INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

**An Investigation of Two
Responsive Learning Automata
In a Network Game with no
A Priori Information**

by

Cynthia Sawchuk

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Master of Computer Science

School of Computer Science

Carleton University
Ottawa, Ontario
December 1997

©
Cynthia Sawchuk
1997

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisitions et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-26982-5

The undersigned hereby recommend to
the Faculty of Graduate Studies and Research
acceptance of this thesis,

**An Investigation of Two
Responsive Learning Automata
In a Network Game with no
A Priori Information**

submitted by
Cynthia Sawchuk
in partial fulfillment of the requirements
for the degree of Master of Computer Science

Director, School of Computer Science

Thesis Supervisor

Carleton University
December 1997

Abstract

Conflict arises when computer network users try to obtain network resources in accordance with their respective quality of service requirements. Game theory and learning automata can be combined to provide a formal framework for studying such conflict.

In this thesis, the learning automaton of Thathachar and Sastry [TS85] is extended in the manner of Shenker and Friedman [S96]. The responsive learning automaton (RLA) of Shenker and Friedman [SF96] is based on the reward-inaction learning automaton. Since the Thathachar and Sastry [TS85] automaton has better convergence properties than the reward-inaction automaton, this thesis investigates whether the asymptotic properties proven for the RLA also hold for the extended Thathachar and Sastry [TS85] automaton. The behaviour of the extended automaton in finite horizon games is also investigated.

Acknowledgments

I would like to thank Danny Krizanc, my supervisor, for his inspiration, ideas, and enthusiasm. His thoughtful advice and insights were invaluable. I am especially grateful for the extra time and effort he offered to ensure that I finished my thesis on time.

I also wish to thank my husband, Kevin Fettig, for his unwavering support. He provided me with a great deal of encouragement and support throughout my undergraduate and graduate studies at Carleton. This thesis would not have been finished otherwise.

I want to thank my employers, The Object People. Whenever possible, they made it easier for me to balance the demands of work and of my thesis.

Finally, I would like to extend my thanks to the School of Computer Science. Over the years, I have come to know the faculty members and staff as individuals whom I respect and appreciate.

Contents

Abstract	iii
Acknowledgements	iv
1 Introduction	1
2 An Introduction to Game Theory	6
Extensive Form Games	8
Strategic or Normal Form Games	13
Cooperative Games	15
Non-Cooperative Games	16
Repeated Games	20
Learning in Repeated Games.....	23
3 Learning Automata	28
The L_{R-I} Scheme.....	34
The L_{R-P} Scheme	35

4	The Responsive Learning Automaton of Shenker and Friedman	38
	Synchronous Automata	43
	Asynchronous Automata	47
5	An Alternative Reinforcement Scheme for Learning Automata	53
	Theorem 1	57
	Claim 1	58
	Claim 2	61
	Claim 3	62
	Claim 4	63
	Claim 5	65
	Claim 5a	67
	Claim 6	70
	Claim 7	71
	Theorem 2	72
	Synchronous RLATS	73
	Theorem 3	73
	Theorem 4	74
	Asynchronous Automata	74
	Theorem 5	75
	Theorem 6	76

Theorem 7	76
6 Network Game Experiments	77
Description of the Network Game	78
Experiments with the Game	82
7 Conclusions	90
Appendix A. Graph of Results	94
Figure 1	95
Figure 2	96
Figure 3	97
Figure 4	98
Appendix B. Source Code	99
References	117

Chapter 1

Introduction

Despite the increasing power of computer systems, resource allocation is an important problem faced by distributed computer systems. It is important that we continue to study how to efficiently allocate resources, such as bandwidth, across the many users of a computer network.

Almost all of the early research in network design assumed that computer networks would be centrally controlled. Resource allocation across network users could then be dictated by a centrally mandated algorithm. The goal of the resource allocation algorithm was to achieve certain broad goals such as high utilization or small delay.

With the growing prominence of the Internet and other very large networks, network designers realized that a centrally controlled framework for a network was not always feasible. The performance of a decentralized network would result solely from the behaviour of the users. An individual user's behaviour would be motivated by that user's self-interests, not by any desire to achieve system-wide goals.

As a result, the focus of the network designer has shifted from centralized control algorithms to a study of the interaction of groups of non-cooperative users with conflicting goals. Many network designers have turned to game theory to aid their research. Game theory concentrates on the formal mathematical study of conflict, such as would occur when network users with different requirements attempt to gain network resources.

The essential components of a game include the players, their actions, the information they possess, and the payoffs they receive from their behaviour. The player's objective is to maximize her payoff during the game. However, with no a priori information about the game, which is typical for network users, a player will have to experiment, i.e., use trial and error, to find out about the structure of the game.

