Scientific Research and Essays Vol. 7(42), pp. 3609 -3620, 31 October, 2012
Available online at http://www.academicjournals.org/SRE
DOI: 10.5897/SRE11.1860
ISSN 1992-2248 ©2012 Academic Journals

Full Length Research Paper

Adaptive automata model for learning opponent behavior based on genetic algorithms

Sally Almanasra[1], Khaled Suwais[2]* and Muhammad Rafie Arshad[1]

[1]School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia.
[2]Information Technology and Computing Department, Arab Open University, Riyadh, Saudi Arabia.

Accepted 9 September, 2012

The purpose of this research is to study how genetic algorithms (GA's) are applied in the field of Game Theory. GA's are effective approaches for machine learning and optimization problems. In this work, genetic algorithm is utilized to determine the behavior of an opponent in Prisoners' Dilemma. The opponent behavior will be modeled by means of adaptive automaton. The basic problem of this study is the well-known Prisoner Dilemma. The primary purpose of this research is to determine the opponent behavior towards finding a better strategy to be followed by the player, since the best strategy to be followed depends on the opponent behavior. The results of our proposed model showed the capability of our model to identify the opponent model efficiently. Based on the provided knowledge about the opponent model, the dynamic strategy showed better results when compared to other well-known strategies.

Key words: Game Theory, Prisoner's Dilemma, genetic algorithms, adaptive automata.

INTRODUCTION

In recent years, Game Theory and Decision Theory were tightly related to the field of Artificial Intelligence through the use of intelligent agents. Game theory was adopted to represent and model multi-agent systems. Several researches were conducted for the purpose of incorporating learning in game theory. Some modern approaches of opponent modeling, in multi-agent systems, have modeled an agent by classifications (Riley and Veloso, 2000; Steffens, 2002). GA's have been used in game theory in several forms, such as searching for optimal strategy using automata with multiplicities (Genetic Automata) (Ghnemat et al., 2005) and characterizing a form of social learning. GA's have become a tool for representing and modeling particular types of economic problems. One reason for that is the ability of simulating the behavior of the individual's action efficiently. Several applications of GA in economic modeling have been introduced (Birchenhall, 1995; Miller, 1996; Axelrod, 1997; Curzon, 1997). They were applied in well-known game models such as cobweb-type and prisoner's dilemma models.

The expression 'Game Theory' was created by Von Neumann and Morgenstern in 1944. Since then, the applications of game theory are found in economics, politics, biology, computer science, psychology and sociology. Game theory provides a language to formulate structure, analyze, and understand strategic scenarios (Turocy, 2001). In Game theory, the actions of several agents are interdependent. These agents can be individuals, groups, firms, or any combination of these.

The applications of game theory have been utilized in economics; at which such applications have been used in studying competition for markets, advertising, and planning under uncertainty (Martin, 1978). In addition, the applications of game theory are extensively utilized in the field of computer science. They have been used in interface design, discourage understanding, network routing, load sharing, resource allocation, and service transactions on the internet. The best reason for studying games is that games can model many real-life situations

*Corresponding author. E-mail: khaled.suwais@arabou.edu.sa.

to achieve the individual goals and maximize its payoffs.

Learning algorithms in game theory

Learning is an active area of research in many disciplines and it includes several concepts such as self-organizing neural networks, learning automata, and reinforcement learning (Lucas, 2005). The main objective of learning algorithms is to have systems that are capable of adapting their behavior in real time within complex environment without explicating learning signals.

Learning techniques can be directly applied to game theory. Game characters can exploit some sorts of learning characteristics that make the game much more playable. This capability enables characters to invest in complex and long-term dynamics. Until now, there is no such a perfect game that fully exploits these characteristics (Lucas, 2005). However, Yannakakis (Georgios and Hallam, 2005) used what he called an objective measure of interestingness in order to evolve behaviors of Pac-Man and other similar games.

Intelligent agents have strong ties with the problem of opponent modeling and inherently inhibit learning models. Intelligent agents are required to interact with their environment in an intelligent way. Interaction between agents includes information gathering, conflict resolution, task allocation, resource sharing and cooperation. This interaction is similar to the payoff matrix in the context of Game Theory. However, finding a strategy for interaction is a hard problem and an adaptive strategy is required based on the agent’s interaction experience (Carmel and Markovitch, 2000).

In Carmel and Markovitch (2000), a model-based approach was presented to learn a strategy for interaction in which the adaptive agent holds a model of its opponent’s strategy. This model is used to determine which strategy that the agent should respond to a given opponent’s strategy. It requires an adaptation method to cope with the unknown strategies of the opponent. The model could infer the opponent’s model based on its input/output behavior.

The idea of using evolution computing for developing adaptive strategies rather than searching for a sub-optimal strategy was introduced and discussed in (Gallagher and Ryan, 2003). As pointed out in Outkin (1998), game theory can be viewed as the theory of strategic interaction, since it considers at the tactical value of a decision without taking the future into account. It also pre-assumes the equilibrium condition, such as Nash equilibrium, where time is not assumed to play a role. On the contrary, human politics, economics and business history provide us with examples which ascertain that dynamic strategies are dominant in our world.

The model presented in Outkin (1998) aims to expand the domain of game theory to investigate evolving and non-equilibrium strategies and including non-linearity and non-equilibrium properties. Subsequently, it models the process of decision making in this kind of environment. In this new configuration, temporary equilibrium, and chaotic behavior could arise; hence in the economic context, it needs local interactions and learning by agents. The adaptive strategy model is based on automata networks as a modeling tool for game theory. It studied cooperation and local interaction in the Prisoner's Dilemma Game. The deterministic and stochastic best response is played, when local interactions are investigated. For dynamic strategies, it is a world of disequilibrium, incomplete information and a rational decision maker who cannot rely on static or bounded strategies, rather develop an adaptive and dynamic strategy. One possible solution is that agent can search all possible scenarios based on historic information and then picks up the best one. However, this solution could be hard to implement since it is hard to predict the future evolution of the system. On the other hand, the search space will be exponential if it is required to optimize *n* periods ahead, where these periods cannot be solved analytically or even numerically.

However, from the agent point of view, it is better to follow heuristic strategy that is based on some acquired knowledge, hoping that the embedded experience is sufficient. Another important point is that the collective problem solving behavior could lead to learning in the evolutionary system, where each individual can hold a simple strategy and the global performance of the system as a whole is powerful. Therefore, it is important to define the interaction between players and their information sharing.

