SOLVING THE ITERATED PRISONER'S DILEMMA USING LEARNING AUTOMATION

by

Yaojun Wu, B.A.

A thesis submitted to the

Faculty of Graduate Studies and Research Office

In partial fulfillment of the requirements for the degree of

Master of Information and System Science

Department of Systems and Computer Engineering

Faculty of Engineering

Carleton University

Ottawa, Ontario

Your file Votre référence
ISBN: 0-494-13475-5
Our file Notre référence
ISBN: 0-494-13475-5

DEDICATION

To my beloved parents Quanrong and Liqun, and my lovely wife Xiaoying.

ABSTRACT

The Prisoner's Dilemma (PD) has been discussed extensively to model the conflict between competition and cooperation, or between individual and collective rationality [1][8][9][10][11]. The machine learning community has an interest in the iterated PD (IPD) game, but has special interest in the behavior of the individual in the IPD game.

A family of estimate-based strategies, which are based on the pursuit scheme and interconnected Learning Automaton (LA) structure, is developed in this Thesis. These achieve a high performance in playing the IPD game in Nonstationary Environments. Simulation results show that, our proposed scheme, the IPPP (Interconnected Learning Automata with the Preceding Penalty Limit Pursuit), achieves 4.86% lower costs than the SLASH (Stochastic Learning Automata with States of History) strategy, which is the most efficient learning strategy reported in the literature. The advantages of IPPP are its quick detection of strategy switches, fast convergence, and its good generalization capability.

ACKNOWLEDGMENTS

I have to express my gratitude to my Professor, Dr. John Oommen, for being my guide during the term of this project, for editing this Thesis earnestly, and for his continuous support and encouragement. He is genuinely understanding, and very supportive of me. I learned a great deal from him about Learning Automata (among other fields) and about academic life. Without his guidance, advice, and help, this Thesis would not have been completed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

An automaton is a machine or simple mechanism that operates according to a series of states or configurations [12]. Learning can be defined as the process of arriving at any relatively permanent change in behaviour which may improve the performance over time on tasks done previously. Therefore, a learning system has the ability to improve its behaviour with time, according to a defined performance measure. The automaton can learn from its decision if the Environment provides some forms of feedback. The feedback would inform the automaton of whether the decision or selection it made was good or bad. Thus, an LA can be defined as a decision maker which operates in a random Environment, updating its strategy for choosing actions on the basis of the Environment's response [2].

Learning automata (LAs) are powerful in that they can improve their performance by a learning process which combines rapid and accurate convergence with low computational complexity. Recent applications of learning automata to real life problems include process control, pattern recognition, task scheduling, optimization problems, diagnosis, concept learning, and routing and bandwidth allocation in computer communication networks[12][17][18] [19].

The Prisoner's Dilemma (PD) has been discussed extensively to model the conflict between competition and cooperation, or between individual and collective rationality [1][8][9][10][11]. The dilemma was originally introduced by Merrill Flood and Melvin Dresher in 1950. In the same year, the concepts of the "Prisoner's Dilemma" and "Sentence Payoff" were invented by the mathematician Albert W. Tucker. Since then, the Prisoner's Dilemma has been studied in depth as a "non-zero sum" game in areas such as economics, political science, evolutionary biology, and game theory.

The machine learning community has an interest in the Iterated PD (IPD) game. The interest is in adaptive participants who adaptively learn to play these games. This is in contrast to game theory, where the participants are not assumed to be perfectly rational. It is also in contrast to the study of evolutionary game theory, where the focus is on populations but not on an individual participant whose behaviour would change as he/she learns more about the other participants' actions.

Although there is no single optimal strategy for an individual against every opponent's strategies, it is possible to have a strategy which can play sub-optimally against quite a number of strategies; especially one which can achieve relatively better results in Nonstationary Environments. We believe, therefore, that it is profitable to seek a more efficient and adaptive algorithm for playing the IPD game.

1.1 Thesis Objective

There are four objectives for this Thesis. The first objective is to develop efficient and adaptive strategies for playing the IPD game in Nonstationary Environments. The second objective is to implement these new strategies and to construct Nonstationary

Environments for testing these different strategies. The third objective is to compare the newly developed strategies with prior strategies applicable for Nonstationary Environments, especially with the state-of-the-art schemes, namely the SLASH/ISLASH strategies. The last objective is to compare our newly developed strategies with the existing ones, and to achieve a competition between them.

1.2 Contributions

We developed a family of efficient estimate strategies for achieving efficient and adaptive learning in playing the IPD game. We have also implemented the newly developed strategies and compared them with the best-reported learning strategies. The simulation results show that our newly-proposed method, the IPPP strategy, has the lowest cost when compared to the existing learning strategies when playing in the Nonstationary Environment described in this Thesis.

The important contributions of this research when compared to the state-of-the-art are briefly summarized as follows:

- We have developed a family of estimator-based strategies which are based on the Learning-Automata (LA) pursuit scheme and the interconnected LA structure.
- In order to transform the problem to a supervised learning problem, we have proposed two reward reorganization mechanisms: the so-called Dynamic Reward Limit and Previous Reward Limit mechanisms.
- The IPPP strategy achieves 4.86% less cost than the SLASH strategy in its overall performance in the Nonstationary Environment described in this Thesis.

1.3 Organization of Thesis

Chapter 2 introduces some background information related to this Thesis, namely an overview of the field of Learning Automata (LA), and an introduction to the Prisoner's Dilemma problem. Some methodologies used to solve the Prisoner's Dilemma problem, as well as some of the most relevant concepts, are reviewed in Chapter 3. In Chapter 4, we present three learning strategies of the IPD game when played in Nonstationary Environments. In Chapter 5, we discuss the potential factors which may affect the convergence behaviour of learning strategies, and then propose three new approaches which exhibit outstanding achievements in playing the IPD game in such Nonstationary Environments. The results of the experiments involving the different strategies when working in Nonstationary Environments are presented in Chapter 6. A summary of the research done and a discussion of future directions is included in the last Chapter.

Chapter 2

BACKGROUND

In this Chapter, we present some background information related to this Thesis. First, we present an overview of Learning Automata (LA), and then present an overview of the Prisoner's Dilemma problem.

2.1 Overview of Learning Automata

The concept of LA grew out of a fusion of the works of psychologists in modeling observed behaviour, the efforts of statisticians to model the choice of experiments based on past observations, the attempts of operation researchers to implement optimal strategies in the context of the two-armed bandit problem, and the endeavors of system theorists to make rational decisions in random Environments [2].

An automaton is a machine or simple mechanism that operates according to a series of states or configurations [12]. Learning can be defined as the process of arriving at any relatively permanent changes in behaviour which may improve the performance over time on previously completed tasks. Therefore a learning system has the ability to

improve its behaviour with time, according to a defined performance measure. The automaton may learn from its actions if the Environment provides some forms of feedback that informs it whether the decision or selection it made was good or bad. Thus, an LA can be defined as a decision maker which operates in a random Environment, updating its strategy for choosing actions on the basis of the Environment's response [2].

Learning automata (LAs) are powerful in that they can improve their performance by a learning process, which combines rapid and accurate convergence with low computational complexity. In practice, learning automata are used to solve many real life problems such as process control, pattern recognition, task scheduling, optimization problems, diagnosis, concept learning, and routing and bandwidth allocation in computer communication networks [12][17][18][19].

2.1.1 Basic Concepts

Automata operate in random Environments and the learning process involves the determination of an optimal action out of a finite number of allowable actions. For each action, the Environment provides a random response which is either favourable or unfavourable. The Environment and automaton are described in the following sections.

2.1.1.1 The Environment

In the LA paradigm, the Environment is defined by a triple $\{\underline{\alpha}, \underline{c}, \underline{\beta}\}$ where $\underline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents a finite input set, $\underline{\beta} = \{\beta_1, \beta_2, \ldots, \beta_m\}$ represents an output set, and $\underline{c} = \{c_1, c_2, \ldots, c_r\}$ a set of penalty probabilities, where each element c_i of $\underline{c}$

corresponds to one input action α_i of the set $\underline{\alpha}$. The automaton's interaction with its Environment is shown in Figure 2-1 [2].

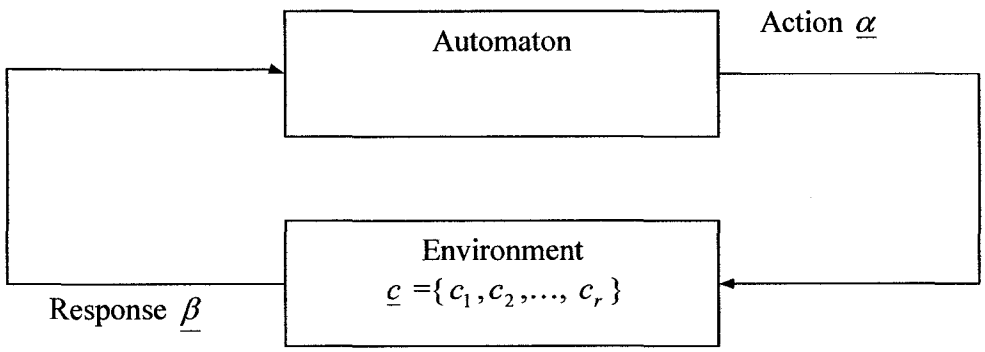

Figure 2-1: The Automaton's Interaction with Its Environment

In the simplest case, the values β_i are usually chosen to be 0 and 1. In the literature '0' is associated with a reward, and '1' with a penalty. With this understanding, the element c_i of $\underline{c}$ may be defined by: $c_i = \Pr\{\beta(n) = 1 | \alpha(n) = \alpha_i)\}$ (i =1,2,...,r), where n represents time. Therefore c_i is the probability that choosing the action α_i will result in a penalty output from the Environment. When the penalty probabilities c_i are constant, the Environment is called a "stationary" Environment. If the penalty probabilities change over time, the Environment is a "Nonstationary" Environment.

There are several models characterised by the response set given by the Environment. Models in which the output from the Environment can take only one of two values, "0" or "1", are referred to as P-models [2]. A further generalization of the Environment allows finite response sets with more than two elements that may take on a finite number of values in an interval [0,1]. Such models are called Q-models [2]. When the output from the Environment is a continuous random variable with possible values in

an interval, say [0, 1], the model is called the S-model [2]. Q-models and S-models are of greater practical utility because they improve the discrimination of the nature of the response of the Environment.

2.1.1.2 The Automaton

The automaton is represented by the quintuple $\{\underline{\phi}, \underline{\beta}, \underline{\alpha}, F\{..\}, H\{...\}\}$[3] where:

i. $\underline{\phi}$ is a set of internal states. At any instant n, the state $\phi(n)$ is an element of the finite set $\underline{\phi} = \{\phi_1, \phi_2, \ldots \phi_s\}$.

ii. $\underline{\alpha}$ is a set of actions (or outputs of the automaton). The output or action of an automaton at the instant n, denoted by α (n), is an element of the finite set $\underline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$.

iii. $\underline{\beta}$ is a set of responses (or inputs from the Environment). The input from the Environment $\beta(n)$, is an element of the set $\underline{\beta}$ which could be either the binary set {0,1}, a finite set, or an infinite set, such as an interval:

$\underline{\beta} = \{\beta_1, \beta_2, \ldots, \beta_m\}$or $\underline{\beta}$ =[a,b] where a, b are real numbers

iv. The transition function $F(.,.)$: $\underline{\phi} \times \underline{\beta} \rightarrow \underline{\phi}$ maps the current state and current input into the next state. It determines the state at the instant $(n+1)$ in terms of the state and input at the instant n

$$\phi(n+1) = F\{\phi(n), \beta(n)\},$$

where the transition function could be either deterministic or stochastic.

v. The output function H(.,.): $\underline{\phi} \times \underline{\beta} \rightarrow \underline{\alpha}$ maps the current state and input into the current output. It determines the output of the automaton at any instant n in

terms of the state at that instant. If the current output depends only on the current state, the function H(.,.) could be replaced by $G(.): \underline{\phi} \rightarrow \underline{\alpha}$. That is

$$\alpha(n) = G(\phi(n))$$

The output function could be either deterministic or stochastic, but without loss of generality, it can be purely deterministic.

The automaton is shown in Figure 2-2 [2].

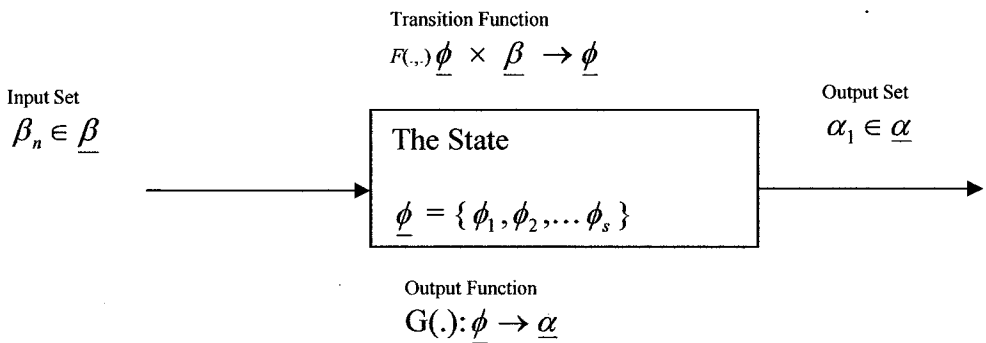

Figure 2-2: The Automaton

2.1.1.3 The Deterministic Automaton

Tsetlin introduced deterministic automata operating in random environments as a model for learning [13]. If the transition function F and output function G are both deterministic, the automaton is called a Deterministic Automaton. In such cases, the succeeding state and action are completely determined by the starting state and the initial inputs. Given a finite input set, a deterministic automaton can be represented by graphs or matrices. An example of a deterministic automaton which has a finite binary input set is shown in Figure 2-3.

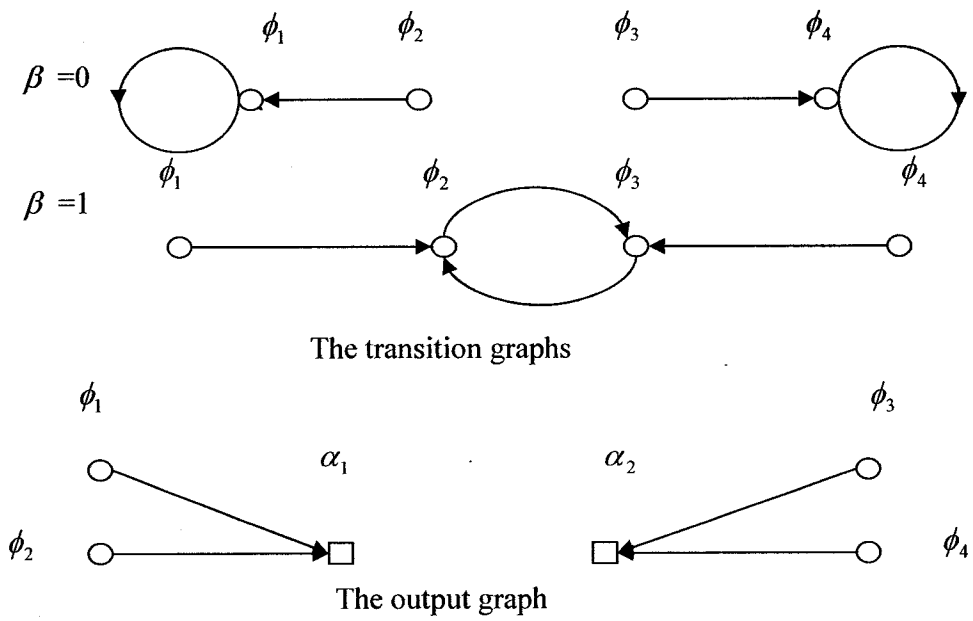

Figure 2-3 A Deterministic Automaton

2.1.1.4 The Stochastic Automaton

If either the transaction function F or the output function G is stochastic, the automaton is stochastic. If the transition function F is stochastic, the element f_{ij}^{β} represents the probability that the automaton moves from state ϕ_i to ϕ_j following an input β. Thus,

$$f_{ij}^{\beta} = \Pr\{\phi(n+1) = \phi_j \mid \phi(n) = \phi_i,\ \beta(n) = \beta\}\ i,j = 1,2,\ldots,s\ \beta(n) \in \underline{\beta}$$

For the output function G, the element g_{ij} represents the probability that the automaton chooses action α_j in state ϕ_i.

$$g_{ij} = p_r\{\alpha(n) = \alpha_j \mid \phi(n) = \phi_i\}\ i,j = 1,2\ldots,r$$

Since g_{ij} and f_{ij}^{β} are probabilities, they lie in the closed interval [0,1], and so to conserve probability measure we have :

$$\sum_{j=1}^{s} f_{ij}^{\beta} = 1 \text{ for each } \beta \in \underline{\beta} \text{ and } i,$$

$$\sum_{j=1}^{r} g_{ij} = 1 \text{ for each } i.$$

The stochastic automaton attempts a solution of the problem without any information of the optimal action. The LA works as follows: One action is selected at random, the response from the environment is observed, action probabilities are updated based on that response, and then the procedure is repeated.