Learning automata are inherently simple but are designed to work in stochastic environments. This makes them an ideal mechanism for exploring the unknown structure of a game. Thus network designers can explore networks games by using learning automata to represent the users or players.

A major shortcoming of many learning automata, however, is that there is a non-zero probability that the automaton's strategy set will converge to a non-optimal strategy. To overcome this problem and to also allow automata to perform well in non-stationary environments, Shenker and Friedman [SF96] provide a new class of learning automata called the responsive learning automata (RLA).

Shenker and Friedman [SF96] prove that in a game between a group of synchronous RLA, where there is no a priori information, the strategy space of the game will converge asymptotically to the special set known as the serially undominated set. If the game is between a group of asynchronous RLA, the strategy space will converge asymptotically to a superset of the serially undominated set, known as the serially unoverwhelmed set.

Building on the work of Shenker and Friedman [SF96], this thesis examines an alternative responsive learning algorithm (RLATS) which is based on an automaton developed by Thathatchar and Sastry [TS85]. The automaton of Thatachar and Sastry [S85] is known to converge faster than the linear reward-inaction learning

automation upon which the RLA is based. This thesis shows that in a game between a synchronous group of RLATS, the strategy space will converge asymptotically to the serially undominated set, and to the serially unoverwhelmed set in the asynchronous case.

After the theoretical results mentioned above were established, some experiments were conducted to investigate the speed at which the strategy space would converge to the serially undominated set in synchronous games involving RLA or RLATS. If the speed of convergence of the strategy space is so slow that the serially undominated set cannot be identified before the game environment changes, then the learning algorithm is not useful in finite horizon games. That is because the strategy space will not have converged to a given serially undominated set before another serially undominated set replaces it. Changes in the strategy space will not occur fast enough, and thus the strategy space will always contain dominated strategies with relatively large probabilities of being played.

The thesis is organized as follows. Chapter 2 provides an introduction to game theory, while Chapter 3 provides an introduction to learning automata. Chapter 4 discusses the responsive learning automata (RLA) of Shenker and Friedman [SF96]. Chapter 5 contains the main theoretical contribution of this thesis. In this chapter, a series of proofs demonstrate that the responsive learning automaton (RLATS) based on the automaton of Thathachar and Sastry [TS85] has the same asymptotic

properties proven for the RLA of Shenker and Friedman [SF96]. Chapter 6 describes the game to be used for investigating the convergence of the strategy spaces of the repsective learning automata. As well, it contains the main empirical contributions of this thesis, i.e., the results of the investigations. Chapter 7 provides a summary of the results developed in this thesis.

Chapter 2

An Introduction to Game Theory

In the context of sports, a game is usually an interactive and competitive event governed by a set of rules which dictate how play can proceed. These rules govern the order of player actions, the range of legal player actions, and the relationship between these actions and the outcome of the game.

In such a game, each of the two or more players is trying to maximize the outcome resulting from their actions. This outcome depends on the actions of the other players, and all players are aware of this, and are also aware that all other players are aware of this. Choosing the best action in a given game situation thus requires an evaluation of the actions likely to be taken by a player's opponents.

Of course, the preceding description of a game characterizes many real-life situations which are not games of sport. The computer science problems which have been studied from the perspective of a game include flow control ([HL91] [MMD 91] [ZD92] [KL 94] [KL95]), routing ([ES91] [ORS93]), pricing in modern networks [CESZ93], and bandwidth allocation [LOP95].

For example, Lazar et al. [LOP95] consider the problem of allocating limited bandwidth in a network where the users have different quality of service requirements. Each user makes a bandwidth request and the resulting bandwidth allocation is based on these requests. The users have conflicting goals due to their different quality of service requirements, and thus game theory provides a means of studying what outcomes result in such a situation.

The game studied by Lazar et al. [LOP95] is an example of a non-cooperative game. In non-cooperative games, the players are prevented, by choice or circumstance, from colluding with each other. In cooperative games, however the players or any subgroup of them can enter into binding agreements about their future actions.

Non-cooperative games can be formally described in two ways. The extensive form of a game explicitly describes each individual move available to a player, while the normal or strategic form of a game describes the overall plans or strategies available to the players, but not the individual moves.

Extensive Form Games¹

As mentioned, non-cooperative games in extensive form explicitly detail every move available to each player. The most convenient way to present such a description is via a game tree.

A game tree is a finite set of nodes connected by edges so as to form a connected structure. The game tree contains no simple closed path, so any two given nodes are connected by a unique path.

