10-tuple **R** as follows:

$$\mathbf{R}= (S; \alpha; I_0; F; \delta; U; \Gamma; H; Q; \Delta) \quad (4)$$

where the tuples of R are classified into the following categories:

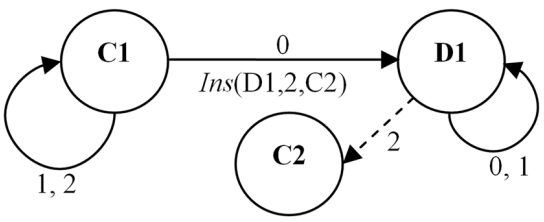
- **State Tuples:** This category includes tuples S, I_0 and F, such that:
  - S is a set of states.
  - I_0 is the initial state of the automaton.
  - F is the final (acceptance) state.
- **Input Tuples:** This category includes tuples α and Γ, such that:
  - α is a set of input symbols.
  - Γ is a set of parameters and variables.
- **Transition Tuples:** This category includes tuples δ, U, H, Q and Δ, such that:
  - δ is the transition relation, where this relation takes two elements: an element from U and a set of mapped parameters from Q.
  - U is a set of adaptive function labels.
  - H is a set of generators.
  - Q is used for mapping parameters, variables and generators to **U**.
  - Δ is a set of adaptive actions {+,-,?}.

Each adaptive action consists of the *type* and the *transition* of the adaptive action. The action type can be either a query, remove or insert actions, represented by (?), (-) and (+) respectively. Adaptive actions are formulated as calls to adaptive functions with a set of parameters. These actions describe the modifications which should be applied to the adaptive automaton whenever they are called. Technically, simple finite automaton can be turned into an adaptive automaton by allowing its rules to change dynamically (adaptive mechanism). Figure 4 illustrates the concept of adaptive automata, where the discrete arrow between D1 and C2 is created when the transaction between C1 and C2 is accomplished.

Cellular Automata

The structure of Cellular Automata (CA) is based on lattice of cells where each cell may be in a predetermined number of discrete states. Updating the states of CA in each time step requires a pre-determination process of the neighborhood relation, indicating the communication structure between the cells. Updating the state of each

Figure 4. Adaptive automata



cell is carried out using a transition rule. Each transition rule takes the states of all the cells in its neighborhood as input to the automaton, and updates all cells in CA synchronously.

For simplicity, the following are the basic properties and characteristics of a given CA (Hegselmann & Flache, 1998):

- The cells are regularly arranged in *n*-dimensional grid.
- Each cell has an initial state selected from a finite set of states.
- Update is carried out in a discrete time step.
- Changing the state of the cells depends on the set of local rules.
- The selected transition rule is applied to all cells.
- The update process is carried out simultaneously for all cells.

Mathematically (Kari, 2011), a cellular automaton is defined as a 4-tuple **R** as follows:

$$\mathbf{R}= (n; S; P; F) \quad (5)$$

where

- n is the dimension of the lattice space.
- S is a finite set of states.
- P is a finite set of neighborhoods.
- F is the local rule of CA.

However, cellular automata are considered easy to implement and emulate on a computer. In addition, CA exhibits a wide range of possible nonlinear behavior and thus it is capable of producing output at the highest complexity level of formal language theory (Albin S., 1998).

Stochastic Automata

Stochastic learning automaton can be defined as an element which interacts with a random environment to improve a specified performance (Narendra, 1974). This is possible by changing the automaton action probabilities based on the responses received from the corresponding environment. The stochastic automaton works as follows:

- The automaton starts with no information about the optimal action.
- The probabilities are equally attached to all actions.
- One action is selected at random.
- Based on the response of the environment, the action probabilities are changed.
- New action is selected according to the updated action probabilities, and the procedure is repeated.

Mathematically (Chandrasekaran & Shen, 1972), a cellular automaton is defined as a 6-tuple **R** as follows:

$$\mathbf{R}= (S; F; \Phi; G; \pi; \Gamma) \quad (6)$$

where

- S is a set of inputs to the automaton.
- F is a set of outputs from the automaton.
- Φ is a finite set of states.
- G is the output function which maps the state to a specific output (one-to-one).
- π is the state probability.
- Γ is the reinforcement operator that specify the manner in which $\boldsymbol{\pi}$ is to change according to the environment.

Another form of stochastic automata is the Multiplicity automata (Oncina, 2008). Multiplicity automata are stochastic automata with only one initial state and no restrictions to force the normalization (distribute probabilities). In fact, automata with output are called automata with multiplicities (Ghneamat, 2005).