2.1.1.5 Norms of Learning

Based on the interaction with the Environment, a LA generates a sequence of actions. To assess the performance of the learning automaton, it is necessary to introduce quantitative norms of learning. In this section, we will only discuss the definitions for norms of learning for the P-model in stationary random Environments. Further definitions of the LA in Nonstationary Environments will be presented in Section 2.3

To set a quantitative standard, a "pure-chance automaton" can be introduced as a norm for comparison. In the pure chance situation, all the action probabilities are equal. For an *r*-action automaton, the action probability vector $p(n) = p_r\{\alpha(n) = \alpha_i\}$ is given by:

$$p_i(n) = \frac{1}{r} \quad i = 1,2\ldots,r$$

If an automaton has a learning capability, it must do at least better than a pure-chance automaton.

To make this comparison we consider a stationary random environment with penalty probabilities $\{c_1, c_2, \ldots, c_r\}$. If two automata operate in this environment, the one that results in the environment emitting favourable responses more often is to be considered better. A quantity *M(n)* can be defined as the average penalty for a given action probability vector:

$$M(n) = \sum_{i=1}^{r} c_i p_i(n)$$

For the pure-chance automaton, *M(n)* is a constant denoted by M_0 and given by:

$$M_0 = \frac{1}{r}\sum_{i=1}^{r} c_i$$

Also note that: $E[M(n)] = E\{E[\beta(n) \mid p(n)]\}$

$$= E[\beta(n)],$$

$E[M(n)]$ is the average input to the automaton. Using the above definitions, we have the following:

Definition2-1: A learning automaton is *expedient* if $\lim_{n\to\infty} E[M(n)] < M_0$.

Since $\sum_{i=1}^{r} p_i(n) = 1$, we have $\inf M(n) = \inf_{p(n)} \{\sum_{i=1}^{r} c_i p_i(n)\} = \min_i \{c_i\} \equiv c_l$.

Definition2-2: A learning automaton is *optimal* if $\lim_{n\to\infty} E[M(n)] = c_l$,

where $c_l = \min_i \{c_i\}$

Optimality implies that the action α_l associated with the minimum penalty probability c_l is chosen asymptotically with probability one. Although the optimal behaviour is always desirable, literature shows that it is not possible to achieve optimality in any given situation. So, a sub-optimal behaviour is defined [29].

Definition 2-3: A learning automaton is $\varepsilon - optimal$ if $\lim_{n\to\infty} E[M(n)] < c_l + \varepsilon$, where ε is an arbitrarily small positive number.

There are two factors that affect the performance of the automaton: The first is the initialization of the automaton and the second is the set of penalty probabilities of the environment. Some automata satisfy the conditions stated in the definitions for certain initialization and for sets of penalty probabilities. However, the automata for arbitrary initialization in arbitrary environments are more interesting in practical research. These requirements are partially met by an absolutely expedient automaton [29] which has the property defined as below:

Definition 2-4: A learning automaton is *absolutely expedient* if

$$E[M(n+1)|p(n)] < M(n)$$

for all n, all $p_i(n) \in (0,1)$ and for all possible sets $\{c_1, c_2, \ldots, c_r\}$. Taking expectations of both sides, we have

$$E[M(n+1)] < M(n)$$

and thus, $E[M(n)]$ is strictly monotonically decreasing with n in all stationary random Environments.

Therefore, absolute expediency implies expediency and is a stronger condition on the automaton in the stationary random Environments.

2.1.1.6 Fixed Structure Automaton

If the conditional probabilities $\{ g_{ij} \}$ and $\{ f_{ij}^{\beta} \}$ are assumed to be fixed, independent of the time and the input sequence, such a stochastic automaton is referred to as a Fixed-

Structure Stochastic Automaton (FSSA). Tsetlin studied the behaviour of FSSA in the 1960's [13]. An example of his automaton is shown below:

The automaton ($L_{4,2}$) has four states $\{\phi_1, \phi_2, \phi_3, \phi_4\}$ and two actions $\{\alpha_1$ and $\alpha_2\}$. The states ϕ_1, ϕ_2 correspond to the action α_1, while states ϕ_3, ϕ_4 correspond to the action α_2. The automaton accepts inputs from a set {0, 1} and updates its states depending on the input. The state transition graphs are shown in Figure 2-4

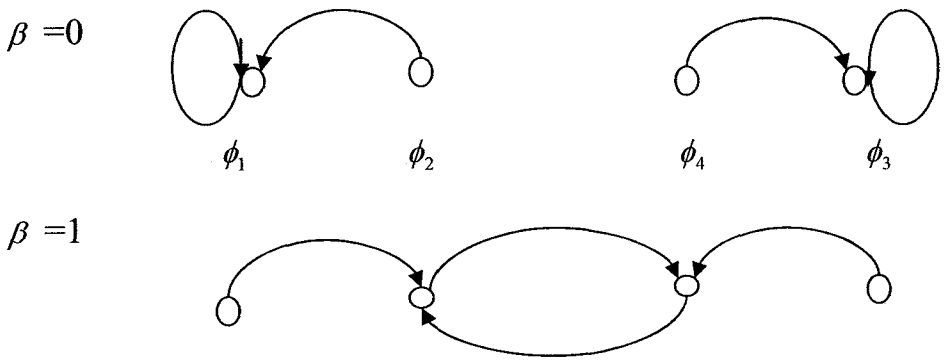

Figure 2-4 The State Transition Graphs

Consequently the transition matrices and output matrix of the automaton can be expressed as:

$$F^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad F^1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad G = \begin{matrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{matrix} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

We use the notation that $d_i = 1 - c_i$, which is the probability of obtaining a favourable response from the environment to an action α_i. Then, the matrix of total transition probabilities can be represented by:

$$\widetilde{F} = \begin{bmatrix} d_i & c_1 & 0 & 0 \\ d_1 & 0 & 0 & c_1 \\ 0 & 0 & d_2 & c_2 \\ 0 & c_2 & d_2 & 0 \end{bmatrix}$$

The matrix of transition probabilities represents an ergodic Markov Chain. The final probabilities of the four states $\phi_i (i = 1,2,3,4)$ can be obtained as $\{\pi_i^*\}$, where π_i^* is the element of the so-called equilibrium (or stationary) probability vector, π^*, which satisfies the equation:

$$\widetilde{F}^T * \pi^* = \pi^*$$

By solving this equation, the action probabilities vector $\underline{P}$ of choosing the action asymptotically can be obtained as below:

$$P = \begin{bmatrix} p_1(\infty) \\ p_2(\infty) \end{bmatrix} = \begin{bmatrix} \pi_1^* + \pi_2^* \\ \pi_3^* + \pi_4^* \end{bmatrix} = \begin{bmatrix} \dfrac{c_2^2}{c_1^2 + c_2^2} \\ \dfrac{c_1^2}{c_1^2 + c_2^2} \end{bmatrix}$$

The action probabilities p_1 and p_2 converge, and the final state probabilities are functions of the penalty probabilities c_1 and c_2.

FSSA in stationary Environments are often modelled using egodic Markov Chains, where the asymptotical probabilities of converging in the various actions are completely dependent on the penalty probabilities of the Environment and the meaning of the machine.

2.1.1.7 Variable Structure Automaton

In a learning automaton model, if the transition probabilities f_{ij}^{β} are updated at every step by a reinforcement scheme, the automaton is called a Variable-Structure Stochastic Automaton (VSSA). In 1963, Varshavskii and Vorontsova first introduced VSSA which updated these transition probabilities [14]. Further, Fu and his associates made extensive use of action probabilities as these were found to be mathematically more tractable [3][15].

VSSA are represented by the quintuple:

$$\{\underline{\phi}, \underline{\alpha}, \underline{\beta}, \mathrm{A}\}$$

where A is the updating algorithm (also known as the reinforcement scheme).

In general, updating algorithms or reinforcement schemes can be represented as follows:

$$P(n+1) = T[P(n), \alpha(n), \beta(n)]$$

The reinforcement schemes changing action or transition probability bring greater flexibility into the learning process. In the next Section, several reinforcement schemes which are the basis of learning in variable structure automaton will be presented.

2.1.2 Reinforcement Schemes

Reinforcement schemes can be classified as being linear, nonlinear, or hybrid based on the nature of the mapping *T*. In this Thesis, for reasons of analytical simplicity, we will place a strong emphasis on the linear schemes.

2.1.2.1 Linear Reward-Penalty (L_{RP}) Scheme

This scheme updates the probabilities for both a favourable and unfavourable response from the Environment. The probabilities are updated as follows:

Reward: $\alpha(n) = \alpha_i$, $\beta(n) = 0$:

$$p_{j \neq i}(n+1) = (1-a)p_j(n)$$

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$

Penalty on $\alpha(n) = \alpha_i$, $\beta(n) = 1$:

$$p_{j \neq i}(n+1) = b/r-1 + (1-b)p_j(n)$$

$$p_i(n+1) = (1-b)p_i(n)$$

where a and b are the respective reward and penalty parameters, and $0<a<1$, $0 \leq <b<1$.

The L_{RP} scheme is expedient for all initial conditions and all stationary Environments (with $c_1 \neq c_2$)[2]. The action probability vector $P(n)$ has been shown to converge in distribution. The scheme is also ergodic, so that this distribution function is independent of the initial probability vector $P(0)$.

2.1.2.2 Linear Reward-Inaction (L_{RI}) Scheme

The Linear Reward-Inaction L_{RI} scheme is derived from the L_{RP} by setting the penalty parameter b to zero. Using the L_{RI} scheme, the automaton ignores the penalty but updates the action probabilities when it receives a favourable response. The updating equations are as follows:

Reward: $\alpha(n) = \alpha_i$, $\beta(n) = 0$:

$$p_{j \neq i}(n+1) = (1-a)p_j(n)$$

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$

Penalty: $\alpha(n) = \alpha_i$, $\beta(n) = 1$:

$$p_i(n+1) = p_i(n) \text{ for all } i$$

The L_{RI} scheme is absolutely expedient and $\varepsilon - optimal$ in stationary random Environments. But since the automaton is not ergodic, it is possible for it to get "stuck" in one of the absorbing states. This makes it sensitive to the starting conditions and probabilities and unsuitable for the Nonstationary Environments. The probability vector may not adapt to the new optimum for a long time when the action probabilities are close to unity.

2.1.2.3 Linear Inaction-Penalty (L_{IP}) Scheme

The Linear Inaction-Penalty L_{IP} scheme is derived from the L_{RP} by setting the reward parameter a to 0. Using the L_{IP} scheme, the automaton ignores the reward but updates the action probabilities when an unfavourable response is obtained. The update equations follow from this description, and are omitted.

The L_{IP} scheme is not $\varepsilon - optimal$ in stationary random Environments.

2.1.2.4 Discrete Linear Reward-Penalty (DL_{RP} scheme)

In a stationary Environment, all the schemes approach the optimal action probability asymptotically. By discretizing the probability space to assume only discrete values in the

interval [0,1], the convergence can be enhanced when the action probability of any choice is close to unity. The discretisation is termed linear if the allowable values are equally spaced; otherwise it is termed non-linear. In general, the discretized version of a continuous scheme retains the property of the original VSSA, namely $\varepsilon - optimal$ or ergodicity [4]. In this section we give the DL_{RP} scheme, which is achieved by discretizing, the probability space of the L_{RP} scheme.

The updating algorithm is fairly similar to that of the continuous case, except that the automaton has (N+1) probable values. The updating formulas are as follows:

Reward: $\alpha(n) = \alpha_i$, $\beta(n) = 0$:

$$p_{j \neq i}(n+1) = p_j(n) - \frac{1}{N(r-1)},$$

$$p_i(n+1) = p_i(n) + \frac{1}{N},$$

Penalty: $\alpha(n) = \alpha_i$, $\beta(n) = 1$:

$$p_{j \neq i}(n+1) = p_j(n) + \frac{1}{N(r-1)},$$

$$p_i(n+1) = p_i(n) - \frac{1}{N},$$

where the automaton has N +1 states and r available actions.

Discretised automata increase the probability of choosing the desired action to the value of unity directly, instead of approaching that value asymptotically as in the case of the continuous schemes [4].

2.1.3 Nonstationary Environments

A Nonstationary Environment can be defined as an Environment $\{\underline{\alpha}, \underline{c}, \underline{\beta}\}$ in which the penalty probabilities $\{ c_i \}$ (i=1,2,..,r) corresponding to the various actions vary with time. We present below an example of a hierarchical system of LA to illustrate the concept of a Nonstationary Environment.

The hierarchy consists of a single automaton with two actions at the top level, each of which is connected to an automaton at the second level. The second-level automata interact with a stationary Environment. From the point view of the automata at the top level, they are operating in a composite Environment consisting of the second level automata and the Environment. Thus, the penalty probabilities depend on the action probabilities of automata in the second-level and the process as it is updated with time. The composite Environment is thus a Nonstationary Environment. The hierarchical system is shown in Fig. 2-5

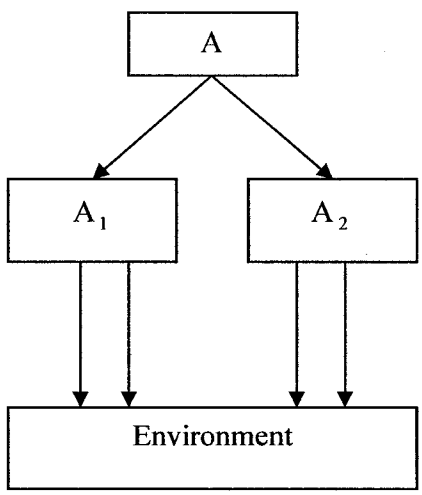

Figure 2-5: A Hierarchical System of LA and a Switching Environment

From both theoretical as well as practical viewpoints, learning in Environments which changes with time is more complex. In such environments, LA with more flexible

and adaptive schemes may be more advantageous than those which operate with a fixed strategy. It is not certain whether we can obtain a single strategy which can perform efficiently in the specific Nonstationary Environment. In this Section, we will discuss four types of Nonstationary Environments.

2.1.3.1 Markovian Switching Environments

In the case of a Markovian Switching Environment (MSE), we assume that the composite Environment consists of separate Environments, E_i (i =1,...,d) which themselves obey a Markov chain. Varshavskii and Vorontsova analyzed VSSA in a MSE in detail and proved that it is possible to obtain a good performance in a MSE by using such LA. Further, it is possible to view the VSSA itself in the MSE as a homogeneous Markov chain. A FSSA in a Nonstationary Environment, in general, is described by a non-homogeneous Markov chain, but when the Nonstationary Environment is MSE, the overall system is equivalent to a homogeneous Markov chain [2].

2.1.3.2 State Dependent Nonstationary Environments

A State Dependent Nonstationary Environment is defined by E = $\{E_1, E_2, \ldots, E_d\}$ which is the finite state space of the Environment when the Environment is Nonstationary, where states vary with a stage number n. Such variations may depend either explicitly or implicitly on n [2]. Three models for State Dependent Nonstationary Environments were introduced by Narendra, and are listed below:

- In Model A, the penalty probability c_i of the corresponding action α_i increases while c_j (j $\neq$ i) decreases. Obviously, the action α_i becomes worse as time

proceeds.

- In Model B, the penalty probabilities are monotonically increasing functions of the corresponding action probabilities. In this case, the Automaton-Environment interaction can be described by a homogeneous Markov process.
- Model C represents a dynamic Environment in which the penalty probabilities are the function of the previous penalty probabilities and action probabilities. Compared with Model B, this model is more effective in predicting the transient behaviour of systems accurately.

2.1.3.3 Multiple Environments

Multiple Environments are defined by a composite Environment E which is made up of N Environments operating in parallel; the response of the Environment to an action of the automaton is an n-vector, each of whose elements is a "0" or "1". An automaton interacting in multiple Environments is shown in Figure 2-6.

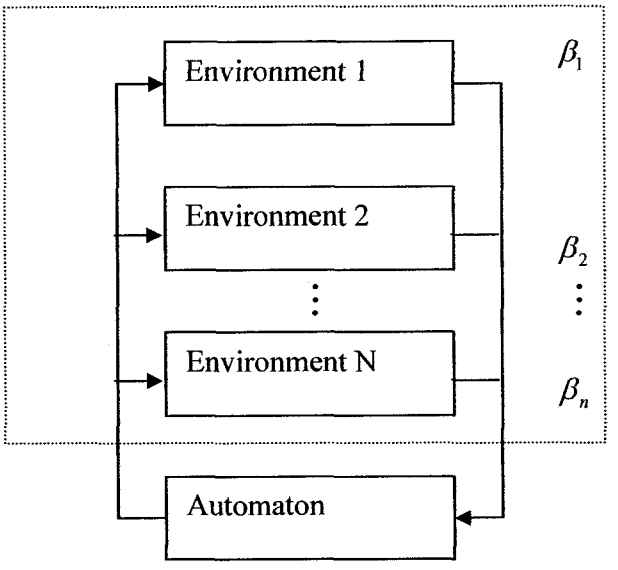

Figure 2-6 An Automaton Acting in Multiple Environments

2.1.3.4 Periodic Environments

A periodic Environment consist of d stationary Environments $\{E_1, E_2, \ldots, E_d\}$, and the set of Environments switches periodically between them. If the period is known *a priori*, we can start d LA and a switching device that connects the automata in sequence to the specific Environment at every time instant, and thus each automaton effectively operates in a stationary Environment. In the case when the period is unknown, the solution can be obtained by using either a deterministic LA or a VSSA.