In Game Theory-based adaptive strategies, it is important to address that the player whom the agent plays with, was recognized in the domain of business more than the theoretical domain. One possible definition for an adaptive strategy, in the context of Game Theory, is a strategy that uses rational as well as heuristic rules and changes them according to the player's experiences in the game (Outkin, 1998). Opponent modeling in the context of games has been investigated in Carmel and Markovitch (1995), Smith (1982), Jansen (1992) and Findler (1977). Opponent modeling did not achieve much improvement in practice, as in chess, where it is not essential to achieve high performance, while it is essential to success in poker as illustrated in Billings et al. (1998).

Prisoner’s dilemma

Prisoner’s Dilemma is a game which models the story of two prisoners held suspect of a serious crime. If one of the prisoners testifies, or becomes a witness against the other, he will be rewarded, or goes free (a payoff of 5), while the other will serve a long prison sentence, (payoff 0). If both testify, their punishment will be relatively less,

Table 1. The philosophy of Prisoner's Dilemma.

Prisoner	Prisoner B stays silent	Prisoner B betrays
Prisoner **A** Stays Silent	Each serves six months	Prisoner **A** serves ten years Prisoner **B** goes free
Prisoner **A** Betrays	Prisoner **A** goes free Prisoner **B** serves ten years	Each serves five years

Table 2. The Payoff Matrix of Prisoner's Dilemma.

Matrix		Player II	
		Cooperate	Defect
Cooperate	Player I	3.3	0.5
Defect		5.0	1.1

five years each (payoff 1 for each). However, if they both *cooperate* with each other, they will only be imprisoned for a short term or just six months as shown in Table 1.

Prisoner's Dilemma is a strategic game between two players. Each player has two strategies, called "Cooperate" and "Defect," which are labeled C and D respectively. The game is played as follows: Player (I) chooses a row, either C or D and player II chooses either C or D simultaneously. Each choice has a payoff for the two players. For example, the strategy combination (C; C) has payoff 3 for each player, and the combination (D; D) gives each player payoff 1. The combination (C; D) results in payoff zero for player (I) and payoff 5 for player (II). When (D; C) is played, player I gets 5 and player II gets 0.

In matrix games, any two-player game in strategic form can be described by a table shown in Table 2. Generally, a player may have more than two strategies. Each strategy combination defines a payoff pair, like (D; C), which is given in the respective table entry. From the other perspective, Prisoner's Dilemma game is found to be symmetric, where the game stays the same when the players are exchanged. For this game, the payoffs matrix is said to be transparent, where both players can act simultaneously without knowing the other's action, which makes the symmetry possible. *Defect* is a strategy that dominates *Cooperate* in Prisoner's Dilemma. No rational player will choose a dominated strategy, since the two players will lose if they continue playing with it. What makes up the dilemma is that there is a risk for each prisoner if he does not defect and, at the same time, the other testifies. On the other hand, if the prisoner testifies and the other does not, he will have fewer payoffs. In such a repeated game, patterns of cooperation can be established as players' rational behavior outweighs their gain from the current defecting.

Prisoner's Dilemma has various applications where individual's defections at the expense of others lead to overall less desirable outcomes. Examples include arms races, litigation instead of settlement, environmental pollution or cut-price marketing. The game-theoretic justification of Prisoner's Dilemma on individual sides is sometimes taken as a case for treaties and laws, which imposes cooperation. However, there is some inefficiency on the outcome of the Prisoner's Dilemma game. For example, the game is fundamentally changed by playing it more than once and the payoffs matrix can be a subject to change.

Prisoner's Dilemma will be considered as a standard model in this study. Prisoner's Dilemma makes a useful framework for manifesting the significance of decision-making in ethical way by social environment (Turocy, 2001). However, the main goal of this research is to build up an adaptive automata-based model which is capable of determining the opponent behavior. Once the opponent behavior is determined, the model will find the best strategy to be chosen in order to achieve higher payoff compared to the opponent's strategy.

RELATED WORKS

Genetic algorithms (GA)

GA is efficiently used in solving complex functions and combinational optimization problems (Spears, 1993). GA can be described by fixed length strings which encode the problem within binary string (Abraham and Jainm, 2005). The other techniques in Evolutionary Algorithms are phenotypic algorithms which are used directly in the parameters of the system itself (AL-Salami, 2009). The commonly used method in GA's is the roulette wheel selection. The main operation of the GA is the recombination operator which acts as a natural recombination. However, GA is found easy to be used and can be operated in many kinds of optimization problems (Spears, 1993). From the other perspective, genetic algorithms are stochastic search and global minimization technique, for systems with complicated potential energy surface. GA can be viewed as a system of individuals that constitutes a population in which each individual represents a solution to the target problem. The population is allowed to evolve and emerge towards sub-optimal solutions. GA's have means to represent the population, perform mating (crossover) between

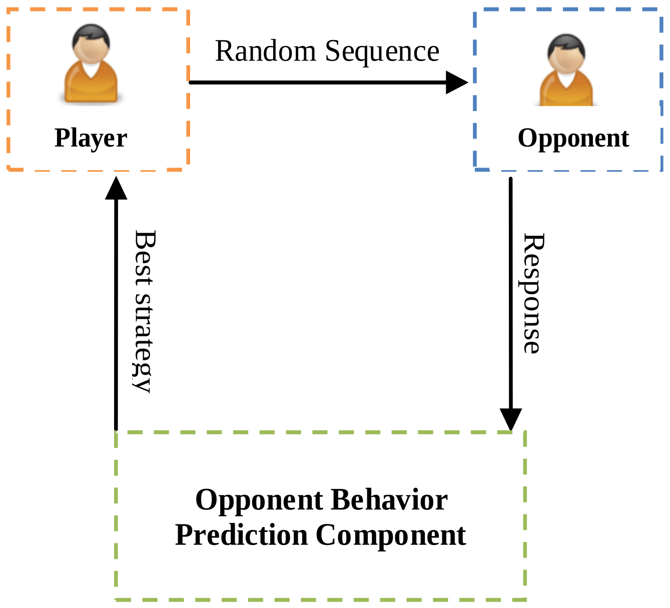

Figure 1. The overall structure of the proposed model.

individuals, mutate the population members and select the best members for a new generation (Abraham and Probert, 2006).