We can conclude that automata theory provides us with a simple presentation for a given problem. Automata theory is found to depend on simple rules

instead of random behavior. However, automata theory shows that the simplicity of the structure of automata can obviously allow them to handle complex problems. Therefore, automata theory has been widely applied to different games in the field of game theory.

THE APPLICATIONS OF AUTOMATA IN GAME THEORY

Automata theory has presented various types of automata with different characteristics. In this section we will go through the applications which have successfully utilized the concept of automata in game theory.

Applications of Finite Automata

Back to 1985 (Neyman, 1985), Neyman tries to model the fact that players can do only bounded computation, by modeling the iterated prisoner's dilemma players as finite automata. One year later, Rubinstein (Rubinstein, 1986) tries to show that costly computation may affects an agent's utility. Rubinstein assumed that players choose a finite automaton to play the game instead of choosing a strategy directly. It was clearly stated that a player's utility depends on the automaton's moves and the complexity of the automaton structure (number of states of the automaton). In other words, automata that use more states are capable to represent more complicated procedures (Halpern, 2008). In 2000, Dodis et al. (Dodis, 2000) were the first who formalized the idea of finite automata in prisoner's dilemma instead of modeling players as polynomially bounded Turing machine.

Recently, Finite automata have been widely used to represent player strategies in game theory. For instance, the researchers of the work mentioned in (Bó & Sichman, 2004) have conducted several experiments and the results showed that finite automata have significantly improved the performance of n-player Prisoner's Dilemma. The study presented in (Maenner, 2008) showed that infinitely repeated games, such as prisoner's dilemma and Matching Pennies, have problems in learning and presenting the strategies. Therefore, the study introduced dynamic systems where agents start the game with randomly chosen strategies which are represented by finite state automata. The agents are also allowed to change their strategies during the game.

In (Lindgren & Nordahl, 1994) the authors have presented a model which utilizes genetic algorithms and finite automata for presenting the strategies of Prisoner's Dilemma with changeable memory size. The work presented in (Bouhmala & Granmo, 2010) showed the benefits of finite learning automata which help agents to find the action that minimizes the expected number of penalties received, or maximizes the expected number of payoffs received.

On the other hand, there is a difference between the environments for representation the players in lattices, trees, or graphs using EAs techniques, based on many factors that in turn affect back the outcome of the evolution of strategies. Some of these factors include the number of players, technique of evolutionary computation, type of automata and neighborhood structures. In (Dworman, Kimbrough, & Laing, 1995) the authors have reported that they build their 3-players model using a genetic programming technique to discover high-quality negotiation policies, where the behavior of the players was presented by finite automata.

However, finite automata are found efficient in representing simple strategies in game theory. Nevertheless, game theory is interested in representing simple strategies since many real life applications do not require complex strategies to achieve their goals.

Applications of Adaptive Automata

Adaptive automata-based devices have been presented as powerful tools for defining complex languages. Adaptive automata have the following important features (Neto & Iwai, 1998):

- The accepting procedure performed by an adaptive automaton is done efficiently, due to their structure which is based on finite-state or structured pushdown automata.
- Adaptive automata can handle context dependencies in a strict syntactical way, without the aid of auxiliary semantic procedures.
- The structure of adaptive automata can extended to implement transducers, allowing the generation of parse trees as a side effect of parsing sentences.
- Generally, adaptive automata have cheaper implementations compared to other mechanisms.
- Adaptive automata can be considered as alternative to existing solutions of natural language processing.

In order for adaptive automata to do self-modification, adaptive acts adhered to their state-transition rules are activated whenever the transition is used. Adaptive mechanism can be defined as Adaptive actions which change the behavior of adaptive automata by modifying the set of rules defining it. The simple notation for representing adaptive automata should have some features such as, being, at least, compact, simple, expressive, unambiguous, readable, and easy to learn, understand and maintain (Neto & Iwai, 1998).

The work presented in (Ghnemat, Oqeili, & Bertelle, 2006) has paid more attention to the iterated prisoner dilemma. An original evaluative probabilistic automaton was created for strategy modeling. It has been shown that genetic automata are well-adapted to model adaptive strategies. As a result, we noticed that modeling the player behavior needs some adaptive attributes. However, the computable models related to genetic automata are good tools to model such adaptive strategy. In (Ghneamat, 2005) the authors used genetic algorithms to generate adaptive behaviors to be applied for modeling an adaptive strategy for the prisoner dilemma. They used adaptive automata-based model for the modeling agent behavior.