In this Thesis, we will propose to use the concept of periodic Environments to tackle the Iterative Prisoner's Dilemma game.

2.1.3.5 Norms of Learning in Nonstationary Environment

Since the optimal action can change with time, the fundamental concepts of norms of learning, such as expediency, optimality, and absolute expediency should be re-considered in the Nonstationary Environments. We will consider the P-model for stating the definitions.

The average penalty *M(n)*:

$$M(n) = \sum_{i=1}^{r} c_i(n) p_i(n)$$

The pure-chance automaton has an average penalty M_0 as a function of time:

$$M_0(n) = \frac{1}{r} \sum_{i=1}^{r} c_i(n)$$

Now we make the following definition:

Definition 2-5: A learning automaton operating in a Nonstationary Environment is *expedient* if there exists a n_0 such that for all $n > n_0$, we have:

$$E[M(n)] - M_0(n) < 0$$

Definition 2-6: In a Nonstationary Environment where there is no fixed optimal action, an automaton can be defined as optimal if it minimizes:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{n=1}^{T} E[\beta(n)]$$

The definition of absolute expediency can be achieved as before, with the time-varying penalty probabilities.

2.2 Prisoner's Dilemma

2.2.1 Introduction

The Prisoner's Dilemma (PD) has been discussed extensively to model the conflict between competition and cooperation, or between individual and collective rationality [1] [8][9][10][11]. The dilemma was originally introduced by Merrill Flood and Melvin Dresher in 1950. In the same year, the concepts "Prisoner's Dilemma" and "Sentence Payoff" were invented by the mathematician Albert W. Tucker. Since then, the Prisoner's Dilemma has been studied in depth as a "non-zero sum" game in areas such as economics, political science, evolutionary biology, and game theory.

Zero-sum describes a situation in which a participant's gain (or loss) is exactly balanced by the losses (or gains) of the other participant(s) [5]. It is named after the fact that the net sum will be zero when the total gains of the participants are added and the

total losses are subtracted. Non-zero sum games are interesting since there are options that bring gains to both players.

Here is a detailed description of the classical "one-shot" PD game [1]:

Two people, call them A and B, are accused of having committed a crime. The prosecutor makes A the following proposition. "There is circumstantial evidence against both you and your accomplice. If both of you plead innocent you will be convicted anyway, and each will receive a sentence of two years imprisonment. However, if you give evidence against your accomplice, we will have a better case against him and he will be convicted to a harder sentence: five years imprisonment. In exchange for your help, we will let you go free for giving evidence to the State. The prosecutor gives B the exact same proposition. He also tells both A and B that the proposition was made to the other person. Finally, he tells them that if they both confess, each will receive a sentence of four years. A and B cannot communicate with each other; each must decide independently whether to incriminate his partner, or to cooperate with him.

Consider the situation above. The two prisoners can be viewed as two players in the PD game. Obviously, it is better off for them to cooperate with each other. From the point view of Player A, if Player B chooses cooperation, the rational choice of Player A is to defect since he would rather be free and not in jail for two years. Suppose that Player B chooses defection, it is again preferable for him to defect otherwise he would stay in jail for five years. So, no matter what Player B does, it is rational for Player A to defect all the time. In a similar way, Player B has the same reason to defect all the time. So, both players defect even though they would do worse than if they both had cooperated. Therefore, the selfish choice of defection brings about a higher payoff than that of cooperation. Here lies the dilemma!

2.2.2 Generalization and Constraints

After refining the PD game in different situations, we can gain a more general definition of the game as follows:

Two players simultaneously choose an action, to either cooperate (C) or defect (D). There are four possible outcomes for each round: both cooperate (CC), the first player cooperates, while the second defects (CD), the opposite of the latter (DC), and both players defect (DD). Each player receives a payoff after each round, as shown in Table 2-1:

B Actions / A Actions	C	D
C	R, R	S, T
D	T, S	P, P

Table 2-1: Payoff for Prisoner's Dilemma

In the above table, T denotes the Temptation to defect, R denotes the Reward for mutual cooperation, P denotes the Punishment for mutual defection, and S denotes the so-called Sucker's payoff. The payoff for each player depends on the combined actions of both the players.

To render the dilemma meaningful, the constraint $T > R > P > S$ should hold.

It should be noted that, to avoid getting out of the dilemma by switching actions during the game, the reward for mutual cooperation should be greater than the average of the payoff for the Temptation and the Sucker's payoff. This leads to the inequality: $R > (S + T)/2$.

There are many examples in human interaction which have a similar payoff matrix. In politics, the PD scenario is often used to illustrate the problem of the arms race between two rival countries [20]. Another interesting example concerns the position trade-off in cycling races [21]. In this case, the two cyclists encounter the dilemma when

they have to decide whether or not to cooperate with each other so as to share the tough load of the front position. Furthermore, the PD scenario also can be found in many special problems.

Consider the one shot Prisoner's Dilemma. The most rational choice for each player is to defect against each other. But in real life instances of the PD, cooperation is often observed. One possible explanation is that people in real-life scenarios expect the opponent to understand his kindness, hoping to meet the opponent in the future again [2]. This is the reason why it is more interesting to repeat the PD game many times, which leads to the Iterated Prisoner's Dilemma (IPD) explained below.

2.2.3 Iterated Prisoner's Dilemma (IPD)

In an IPD game, players follow strategies which respectively determine a player's next action in any given game situation. The players repetitively perform actions which make it possible to memorize the previous behaviours of the opponent for future actions.

2.2.3.1 Fixed number of Iterations

If an IPD game operates exactly N times, in which N is a known constant, then the game can be reduced to the one-shot PD game, which is easily proved by backward induction. As a rational player, a reasonable person might defect at the last iteration, since the opponent will not have a chance to punish him. Therefore, both players will defect at the last iteration. Similarly, each player might defect at the second-to-last iteration. This logic can be applied all the way back to the first iteration. Thus, the both players will perform mutual defection during the iterations.

2.2.3.2 Infinite Iterations

Backward induction depends on the players knowing the number of iterations in the IPD game. So, to avoid backward induction, we let the IPD game operate infinitely. Since there is no "last round", it is obvious that backward induction does not apply to the infinite IPD. Thus, each player has to determine the successive actions based on the previous actions, and the behaviour of the opponent. In this situation, we can use the average value of the payoff during a certain number of iterations to evaluate the strategy of the player.

2.2.3.3 Indefinite Iterations

Most contemporary investigations in the IPD game assumes that the number of the iterations is neither infinite nor of fixed finite length, but rather of indeterminate length [6]. A probability P is introduced in the indefinite IPD so that the game will continue to operate at each iteration with the probability P. If the probability P is set to zero, the IPD becomes a one shot PD game. If the probability P is set to unity, the IPD becomes an infinite IPD, and the value of defection decreases. If the probability P is between zero and one, the players might not know when the last interation occurs, implying that there is no start point for the backward induction. In this case, both players have to use different types of strategies which only look forward. By playing the IPD indefinitely, cooperation is a rational outcome.

2.2.4 Multiplayer IPD game

In the real world, the situation is more complex because the individual interacts with ALL of the other participants at once. If an IPD game involves more than two players, the IPD game is called the Multiplayer IPD game.

A classical and realistic Multiplayer PD circumstance was described by Garret Hardin in "The tragedy of the commons" [7]. "The tragedy of commons" was described in the scenario in which a group of farmers graze their cows on common land. Each farmer prefers to graze more of their cows in the common area rather than on his own land. However, the common area will be rendered unsuitable for grazing if more than a certain number of cows are on it. We might thus formulate the payoff matrix as shown in Table 2-2.

Action	N or fewer Defect	More than N Defect
C	R, T	S, P
D	T, R	P, P

Table 2-2: Payoff for Multiplayer Prisoner's Dilemma

where T denotes the temptation to graze more cows, R denotes the reward for grazing an adequate number of cows in common area, P denotes the punishment for destroying the common land by over grazing, and S denotes the punishment for destroying the common area and the missed benefit of grazing more cows there. Obviously, the constraint $T > R > P > S$ still holds.

In the multiplayer PD, the equilibrium not only lies in the universal defection, but also before the threshold *N*. When the number of defectors exceeds the threshold *N*, the payoff of the defectors will move from T to P. Furthermore, in a two-player PD, while it

is better off to cooperate, in a multiplayer PD, the optimal outcome depends on the threshold. The multiplayer PD is not only much more complex, but also more realistic than the two player PD. In this Thesis, we will focus on the two players PD game and marginally visit the concept of the multiplayer PD.

Chapter 3

PRIOR ART IN THE ITERATED PRISONER'S DILEMMA

In order to tackle the problems described in the previous Chapter, this Chapter will review some methodologies used to solve the PD and IPD, as well as some of the most relevant concepts.

3.1 Axelrod's Tournaments

Around 1980, Robert Axelrod conducted a computer simulation tournament designed to investigate the PD game [10][11]. The participants in the tournament submitted computer programs that would compete in an IPD game. In order to get a more stable estimate of the scores for each pair of players, the tournament ran five times, and every program competed against each other in exactly 200 iterations.

3.1.1 The First Round Tournament

In the first round, there were fourteen entries in the tournament which came from different disciplines. The analysis of the results showed that the strategy Tit for Tat submitted by Anatol Rapoport got the highest average score [10]. Tit for Tat starts with a cooperative action and replicates the actions of the opponent in succession. It is very powerful and does well when playing against its twin although it seems too generous when it plays the tournament against "Random" strategy. Furthermore, this simple rule cannot be exploited too easily.

In the literature, Axelrod claimed that niceness, forgiveness and optimism are essential characters of a winning strategy. Furthermore, although Tit for Tat won in this tournament, the result was not definitive because the effectiveness of a particular strategy depends not only on its own characteristics, but also on the nature of other strategies with which it must interact [1].

3.1.2 The Second Round Tournament

In the same year, Axelrod conducted the second round which made up for the imperfection in the first round and involved more strategies [11]. There were 62 strategies implemented in the tournament and the length of the game was determined by a probability 0.00346 to end the game with each given iteration.

Just as it had won in the first round, Tit for Tat won again in the second tournament, despite the fact that many of its competitors were far more intricate [11]. In the tournament, Tit for Tat exhibited favorable characteristics of a winning strategy such as niceness, forgiveness, and provocability. It was never the first to defect. It also forgave an

isolated defection after a single response. However, it was always provoked by a defection no matter how good the interaction had been so far [2].

Axelrod proposed another way to examine the robustness of the results by conducting a whole sequence of hypothetical rounds in the tournament. During the series tournaments, the unsuccessful strategies died away, while the successful ones became popular in the later tournaments.

In the process of robustness examination, the tournament was regarded as a single generation of strategies. The average score for a strategy in the current generation corresponded to its degree of success in the generation The numbers of a given strategy in the next generation were taken to be proportional to the result of its numbers and its average score in the current generation. Thus, the effective strategies became more widespread and the less effective strategies became less common in the tournaments. Since there were not any new strategies generated, the validation approach was still not a strictly evolutionary approach.

The Tit for Tat is a very robust strategy. It does very well over a wide range of environments, but it is not the best strategy for the IPD game. The Tit for Tat could be beaten by any strategy which is able to identify the "Random" strategy and never cooperate with it. Furthermore, in the evaluation of robustness, the Tit for Tat would not come in first place if there were only specific strategies in the environment [11].

3.1.3 The Strategies in the Tournaments

Here are some of strategies in the Axelrod's Tournaments, some of which we will use in our experiments.

Defect-All

This strategy always defects. To avoid getting the Sucker's payoff, an opponent's best choice is to defect as well.

Cooperate-All

This strategy always cooperates. To get as low a penalty as possible, the opponent player should defect, hence giving the cooperative player the Sucker's payoff.

Tit for Tat

This strategy cooperates in the first round, and then repeats the opponent's actions in the following rounds.

Tit for Two Tat

This strategy plays like Tit for Tat but forgives one defection. It defects after two consecutive defections.

Random

A Random player makes random actions. It defects or cooperates with a probability of 0.5 in every move.

Grim

A Grim Player cooperates in the first round, but will always defect after one single defection of the opponent. This player is also known as the Friedman.

Tester

A Tester always starts with defection in the first round, and cooperates in the second round. It tests the response of the opponent. If the opponent reciprocates the first defection, the Tester will go on playing Tit for Tat. However if the opponent continues to

cooperate, the Tester will continue to alternate between cooperation and defection until the opponent defects, whereby it will then continue with the Tit for Tat strategy.

Joss

This strategy is a variation of Tit for Tat. Like Tit for Tat, it always defects immediately after the other player defects. And after the other player cooperates, 10% of the time it defects. Thus it tries to get away with an occasional exploitation of the other player.

3.2 Evolutionary Cooperation Approach

3.2.1 Evolutionary Game Theory

Evolutionary game theory is an adaptation of game theory that concerns games played by populations of players, in which expected payoffs are frequency dependent [22]. The players in the evolutionary game model are not assumed to be rational. Instead, they play whatever strategy their genes tell them to play. During the evolutionary process, each player might adjust its strategy according to the payoff achieved at the end of each interaction.

3.2.2 The Evolution of Strategies in the IPD

In 1987, Axelrod demonstrated a genetic algorithm to simulate the evolution of strategies for an IPD game [8]. The environment, in which the evolutionary simulation was run, consisted of eight representative strategies for playing the IPD game. The genetic algorithm worked in five steps as below:

- A population is initialized by random strings consisting of seventy C's and D's, which represents each allowable strategy.

- The effectiveness of each strategy is determined by its average score obtained by playing the IPD game in the current environment.
- Using the average score as a fitness criterion, the relatively effective strategies are selected to have more offspring.
- The relatively effective strategies are used to produce offspring by using two operators such as mutation and crossover. Thus, each offspring not only inherits part of its genetic material from its parents, but also occasionally brings in new genes through mutation.
- After many generations of selection for relatively effective strategies, the new population might be more likely to use the most successful strategy than the original population does in the tournament.

During the computer simulation from a strictly random start, the genetic algorithm evolved to yield populations whose median member was just as successful as the strategy Tit for Tat. Most of the strategies that evolved actually resembled Tit for Tat.

3.2.3 The Limitations of Axelrod's Evolutionary Strategies

The genetic algorithm in Axelrod's evolutionary approach is a highly effective method of searching for effective strategies in a huge space of possibilities. However, it still has some intrinsic limitations.

In Axelrod's evolution simulation, the initial population consists of twenty strings of characters C and D, which are generated randomly without a systematic selection criterion. However, since the effectiveness of the strategy depends on the opponent's behaviour, results of the simulation may depend on the composition of the initial population.

It should be noted that the simulation results are sensitive to the environment in which the computer simulation was run. In Axelrod's evolution simulation, the environment was formed by eight strategies, which were representative in his second tournament. The literature has confirmed the sensitivity of Axelrod's results with respect to the alternative environment [23] [24].

3.3 Machine Learning

The machine learning community also has an interest in the IPD game and other games. Their interest is in adaptive players who learn to play these games [22]. In contrast to game theory, the players are not assumed to be perfectly rational, and in contrast to the evolutionary game theory, the focus is not on populations, but on an individual player, whose behaviours would change as he/she learns more about the other players' actions.

Consider an LA that interacts with the surrounding environment. The connection from the PD game to the LA can be set by the following description: take the LA to be a PD player, the possible actions to be C and D, and the cost of each action to be the corresponding prison sentence; a short (or zero) sentence is a reward, a long sentence is a punishment.

In this section, we will present Kehagias IPD played by Probabilistic Learning Automata (PLA) [1] and several learning approaches based on the different learning schemes.

3.3.1 Kehagias IPD and PLA

In general, the LA should choose its actions in a way that maximize rewards and minimizes punishments. While playing the IPD game, the LA should try to minimize the cost of playing the game. Kehagias designed the LA based on the L_{RP} scheme [1]. The LA will act based on the action probability vector which changes with time.

3.3.1.1 Kehagias Learning Scheme

In the Kehagias learning scheme, a' represents the learning rate, $p(t)$ represents the probability of choosing cooperation (C) and $q(t)$ represents the probability of choosing defection (D) at time t. Thus, $p(t)$ and $q(t)$ change according to the Environment's responses.