An n-person game in extensive form thus consists of:

- 1) a game tree Γ with a root node A which is called the starting point of Γ
- 2) a function, called the payoff function, which assigns an n-vector to each terminal vertex of Γ
- 3) a partition of the non-terminal nodes of Γ into $n+1$ sets, $S_0, S_1, \dots, S_n$, called the players sets,

¹ The following discussion is based largely on Friedman [Fri86].

- 4) a probability distribution, defined at each node of Γ across the immediate followers of the node,
- 5) for each $i = 1, 2, \dots, n$, a sub-partition of S_i into subsets S_i^j , called the information sets, such that two nodes in the same information set have the same number of followers and no node can follow another node in the same information set,
- 6) for each information set S_i^j , an index set I_i^j , together with a 1-1 mapping of the set I_i^j onto the set of immediate followers of each node of S_i^j .

Points 1) and 2) above define, respectively, the starting point and the payoff function of the game. Point 3) divides the moves of the game into chance moves (S_0) and moves for the n players, $(S_1, S_2, \dots, S_n)$, while point 4) defines a randomization scheme at each chance move. Point 5) divides the players' moves into "information sets". A player knows which information set she is in, but not which node of the information set.

Information sets are used to model simultaneous moves. When it is a player's turn to move, she is located at a node, i.e. at a specific decision point. If she knows precisely which node she is at, then that node constitutes an information set. If two players move simultaneously, then the extensive form of the game has player 1 moving first, and both players know which specific node she is moving from. If she has m

possible moves, then player 2 will not know at which of these nodes she is actually located, and so these m nodes constitute an information set.

Games can be distinguished on the basis of what kind of information (complete or incomplete) the players have. Complete information versus incomplete information refers to whether or not each player knows:

- 1) who the set of players is,
- 2) all actions available to all players,
- 3) all potential outcomes to all players.

Complete information implies that each player knows the entire game tree Γ , including all the payoffs consistent with each terminal node. If any player does not know any subset of items 1-3 above, then the game is one of incomplete information.

In the bandwidth allocation game of Lazar et al. [LOP95] mentioned earlier, the game is assumed to have complete information. It is unlikely, however, that a given user of the network knows who the other users are or even how many there are. Thus it is much more realistic to assume that the game has incomplete information.

Games can also be distinguished on the basis of perfect and imperfect information. If each information set in the game consists of just one node, the game is one of perfect information. Otherwise, the game is one of imperfect information. In games with perfect information, each player knows exactly what actions were played in previous moves. In games with imperfect information, there is some uncertainty about previous moves, i.e., a player can get to a particular node in via different combinations of moves.

Since a user in a network does not typically know exactly what decisions other players made in the past, the bandwidth allocation game of Lazar et al. [LOP95] is one of imperfect information.

Finally, games can be distinguished by the relationship between the information sets and the moves of the players. If the information sets reflect that a player remembers all past moves she has played, the game is one of perfect recall. Otherwise, the game is one of imperfect recall.

Since a network user can record the decisions made in the past (and the resulting outcomes), the bandwidth allocation game of Lazar et al. [LOP95] involves perfect recall.

Imperfect information and perfect recall characterize many real-life games. This is because games with simultaneous moves necessarily have imperfect information, and individuals (or firms) usually have centralized information and decision-making which facilitate perfect recall.

While the extensive form of a game displays the complete move, information set, and payoff structure, the rules of the game are still needed to make the game complete. The rules of the games dictate:

- 1) whether information is complete or incomplete,
- 2) the extent of common knowledge – "those things that are known by all players, and known by each to be known by all of them" Friedman [Fri86],
- 3) whether it is possible for players to make binding agreements or commitments.

While it is true that in games of complete information, complete information is usually common knowledge, it is important to note that there exist complete information games where each player does not actually know whether other players have complete information. There is no reason to expect that intelligent behaviour should be the same in both cases.

The ability to make binding agreements or commitments is also included in common knowledge. If players adopt binding agreements or commitments, they agree to voluntarily restrict the actions from which they select. Such restrictions are enforceable. Binding agreements involve two or more players jointly agreeing upon restrictions, while a commitment is a unilateral restriction which is usually backed by a credible threat.

While expressing a game in extensive form is effective for simple games, the game tree becomes difficult to manage for more complex games. As well, some games have uncountably infinite strategy spaces, making an explicit representation of the game tree impossible.

Representing a game in normal or strategic form is an efficient alternative to using the extensive form.

Strategic or Normal Form Games

A strategy is an overall plan of action with respect to a given game. At every point of decision for a player, the strategy dictates precisely what the player does.

Strategies allow a player to randomly determine their choice of action. If player 1 chooses action i on her first move then, based on player 2's choice of action on her

first move, player 1 will choose the action for her second move based on a probability distribution over the available actions.