The work presented in (Zhang J., 2009) has formed the collection of automata in a tree-like structure, and the modification of action possibility continued at different levels according to the reward signs provided for all hierarchical levels. Another work presented in (Bertelle, Flouret, Jay, Olivier, Ponty, & du Havre, 2002) has focused on the models which can be used for simulating Prisoner's Dilemma. The work showed how existing models and algorithms, in game theory, can be used with automata for representing the behaviors of players. The dynamical and adaptive properties can be described in term of specific operators based on genetic algorithms. In addition, the work showed that genetic operators on probabilistic automata enable the adaptive behavior to be modeled for prisoner dilemma strategies.

Adaptive automata have computational power equivalent to a Turing Machine (Guerberoff, Queiroz, & Sichman, 2010). Thus, strategies presented by adaptive automata may show more complex behaviors than the ones described by finite automata. For instance, learning mechanisms can be constructed using adaptive automata to represent adaptive learning mechanism based on specific input parameters.

However, finite automata are a particular case of adaptive automata. If the automata have no rules associating adaptive functions to transitions, the model can be reduced to finite automata. This characteristic is considered important to use adaptive automata naturally where finite automata are required.

Applications of Cellular Automata

Cellular automata can be complex systems by themselves, or they can offer good ways of studying the behavior of complex systems (Schut, 2010). In this section we overview the existing models and

algorithms which have successfully utilized cellular automata for analyzing or achieving desired dynamics of a complex system.

The presented work in (Szilagyi, 2008) has modeled the participated agents, in a given game, as stochastic learning cellular automata. Stochastic learning means that behavior is not determined it is shaped by its consequences. In the same context, other works have modeled the agents differently, such as modeling agents as a combination of cellular automata (Wolfram, 1994; Hegselmann, Flache, & Möller, 1998) and as stochastic learning automata (Narendra & Thathachar, 1989; Flache & Macy, 1996).

Back to 1975, Albin was the first person who models the checkerboard models as cellular automata (Albin, 1975). In his book The Analysis of Complex Socioeconomic Systems, Albin has showed the cellular automata can provide enormous potentials for understanding social dynamics. In 1990 (Nowak & Latané, 1990), Nowak et al. have developed a two-dimensional cellular automata-based model of evolution for social attitudes. In the field of economics, a one-dimensional cellular automata model has been proposed to model and analyze pricing in a spatial setting (Keenan & O'Brien, 1993).

Recently, cellular automata have been used to simulate agents with different strategies (Chiong & Kirley, 2009). In that work, the researchers showed the ability of co-evolutionary learning used to evolve cooperative strategies in structured populations using the N-Player Iterated Prisoner's Dilemma (NIPD). In addition, they examined the effects of both fixed and random neighborhood structures on the evolution of cooperative behavior in a lattice-based NIPD model. Ghnemat et al. have presented a model to simulate and solve complex systems (Ghnemat, Bertelle, & Duchamp, 2009). The model is based on cellular automata with some approaches for agents programming, such as artificial intelligence, automata modeling, distributed computing, swarm intelligence, and genetic algorithms.

Simulating the n-player iterated prisoner's dilemma as a bidding game using cellular automata is one way of incorporating cellular automata in game theory (Ishibuchi, Namikawa, & Ohara, 2006). In that work, the authors intended to show why cooperation only emerges in the triangular neighborhood structure but not in the rectangular and the random pairing neighborhood structures. The authors in (Zhang, Zhang, Li, Gao, & Li, 2009) and (Szilagyi, 2003) have investigated how the evolution of cooperation is influenced in Prisoner's Dilemma model using cellular automata. In (Su, Hui, Zhang, & Li, 2009) the researchers examined how the spatial structure of environment loss affects the eco-epidemic. They simulated the heterogeneous lattice landscape by defining two environment states, i.e. suitable and unsuitable.

We can conclude that representing the behavior of n-players prisoner's dilemma models using cellular automata has a great benefit, since most of the researchers can utilize it to distinguish the effects of neighborhood structure on the evolution of cooperative behavior.

CONCLUSION

Presenting players strategies in game theory is one of the factors which affect the players' performance significantly. In this work we have analyzed the characteristics and the underlying structure of existed types of automata. Consequently, we have explored the applications of automata in game theory.

However, this study revealed that only few types of automata have been utilized in game theory. We found that finite automata have been widely used to represent simple agents' behavior. On the other hand, adaptive automata and cellular automata are mainly used to represent complex agents' behavior. The complexity level of both adaptive automata and cellular automata is determined by the number of states in the automata. The more states we have, the higher is the capability for those automata to handle complex strategies.

ACKNOWLEDGMENT

The authors would like to express their thanks to Universiti Sains Malaysia for supporting this study under the USM Fellowship.