In the case where action C is chosen, if the environment rewards the action, then the probability of choosing C is increased. The action probability updating scheme is then as shown below:

$p(t+1) = p(t) + (1 - p(t)) \times a'$ if at time t action = C, response = Reward.

$p(t+1) = (1 - a') \times p(t)$ if at time t action =C, response = Punish.

In the case where action D is chosen, a similar updating scheme is used. This is shown below:

$q(t+1) = q(t) + (1 - q(t)) \times a'$ if at time t action = D, response = Reward,

$q(t+1) = (1 - a') \times q(t)$ if at time t action = D, response = Punish.

3.3.1.2 The Learning Rate and the Environment Responses

In Kehagias' scheme, the learning rate a' is not fixed but depending on the so-called opportunity cost of the differences S-P, P-R, R-T. Figure 3-1 shows the interaction and the opportunity costs of two players in playing the IPD game.

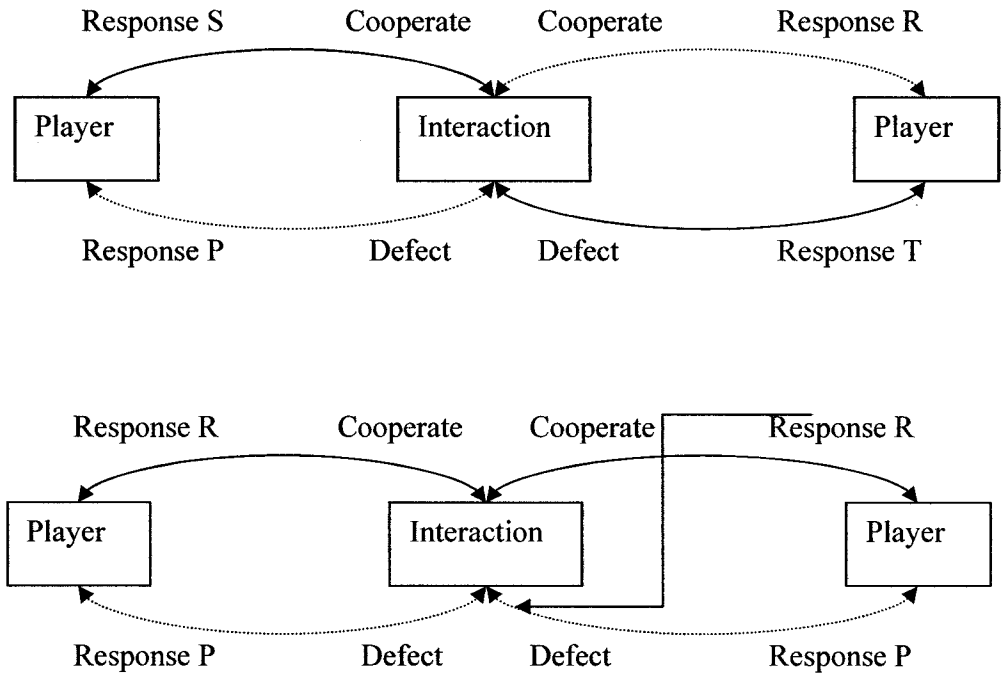

Figure 3-1 The Interaction and the Opportunity Costs of Two Players in Playing the IPD Game

In the Figure 3-1, there are two players A and B. The learning rate for Kehagias' scheme and the interactions between the two players are specified as below:

- If both players choose C, the Kehagias environment will reward them for their cooperation. Thus, both players will compute their gain as P-R and increase their

cooperation probability by a learning rate a' = (P-R)/100.

- If player A cooperates and player B defects, the Kehagias environment will reward player B for its defection and punish player A for its cooperation. In this situation, player A will receive a cost of S and player B will receive a cost of T. From the viewpoint of player A, it might obtain a cost of P if it (i.e., Player A) chose defection. So, the difference of the opportunity cost of player A is a loss of S-P and the learning rate should be a'=(S-P)/100. Similarly, the difference of the opportunity cost of player B is a gain of R-T and the learning rate would be a'=(R-T)/100.
- If they both choose D, the Kehagias environment will punish them for their defection. Thus, both players will compute their loss as P-R and decrease their defection probability by a learning rate a' = (P-R)/100.

3.3.1.3 Kehagias' Experiments

Kehagias designed an experiment of the IPD with two PLA's, in which one LA represents player A and the other LA represents player B. The computer simulation performed 700 iterations during the experiment. Since the learning process described in the previous section depends not on T, R, P or S themselves but on the differences S-P, P-R and R-T, Kehagias ran several experiments with various combinations of P, R, S, T values.

The results of the experiments show that with certain combinations of values of P, R, S and T, the cooperation probabilities of both players become unity after 700 iterations, which means that in some cases, both players obtain a pure cooperation strategy. Since Kehagias' scheme encourages both players cooperating and punishes both playing defectively, there is no pure defection strategy in the experiment. In general, as expected,

a large value of P and a small value of R promote cooperation, and a large value of S and a small value of T promote defection [3].

3.3.2 Reward-Inaction LA Approach

This LA is based on the L_{RI} scheme which ignores the penalty response, meaning that learning is only achieved when a reward from the environment is given. When the action probability variable is updated due to a reward, it is calculated using the same formula as Kehagias' L_{RP} scheme. The Environment rewards the LA in such situations as below:

- The Reward Inaction LA defects alone.
- Both LA cooperate.

3.3.3 Inaction-Penalty LA Approach

This LA is based on the L_{IP} scheme which ignores the reward response, meaning that learning is only achieved when a penalty from the environment is given. When the action probability variable is updated due to a penalty, it is calculated using the same formula as Kehagias' L_{RP} scheme. The Environment punishes the LA in such situations as below:

- The Inaction Penalty LA cooperates alone.
- Both LAs defect.

So, the Inaction-Penalty player never converges toward either cooperation or defection.

3.4 Conclusions

In this Chapter, we reviewed some methodologies used to solve the IPD game as well as some of the most relevant concepts. We focused on optimizing the behaviour of individual players and showed how we could design new adaptive strategies by using LA. In the next Chapter, we will present three learning approaches which perform well, especially in Nonstationary Environments.

Chapter 4

BASIS FOR NEW METHODS FOR IPD GAMES

In this Chapter we present three IPD game players in a Nonstationary Environment, they are implemented by an interconnected PLA. Based on these players, we will introduce our new approaches for playing an IPD game in the next chapter.

4.1 The Nonstationary Environment in an IPD Game

In an IPD game, an optimal solution depends completely on the strategy of the opponent. This means that a single fixed strategy has the potential to be an optimal solution for playing the IPD game. This does not imply that there is an optimal strategy during the play, and furthermore it is practically impossible for a single fixed strategy to achieve an optimal result when the opponent periodically changes strategies during the game. To simulate the behaviour of the learning player in this situation, we have constructed a periodic environment by varying the available strategies for the opponent. The periodic environment for the IPD game is shown in Figure 4-1

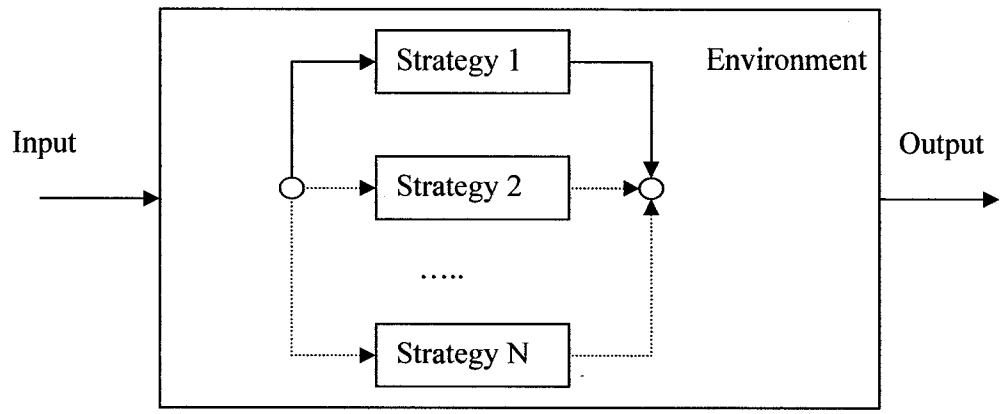

Figure 4-1: Periodic Environment for the IPD Game

The periodic environment above is formed by using N opponent strategies for playing the IPD game.

While playing the IPD game in a periodic environment, if the player achieves an optimal result for each period, the player would obtain an overall optimal outcome. Since it is impossible for a single fixed strategy to be optimal against any strategy of the opponent, the player should not adopt a single fixed strategy all the time. Rather, they should adjust his/her behaviour as the opponent's strategy changes. Thus, to improve success in the game, the player should not only perform efficiently with each opponent's strategy but also discover the switches in time, and adjust his/her behaviour according to the opponent's strategies.

4.1.1 Tit for Tat in the Periodic Environment

The fixed strategy Tit for Tat begins with cooperation in the first round, and is followed by a replica of his opponent's last action, which means that no matter how the environment changes, a player using Tit for Tat will always repeat the opponent's actions during the play. However, although Tit for Tat is able to discover the changes in an opponent's action and adjust his/her behaviour immediately, this strategy does not ensure

an optimal outcome. Figure 4-2 demonstrates Tit for Tat performing in a periodic environment (Defect-All/Tit for Tat).

Tit for Tat	*C D D ... D D D C D C D ...*
Environment	
(Defect-All/Tit for Tat)	*D D D ... D D C D C D C ...*

Figure 4-2: Tit for Tat Performing in a Periodic Environment

In the periodic environment (Defect-All/Tit for Tat), in where the opponent switches between Defect-All and Tit for Tat, the optimal solution is to defect when the environment switches to the Defect-All strategy, and to cooperate when the environment switches to Tit for Tat. Figure 4-2 shows that after the periodic Environment's strategy changes into Tit for Tat from Defect-All, the Tit for Tat player will alternate actions between Defection and Cooperation. It is clear that the Tit for Tat player can not obtain the optimal solution in this environment.

4.1.2 The Linear Reward-Penalty Player in the Periodic Environment

The Linear Reward-Penalty player increases reward and minimizes punishment by choosing actions probabilistically, while the action probabilities are updated following the L_{RP} scheme. On the one hand, the player tries to maximize its immediate rewards; subsequently, it tries to learn the behaviour of the environment and use this knowledge for successive action.

In the L_{RP} scheme, when the learning rate is large, the action probabilities change greatly in a single step. Conversely, when the learning rate is small, the action probability

changes marginally with each step. In order to achieve an optimal or sub-optimal result, the learning rate of the L_{RP} player might vary with different periodic environments.

When the learning rate is very small in the periodic environment, the L_{RP} player can hardly detect changes in the opponent's strategy and the learning process is thus longer. If the learning rate is large, the L_{RP} player would be excessively sensitive to the behaviour of the environment. It is difficult to set an appropriate learning rate for the L_{RP} player in the periodic environment. Therefore, the L_{RP} player does not play well in the periodic environment.

Henning Hetland and Tor-Øyvind Lohne Eriksen introduced three players based on the concept of an interconnected LA, which perform well in a Nonstationary Environment [25]. We will present the concept of the interconnected LA and the three players in the following sections.

4.2 Interconnected LA

Automata can be interconnected in certain configurations so as to exhibit group behaviour for complex and realistic decision-making problems. In this case, the Environment reacts to the actions of multiple automata, and the environmental output is a result of the combined actions chosen by all automata [2]. In general, there are two fundamental models of interconnected automata: namely, the synchronous and the sequential model.

4.2.1 Synchronous Models

For this type of model, all automata choose their actions synchronously. An important consequence of the synchronous model is that the resulting configurations can be viewed as games of automata with particular payoff structures [2]. Figure 4-3 shows the simplest case, in which two automaton-environment pairs, $A_1 - E_1$ and $A_2 - E_2$, are connected in a feedback configuration.

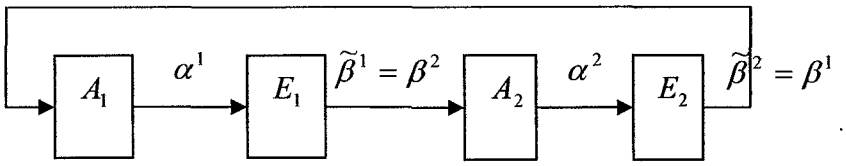

Figure 4-3: Synchronous Models

In this model, each automaton is assumed to have two actions and acts according to their respective environments at each step. The response $\tilde{\beta}^i(n)$ of the pair $A_i - E_i$ becomes the input for the other automaton.

4.2.2 Sequential Models

In the sequential model, only one automaton acts at any one time, and the action chosen determines which automaton will act in the next step. Such a sequential model can be viewed as a network of automata in which control passes from one automaton to another. At each step, one decision maker controls the state transitions of the decision process, and the goal of the whole group is to optimize the overall performance criterion.

There are three main structures in the sequential model: the tree structure, the directed network, and the general network.

4.2.2.1 Tree Structure

Figure 4-4 shows a tree structure of LA. In this model, the essential objective is to choose the best action at the last step. The LA A_0 chooses an action that determines which automaton will act, which in turn decides which of the successive automata will act on the lower levels.

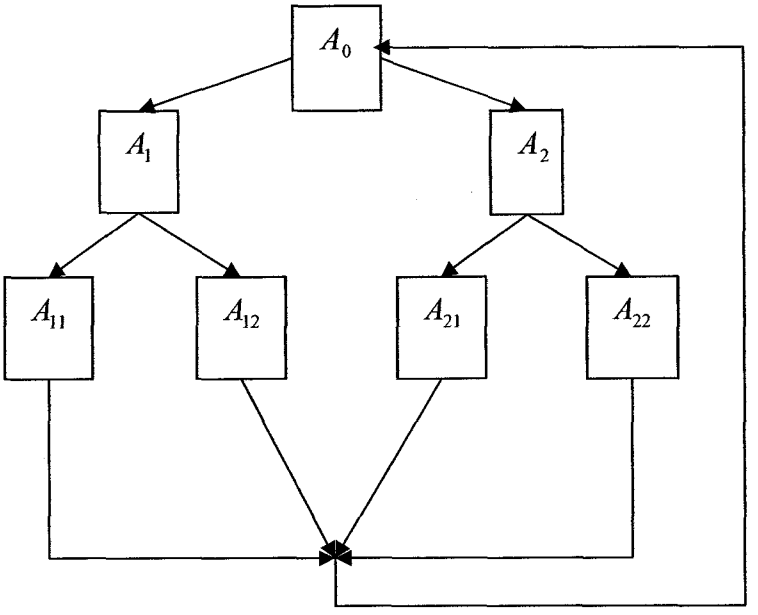

Figure 4-4:A Tree Structure Involving Interconnected LA

The tree structure is used mainly to improve the speed of the search procedure. In this Thesis, every automaton in a path updates its probabilities only after a complete path has been established.

4.2.2.2 Directed Network

Figure 4-5 shows a directed network. In this model, every automaton at any level can choose any of the automata on the next level.

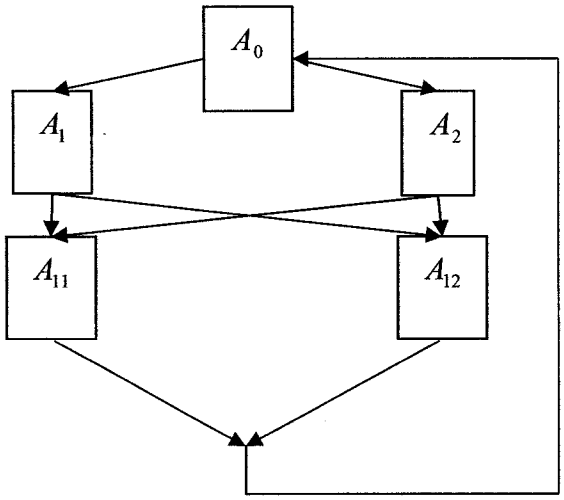

Figure 4-5: A Directed Network of Interconnected LA

4.2.2.3 General Network

Figure 4-6 shows a general network. In this model, any automaton can choose any other automaton in the entire network. There is no ordering of an action sequence giving rise to a cycle or path through the network.

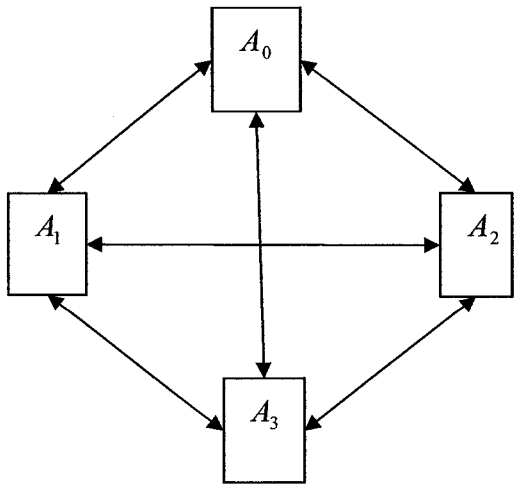

Figure 4-6: General Network

4.3 Three IPD Players Using Interconnected LA

In this Section, we will present the three players, who are based on the concept of Interconnected LA.