Applications of genetic algorithms in game theory

Here, we overview the applications and utilization of GA in game theory. Periaux et al. presented two works on applying GA-based approaches to solve different optimization problems. In their first work Périaux et al. (2001) they applied GA approach on DDM-nozzle optimization problems (Domain Decomposition Method-nozzle). The new GA approach is presented as a new evolutionary strategy for the multiple objective designs optimization of internal aerodynamic shape operating with transonic flow using non-cooperative game. Later, Periaux et al. (1997) presented another GA-based approach to solve bi-objective optimization problem by adopting the concept of Game Theory (Périaux et al., 1997). Ismail et al. (2007) used GA-based approach for finding the optimal strategy in Game Theory of two players. The approach is mainly focuses on utilizing the relationship between Game Theory and linear programming in implementing the fitness function. The utilization of GA's-based approaches was also presented by Chen et al. (2002) and Hamblin and Hurd (2007). The later used an alternative method of searching for optimal solutions in biological communication game, since it is particularly difficult to model the game using evolutionary stable strategy. On the other hand, Jen et al. combined two approaches of game theory and co-evolutionary computation to support a negotiation model to resolve agent's conflict. The main idea of Jen's approach is to use Nash equilibrium and Pareto concept to optimize the agent's resolution regardless the other agent's strategy.

More interesting GA's-approaches in Game Theory were presented by Macy (1996) and Phelps et al. (2005). Macy used evolutionary game theory to study the viability of cooperation in a predatory world. The work is differentiating the selection and learning processes whether it is hard-wired or soft-wired based on the organism that carries them. However, the differentiation was operationalized using artificial neural networks. Phelps has considered the use of evolutionary optimization with a principled game-theoretic analysis for automatic double-auction strategies acquirement. His approach relies on utilizing GA's for searching for a hitherto-unknown best response.

MATERIALS AND METHODS

The idea of the proposed model is to learn the opponent behavior, through opponents' actions, and then play the suitable strategy against it. The learning is based on keeping tracks of opponent's actions and then evolves GA's to estimate opponent's strategy. Exploiting the standard GA's to our problem is explained in details.

The proposed model is based on learning and adaptation, where the learning process is based on GA's. The proposed model is carried out as follows:

1 Generate initial sequence.
2 Receive response (feedback) from the opponent.
3 Evolve GA-based strategies in order to find the best matched strategy that is much closer to the opponent strategy.
4 The player decides the next action based on the output in step 3.

With reference to Figure 1, feedbacks will be received from the opponent and the player attempts to estimate the opponent strategy based on the received feedbacks. This estimation is based on learning through time from the opponent's behavior.

The model consists of two main phases. The first phase is responsible for modeling the behavior of the opponent, while the second phase is responsible for determining the best strategy to play. The initial sequence will be randomly initialized. This will guarantee that we span the states and transitions of finite automata which represent the opponent model as will be described later.

In Prisoners' Dilemma, the possible actions are either Cooperate (C) or Defect (¬C). The actions can be represented by the binary values 1 and 0, respectively. Therefore, the player's sequence and feedback from the opponent can be represented by a stream of zeros and ones. We assume that the opponent plays a deterministic strategy (stochastic strategies are not considered) and the opponent model will be represented in adaptive automaton.

The initial sequence is virtually applied to each individual's strategy. Consequently, the output of each strategy is compared with the feedback of the opponent, and hence the fitness is evaluated. The fitness evaluation is carried out by matching the player's strategies and the opponent's strategies, at which better strategies will be rewarded. The cycle of GA is then applied (selection, replication, crossover, mutation, fitness assignment) in order to evolve the strategies and find the best matched strategy. Our strategies are modeled in an adaptive automaton. Each state represents the state of the player based on previous input. Transition from one state to another is based on the feedback from the opponent and the taken action by the player itself.

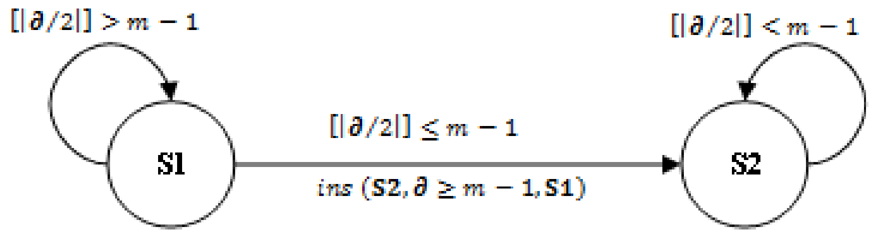

Figure 2. The proposed adaptive automata.

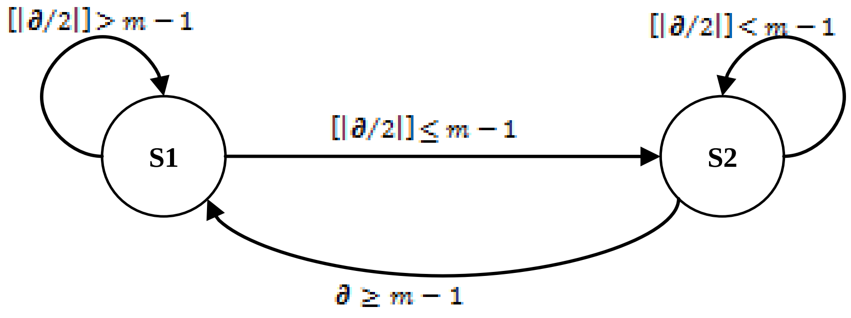

Figure 3. Adding new transition from S2 to S1.

Modeling strategies in finite automata

Automata refer to self-organized machines which combine current inputs with the history of previous inputs, in order to formulate the resulting behavior. There are several types of automata, including: deterministic finite automata, non-deterministic automata, pushdown automata, adaptive automata and so on.

A finite automaton is a dynamic system that changes its behavior at discrete periods. Finite automata are constructed by fixing a set of a finite memory. Regardless the length of the input string or whether the input string is simple or complex, the automata should be, efficiently, able to give the right decision. There are two classes of finite automata: deterministic finite automata (DFA) and non-deterministic finite automata (NFA). In contrast to NFA, each pair of state and input symbols in DFA will lead to one and only one transition to the next state (Geffert et al., 2007).