REFERENCES

Albin, P. (1975). *The analysis of complex socio-economic systems*. London: Lexington Books.

Albin, S. (1998). *Barriers and bounds to rationality: Essays on economic complexity and dynamics in interactive systems*. Princeton, NJ: Princeton University Press.

Almanasra, S. (2007). Learning opponent behavior in game theory using genetic algorithms. Jordan: Al-Balqa' Applied University.

Almanasra, S., & Rafie, M. (2010). Comprehensive survey on automata-based game theory. In *Proceedings of the 1st International Symposium on Computing in Science & Engineering* (pp. 643-647). IEEE.

Bertelle, C., Flouret, M., Jay, V., Olivier, D., Ponty, J., & du Havre, L. (2002). *Adaptive behaviour for prisoner dilemma strategies based on automata with multiplicities*. Dresden, Germany: Academic Press.

Bó, I. G., & Sichman, J. S. (2004). *Strategy representation complexity in an evolutionary n-players prisoner's dilemma model.* Retrieved March 20, 2012, from www.econ.fea.usp.br/complex/Artigos/24_04_08_Ináciol.pdf

Bouhmala, N., & Granmo, O. (2010). Combining finite learning automata with GSAT for the satisfiability problem. *Engineering Applications of Artificial Intelligence*, *23*, 715–726. doi:10.1016/j.engappai.2010.01.009.

Chandrasekaran, B., & Shen, D. (1972). Stochastic automata games. *IEEE Transactions on Systems, Man, and Cybernetics*, 145–149.

Chiong, R., & Kirley, M. (2009). Co-evolutionary learning in the n-player iterated prisoner's dilemma with a structured environment. In *Proceedings of the 4th Australian Conference on Artificial Life: Borrowing from Biology* (pp. 32-42). IEEE.

Dodis, Y. S. (2000). A cryptographic solution to a game theoretic problem. In *Proceedings of CRYPTO 2000: 20th International Cryptology Conference* (pp. 112-130). Springer-Verlag.

Dworman, G., Kimbrough, S., & Laing, J. (1995). On automated discovery of models using genetic programming: Bargaining in a three-agent coalitions game. *Journal of Management Information Systems*, *12*(125).

Flache, A., & Macy, M. (1996). Weakness of strong ties: Collective action failure in a highly cohesive group. *Mathematical Sociology*, *21*, 3–28. doi:10.1080/0022250X.1996.9990172.

Ghneamat, R. (2005). *Genetic algorithms and application to adaptive automata for game theory*. Al-Balqa University.

Ghnemat, R., Bertelle, C., & Duchamp, G. (2009). A methodology for urban and land-use management simulation using spatial self-organization processes. *Dynamics of Continuous, Discrete and Impulsive Systems-series B*, *16*, 501–513.

Ghnemat, R., Oqeili, S., & Bertelle, C. (2006). Automata-based adaptive behavior for economic modelling using game theory. In *Emergent Properties in Natural and Artificial Dynamical Systems* (pp. 171–183). Academic Press. doi:10.1007/3-540-34824-7_9.

Guerberoff, I., Queiroz, D., & Sichman, J. S. (2010). On the effect of the expressiveness of strategy representation languages in multiagent based simulations: an experiment with an evolutionary model of the iterated n-players prisoner's dilemma. *1re Soumission à RIA Special Issue on Agent Based Social Simulation.*

Halpern, J. Y. (2008). *Algorithmic rationality: Game theory with costly computation*. Academic Press.

Hegselmann, R., & Flache, A. (1998). Understanding complex social dynamics: A plea for cellular automata based modelling. *Journal of Artificial Societies and Social Simulation*, *1*(3).

Hegselmann, R., Flache, A., & Möller, V. (1998). Solidarity and social impact in cellular worlds - Results and sensitivity analyses. In *Proceedings of Social Science Microsimulation: Tools for Modeling, Parameter Optimization and Sensitivity Analysis*. Berlin: Springer.

Ishibuchi, H., Namikawa, N., & Ohara, K. (2006). Effects of spatial structures on evolution of iterated prisoner's dilemma game strategies in single-dimensional and two-dimensional grids. In *Proceedings of CEC 2006 IEEE Congress on Evolutionary Computation* (pp. 976 - 983). IEEE.

Izquierdo, L. R. (2007). *Advancing learning and evolutionary game theory with an application to social dilemmas*. Academic Press.

Kari, J. (2011). *Cellular automata*. Lecture Note.

Keenan, D. C., & O'Brien, M. J. (1993). Competition, collusion, and chaos. *Journal of Economic Dynamics & Control*, *17*, 327–353. doi:10.1016/0165-1889(93)90001-9.