<u>Player 2's choice</u>	<u>Player 1's choice</u>	<u>with probability</u>
action ₁	action ₁	p ₁
	.	.
	action _m	p _m
.		
.		
.		
action _m	action ₁	p ₁
	.	.
	action _m	p _m
.		
.		

A pure strategy has no randomly determined choice, while a mixed strategy does.

If a strategic game has few moves, a payoff matrix corresponding to the various strategies can be calculated. It illustrates the payoffs associated with every sequence of strategies undertaken by the players.

Unlike the extensive form, the strategic form of a game suppresses the underlying move structure of the game. There are times however, especially in the context of non-cooperative games and their equilibrium points, when the move structure may be of interest.

Cooperative Games

As mentioned earlier, cooperative games allow the players (or any subgroup of them) to make binding contractual agreements. A subset of players which enters into such an agreement is called a coalition.

In a cooperative game, more attention is paid to the payoff opportunities available to each player and to each coalition than to the available strategies. The payoff possibilities facing each coalition are described by a characteristic function, $v(K)$, i.e., each coalition K receives a total payoff of $v(K)$ when its members act as a group.

It is unlikely that knowing the actions or strategies that the players use to obtain the various enforceable outcomes will provide any additional useful information. Thus cooperative games can be fully expressed in terms of the characteristic function.

The game which will be used for illustrative purposes later in this thesis assumes that players have no a priori information about the game, other than what actions they themselves may choose and the probability distribution over those actions. The common knowledge is severely restricted. Players may suspect that others are playing the game, but know neither the number of other players (if any), their action

sets, nor their resulting payoff from having chosen and played an action at a given time.

As a result, the game discussed in this thesis is necessarily a non-cooperative game and the discussion of game theory from this point will concentrate on such games. As well, since the extensive form for a game is only useful for small, simple games, the games will be discussed in strategic form.

Non-Cooperative Games

Weber [Web79] states that game theory in general asks two questions:

- 1) How do players behave in strategic situation?
- 2) How should players behave in strategic situations?

He notes that while research has shown that the answers to the descriptive and prescriptive questions do not always coincide, the equilibrium approach to non-cooperative games has had some success in answering both questions.

The study of non-cooperative games can be based on the equilibrium point concept. An equilibrium point of a game is a set of strategies, one for each player, such that no single player can gain by changing from their appointed strategy.

The properties of the set of equilibrium points for a non-cooperative game have been the subject of intense study. Much interest has been focussed on developing a set of rules for choosing a particular equilibrium point as the 'solution' of a game.

Consider the following formal description of an equilibrium point. Let $S^1, S^2, \dots, S^n$ be the strategy sets of the n players of a game. Let $P^1, P^2, \dots, P^n$ be the player's payoff functions, so that $P^k(s^1, s^2, \dots, s^n)$ is the payoff to player k when all of the players follow their indicated strategies. The n -tuple of strategies $(s^1, s^2, \dots, s^n)$ is an equilibrium point of the game if for every player k and every strategy t^k in S^k ,

$$P^k(s^1, s^2, \dots, s^k, \dots, s^n) \geq P^k(s^1, s^2, \dots, t^k, \dots, s^n)$$

Thus an equilibrium point is a set of strategies in which each player's strategy is her best response to the strategies of the remaining players.

As Weber [Web79] points out, the equilibrium point concept can be considered a generalization of the minimax solution of two-person zero-sum games. Let the strategy set of each player be made up of all mixed or randomized strategies over a finite set of pure or non-randomized strategies. A two-person game is zero-sum if the players' interests are strictly opposed so that, if P^i is the payoff to player i , then $P^1 = -P^2$. In 1928, von Neumann proved the famous "minimax" theorem which states that every such two-person game has an equilibrium point.

Nash [Nas50] extended this result by proving that every n -person general-sum game, in which each player has a finite set of pure strategies, has an equilibrium in mixed strategies. While several proofs exist for the existence of Nash equilibrium, the classical proof uses a fixed point theorem. First, a mapping is defined on the set of all n -tuples of strategies. Each n -tuple is then mapped onto another, where each player's strategy in the new n -tuple is a perturbation of her original strategy in the direction of a best reply to the other players' strategies in the old n -tuple.

The mapping described above satisfies the conditions for the Brouwer fixed-point theorem, which asserts that there is an n -tuple which will map to itself. This is only possible if each player's strategy in that n -tuple is her best reply to the strategies of the other players. This implies that the n -tuple which represents the fixed point in the mapping must be an equilibrium point in the game.