Consider a system which consists of a finite set of states, an output function and a transition function. The later determines the next state of the system based on the input and the current state of the system. The output function determines the output of the system based on the state. The system moves to a new state based on the new input value. This process can continue for a finite number of iterations. Modeling strategies in finite automata will enable our model to generate complex strategies. Figure 2 represents the design of the adaptive automata used in our learning model.

The symbol *m* refers to a dynamic memory that is used to store the last *m* actions taken by both players in the game. The size of *m* varies from 3 to 5 moves in any given round. The player's next action will be determined based on the feedback received from the adaptive automaton. The adaptive automaton in Figure 2 shows two states S1 and S2, where the two states refer to Cooperate and Defect, respectively. The adaptive automaton is designed to be biased towards cooperative behavior. The player will start the first few moves by cooperating (at least 4 moves). The input (∂) to the adaptive automaton is the total number of cooperation actions taken by both players in the last *m* moves. The subsequent actions will be computed as follows:

(i) If the total number of *cooperate* actions in the last *m* moves is $[|\partial/2|] > m-1$, then player remain cooperate (where [| |] is a rounding function which rounds the given input to its nearest integer number).
(ii) If the total number of *cooperate* actions in the last *m* moves is $[|\partial/2|] \leq m-1$, then the automaton will move from S1 to S2 and the player start defecting.
(iii) As the number of *cooperate* actions is less than the predefined value
$[|\partial/2|] < m-1$, the player remains in S2.
(iv) The adaptive function attached to the automaton is activated when the number of *cooperate* actions is increased based on the opponent's behavior, where $\partial \geq m-1$.
(v) At this point, new transaction will be inserted [using the function *ins* ()] from S2 to S1 to change the player's behavior from defect to cooperate as shown in Figure 3.

Since the model relies on a variable size memory ($3 \leq m \leq 5$), the model will be able to react against several types of strategies and change the player's behavior irregularly. Consider the tournament involved strategies that remembered three, four or five previous games, then there are 64, 256 and 1024 possibilities for the previous three, four and five games, respectively (Tables 3, 4 and 5). Recall that Cooperate action is encoded by 1, while Defect action is encoded by 0. The sequence CC refers to the player and opponent moves, respectively.

Both player's and opponent's actions will be treated as input to the adaptive automaton. The automaton will move from state *i* to state *j* based on the number of cooperation actions taken in

Table 3. Possible moves when m = 3.

Moves			Code	Case
CC	CC	CC	111111	(Case 1)
CC	CC	CD	111110	(Case 2)
CC	CC	DC	111101	(Case 3)
…	…	…	…	…
…	…	…	…	…
DD	DD	DC	000001	(Case 63)
DD	DD	DD	000000	(Case 64)

Table 4. Possible moves when m = 4.

Moves				Code	Case
CC	CC	CC	CC	11111111	(Case 1)
CC	CC	CC	CD	11111110	(Case 2)
CC	CC	CC	DC	11111101	(Case 3)
…	…	…	…	…	…
…	…	…	…	…	…
DD	DD	DD	DC	00000001	(Case 255)
DD	DD	DD	DD	00000000	(Case 256)

Table 5. Possible moves when m = 5.

Moves					Code	Case
CC	CC	CC	CC	CC	1111111111	(Case 1)
CC	CC	CC	CC	CD	1111111110	(Case 2)
CC	CC	CC	CC	DC	1111111101	(Case 3)
…	…	…	…	…	…	…
…	…	…	…	…	…	…
DD	DD	DD	DD	DC	0000000001	(Case 1023)
DD	DD	DD	DD	DD	0000000000	(Case 1024)

the last m games. Table 6 shows an example for some encoded strategies when the model remembers the last m games.

However, our adaptive automaton differs from other automata-based models from different perspectives. Our automaton is fully based on a set of dynamic parameters (∂, m). These parameters are designed to monitor the cooperation level between players. From the other perspective, the automaton is designed to encourage the players to show more cooperative behavior rather than simply model their strategies. This is possible as the transition rules are formulated such that the players keep cooperating until half of the number of cooperative moves drop to m-1.

Mapping the model to GA operations

The structure of chromosome

The chromosome represents the state transition table as shown in Table 7. The rows of the state transition table are lined up in the chromosome structure. Each line represents one state and its possible input (C, ¬C). Under each input, two entries must exist (the next state and the output). The Cooperate (C) is denoted by 1, and the Defect (¬C) is denoted by 0.

Fitness function

Strategies are evaluated by the designated fitness function **F**(x,y). The fitness function is operated by XOR'ing the generated strategy by the player (represented by a chromosome **C**p) and the opponent's strategy (**C**o). The resulted value will be stored in f. In prior to the evaluation process, the chromosome will be converted (using the converter T) into a binary string as follows:

$$\mathbf{B}p = \mathrm{T}(\mathbf{C}p) \quad (1)$$
$$\mathbf{B}o = \mathrm{T}(\mathbf{C}o) \quad (2)$$
$$f = \mathbf{F}(\mathbf{B}p, \mathbf{B}o) = x_i \wedge y_i \quad (3)$$

where **B**p and **B**o denote the binary equivalent of **C**p and **C**o, respectively. The aim of the XOR is to construct a difference between the compared chromosomes. The fitness function is a weighted evaluator, where the difference in most significant bit (MSB) will result in a big value in the fitness, which technically refers to low fitness. Obviously, completely matched chromosomes will have a fitness value of zero, since the outputs are identical.

For instance, the following two chromosomes represent the strategies of one player and one opponent:

Cp =

Output 0	Output 1	Output 2	Output 3	Output 4	Output 5	Output 6	Output 7	Output 8	Output 9
1	0	1	0	0	0	1	1	1	0

Co =

Output 0	Output 1	Output 2	Output 3	Output 4	Output 5	Output 6	Output 7	Output 8	Output 9
0	0	0	1	0	1	1	0	0	1

According to equations (1 and 2), both of **B**p and **B**o are initialized such that:

Bp = 1010001110
Bo = 0001011001

Table 6. Encoded strategies using random memory size *m.*

Size of (*m*)	Player's current state (S*i*)	History	Number of cooperation actions (∂)	Next move
3	D	CCDDDC	3	C
3	C	DDCDCD	2	D
5	D	CCDDCCDCCD	6	C
4	D	CDCCDDDD	3	C
3	C	CCDDCD	3	D
5	D	DDCCDCCDDD	4	C
4	C	CCDCDDCD	4	D

Table 7. The structure of the chromosome

Starting state	**State0**				**State1**			
	C		**¬C**		**C**		**¬C**	
	Output	**Next state**	**Output**	**Next state**	**Output**	**Next state**	**Output**	**Next state**
Example on representing the Tit-for-Tat strategy is illustrated as follows:								
State 0	**Output 0**	**State 1**	**Output 1**	**State 2**	**Output 2**	……..	**State *n***	**Output *m***
0	1	0	0	1	1		0	1

Based on Equation (3), the integer representation of the fitness value *f* of these two strategies is 727. This value is relatively high, which means that the player's strategy has low fitness.