Lindgren, K., & Nordahl, M. (1994). Evolutionary dynamics of spatial games. *Physica D. Nonlinear Phenomena*, *75*, 292–309. doi:10.1016/0167-2789(94)90289-5.

Maenner, E. (2008). Adaptation and complexity in repeated games. *Games and Economic Behavior*, *63*(1), 166–187. doi:10.1016/j.geb.2007.07.008.

McMillan, J. (1992). *Games, strategies, and managers: How managers can use game theory to make better business decisions*. New York: Oxford University Press.

Narendra, K. (1974). Learning automata - A survey. *IEEE Transactions on Systems, Man, and Cypernetics, 4*.

Narendra, K., & Thathachar, M. (1989). *Learning automata (an introduction)*. Englewood Cliffs, NJ: Prentice–Hall.

Narendra, K. S., & Wright, A. (1977). Application of learning automata to telephone traffic routing and control. *IEEE Transactions on Systems, Man, and Cybernetics*, *7*(11), 785–792. doi:10.1109/TSMC.1977.4309623.

Neto, J., & Iwai, K. (1998). Adaptive automata for syntax learning. In Proceedings of Anais da XXIV Conferencia Latinoamericana de Informática - CLEI 98 (pp. 135–149). Quito, Equador: CLEI.

Neto, J. J., & Bravo, C. (2003). *Adaptive automata - A reduced complexity proposal*. Lecture Notes in Computer Science Berlin: Springer.

Neyman, A. (1985). Bounded complexity justifies cooperation in finitely repeated prisoner's dilemma. *Economics Letters*, *19*, 227–229. doi:10.1016/0165-1765(85)90026-6.

Nowak, A. S., & Latané, B. (1990). From private attitude to public opinion - Dynamic theory of social impact. *Psychological Review*, 362–376. doi:10.1037/0033-295X.97.3.362.

Oncina, J. (2008). Using multiplicity automata to identify transducer relations from membership and equivalence queries. In *Proceedings of 9th International Colloquium on Grammatical Inference,* (vol. 5278, pp. 154–162). Berlin: Springer.

Pistori, H., & Martins, P. S. (2005). Adaptive finite state automata and genetic algorithms: Merging individual adaptation and population evolution. In *Proceedings of the International Conference on Adaptive and Natural Computing Algorithms – ICANNGA*. Coimbra: ICANNGA.

Platkowski, T., & Siwak, M. (2008). Mean-field approximation for two-and three-person prisoner's dilemmas. *Physica A: Statistical Mechanics and its Applications,* 387.

Rubinstein, A. (1986). Finite automata play the repeated prisoner's dilemma. *Journal of Economic Theory*, *39*, 83–96. doi:10.1016/0022-0531(86)90021-9.

Schut, M. (2010). On model design for simulation of collective intelligence. *Information Sciences*, *180*, 132–155. doi:10.1016/j.ins.2009.08.006.

Sipser, M. (1997). *Introduction to the theory of computation*. Boston: Academic Press.

Su, M., Hui, C., Zhang, Y., & Li, Z. (2009). How does the spatial structure of habitat loss affect the ecoepidemic dynamics? *Ecological Modelling*, *220*, 51–59. doi:10.1016/j.ecolmodel.2008.09.009.

Szilagyi, M. (2003). An investigation of n-person prisoners' dilemmas. *Complex Systems, 14*(2).

Szilagyi, M. N. (2008). Agent-based simulation of n-person games with crossing payoff functions. *Complex Systems*, *17*(4), 427.

Thakkar, D. (2004). *Game theoretic models of computation*. New York: Academic Press.

Thathachar, M. A. (1985). Learning systems: Stochastic automata models. *Defence Science Journal*, *35*(3), 361–366.

van Leeuwen, J., & Wiedermann, J. (2001). *The turing machine paradigm in contemporary computing*. Mathematics Unlimited. doi:10.1007/978-3-642-56478-9_30.

Wolfram, S. (1994). *Cellular automata and complexity* (*Vol. 10*). Reading, MA: Addison–Wesley.

Zhang, H., Zhang, F., Li, Z., Gao, M., & Li, W. (2009). Evolutionary diversity and spatiotemporal dynamics of a spatial game. *Ecological Modelling*, *220*, 2353–2364. doi:10.1016/j.ecolmodel.2009.06.005.

Zhang, J. (2009). Adaptive learning via selectionism and Bayesianism: Part I: Connection between the two. *Neural Networks*, *22*(3), 220–228. doi:10.1016/j.neunet.2009.03.018 PMID:19386469.