In the previously mentioned two-person zero-sum games, the equilibrium point(s) can be calculated using the simplex algorithm of linear programming. The equilibrium points of two-person general-sum games can be calculated by the complementary pivot algorithm of Lemke and Howson [LH64]. As well, the equilibrium points of n-person general-sum games can be calculated by generalizations of the two-person procedures, methods for approximating the fixed points of mappings, and the solution of dynamic systems converging to equilibrium points. Research has also led to the discovery of general conditions for the existence of equilibrium points in games with infinite pure strategy sets.

It is important to note that some properties of the equilibrium points for zero-sum games do not apply to the equilibrium points for general-sum games. The most important example is that of Pareto-optimality. If a game outcome is Pareto-optimal, other outcomes of the game can make some players better off (via higher payoffs) only at the expense of other players (via lower payoffs). This property is not upheld in general-sum games. The classic example is the game known as Prisoners' Dilemma, where there exists a unique equilibrium point yet both players would receive higher payoffs if they used another pair of strategies.

The preceding paragraph indicates, therefore, that not all Nash equilibria are equally attractive. A significant amount of research has been focussed on identifying the characteristics which equilibrium points must have in order to be considered solutions to a game. This research has given rise to several equilibrium refinements, such as perfect equilibria, proper equilibria, sequential equilibria, and stable sets.

Repeated Games

A repeated game is a special case of a game in extensive form in which the game tree consists of a sequence of strategic form games. It is possible to analyze repeated games with the notation of extensive form games, but the special structure of the games does allow for a simpler alternative notation.

Consider a game in strategic form which is merely the constituent game G repeated T times. It is easy to determine the unique path leading to a particular decision node in the t^{th} repetition of the game by the sequence of strategy combinations played in the stage games up to period t .

Each decision node at which a player has to move is identified by the history that led to this node. A history up to period t , h^t , (or a t -history) is a sequence of strategy combinations played up to this stage $(s^1, s^2, \dots, s^{t-1})$. The set of possible

histories up to the period t is the $(t-1)$ -fold Cartesian product of the set of all strategy combinations in the stage game, i.e., $S^{t-1} = S \times S \times \dots \times S$ $t-1$ times.

Because a strategic or normal form game is played at the beginning of each stage, the information of all players is the same at the beginning of each stage, although there may be imperfect information within a stage game. The strategic implications of imperfect information within a stage game are, however, captured by the strategic combination played in the stage game. Thus only the sequences of strategy combinations, one for each stage, need be recorded.

The strategies $s_i \in S_i$ of a player i in the constituent game G are called actions in the repeated game to distinguish them from the strategy of a repeated game, which is denoted by σ_i . A strategy σ_i in the repeated game is a sequence of action choices that depend on the history that led to this stage of the game.

An action at stage t , $a_i^t(\cdot)$, is a function that associates with each history in S^{t-1} an action from S_i . The set of possible actions in stage t is, therefore, the set of functions $A_i^t = \{a_i^t \mid a_i^t: S^{t-1} \rightarrow S_i\}$. Period 1 has no history so $S^0 = \{\emptyset\}$. The action choice in period 1 selects a unique action $a_i^1(h^1) \in S_i$ for the first stage of the game.

In a repeated game, choosing a strategy requires choosing a sequence of functions that specify what the player will do, given any possible history. A player i 's strategy set, Σ_i , in a repeated game is thus the T -fold Cartesian product of action function sets A_i^t , $\Sigma_i = A_i^1 \times A_i^2 \times \dots \times A_i^T$. The set of strategy combinations in a repeated game can then be described as $\Sigma = \Sigma_1 \times \Sigma_2 \times \dots \times \Sigma_n$, with typical elements $\sigma = (\sigma^1, \sigma^2, \dots, \sigma^n)$.

The strategy combination σ holds a plan for each player that specifies the action of the player after any possible history. This set of plans uniquely determines the sequence of action combinations that is actually played, known as the path of play and denoted by $\pi(\sigma) = (\pi^1(\sigma), \pi^2(\sigma), \pi^3(\sigma), \dots)$. The path of play $\pi(\sigma)$ is the actual history of the game if strategy combination σ is used.

In a repeated game, the payoff can be considered to be the sum of the discounted or undiscounted payoffs of the stage games. A discount factor $\delta \in [0, 1]$ allows for the extreme cases where a player only cares about the present, $\delta = 0$, and where there is no discounting of the future at all, $\delta = 1$.

In a repeated game, it is also necessary to consider the relationship between the payoffs in the stage games and the payoff from the repeated game as a whole. For example, maximizing the average payoff over the number of stages of the game leads to the same result as maximizing the sum of the payoffs because the number of stages or rounds in a repeated game is fixed.