Crossover operation (mating)

Crossover is a genetic operator that combines two chromosomes (parents) to produce a new chromosome (offspring). The new chromosome might be better than the parents if it takes the best characteristics from both of them. In this model, crossover occurs during the evolution and according to a crossover probability (uniform distribution for determining the crossover point). The offspring is formed by concatenating the bits string before point *t* (from parent *i*) and the bits string after the crossover point *t* (from the other parent *j*).

Mutation

Mutation is a genetic operator that changes one or more gene values in a chromosome from its initial state, which might results in generating a completely new gene added to the population. This operation enables the genetic algorithm to reach a better solution. The favor of mutation operation is that it prevents the population from stagnating at any local optima. In our model, the mutation occurs during the evolution and according to a certain mutation probability (specifically 0.01). This probability should usually be set fairly low. If it is set too high, the search will be considered a primitive random search.

Selection

Selection is a genetic operator that selects a chromosome from the current generation's population and includes it in the next generation's population. The selection is based on the value of the fitness. Each individual receives a reproduction probability based on the individual's own objective value and the objective value of all other individuals. Hence, it gives preference to better individuals, allowing them to pass to the next generation.

RESULTS

As a part of evaluating our model, we have initialized the model as shown in Table 8. These parameters will be tuned to measure the performance of the model from several perspectives.

The simulator is initialized with the following parameters: Number of Chromosomes = 30, Mutation Probability = 0.01, Population size = 200 and Maximum Random Sequence = 1000. Figure 4 shows that the model could achieve good results, that are closed to the *BestFit* value (*BestFit* refers to the fitness of the best chromosome or individual), after 55 generations. It can also be noticed that the average fitness *AvgFit* (the average fitness of the whole population) was improved through time, which means that learning is continuously improved over time.

The model was evaluated using different values of population sizes, specifically *n* = 10, 50, 100, 150, 200. Figure 5 shows the performance of the model for different generations' sizes. The *y*-axis represents the best individual fitness at each population size, while the *x*-axis represents the generation number.

Table 8. Parameters Initialization.

Parameter	Initialized value
Number of Chromosomes	30
Number of States	2
Size of Memory	$3 \leq Random \leq 5$
Mutation Probability	0.01
Maximum Length of Random Sequence	1000
Maximum Length of Generation	100
Number of Simulations	20
Opponent Strategy	Toggle
Cross Over Probability	0.0 <*Random*< 1

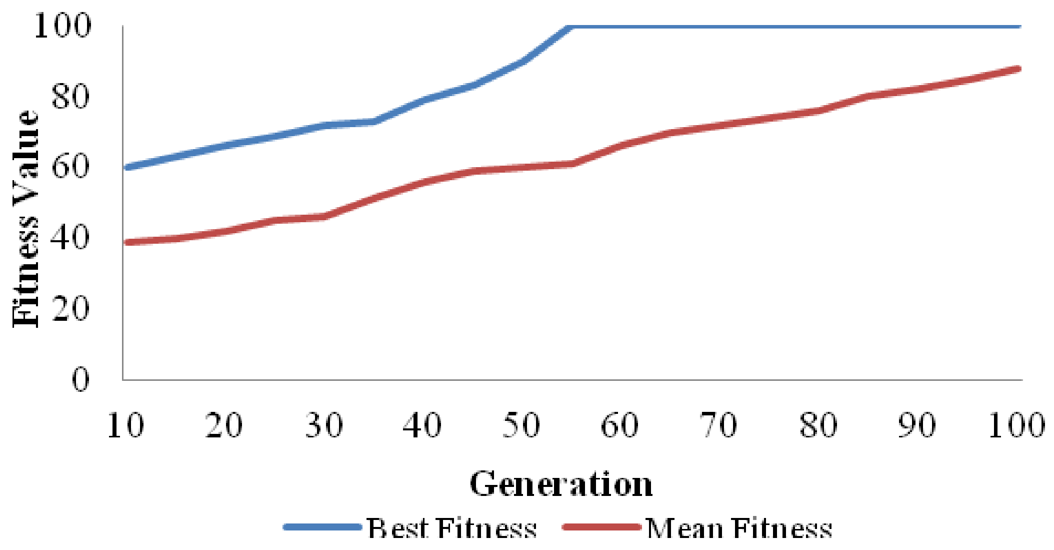

Figure 4. The performance of the model in 100 generations.

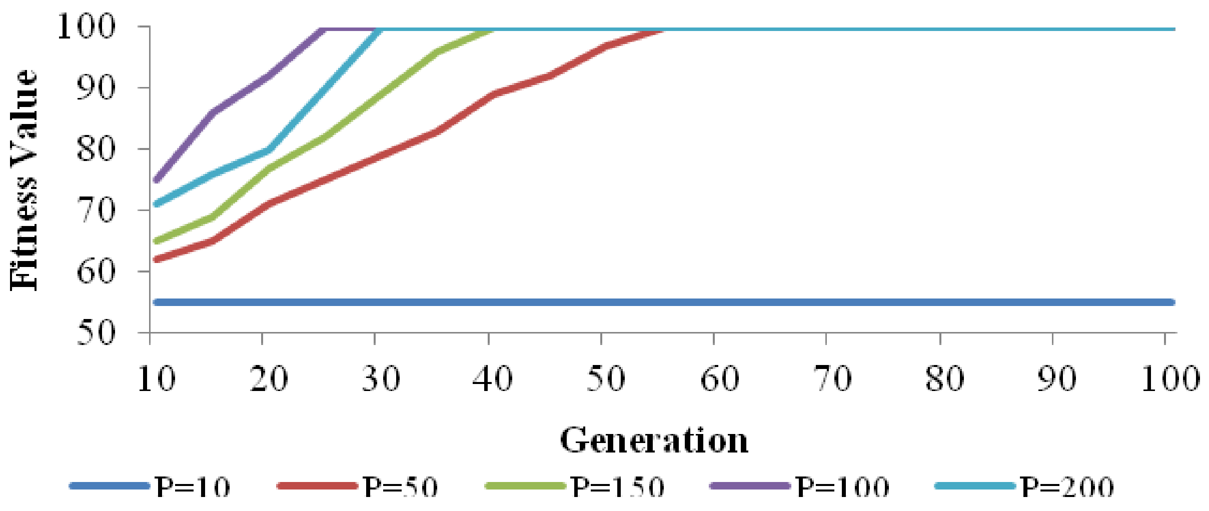

Figure 5. The effect of population size *P* on reaching the optimal solution.