4.3.1 The Hierarchical Player

As stated, playing the IPD game with a single fixed strategy is always a possibility, but is not necessarily the optimal solution. The optimal strategy against different opponents usually differs. A single fixed strategy will likely be even more inefficient if the opponent changes strategy during the play. To handle these problems, Henning Hetland and Tor-Øyvind Lohne Eriksen proposed a hierarchical player based on the concept of interconnected LA [25]. The idea is to have a player who chooses an optimal strategy against the opponent's original strategy, but if necessary, alters the optimal strategy if the strategy of the opponent changes. In the ideal situation, if the Hierarchical player can immediately detect the switching of the opponent's strategy, he /she will always play optimally against any opponent.

The Hierarchical player is organized in a tree structure. At the top of the hierarchy is a learning automaton, which acts as a coordinator, choosing between different strategies. At the bottom of the hierarchy, there can be either fixed strategies or learning automata, which are referenced as sub-players.

4.3.1.1 The Coordinator LA

Each action of the coordinator LA corresponds to one sub-player. Different reinforcement schemes can be used to update action probabilities. In a Reward Penalty

scheme, if the average penalty of a sub-player is below the penalty limit, the action of choosing this sub-player will be rewarded; otherwise, it will be punished. In the favourable response case, the Reward-Penalty scheme will increase the probability of the action, which chooses the current strategy, and decrease the probability of the action, which chooses other strategies. In the unfavourable response case, the Reward-Penalty scheme will increase the probability of choosing another strategy, and lower the probability of maintaining the current strategy.

4.3.1.1.1 The Penalty Limit

In the IPD game, each response from the environment can be perceived as a penalty since it corresponds to a certain sentence. In order to update the action probabilities, Hetland and Eriksen set a penalty limit to map the penalties from IPD over to rewards and penalties in the LA scheme. The penalty limit influences the learning rate since the learning rate is calculated from the average penalty. In their experiments [29], the penalty limit was set to 2.5.

4.3.1.1.2 The Learning Rate

In Kehagias' scheme, the learning rate is not fixed. It is variable depending on the opportunity cost of the differences S-P, P-R, R-T. In the Hierarchical player implementation, the learning rate is also variable and is determined by the difference between the average penalty and the penalty limit, which is associated with the payoff matrix. So, the learning capabilities of these players depend on the penalty limit and vary with the payoff matrices.

4.3.1.1.3 The Probability Updating Space

If the coordinator updates the action probabilities and selects sub-players after each iteration, the Hierarchical player will converge toward a totally random strategy. If each sub-player plays too many iterations, the learning rate of the Hierarchical player will be slowed down. To avoid such situations, each sub-player plays 5 iterations before the coordinator updates the action probabilities. This results in the Hierarchical player converging on the same strategy after several iterations and achieving a fairly quick learning rate.

4.3.1.1.4 The Implementation of the Coordinator LA

During the initialization, the action probabilities - according to which the coordinator makes the selections - are set equal. The chosen sub-player will then play a set amount of iterations against the opponent. Based on the strategy of the sub-player and the opponent's actions, the penalty will be calculated after the set of iterations. The average penalty during the set of iterations is then compared to the penalty limit to determine whether the response from the environment is favorable or not. After updating the action probabilities, the coordinator will choose a new sub-player based on the new action probabilities.

4.3.1.2 The Sub-Players

The Hierarchical player has no limit as to the amount of sub-players in which to choose from. However, the more sub-players there are, the longer the time it takes to find the optimal one, since each sub-player has to be tested. As well, each sub-player could be

either a fixed strategy player or a learning player. Each sub-player initially has an equal probability of being chosen, and the sum of all the probabilities of choosing a sub-player is equal to unity. Because the choice of a sub-player has a probability of one, the Hierarchical player converges toward a single strategy.

4.3.2 Stochastic Learning Automata with States of History (SLASH) Player

A Stochastic Learning Automaton with States of History (SLASH) is an efficient player, based on interconnected automata with a history stack [25]. The history stack makes it possible to outsmart simpler strategies, such as some fixed-structure automata. For example, if a player is playing against Tit for Two Tat, the best strategy would be to alternate between cooperation and defection. A player with states of history can recognize and exploit this pattern. One state should always defect while another should always cooperate according to the previous steps.

4.3.2.1 The Structure of SLASH

A SLASH player consists of various states that represent the former steps. This concept is based on a sequential model in a general network structure, where one automaton (state) may lead to any of the other automata.

The number of states depends on how many steps are in the history stack. Each step is represented by two capital letters which designate the actions of the two players. For example 'CD' signifies that the player chose to cooperate in its last play, and the opponent chose to defect. The number of states, namely the number of LA, can be calculated as follows:

$$Number\ of\ states = (Number\ of\ steps * 2)^2$$

4.3.2.2 The Learning Scheme

The L_{RI} scheme is used in the LA for each state. With respect to updating the action probability, the LA evaluates the rewards of an action that the player will gain when the player returns to the same state. The average penalty during these iterations will be used to calculate the volume of reward. In this scheme, the penalty limit is set to be equal to the highest penalty (S). Thus, the volume of reward (γ) is equal to the difference of the penalty limit and the average penalty. The action probability updating formulas are as shown:

$p(t+1) = p(t) + (1 - p(t) \times a$ Increase in the probability of defection

$p(t+1) = (1-a) \times p(t)$ Increase in the probability of cooperation,

where $p(t)$ is the probability of defection at time t, and a is the learning rate.

The learning rate is defined by the formula:

$a = 0.01 \times \gamma$,

where γ is the volume of the reward.

4.3.2.3 The Learning Process

To demonstrate the learning process, an example of a SLASH player with 16 states is presented below.

A history length of 2 will result in 16 states, each of which is represented by a four-letter string, such as "CCDC". The first letter describes the action that SLASH had completed two steps ago. The second letter describes the action that the opponent had completed two steps ago. The third and fourth letter describes the former actions of the two players.

Figure 4-7 shows the states and the movement from one state to the other. Each state uses the L_{RI} scheme to update its action probabilities. In this example, the player starts in state 0. When it returns state to 0, the average penalty is calculated. The volume of reward can be calculated based on the penalty limit. Therefore, the LA in state 0 will update its action probabilities according to the L_{RI} scheme described above.

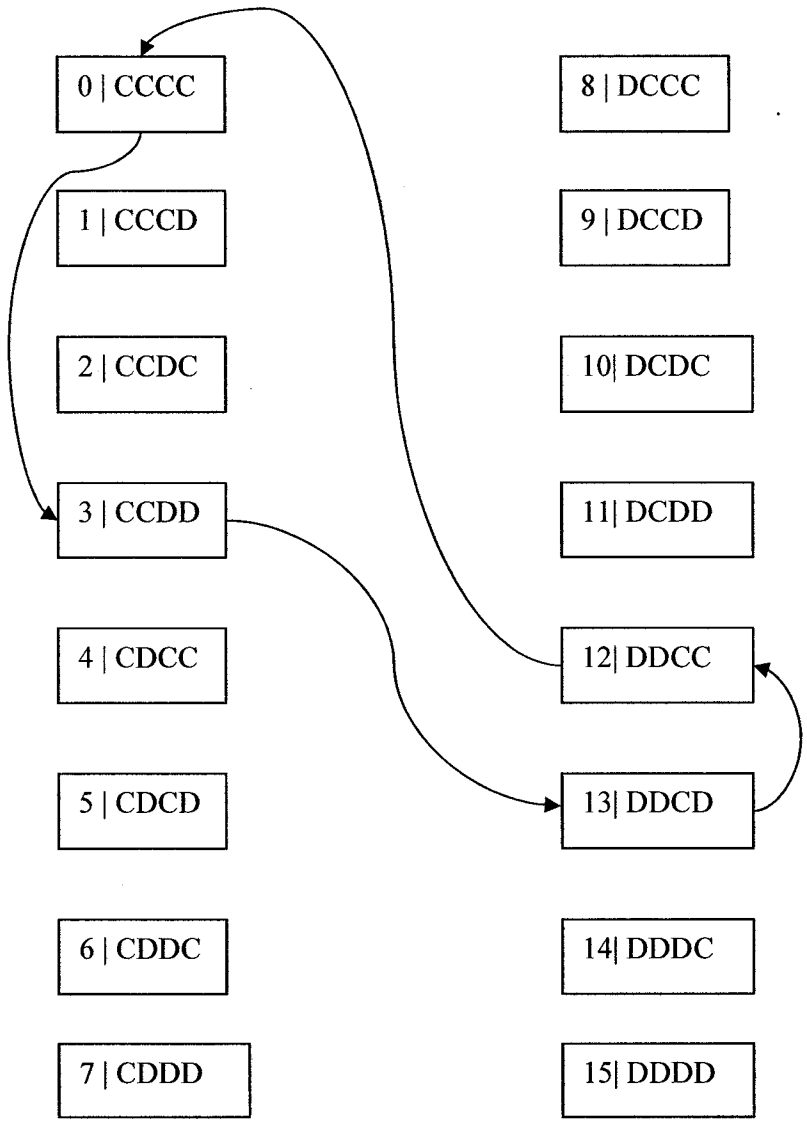

Figure 4-7: The States and the Movement from One State to the Other

4.3.3 Interconnected SLASH (I-SLASH) Player

In fact, the I-SLASH player is a Hierarchical player which consists of at least three SLASH players. In the hierarchy structure, the coordinator is a 16-state SLASH player whose task it is to predict the opponent's action. If the coordinator predicts that the opponent will cooperate, it will let the first automaton play; similarly, if it predicts that the opponent will defect, it will let the other automaton play. If the coordinator predicts correctly it will be rewarded and if it predicts mistakenly it will be punished. Since the Environment knows exactly what action is considered correct, this learning is called supervised learning. With a correct response from the Environment instead of a graded feedback, the coordinator in I-SLASH can increase its learning rate. This gives faster update frequencies and more accurate predictions, especially against fixed-structure automata.

The two sub-players, which are specialists on either a cooperative or defective opponent, have only 4 states. Tests show that this configuration is able to learn up to the same complexity level as a 16-state- SLASH player. Furthermore, 4-state sub-SLASH players make the learning process faster than in a traditional SLASH player.

4.3.4 Discussion of the Converge Behaviour of Hierarchical and SLASH/I-SLASH Strategies

Although the Hierarchical player may include sub-players which perform optimally against each opponent's strategies in the Nonstationary Environment, when the opponent's strategy switches, the coordinator will take considerable time to detect the switches and then change its strategy. It also should be noted that the probabilities of

choosing sub-players are updated only after a certain number of iterations, which prolongs the learning process. Furthermore, the Hierarchical player may converge, but the convergence depends not only on the opponent's strategies in the Environment, but also on the component strategies themselves. Theoretically, the Hierarchical player can contain as many sub players as possible, but this would largely prolong the learning process and lead to low efficiency.

Although, in theory, the Hierarchical player can perform optimally in a Nonstationary Environment, the unavoidable issues mentioned above might lead to an unstable performance and low efficiency in practice.

The SLASH/I-SLASH players will take time to learn in the beginning, and after enough iterations, they are able to handle a switch of the opponent's strategy. But, during the period where detecting the switch and adjusting behaviour occurs, the penalty tends to increase.

4.4 Conclusions

In this Chapter, we reviewed interconnected LA structures and presented three approaches for solving the IPD game. Based on these approaches, in the next Chapter, we will propose our new approaches which perform better in Nonstationary Environment.

Chapter 5

NEW APPROACHES FOR THE IPD

In this Chapter, we will discuss the potential factors which may affect the convergence behaviour of learning strategies, and then propose three new approaches which exhibit outstanding achievements in playing the IPD game in Nonstationary Environments.

5.1 Generalization of Learning Strategies

It seems to be impossible to have any one strategy which can perform optimally at all times, while competing with any other opponent strategies. However, there are some aspects of learning strategies, such as the learning scheme and the structure of the automaton, which may have impacts on whether a strategy can attain to a better payoff in the IPD game, especially in Nonstationary Environments. In this Section, we will discuss the role of the learning scheme and structure of the automaton in the learning strategy for the IPD game.

5.1.1 The Payoff Matrix

According to Nachbar [26] and Fogel [27], payoff setting has considerable effects on the converge behaviour of the learning strategy. When the payoff of mutual defection is close to the payoff of mutual cooperation, the learning strategy will approach that of pure defection. For example, while the Kehagias' L_{RP} strategy competes with the Cooperate-All strategy, in the case of the payoff matrix {5,3,1,0}, the Reward Penalty strategy will converge to pure cooperation. However, in the case of the payoff matrix{5,1.5,1,0}, the Reward Penalty strategy will converge to pure defection [3]. In this Thesis, we are not going to study the conditions for which the learning strategy will converge to cooperation or defection, but rather we will put our emphasis upon designing an efficient strategy that will perform better than other strategies in such Nonstationary Environments.

5.1.2 The Learning Scheme

In the previous Chapters, we presented Kehagias' learning strategy and the Hierarchical strategy. Both of these are based on the L_{RP} scheme, the L_{RI} scheme, and the SLASH/ISLASH strategies which implement the L_{RI} scheme for the learning process. However, implementing such schemes with different mechanisms indicates that these learning strategies exhibit dissimilar convergence phenomena.

- In the Kehagias L_{RP} strategy, since mutual defection is treated as a penalty, the player with a Reward-Penalty strategy will never move toward a full defection. Even if the player competes with a Defect-All player, it will do some cooperation and thus obtain a Sucker's payoff.
- In the Hierarchical strategy, in certain experiments, a penalty limit was set to

have a value equal to the average of the Temptation payoff and the Sucker's payoff. Compared with the penalty limit, the feedback from the Environment can be determined to be either a reward or a penalty. Although the Hierarchical strategy does not value cooperation any more, there still are chances to converge to a full cooperation sub strategy while competing with the Cooperate-All strategy if the initial probability of cooperation is large enough.

- In the L_{RI} strategy, the learning scheme ignores the penalty and thus updates the probability of cooperation only after the player defects alone or both players cooperate. Although this scheme can converge not only to pure defection but also to pure cooperation, the learning process will be decelerated since only rewards are considered in the play. Furthermore, in a Nonstationary Environment, while the L_{RI} strategy converges to pure cooperation, if the opponent switches to pure defection, the Reward-Inaction player would suffer the Sucker's payoff without learning anything.
- In the SLASH/ISLASH strategy, each automaton in the interconnected structure implements the L_{RI} scheme, and each feedback from the Environment is treated as a reward. The learning scheme evaluates the intensity of the reward by calculating the difference of the Sucker's payoff and the feedback from the Environment. As a result of this, any feedback except the Sucker's payoff will lead to updating the action probabilities. Therefore, theoretically speaking, this strategy can detect the opponent's behaviour rule in less time than the Reward-Inaction strategy although they are both based on the L_{RI} scheme. It is also to be noted that because any feedback is recognized as a reward, the way to the

optimal behaviour may be prolonged.

5.1.3 The Structure of LA

According to the number of LA interacting in the learning strategies, we classify the learning strategies as being a single LA-learning strategy or an interconnected-LA learning strategy.

5.1.3.1 The Single LA Learning Strategy

While playing an IPD game in a stationary Environment, a single LA learning strategy updates the action probability in the learning process and performs actions according to the action probability so as to have a better payoff. After a number of iterations, the action probability will converge to a certain value. For the same reason, in a Nonstationary Environment which consists of two distinct opponent strategies, the action probability will continuously adjust between two certain converging values. Thus, the inherent characteristic of the single LA learning strategy makes it difficult to play optimally for each opponent's strategy in the Nonstationary Environment.

5.1.3.2 The Interconnected LA Learning Strategy

In playing the IPD game, the learning player performs according to the action probabilities, which are achieved based on the previous experience in the play. In other words, the player selects an action following the rule that the action chosen can favor the consequences while competing against a possible choice made by the opponent. Thus, if the player can exploit patterns of the opponent's behaviour and the action probabilities

are based on the patterns respectively, the action chosen will be more accurate for favoring the consequences.

By implementing the interconnected LA in playing the IPD game, each state of the strategy is represented by an LA. Although it will take a longer time for each LA to converge than the single LA strategy in the beginning, the interconnected LA learning strategy can detect the switches of the opponent's strategies by jumping from one state to another, but not by continuously changing the action probability. Figure 5-1 demonstrates the behaviour of the single LA learning strategy and the interconnected LA learning strategy in such Nonstationary Environments.