Independent of whether the repeated game has a finite horizon or not, the following result holds:

Let s^* be a Nash equilibrium of the constituent game G and define the repeated game strategy σ^* by $a_i^t(h^t) = s_i^*$ for all $t = 1, 2, \dots, T$ and all $i \in I$, then, for all $\delta \in [0, 1]$, δ^* is a Nash equilibrium of $\Gamma^T(\delta)$.

Thus the repeated play of a Nash equilibrium of the stage game forms a Nash equilibrium of the repeated game. The average payoff from the repeated Nash equilibrium play is, as expected, exactly the payoff of the Nash equilibrium in the stage game.

Learning in Repeated Games

As Shenker and Friedman [SF96] note, the standard game theoretic literature on learning in repeated games covers four basic theories about learning:

- 1) Game participants can play their best response to the previous play, as discussed by Cournot [C38].
- 2) Game participants can play the best response to some average of historical play, an approach which has been studied by Carter and Maddock [CM84].
- 3) Game participants can play according to a complex Bayesian updating scheme based on the opponents' prior strategic behaviour. Fudenberg and Kreps [KF88] and Kalai and Lehrer [KL93] have investigated this theory.
- 4) Game participants can play according to a complex Bayesian updating scheme based on the structure of the game. Jordan [J91] discusses this theory.

In many of the game theory applications to computer science, the first three theories mentioned above are not applicable since they require that players' actions and the payoff matrix be common knowledge. A user of a computer network seldom knows how many 'opponents' she faces during network use, let alone who they are and what their actions are. It is also unlikely that a user will know the payoff matrix, but rather can only observe her specific payoffs resulting from actions undertaken. For example, Lazar et al. [LOP95] assume complete information in the bandwidth allocation game that they study, but it is much more likely that a network user will only know her own strategy set and will only observe payments accruing to her from having played a given strategy.

The fourth theory listed above is typically impractical to use in computer science games.

As well, none of the learning theories mentioned above allow for the possibility that the payoff matrix can change over time. This limits their use in computer science applications since, for example, users can join and/or leave a network during the course of a game, causing the payoff matrix to change.

In the absence of any knowledge about the payoff matrix, players are forced to use trial and error to learn about the game. To study such situations, Shenker and Friedman [SF96] combine the incentive issues normally addressed by game theoretic discussions of learning with stochastic sampling theory, the latter being a fundamental concept in the theory of decentralized control.

In order to understand learning in such a context, Shenker and Friedman [SF96] address two basic questions:

- 1) What properties should a reasonable learning algorithm have in this context?
- 2) How can one describe the set of asymptotic plays which results when all players use a reasonable learning algorithm?

When play occurs in a stationary random environment, such that the payoff for a given action in any period is drawn from a fixed probability distribution, a reasonable learning algorithm should eventually learn to play the strategy (or strategies) with the highest average payoff. Shenker and Friedman [SF96] suggest that a stronger condition be met:

a) Even in a non-stationary environment, a reasonable learning algorithm should eventually learn an action that is optimal (in the sense of the expected payoff).

As well, Shenker and Friedman [SF96] contend that if players cannot directly observe the payoff function, then changes in the game can **only** be detected by experimentation within the learning algorithm. Thus Shenker and Friedman [SF96] add the requirement that:

b) A reasonable learning algorithm must be able to adapt to changes in the environment.

Shenker and Friedman [SF96] provide a new class of responsive learning automata which in fact satisfy conditions a) and b) above.

The second question raised above asks whether the asymptotic set of plays will be a Nash equilibrium or some other type of previously identified subset of the plays. In the case of Shenker and Friedman's [SF96] responsive learning automata, the answer depends critically on the timing of the game. Before looking at the responsive learning automata in detail, a short review of learning automata may be useful.

Chapter 3

Learning Automata

Thathachar and Sastry [TS85] describe a learning automaton as a stochastic automaton that is connected via a feedback loop to a random environment. They define a stochastic automaton as a quadruple $\langle A, Q, T, R \rangle$ where:

$A = \{ a_1, a_2, \dots, a_r \}$ is the set of actions of the automaton, $2 \leq r < \infty$,

$a(k)$ denotes the action chosen by the automaton at time k and $a(k) \in A$ for all k ,

Q is the vector state of the automaton, where $Q(k)$ is the state at time k ,

$p(k) = [p_1(k) \dots p_r(k)]$ is a probability distribution over A at time k
where $p_i(k) = \text{prob}[a(k) = a_i]$, $p_i \geq 0$, and $\sum p_i(k) = 1$ for all k ,

R is the set of payoffs or rewards, i.e., reactions of the environment to the automaton's actions ($R \in [0,1]$ is a normalized game)

$r_i(k)$ represents the payoff at time k , where $r(k) \in R$ for all k ,

T is the updating operator defined by $Q(k+1) = T(Q(k), a(k), r_i(k))$,

T is almost always an algorithm, i.e., the learning, update or reinforcement algorithm.