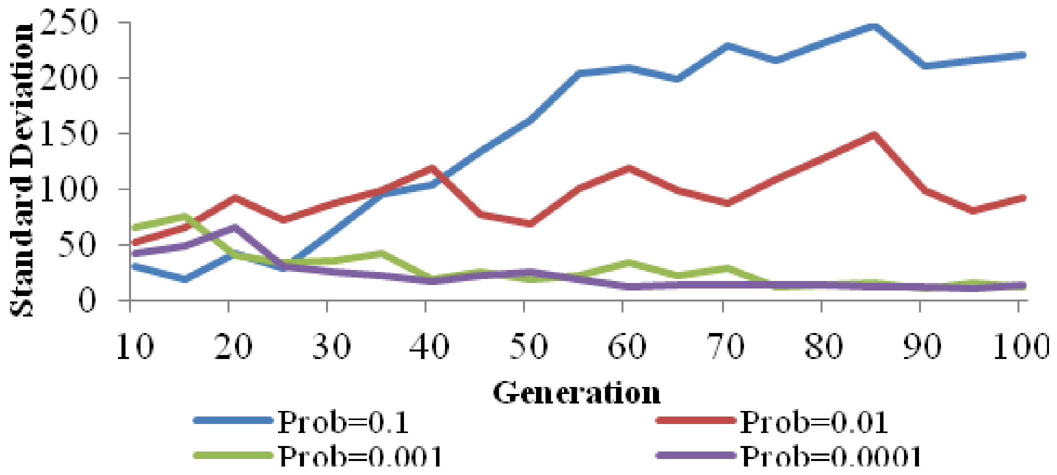

Figure 6. The effect of mutation probability (*Prob*) on the diversity of the individual's fitness.

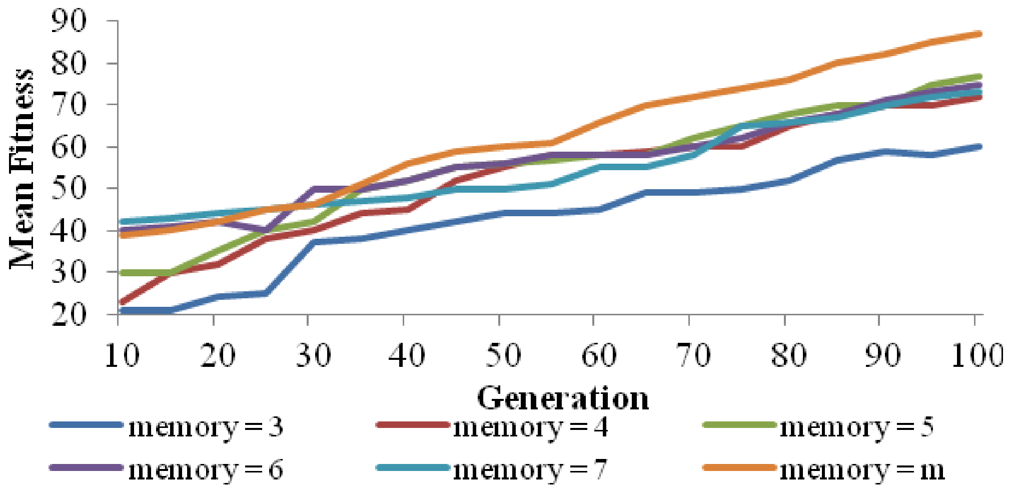

Figure 7. The effect of memory size on the performance of players.

From the other perspective, we have tested the effect of mutation probability on the performance of the proposed model. We found that the mutation probability have a direct affects on the diversity of the solutions. The results presented in Figure 6 show that as the mutation probability increases, the diversity of the solutions increases.

In term of memory, experiments shows that considering variable-memory size during the run of the model has outperformed the fixed-memory size technique. The results presented in Figure 7 shows that assigning a variable value to the memory size has significantly enhanced the mean fitness of the whole population. The results have also showed that the larger is the memory size, the higher is the mean fitness in early generations.

At the same time, large memory size has affected the learning process after performing approximately 30% of the total number of generations.

The proposed model initially generates a random sequence of *cooperate/defect* actions. The aim of this random sequence is to identify the model of the opponent by learning its behavior based on its feedback. The length of the random sequence should not be so long, since it is required to learn the opponent model as fast as possible. One hundred experiments were conducted and the length of the initial sequence on each experiment was determined. Based on these results, the best sequence length is found to be of the range 20 to 30 moves, as shown in Figure 8.

In term of strategies performance, Bruno (Beaufils et al., 1996) compared different strategies against themselves as shown in Table 9. Bruno’s experiments showed that each strategy can be defeated by other strategies. For instance, the best strategy to be played against an opponent who plays Spite strategy is the Tit-for-Tat strategy.

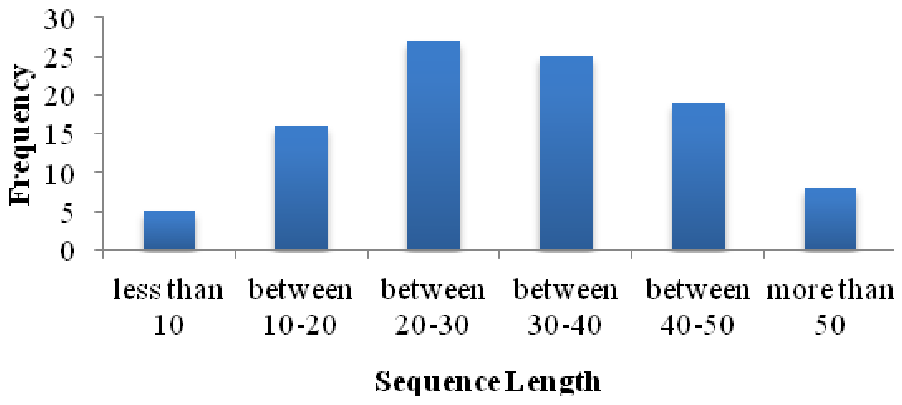

Figure 8. The frequency of sequence length.