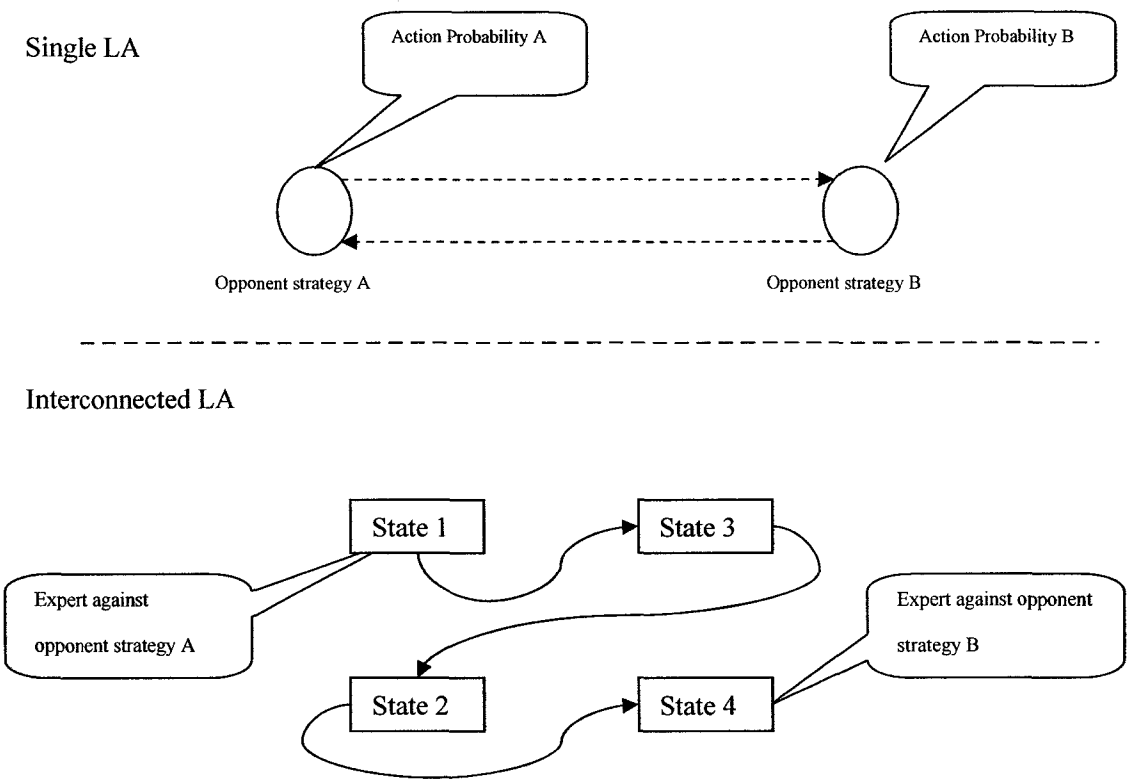

Figure 5-1: Behaviour of the Single LA Strategy and the Interconnected LA strategy in Nonstationary Environments

It is clear that the interconnected LA structure can detect switches of the opponent strategies more rapidly than the single LA. Furthermore, based on the interconnected LA structure, a more efficient learning scheme may accelerate the learning process and thus improve the performance of the learning strategy. So, we will introduce a family of schemes which exhibits fast convergence, and propose three new approaches which are based on it and the concept of the structure of interconnected LA. We will also implement them and study their behaviour in playing the IPD game in Nonstationary Environments.

5.2 The Pursuit Learning Scheme

The fastest LA algorithm currently available falls in the family of estimator algorithms which were first introduced by Thathachar and Sastry [4] [28]. The Pursuit algorithm is a branch of this family. It pursues the action which is currently estimated to be the most optimal. In this section, we will present two important pursuit schemes, which are the CP_{RP} scheme and the DP_{RI} scheme

5.2.1 The Continuous Pursuit Reward – Penalty (CP_{RP}) Scheme

The CP_{RP} scheme is similar to the L_{RP} scheme in the sense that both schemes update the action probabilities for both a favorable and unfavorable response from the Environment. The difference between them is that the CP_{RP} scheme is a pursuit scheme that pursues the action which has the highest reward estimate. However, the L_{RP} scheme updates the action probabilities in the direction of the action which obtains the reward or in the direction of all the actions that are not penalized.

5.2.1.1 Steps in the CP_{RP} scheme

The CP_{RP} scheme involves three steps [28]:

- In the first step, the LA chooses an action $\alpha(t)$ according to the probability distribution $P(t)$.
- In the second step, the CP_{RP} scheme does not adjust the action probabilities like the L_{RP} scheme (which updates the probability of the action according to whether the response from the Environment is favorable or not), but rather increases the probability of the action whose reward estimate is maximal, and decreases the probability of all other actions. The probability updating formula can be expressed as follows:

$$P(t+1) = (1-\lambda)P(t) + \lambda E_m$$

where E_m is the unit vector $[0...1...0]^T$ with the position of unity which represents the action with the maximal reward estimate. This equation shows that the action probability vector $P(t)$ is moved in the direction of the action with the current maximal reward estimate.

- The last step is to update the running estimates for the action which is rewarded in the second step. $D(t)$ is a vector which represents the reward estimates for each action. The scheme involves two more vectors, $W(t)$ and $Z(t)$, where $z_i(t)$ is the number of times the i^{th} action has been chosen, and $w_i(t)$ is the number of times the action α_i has been rewarded.

5.2.1.2 The Initialization and Estimate Updating Formula

The action probability $p_i(t)$ is set to *1/r,* where *r* represents the number of the action. The vector $D(t)$ is initialized by choosing each action a small number of times.

The estimate for the chosen action should be updated in each round. The formula for updating the vector $D(t)$ is as below:

$$w_i(t+1) = w_i(t) + (1-\beta(t))$$

$$z_i(t+1) = z_i(t) + 1$$

$$d_i(t+1) = \frac{w_i(t+1)}{z_i(t+1)}$$

where $\beta(t)$ = 0 if the chosen action was rewarded, otherwise $\beta(t)$=1

The CP_{RP} scheme was proved to be $\varepsilon - optimal$ in every stationary random environment by Thathachar and Sastry [19]. The proof is not included here.

5.2.2 The Discretized Pursuit Reward – Inaction (DP_{RI}) Scheme

In 1990, Oommen and Lanctot introduced the first discretized pursuit estimator algorithm which is denoted by DP_{RI} [4]. The CP_{RP} scheme is similar to the DP_{RI} scheme. Just as its name implies, this pursuit scheme updates the action probabilities only if the Environment rewards the chosen action, and it also changes the action probabilities in discrete steps.

The DP_{RI} scheme was also proved to converge and to be $\varepsilon - optimal$ in every stationary random environment by Oommen and Lanctot.

In the next Section, we will propose three new approaches based on the pursuit scheme and the interconnected LA structure. We also will study their behaviour while playing the IPD game in a Nonstationary Environment.

5.3 The Pursuit Interconnected LA Approaches

As stated in the previous section, the L_{RI} scheme in the SLASH/I-SLASH strategies may prolong the leaning process in playing the IPD game. In the same way, moving to the LA which can perform optimally with regard to an opponent strategy in a Nonstationary Environment will also be delayed. Based on different reward recognition mechanisms, we propose three approaches which integrate the benefit of the pursuit scheme and interconnected LA structure. In the following sections, we will introduce the reward recognition mechanisms and the three approaches in detail.

5.3.1 The Reward Recognition Mechanisms

Since the reward recognition mechanisms play a very important role in the convergence behaviour of the learning strategies in playing the IPD game, we will introduce the concept of immediate and long term rewards, and will propose two ways to recognize the reward of the strategy in the game.

5.3.1.1 The Immediate Reward

In the previous schemes, some have tried to learn the opponent's behaviour based on the immediate reward and use its knowledge for future actions. This kind of player

seems to be shortsighted because the immediate reward does not necessarily benefit the overall score. Any "vicious" actions may be repaid by the opponent in the future.

5.3.1.2 The Long Term Reward

Some schemes update the action probabilities based on the long term rewards. They evaluate the average penalty in several iterations. In this case, although the learning process may take a longer time, the achievement will be closer to the optimal in the long run. Thus, the overall performance of the player averaged over the whole game will be increased, especially in the Nonstationary Environment in which the opponent switches its strategies.

5.3.1.3 The Reward Recognition

Some schemes reward the action in different situations and thus value cooperation but punish the mutual defection. Other schemes determine whether the response from the Environment is a reward or not by comparing the payoff achieved with a fixed penalty limit. In this Chapter, we introduce two new approaches which are based on the past experience in playing the IPD game. They are as shown below:

- Dynamic Average Penalty Limit

  The penalty limit is initialized to be the average of the Temptation payoff and the Sucker's payoff at the start of the game. After a small number of iterations, the penalty limit is updated dynamically to the average of the payoffs in the past iterations. By comparison with the Dynamic Average Penalty Limit, the response from the Environment is determined to decide whether this is to be perceived as a reward of not. In this way, the actions which achieve a lower

penalty than the overall average penalty will be rewarded, and thus the behaviour of the learning strategy will pursue to lower the overall average penalty.

- Preceding Action Average Penalty Limit

  Similar to the Dynamic Average Penalty Limit approach, the penalty limit is initialized to be the average of the Temptation Payoff and the Sucker's Payoff at the start of the game. After a small number of iterations, the penalty limit is updated to be the average of the payoffs which the preceding action achieved. So, the payoff of each action would be compared with the average penalty limit of its preceding action. In this way, the successive actions which gained fewer penalties would be rewarded, and thus the action would pursue to a solution so as to lower the average penalty.

5.3.2 The Interconnected LA with the Immediate Pursuit Scheme (ILIP)

If the player can exploit the strategy of the opponent as time proceeds and performs the optimal actions for the rest of the game, the overall gain of the player may increase. Thus, accelerating the convergence of the LA is a possible way to improve the achievements of the player in playing the IPD game. The Interconnected LA withthe Immediate Pursuit Scheme (ILIP) is an efficient strategy, which is based on the interconnected LA structure. To accelerate the learning speed in pursuing the optimal behaviour, each LA in the structure adopts the CP_{RP} scheme with an immediate reward.

5.3.2.1 The Structure of the Interconnected LA

The structure of the Interconnected LA is similar to the LA structure in the SLASH strategy. The LA in the ILIP are interconnected as a general network and each LA represents a state/pattern. Each state/pattern is a potential combination of interactive actions between two players in a certain number of iterations, but not a history stack in the SLASH strategy. In our implementation, we consider the possible combinations of interactive actions in two iterations.

5.3.2.2 The Learning Scheme

The CP_{RP} scheme with immediate reward is used in each LA for updating the action probability. The LA evaluates the immediate reward estimate for each chosen action and updates the action probabilities to favor the action that has the highest reward estimate.

The CP_{RP} scheme with an immediate reward involves four steps as below:

Initialization: To let each action have the same reward estimate at the start of the IPD game, we set $z_{ic}(0)$ and $z_{iD}(0)$ to 2, $w_{ic}(0)$ and $w_{iD}(0)$ to 1. Here, $z_{ic}(0)$ and $z_{iD}(0)$ are the number of times the action has been chosen up to times 0; $w_{ic}(0)$ and $w_{iD}(0)$ are the number of times the chosen action has been rewarded up to time 0, and thus the reward estimate vector $d_i(0)$ is set to $\{0.5,0.5\}$, i denotes the *ith* state. The action probability for each LA is also set to 0.5.

Update action probability: For the LA in each state, choose the action based on the action probability vector. If the chosen action has the current highest reward estimate, update the cooperation probability $p_i(t)$ as:

$p_i(t+1) = p_i(t) + (1 - p_i(t) \times a$ if the chosen action is cooperative

$p_i(t+1) = (1-a) \times p_i(t)$ if the chosen action is defective

Otherwise, update the cooperation probability $p(t)$ using:

$p_i(t+1) = (1-a) \times p_i(t)$ if the chosen action is cooperative

$p_i(t+1) = p_i(t) + (1 - p_i(t) \times a$ if the chosen action is defective

where a is the learning rate defined by formula:

$a = 0.01 \times \gamma$, and γ is the volume of the reward the current action has gained.

Update the reward estimate vector: The immediate reward has different levels of intensity during the game. Due to this, we will update $d_{ij}(t)$ according to the following equations:

$$w_{ij}(t+1) = w_{ij}(t) + \beta(t)$$

$$z_{ij}(t+1) = z_{ij}(t) + \beta(t)$$

$$d_{ij}(t+1) = \frac{w_{ij}(t+1)}{z_{ij}(t+1)}$$

where $\beta(t) = \gamma / s$, and s denotes the Sucker's payoff in the IPD game, $j \in \{C, D\}$, $0 \le i \le 15$

Move to next state: Based on the interactive actions in previous iterations, the corresponding LA takes the control and repeats the above two steps.

The ILIP strategy exhibits favorable performance in playing the IPD game in the Nonstationary Environment, especially when the Environment consists of fixed strategies.

5.3.3 The Interconnected LA with the Dynamic penalty limit Pursuit Scheme (ILDP)

In playing the IPD game, performing "vicious" actions by simply maximizing its immediate rewards may be repaid by the opponent in the future. So, the ILDP approach evaluates the average payoff in which an action is gained during the trip which starts at one state and ends in the same state. To determine the average payoff, whether it is a reward or not, we set a dynamic penalty limit by calculating the average penalty during a certain number of iterations. In our implementation, we compute the average penalty of the previous 50 iterations.

The structure of LA in the ILDP approach is the same as that of the one in the ILIP approach. The learning scheme also involves four steps as below:

Initialization: Same as in the ILIP approach except we initialize the average penalty by the average of the Temptation payoff and the Sucker's payoff.

Update action probability: Same as in the ILIP approach.

Update the reward estimate vector: Here, we compare the average payoff gained in the round trip with the average penalty in the previous k (say, 50) iterations. If the average payoff is less than the average penalty, the LA will be rewarded, otherwise it will be punished. In this regard, we will update $d_{ij}(t)$ according to the following equations

$$w_{ij}(t+1) = w_{ij}(t) + (1 - \beta(t))$$

$$z_{ij}(t+1) = z_{ij}(t) + 1$$

$$d_{ij}(t+1) = \frac{w_{ij}(t+1)}{z_{ij}(t+1)}$$

where $\beta(t) = 0$ if the action is rewarded; otherwise it is set equal to 1, $j \in \{C, D\}$

Move to next state: Same as in the ILIP approach.

5.3.4 The Interconnected LA with the Preceding Penalty Limit Pursuit Scheme (IPPP)

The Interconnected LA with the Preceding Penalty Limit Pursuit Scheme is the same as the ILDP approach except for the penalty limit, which is used to determine whether the average payoff is a reward or not. Just as its name implies, the penalty limit is set to the average penalty that the previous action achieved in the same state. As stated in the previous section, the successive actions which gained fewer penalties will be rewarded and thus the action will pursue paths which enable it to lower the average penalty in the long term.

5.4 Conclusions

In this Chapter, we discussed the potential factors which may affect the convergence behaviour of learning strategies, and then proposed three new approaches which exhibit outstanding performance in playing the IPD game in Nonstationary Environments. In the next Chapter, we will present the simulations, an analysis of the results and the conclusions we have made based on the comparison of the results achieved by the different strategies.

Chapter 6

EXPERIMENTS AND DISCUSSION

In this Chapter, we will present the results of the experiments performed to compare the effectiveness of the different proposed strategies in the Nonstationary Environments, and to also present the analysis and discussion of the simulation results.

6.1 The Experimental Goals

In Chapter 5, we introduced three new learning strategies for playing the IPD game in a Nonstationary Environment. In this Chapter, we will explain the conducted experiments and demonstrate exactly how beneficial the strategies are.

There are three goals in the experiments: First, to find out how well the three strategies compete with other strategies. Second, to determine how well the three strategies perform in a Nonstationary Environment. Third, by comparing these with the SLASH/I-SLASH strategies, we would like to know what new improvements will be obtained by the new approaches in different situations.

6.2 The Experimental Environment

The implementation of the experiments has been done in the JAVA programming language. There are three main modules in the implementation. The first module implements the strategies explained in Chapters 3 and 4. The second module designs the Nonstationary Environment. Lastly, the third module implements our new strategies, analyzes the experimental results, and compares the performance of our strategies with that of other strategies. Figure 6-1 shows the class diagram for the implementation.