The environment is defined by the triple $\langle A, H, R \rangle$ (where A and R are as defined above):

$H = \{F_1, F_2, \dots, F_r\}$ is a family of distributions,

F_i is the conditional distribution of $r_i(k)$, given $a(k) = a_i$,

$F_i(x) = \text{prob}[W_i \leq x] = \text{prob}[r_i(k) \leq x \mid a(k) = a_i]$ where W_i represents the random variable $[r_i(k) \mid a(k) = a_i]$,

$W_i \in [0, 1]$, and the F_i are assumed to be independent of k .

A learning automaton operates as follows:

- 1) at time k , the automaton chooses an action $a(k) \in A$ at random according to $p(k)$,

- 2) the environment responds with a random reward $r_i(k)$ from the distribution F_i when $a(k) = a_i$,
- 3) the automaton computes $Q(k + 1)$ using the learning algorithm T .

As Narendra and Thathachar [NT89] note, the learning algorithm is constructed so that if the automaton selects action a_i at time k , and a favourable payoff $r_i(k)$ occurs, the probability of playing action a_i at time $k+1$ is increased, i.e., $p_i(k+1) > p_i(k)$. Since $\sum p_i(k) = 1$ for all k , this implies that $p_j(k+1) < p_j(k)$ for at least one $j \neq i$.

The rules which dictate how $p(k+1)$ is to change when $a_i(k)$ is played completely define the learning or reinforcement algorithm. As Narendra and Thathachar [NT89] point out, this in turn determines the resulting behaviour of the system as a whole.

When the possible payoffs in the environment are restricted to 0 or 1, the model is referred to as a P-model. If the payoffs from the environment number more than two but are still a finite number, the Q-model results. The S-model represents an environment which has an infinite number of payoffs although, in the case of a normalized game, these payoffs are still in the range $[0, 1]$.

Thathachar and Sastry [TS85] provide several criteria for judging the performance of an automaton. In the following definitions, let the average reward $M(n)$ be defined as $M(n) = E[r(n) | p(n)]$, where $p(n)$ is the probability distribution over the actions.

A learning automaton operating in a Q- or S-model environment is **optimal** if

$$\lim_{n \rightarrow \infty} E[M(n)] = s_l$$

where

$$s_l = \max\{s_i\}$$

A learning automaton operating in a Q- or S-model environment is **ε -optimal** if

$$\lim_{n \rightarrow \infty} E[M(n)] < s_l + \varepsilon$$

can be obtained for every $\varepsilon > 0$ by a proper choice of the parameters for the automaton.

A learning automaton operating in a Q- or S- environment is **optimal in probability** if

$$p_l(k) \rightarrow 1$$

in probability.

Optimal in probability is a stronger condition than ϵ - optimality. With an ϵ - optimal learning automaton, there is a non-zero probability of converging to a non-optimal solution. Given a learning automaton that is optimal in probability, however, p^t evolves such that, with arbitrarily large probability, $p = 1$ for the optimal strategy.

A learning automaton operating in a Q- or S-model environment is **absolutely expedient** if

$$E[M(n + 1) - M(n) | p(n)] < 0$$

for all n , all $p_i(n) \in (0,1)$ for $i = 1, 2, \dots, r$ and all possible sets $\{s_i\}$ excluding the trivial environments in which all s_i are equal.

A learning automata which is absolutely expedient has absorbing states. If one of the absorbing states is reached, the automaton stays there permanently. Absorbing states are not necessarily optimal, so the automaton may be converging to a suboptimal strategy.

As mentioned earlier, the rules which dictate how $p(k+1)$ is to change when $a_i(k)$ is played completely define the learning or reinforcement algorithm. Narendra and Thathachar [NT89] provide the following description of a general nonlinear reinforcement scheme for a Q- or S-model:

$$p_i(n+1) = p_i(n) - (1 - r(n))g_i(p(n)) + r(n)h_i(p(n)) \text{ if } a(n) \neq a_i$$

$$p_i(n+1) = p_i(n) + (1 - r(n))\sum_{j \neq i} g_j(p(n)) - r(n)\sum_{j \neq i} h_j(p(n)) \text{ if } a(n) = a_i$$

$g_j(\cdot)$ and $h_j(\cdot)$ can be viewed as reward and penalty functions, respectively. They must satisfy some constraints if the $p_i(n), i = 1, 2, \dots, r$, are to remain in the interval $[0, 1]$.