Table 9. Bruno's model of different strategies.

Opponent strategy	Best strategy
Always Cooperate **(AC)**	Always Defect **(AD)**
Always Defect **(AD)**	Mis-Trust **(MT)**
Tit-for-Tat **(TFT)**	Always Cooperate **(AC)**
Spite **(SPT)**	Tit-for-Tat **(TFT)**
Per-Nasty **(PN)**	Spite **(SPT)**
Per-Kind **(PK)**	Always Defect **(AD)**
Soft-Majo **(SM)**	Per-Kind **(PK)**
Mis-Trust **(MT)**	Always Cooperate **(AC)**
Prober **(PRO)**	Mis-Trust **(MT)**
Pavlov **(PAV)**	Per-Nasty **(PN)**

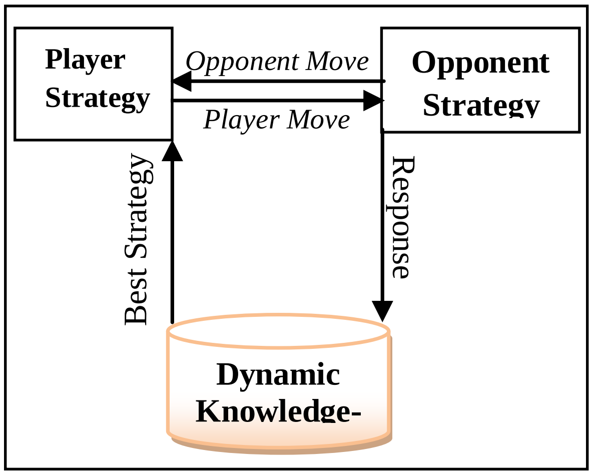

Figure 9. The proposed playing scheme.

Bruno's model inspired us to build a knowledge-base to help our model decide the best strategy to be played when the opponent model is determined. The overall proposed scheme is modeled as shown in Figure 9. The model has two components: the player-opponent component and the knowledge-base that guides the player to change its behavior based on the output of the opponent. The knowledge-base is designed to store all moves during the simulation. The player has an access to this knowledge-base in order to predict the opponents' strategy.

DISCUSSION

With respect to our findings illustrated by Figure 5, we conclude that as the population size increases, we can reach the optimal solution faster. At the same time, when the population size is under certain limit, for example P = 10, the optimal solution could never be obtained. However, we also found that when the mutation probability is set to 0.1 (which is relatively large), the diversity between individuals keep increasing. On the other hand, when the mutation probability is set to less than or equal to 0.001, the individual became quickly the same and sometimes the optimal value is not reached. Therefore, we found that the best setting of the mutation is at the probability 0.01.

In term of memory, experiments showed that the larger is the memory size, the higher is the mean fitness in early generations. At the same time, large memory size has affected the learning process after performing approximately 30% of the total number of generations.

In our second simulation, all the strategies in Table 9 were played one against each other and against our strategies generated by the proposed model. The scores of all strategies against each other are shown in Table 10. The results were obtained from 1000 games. The generated strategies from our model have achieved the highest payoff when played against well-known strategies. The results presented in Table 10 indicate that our model could generate the best fit-strategies based on the total payoff achieved against all other strategies (total of 30571). Figure 10 illustrates the average score

Table 10. Strategies' scores against each others.

Strategy	AD	MT	AC	TFT	SPT	PK	PN	SM	PRO	PAV	Total
AD	0	2997	3000	3000	3000	2001	999	3000	6	3000	21003
MT	1000	1000	5000	1004	1004	3668	2332	1004	1008	3000	20020
AC	999	2500	3000	3000	3000	2667	1998	3000	2999	3000	26163
TFT	999	1003	3000	3000	3000	3663	2331	3000	1007	3000	24003
SPT	667	1999	4334	2003	671	3335	1666	671	2006	3002	20354
PK	333	2664	3666	2667	343	2334	1665	3666	2664	2003	22005
PN	999	2500	3000	3000	3000	2001	2331	3000	2999	3000	25830
SM	1000	1000	3002	2500	1003	2669	1999	2500	3000	2003	20676
PRO	998	2995	4996	2999	1002	2669	1996	2999	1004	1998	23656
PAV	500	1998	3000	3000	3000	2833	1332	3000	2003	3000	23666
Our Model	992	2990	4982	2983	2989	3659	2329	3661	2991	2995	30571

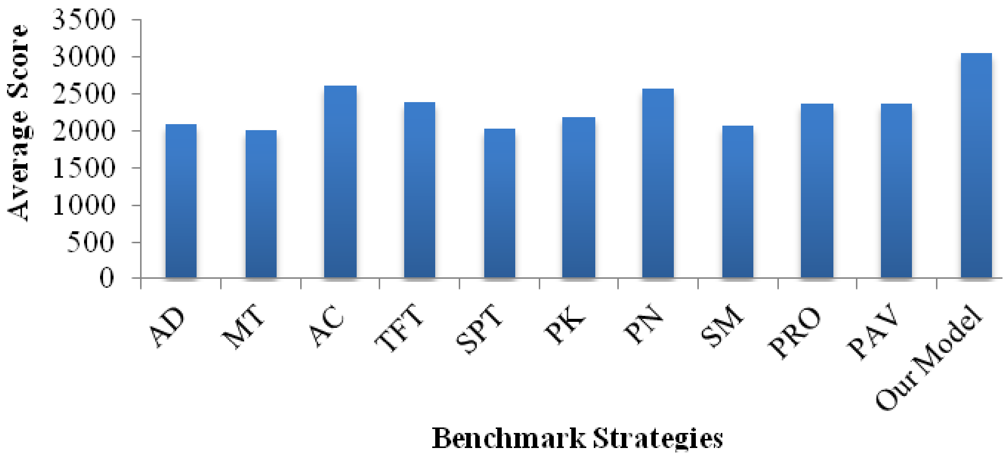

Figure 10. Average score achieved by competent strategies.

differences between benchmark strategies compared to our proposed model.