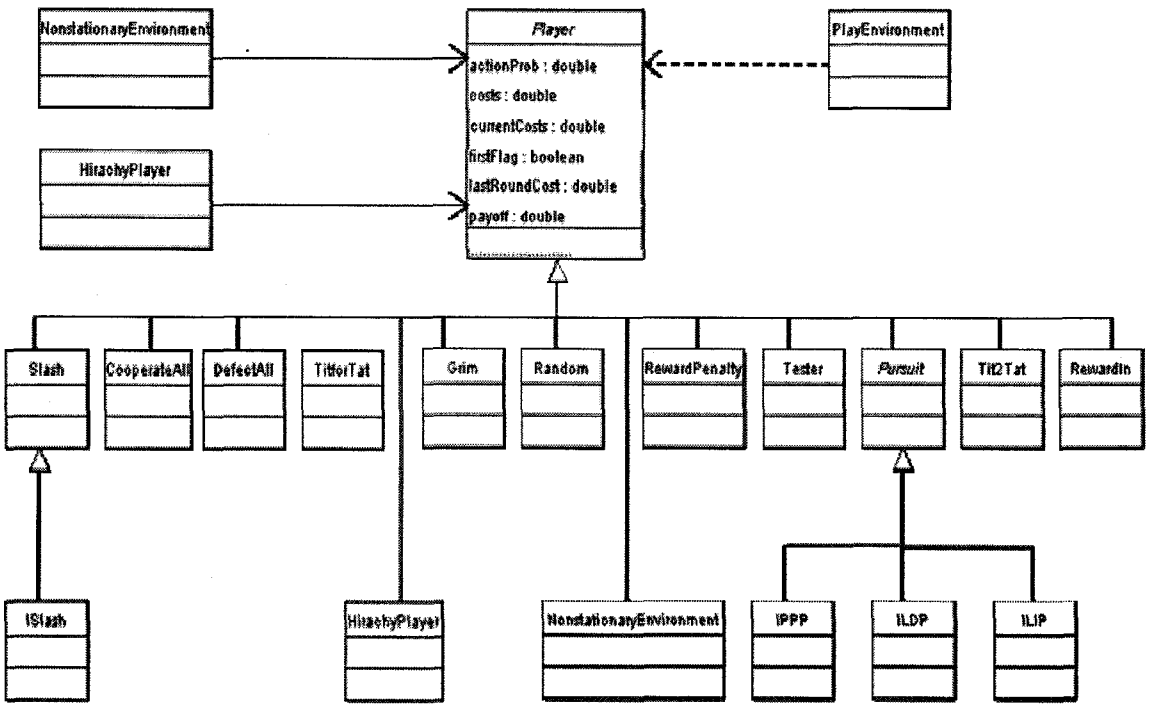

Figure 6-1 The class diagram

6.3 Experimental Results and Discussion

In this Section, we will present the experimental results of the different strategies used for playing the IPD game in different Nonstationary Environments, and also in

competing with other strategies. For each experiment, the IPD game is run 10 times for each, and for 10,000 iterations for each configuration. The payoff matrix for the IPD game is shown in Table 6-1 below, which is same as the benchmark configuration used by SLASH learning strategy in [25]:

B Actions A Actions	C	D
C	2,2	5,0
D	0,5	4,4

Table 6-1: Payoff for Prisoner's Dilemma

6.3.1 Playing IPD Game in Nonstationary Environment

The first question we want to answer is: "How well do the families of Pursuit schemes with interconnected LA structures play in a Nonstationary Environment?" As stated in Chapter 3, we constructed the Nonstationary Environment with two distinct strategies. In the experiments, we chose some classical strategies found in the famous Axelrod's Tournament as candidate opponent strategies for the Nonstationary Environment such as the Tit for Tat, Grim, Random, Cooperate-All, and Defect-All [10][11]. Each Nonstationary Environment switches opponent's strategies 30 times in a game. The distinct strategy combinations which are used to generate the Nonstationary Environment are as shown below:

A. Reward Inaction and Random

B. Grim and Cooperate-All

C. Cooperate-All and Defect-All.

D. Tit for Tat and Defect-All

E. Grim and Random

F. Tit for Tat and Cooperate-All

Furthermore, we also provide the simulation results of the SLASH/I-SLASH (which are very efficient strategies in Nonstationary Environments) as a benchmark for comparison. The average results are shown in Table 6-2

Environment / Player	Reward Inaction /Random	Grim /Cooperate-All	Coop All /Defect-All	TFT /Defect All	Grim /Random	TFT /Cooperate-All
IPPP	24172	20796	20677	30920 *	32538	14781 *
ILDP	23771	20617	31740	31776	30092	20012
ILIP	20218	20003	20075	39779	29797	19656
ISLASH	26879	20846	21160	33032	32403	16026
SLASH	25197	22009	20963	32790	33269	17267
TFT	27845	20000	29980	32376	28770	29960
Grim	20229	20000	20001	39425	29345	20000
Def All	20028 *	19920 *	20000 *	39920	29896	19920
Coop All	34817	20000	35000	35000	27479 *	20000
Random	28770	27379	27556	36231	36184	18761
Reward Inaction	24661	26010	32721	36714	33910	18933

Table 6-2: Performance of the Different Strategies in Nonstationary Environments

In the following Sections, we will discuss the performance of different strategies in Nonstationary Environments with their corresponding histograms.

- **Reward Inaction/Random in a Nonstationary Environment**

The histogram for comparing the total costs of different strategies in using Reward-Inaction/Random schemes in a Nonstationary Environment is shown in Figure 6-2. The Reward Inaction is a learning strategy. Its behaviour changes over time according to the actions of the opponent. Since this strategy never converges to pure defection, intuitively, the optimal strategy against the Reward Inaction strategy is to tempt it to cooperate. It does this by performing cooperation in the beginning and then by performing defection in the rest of game. It is difficult to predict the behaviour of the Random strategy because, by definition, its characteristic is uncertain in nature. But if a strategy always chooses to Defect, it can achieve a cost of 10,000 against the Random strategy. Figure 6-1 shows that the Defect-All gets the best results. The strategies of the pursuit scheme with interconnected LA structure IPPP, ILDP, ILIP are among the best strategies. They attained the cost of 24,172, 23,771, and 20,218 units respectively. Compared with the SLASH strategy, the IPPP strategy has an improvement of 4.20% in the incurred costs.

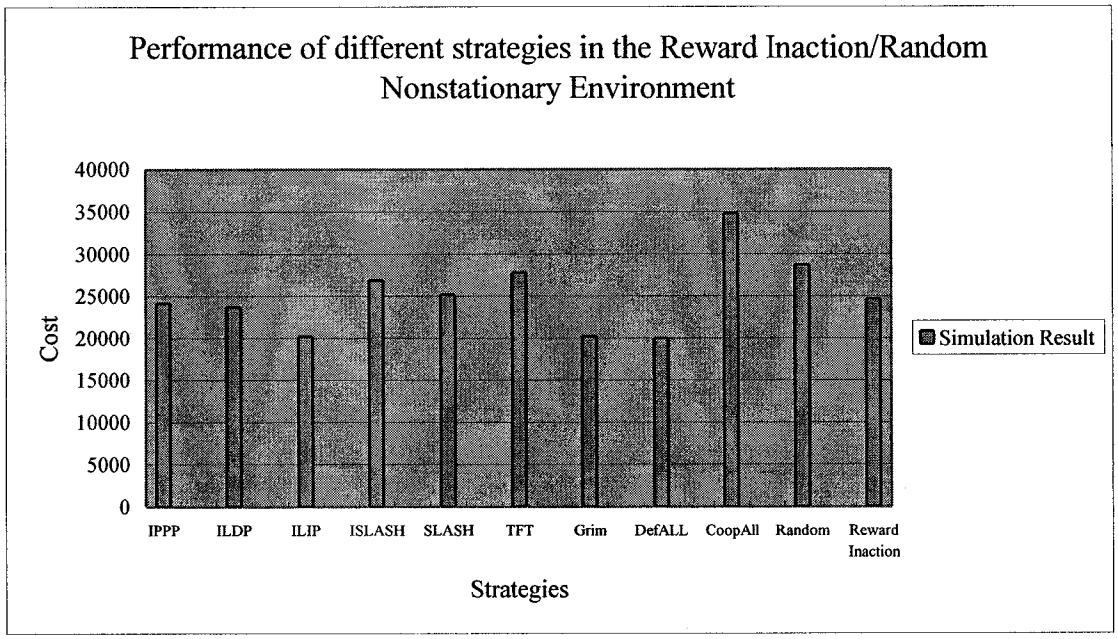

Figure 6-2: Comparing the Cost of Different Strategies in Reward Inaction/Random Nonstationary Environments

- **Grim/Cooperate-All in a Nonstationary Environment**

The histogram for comparing the total cost of different strategies in a Grim /Cooperate-All Nonstationary Environment is shown in Figure 6-3. As shown in the figure, the optimal behaviour is to defect all the time during the game. Although, the learning strategies IPPP, ILDP and ILIP did not get the optimal result, their achievements are very close to the optimal and they received a cost of 20,796, 20,617 and 20,078 units respectively. Compared with the cost that the I-SLASH strategy incurred, our approaches decreased the incurred cost by 0.02%, 1.1% and 4.2% respectively.

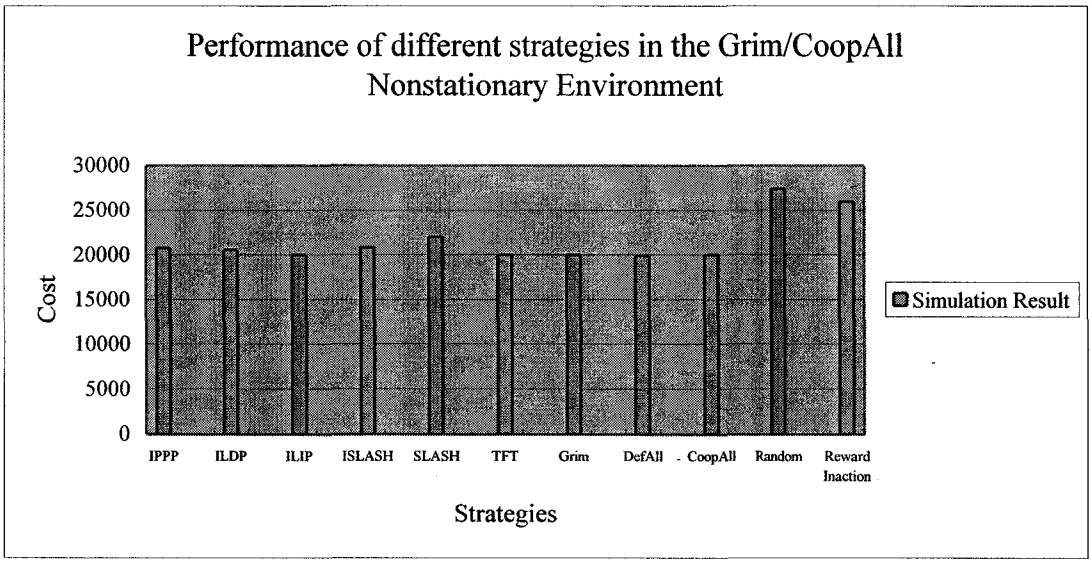

Figure 6-3: Comparing the Cost of Different Strategies in Grim/Cooperate-All Nonstationary Environments

- **Cooperate-All/Defect-All in a Nonstationary Environment**

The histogram for comparing the total cost of different strategies in Cooperate-All/Defect-All in a Nonstationary Environment is shown in Figure 6-4. Playing against Cooperate-All/Defect-All makes Defect-All the best strategy with a score of 20,000 units, whereas the worst strategy is the Cooperate-All strategy. The learning strategies also gained excellent results which are very close to the optimal result. Again, to answer the third question, the IPPP and ILIP pursuit learning strategies played better than the SLASH strategy by decreasing the total costs by 1.38% and 4.48% respectively.

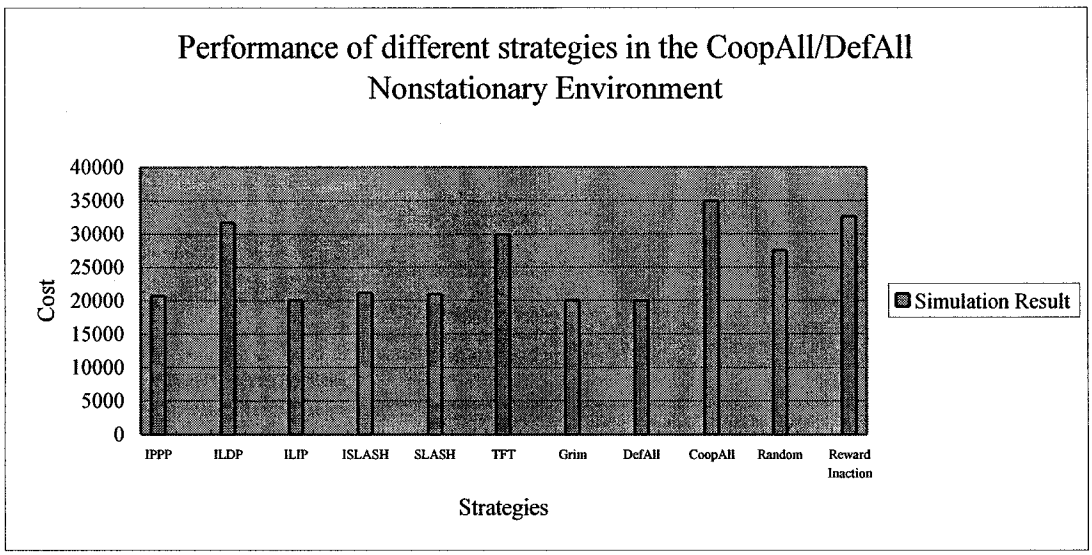

Figure 6-4: Comparing the Cost of Different Strategies in Cooperate-All/Defect-All Nonstationary Environments

- **TFT/Defect-All Nonstationary Environment**

The histogram for comparing the total cost of different strategies using the TFT /Defect-All Nonstationary Environment is shown in Figure 6-5. The optimal behaviour in this Nonstationary Environment is to perform cooperation when the Environment switches to the TFT strategy, but to perform defection when the Environment switches to the Defect-All strategy. The TFT strategy replicates the action of opponent's strategy, and thus can detect the switches of the opponent's strategies immediately. In the case of the Environment switching to the TFT strategy from the Defect-All opponent strategy, the TFT strategy, in both cases, will defect and cooperate with each other. Table 6-3 demonstrates the situation as below:

Environment	Defect-All			Tit For Tat		
	D	D	D	C	D	C
Objective Strategy TFT	D	D	D	D	C	D

Table 6-3: TFT in theTFT/Defect-All Nonstationary Environments

So, although the TFT strategy can change its behaviour to follow the switches of the opponent's strategies immediately, the IPPP and ILDP pursuit learning strategies achieved superior results than the TFT strategy. In this Environment, the IPPP strategy is the best strategy and played better than the SLASH strategy by lowering the total cost by 5.82%.

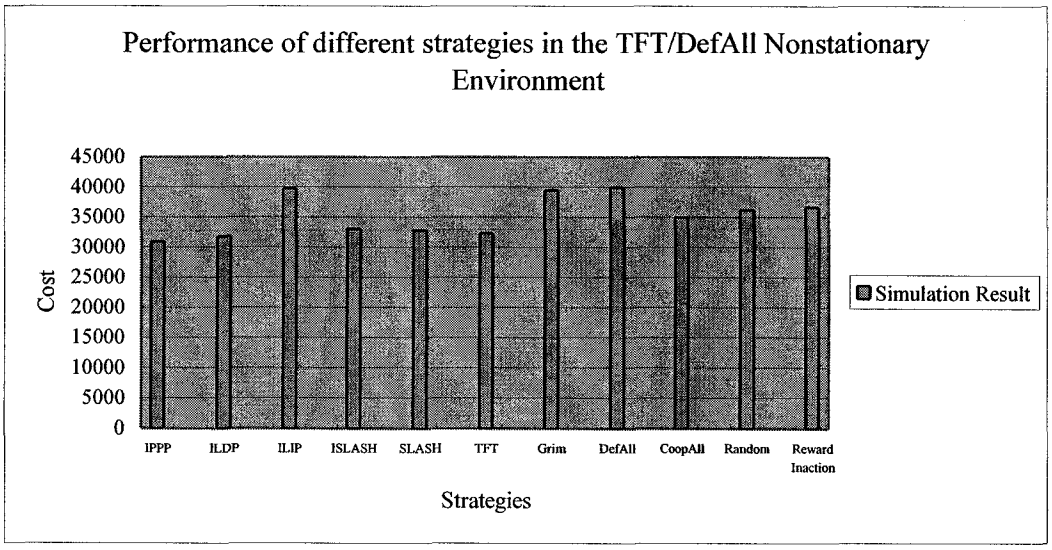

Figure 6-5: Comparing the cost of different strategies in TFT/Defect-All Nonstationary Environments

- **Grim/Random in a Nonstationary Environment**

The histogram for comparing the total cost of different strategies using the Grim /Random Nonstationary Environment is shown in Figure 6-6. Since the Grim strategy will perform a defective action every time once it encounters a vicious action, the

Cooperate-All strategy achieved the best result. The learning strategies such as IPPP, ILDP, ILIP, SLASH and I-SLASH are also among the good strategies. Furthermore, the IPPP, ILDP, and ILIP are slightly better than the SLASH player, by incurring a less cost of 2.07%.

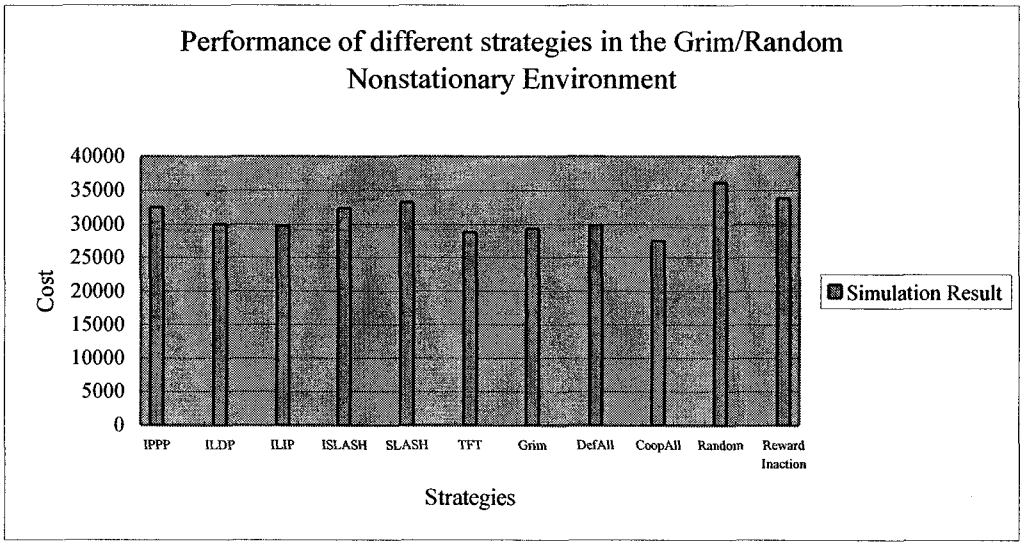

Figure 6-6: Comparing the Cost of Different Strategies in Grim/Random Nonstationary Environments

- **TFT/Cooperate-All in a Nonstationary Environment**

The histogram for comparing the total cost of different strategies using the TFT /Cooperate-All Nonstationary Environment is shown in Figure 6-7. In this Environment, the IPPP strategy achieved the best result. Compared with the I-SLASH strategy, the total costs in the game is decreased by 16.9%, which we believe is quite remarkable.

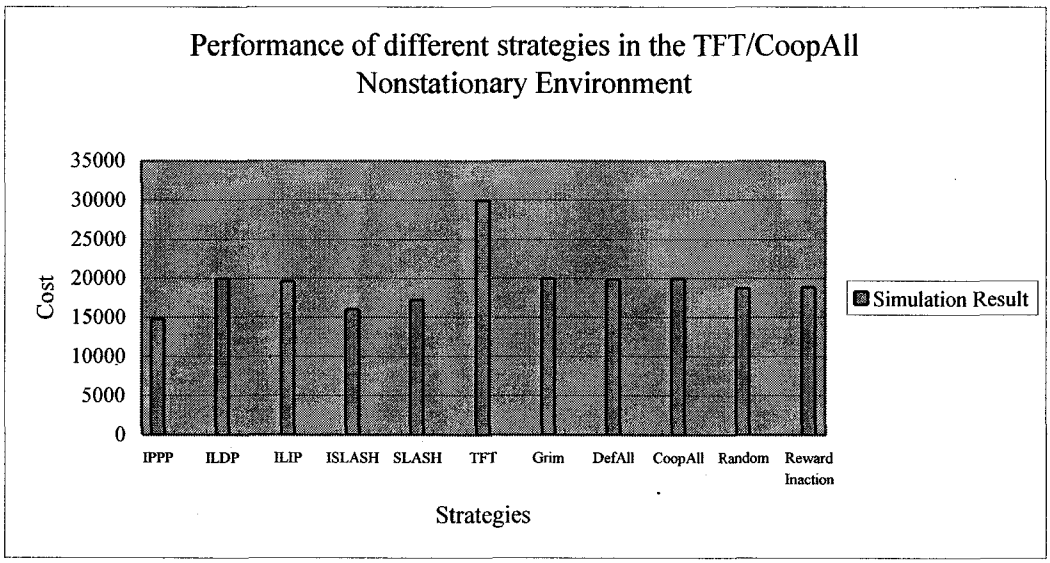

Figure 6-7: Comparing the Cost of Different Strategies in TFT/Cooperate-All Nonstationary Environments

After we tested various kinds of classical strategies in different Nonstationary Environments, we found that the SLASH /I-SLASH strategies yielded the best results as seen in [3]. From Table 6-2, we see that the new estimator-based approaches, which are based on the pursuit scheme and the interconnected LA structure, yielded the best results in playing the IPD game in the above Nonstationary Environments. The overall costs that the strategies achieved in all the Nonstationary Environments described above are shown below in Figure 6-8:

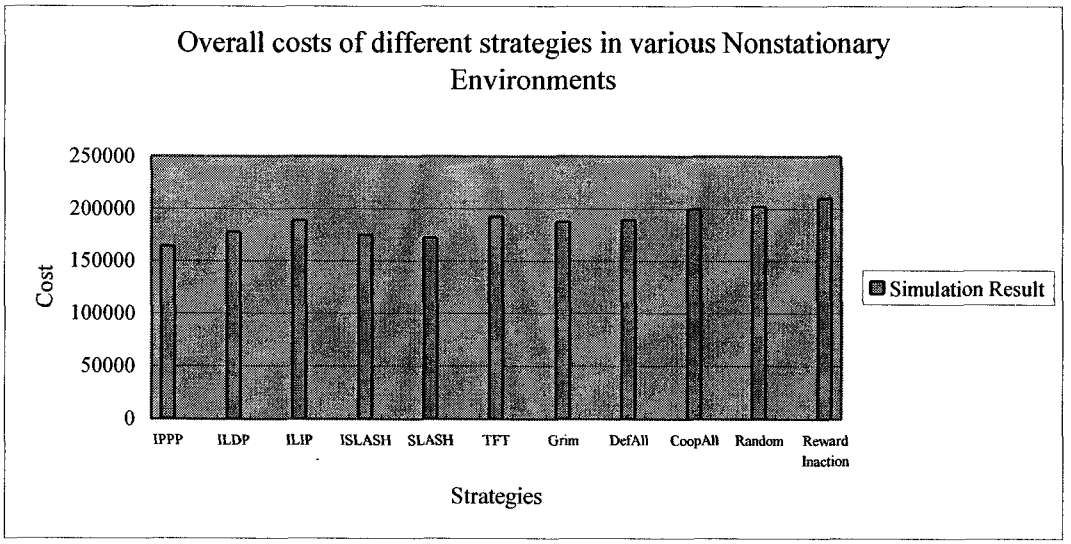

Figure 6-8: Comparing the Overall Costs of Different Strategies

Compared with the SLASH strategy, the IPPP strategy decreased the overall costs by 4.86%.

6.3.2 Playing IPD Games against Each Other

This experiment was designed to see how good the results would be whenever the new strategies competed with other strategies. The results are shown in Table 6-4:

Player / Opponent	IPPP	ILDP	ILIP	SLASH	TFT	Reward Inaction	Grim	Def All	Coop All	Random
IPPP		27591	40483	34657	20732	27387	40514	40364	898	24765
ILDP	30414		44700	30061 *	20012 *	35423	44997	41001	19992	24720
ILIP	37421	19955 *		38298	20074	21110 *	40035	40051 *	30 *	20163 *
ISLASH	30026	34206	40090	36533	23184	34336	40448	40732	776	27126
SLASH	30050	28507	40187		21099	37966	40586	40458	10128	25284
TFT	20868 *	20012	39800 *	32790		35767	20000 *	40001	20000	27537
Reward Inaction	37237	43175	40698	41234	35320	33910	44874	45088	19382	34306
Reward Penalty		23025	45605	35739	25875	35739	45865	45846	17012	28738

Table 6-4: The Results of Strategies Competing with each other

As seen from the Table 6-4, the new estimator-based approaches, which are based on the pursuit scheme and interconnected LA structure, achieved the best results in playing the IPD game against other strategies in most cases. When competing with the fixed strategy TFT, the IPPP, ILDP, and ILIP strategies received a cost of 20,732, 20,012 and 20,074 units respectively. Compared with the cost that the SLASH strategy incurred, they decreased the incurred cost by 1.73%, 5.15% and 4.86% respectively. When competing with the Def All strategy, the IPPP, ILDP and ILIP strategies received a cost of 40,364, 41,001, and 40,051 units respectively. Compared with the cost that the SLASH strategy

incurred, they decreased the incurred cost by 0.23%, -1.31%, and 10.1% respectively. It should be noted that the SLASH/I-SLASH strategies were very efficient and accurate for playing the IPD game. In some cases, the performances of the SLASH/I-SLASH strategies were very close to or even slightly better than the performance of the estimator-based strategies.

6.4 Conclusion

In this Chapter, we have tested the strategies of playing the IPD game in different Nonstationary Environments. We determined that our three initial questions were all answered positively. The simulation results show that the IPPP, ILDP, and ILIP estimator-based strategies performed very well in Nonstationary Environments, and that the IPPP strategy in particular, achieved a performance of 4.86% better than the overall costs of the SLASH/I-SLASH strategies. The final conclusion we have drawn is that in most cases the IPPP, ILDP, and ILIP strategies achieved the best results in playing the IPD game against the other strategies.

Chapter 7

CONCLUSION AND FUTURE WORK

Based on our proposed three new approaches and our simulation results, we now present the conclusions of our Thesis and describe the potential future work.

7.1 Conclusions

This Thesis proposed three new estimator-based strategies with interconnected LA structures for playing the IPD game in Nonstationary Environments. Furthermore, we compared the performance of the three estimator-based strategies with those of other strategies in Nonstationary Environments, and compared the performance of different strategies when competing with each other.

We proposed two reward reorganization mechanisms - the Dynamic Average Penalty Limit and Preceding Action Average Penalty Limit mechanisms - to determine whether a feedback is a reward or not. In this way, an unsupervised learning philosophy is effectively transformed to be a supervised learning method by virtue of the pursuit scheme. In the family of estimator-based strategies, the estimator-based strategy IPPP,

which adopts the Previous Reward Limit mechanism, is able to score fairly better than the learning strategies of the state-of-the-art. The IPPP strategy is not only able to converge quickly in each configuration of the LA, but also able to adjust its behaviour by jumping to other configuration LA when the opponent's strategy switches.

Compared with the SLASH strategy, the IPPP strategy decreased the incurred costs by 4.86% (on average) in the experiments.

7.2 Future Research

The learning automata theory is widely applied to a fairly broad class of problems; especially those that can not be represented by adequate mathematical models. As a "non-zero sum" game in areas such as economics, political science, evolutionary biology, and game theory, the IPD game has been discussed extensively. When we applied the theory of learning automata to design adaptive strategies for the IPD game, some issues that are stated below could be subject to future work:

- In our estimator-based strategies, we adopted the Dynamic Reward Limit and Previous Reward Limit reorganization mechanisms. It is also possible that there are more attractive mechanisms. For example, using a LA to adjust the reward limit to favor the learning process and the switches detection in the game.
- In our estimator-based strategies, we transformed the unsupervised learning to supervised learning by setting a reward limit. Any other approaches which can transform the unsupervised learning to the supervised learning are open to future work.

- The strategies we discussed in this Thesis concern the two players IPD game. It is also possible to design efficient estimator-based strategies for more practical problems, such as the multiplayer IPD game.
- In imperfect Nonstationary Environments, our estimator-based strategies may not be as good as in perfect Nonstationary Environments. Experiments of learning strategies in imperfect Nonstationary Environments are expected to be conducted in future work.
- Any work to "Fine-tune" or "near-optimize" the IPD to solve a specific problem.
- Any other approaches which can accelerate the learning process and improve the performance of learning strategies for playing the IPD game.

BIBLIOGRAPHY

[1] Kehagias, A., "Probabilistic Learning Automata and the Prisoner's Dilemma", *Journal of Liberal Arts*, Vol. 1; pp. 41-55, 1994.

[2] Narendra, K. S. and Thathachar M. A. L., *Learning Automata – An Introduction*, Prentice-Hall, 1989.

[3] Fu, K. S. and McMurtry G. J., "A Study of Stochastic Automata as Models of Adaptive and Learning Controllers", *Technical Report TR-EE 65-8*, Purdue University, Lafayette, IN, 1965.

[4] Oommen B. J. and Lanctot J. K., "Discretized Pursuit Learning Automata", *IEEE Transactions on Systems, Man, and Cybernetics*, July/Aug, Vol. 20; pp. 931-938, 1990.

[5] Wikipedia, "Zero-sum", *http://en.wikipedia.org/wiki/Zero-sum*, [18 May 2005].

[6] Kuhn, S., "Prisoner's Dilemma", *http://plato.stanford.edu/entries/prisoner-dilemma/*, [20 May 2005].

[7] Hardin, G., "The Tragedy of the Commons", *Science*, Vol. 162; pp. 1243-1248, 1968.

[8] Axelrod, R., *The Evolution of Cooperation*, New York: Basic Books, 1984.

[9] Axelrod, R. and Dion, D., "The Further Evolution of Cooperation". *Science*, Vol. 242; pp. 1385-1390, 1988.

[10] Axelrod, R., "Effective Choice in the Prisoner's Dilemma", *Journal of Conflict Resolution*, Vol. 24; pp. 3-25, 1980.

[11] Axelrod, R., "More Effective Choice in the Prisoner's Dilemma", *Journal of Conflict Resolution*, Vol. 24; pp. 379-403, 1980.

[12] Ünsal, C., "Intelligent Navigation of Autonomous Vehicles in an Automated Highway System: Learning Methods and Interacting Vehicles Approach", *Virginia Tech, http://www.cs.cmu.edu/~unsal/diss/etd*,1997.

[13] Tsetlin, M. L., "On the Behavior of Finite Automata in Random Media", *Automat Telemek*, Vol. 22; pp. 1345-1354, 1961.

[14] Varshavskii, V. I. and Vorontsova, I. P., "On the Behavior of Stochastic Automata with Variable Structure", *Automat. Telemek*, Vol. 24; pp. 327–333. 1963.

[15] Fu, K. S., "Stochastic Automata as Models of Learning Systems", *Computer and Information Sciences II*, New York: Academic press, 1967.

[16] Narendra, K. S., and Thathachar, M. A. L., "Learning Automata A Survey", *IEEE Transactions in Systems, Man and Cybernetics*, Vol. SMC-4; No. 4, July 1974.

[17] Oommen, B. J. and De St. Croix, T., "String Taxonomy using Learning Automata", *IEEE Transactions on Systems, Man and Cybernetics,* Vol. SMC-27; pp. 354-365, 1997.

[18] Oommen, B. J. and De St. Croix, T., "Graph Partitioning using Learning Automata", *IEEE Transactions on Computers,* Vol. 45; No. 2, pp. 195-208, 1995.

[19] Hiroyuki, U. and Fei, Q., "Network Load Balancing Algorithm using Ants Computing", *IEEE/WIC International Conference on Intelligent Agent Technology,* pp. 428 – 431, 13-16 Oct. 2003.

[20] Smith, R., Sola, M. and Spagnolo, F., "The Prisoners' Dilemma and Regime-Switching in the Greek-Turkish Arms Race", *Journal of peace research*, Vol. 37; part 6, pp. 737-750, 2000.

[21] Wikipedia, "Prisoner's Dilemma", *http://en.wikipedia.org/wiki/Prisoner's_dilemma*, [20 May 2005].

[22] Hingston, P. and Kendall, G., "Learning versus Evolution in Iterated Prisoner's Dilemma", *Proceedings of the 2004 Congress on Evolutionary Computation*, Vol. 1; pp. 364-372. IEEE, June 2004.

[23] Hirshleifer, J., and Coll, J. M., "What Strategies Can Support the Evolutionary Emergence of Cooperation?", *Journal of Conflict Resolution* Vol. 32(2); pp.367-398, 1988.

[24] Hoffmann, R., "Twenty Years on: The Evolution of Cooperation Revisited", *Journal of Artificial Societies and Social Simulation*, Vol. 3; No. 2, 2000.

[25] Hetland, H., and Eriksen, T. L., *Interconnected Learning Automata Playing Iterated Prisoner's Dilemma*, Masters' Thesis, Agder University College 2004.

[26] Nachbar, J., "Evolution in the Finitely Repeated Prisoner's Dilemma", *Journal of Economic Behavior and Organization*, Vol. 19; pp. 307-326, 1992.

[27] Fogel, D., "Evolving Behaviors in the Iterated Prisoner's Dilemma", *Evolutionary Computation,* Vol. 1(1); pp. 77-97, 1993.

[28] Thathachar, M. A. L. and Sastry, P. S., "Estimator Algorithms for Learning Automata", *Platinum Jubilee Conference on System Signal Processing*, Dept. Elec. Eng., Indian Institute of Science, Bangalore, India, Dec. 1986.

[29] Viswanathan, R. and Narendra, K. S., "Stochastic Automata Models with Applications to Learn1ng Systems", *IEEE, Transactions in Systems, Man and Cybernetics*, Vol. SMC-3, No. l; pp. 107-111, 1973.