Conclusion

In this paper an approach for modeling the behavior of the opponent in game theory was introduced. This approach is based on modeling the opponent behavior as an automaton. A genetic algorithm approach was introduced in order to search for the exact model that the opponent plays with. The results showed the ability of the model to identify the opponent behavior efficiently. The effect of the variation of GA parameters was studied. The second step after identifying the opponent model was to use this information in order to decide which strategy to play. For this issue, a knowledge base was established to determine the best strategy to play against a certain opponent strategy.

A comparative study between our approach and well-known strategies was conducted. The results showed the superiority of our approach over the majority of these strategies. We can conclude that learning in general could be helpful in selecting strategies in game theory. In our case, we use GA in order to learn the opponent behavior. GA proved to be a powerful tool for opponent model learning in game theory. Additionally, the idea of playing a strategy based on estimating the opponent model, proved to be a good approach for playing in game theory, since it gets feedback from the other player and it can adapt its strategy based on this feedback.

This model is not restricted to prisoner dilemma game, but it can be applied to all matrix games. The parameters of our GA-based model have great effect on its performance. For example, if the population size is too small, the optimal solution could not be reached, and the larger the population size is, the faster the best solution is found. On the other hand, the mutation probability controls the diversity of individuals. If the diversity is too small, then the population can have pre-matured convergence. Conversely, if it is too large, then the randomness increases in the population and no learning could occur.

REFERENCES

Abraham A, Jain L (2005). Evolutionary Computation. Handbook of Measuring System Design, Oklahoma State University, USA.

Abraham N, Probert M (2006). A periodic genetic algorithm with real-space representation for crystal structure and polymorph prediction. Physical Review B. APS. 73:224104.

AL-Salami N (2009). Evolutionary Algorithm Definition. Am. J. Eng. Appl. Sci. 2:789-795.

Axelrod R (1997). The Complexity of Cooperation. Agent-Based Models of Competition and Collaboration Princeton, NJ: Princeton University Press.

Beaufils B, Delahaye J, Mathieu P (1996). Our meeting with gradual: A good strategy for the iterated prisoner's dilemma. Artificial life five, Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems, pp. 202-209.

Billings D, Papp D, Schaeffer J, Szafron S (1998). Opponent Modeling in Poker. Proceedings of the National Conference on Artificial Intelligence, pp. 493-499.

Birchenhall C (1995). Modular Technical Change and Genetic Algorithms, Comput. Econ. 8:233-253.

Carmel D, Markovitch S (1995). Incorporating Opponent Models into Adversary Search. AAAI. pp. 120-125.

Carmel D, Markovitch S (2000). Learning Models of Intelligent Agents. Computer Science Department, Technion, Haifa.

Chen J, Chao K, Godwin N, Reeves C, Smith P (2002). An automated negotiation mechanism based on co-evolution and game theory. Proceedings of the 2002 ACM Symposium on Applied Computing pp. 63-67.

Curzon T (1997). Using co-evolutionary programming to simulate strategic behavior in markets. J. Evol. Econ. 7: 219-254.

Findler N (1977). Studies in machine cognition using the game of poker. Comm. ACM 20(4):230-245.

Gallagher M, Ryan A (2003). Learning to play Pac-man: An evolutionary Rule–based Approach In Proceedings of the IEEE Congress on Evolutionary Computation pp. 2462-2469.

Geffert V, Mereghetti C, Pighizzini G (2007). Complementing two-way finite automata. Inf. Comput. 205:1173-1187.

Georgios N, Hallam J (2005). A generic approach for generating interesting Interactive Pac-Man. Proceedings of IEEE Symposium on Computational Intelligence and Games, pp. 94-101.

Ghnemat R, Khatatneh K, Oqeili S, Bertelle C, Duchamp G (2005). Automata-Based Adaptive Behavior for Economic Modeling Using Game Theory. arXiv.org, cs/0510089.

Hamblin S, Hurd P (2007). Genetic algorithms and non-ESS solutions to game theory models. Anim. Behav. 74:1005-1018.

Ismail I, ElRamly N, ElKafrawy M, Nasef M (2007). Game Theory Using Genetic Algorithms. In Proceedings of the World Congress on Engineering pp. 61-64.

Jansen P (1992). Using Knowledge about the Opponent in Game-Tree Search. Ph.D. dissertation, Carnegie-Mellon University.

Lucas S (2005). Evolving a neural network location evaluator to play ms. Pac-man. Proceedings of the IEEE Symposium on Computational Intelligence and Games, pp. 203-210.

Macy M (1996). Natural Selection and Social Learning in Prisoner's Dilemma Sociological Methods and Research, Sage Publications 25(1):103-137.

Martin B (1978). The selective usefulness of game theory. Social Stud. Sci. JSTOR 8:85-110.

Miller J (1996). The coevolution of automata in the repeated prisoner's dilemma. J. Econ. Behav. Org. 29:87-112.

Outkin A (1998). Adaptive Strategies in Game Theory. PSU University: citeseer.ist.psu.edu/outkin98adaptive.html.

Périaux J, Chen H, Mantel B, Sefrioui M, Sui H (2001). Combining game theory and genetic algorithms with application to DDM-nozzle optimization problems. Finite Elem. Anal. Des. 37:417-429

Périaux J, Sefrioui M, Mantel B (1997). GA multiple objective optimization strategies for electromagnetic backscattering. Genetic Algorithms and Evolution Strategy in Engineering and Computer Science: Recent Advances and Industrial Applications, Chapter 11, pp. 225-243. John Wiley and Sons, West Sussex, England.

Phelps S, Marcinkiewicz M, Parsons S, McBurney P (2005). Using population-based search and evolutionary game theory to acquire better response strategies for the double-auction market. Proceedings of IJCAI-05 on Trading Agent Design and Analysis.

Riley P, Veloso M (2000). On behavior classification in adversarial environments. Distributed Autonomous Robotic Systems 4: 371-380.

Smith J (1982). Evolution and the theory of games. Cambridge University Press.

Spears W (1993). An overview of evolutionary computation. In: Proceedings of the European Conference on Machine Learning pp. 442-459.

Steffens T (2002). Feature-based declarative opponent-modelling in multi-agent systems. Master's thesis, Institute of Cognitive Science Osnabr¨uck.

Turocy TL (2001). Bernhard von Stengel, Game Theory, Texas A&M University, CDAM Research Report LSE-CDAM-2001-09.