Assuming that the functions are non-negative, these constraints are:

- 1) $g_j(\cdot)$ and $h_j(\cdot)$ are continuous functions mapping from the simplex

$$S = \left\{ p \mid p_i \geq 0, \sum_{i=1}^r p_i = 1 \right\}$$

to the interval $[0, 1]$.

- 2) $g_j(\cdot) \geq 0$ and $h_j(\cdot) \geq 0$ for all probability vectors $p \in S_r$.

3) for all $p_j \in (0,1)$, for all $i, j = 1, 2, \dots, j$

$$0 < g_j(p) < p_j$$

$$0 < \sum_{j \neq i} (h_j(p) + p_j) < 1$$

Two of the most well-known reinforcement schemes are the linear reward-inaction scheme, L_{R-I} , and the linear reward-penalty scheme, L_{R-P} .

The L_{R-I} Scheme

If the functions $g_j(\cdot)$ and $h_j(\cdot)$ are specified as follows:

$$g_j(p) = \phi p_j \quad 0 < \phi < 1$$

$$h_j(p) = 0,$$

the resulting reinforcement scheme is:

$$p_i(n+1) = p_i(n) - \phi(1 - r(n))p_i(n) \quad \text{if } a(n) \neq a_i$$

$$p_i(n+1) = p_i(n) + \phi(1 - r(n)) \sum_{j \neq i} g_j(p(n)) \quad \text{if } a(n) = a_i.$$

As Narendra and Thathachar [NT89] point out, the idea is to reward a favourable response by increasing the probability associated with the action which generated the response. No probabilities are changed, however, when an unfavourable response occurs. The scheme is thus one of reward and inaction, i.e. R-I.

The L_{R-P} Scheme

If the functions $g_j(\cdot)$ and $h_j(\cdot)$ are specified as follows

$$g_j(p) = \phi p_j \quad 0 < \phi < 1$$

$$h_j(p) = \phi/(r-1) - \phi p_i(n)$$

the resulting reinforcement scheme is:

$$p_i(n+1) = p_i(n) + r(n)\left(\phi / (r-1) - \phi p_i(n)\right) - \phi(1-r(n))p_i(n) \text{ if } a(n) \neq a_i$$

$$p_i(n+1) = p_i(n) - r(n)\phi p_i(n) + \phi(1-r(n))(1-p_i(n)) \text{ if } a(n) = a_i.$$

As Narendra and Thathachar [NT89] again point out, the idea is to reward a favourable response by increasing the probability associated with the action which generated the response, and punish an unfavourable response by decreasing the probability associated with the action. The scheme is thus one of reward and penalty, i.e. R-P.

The L_{R-P} scheme is an example of an ergodic learning algorithm. The sequence $\{p_j(k)\}$, $k \geq 0$, is an ergodic Markov process, which means that the distribution of $p(n)$ converges as $n \rightarrow \infty$. This implies that $p(n)$ has weak convergence, or converges in distribution. Thus we can discuss the asymptotic behaviour of the probability distribution of $p(n)$, but not of $p(n)$ itself.

In contrast, the L_{R-I} scheme is not an ergodic learning algorithm, but is an absolutely expedient algorithm. The sequence $\{p(k)\}$, $k \geq 0$, again a Markov process, has absorbing states. Once the process reaches an absorbing state, it remains there permanently. The process will converge with probability one to one of these absorbing states. Unfortunately, the absorbing states are not necessarily optimal, so there is a nonzero probability that the process will converge to a suboptimal action. Thus the algorithm will be ϵ -optimal.

Learning algorithms exist which are neither ergodic nor absolutely expedient. For example, Thathachar and Sastry [TS85] developed an algorithm in which $Q(k) = \langle p(k), d(k) \rangle$ forms a Markov process, rather than $p(k)$ itself, where $d(k)$ is a sequence that provides additional information to the algorithm. The updating rules of this algorithm ensure that, with arbitrarily large probability, the optimal action, a^* , is the only absorbing state. Thus the convergence of $p^*(k)$ to 1 in probability can be proven.

Chapter 4

The Responsive Learning Automaton of Shenker and Friedman

The major flaw in learning automata that are merely ϵ -optimal is that there exists a non-zero probability that the automata will converge to a non-optimal strategy. While this presents an obvious problem in stationary environments, it is even more troublesome in non-stationary environments, i.e., games where the payoff matrix changes, perhaps because players joined or left the game, or other players' strategy sets changed. Convergence to a single strategy, optimal or not, means that the other strategies are discarded and never played again. If the environment changes such that a discarded strategy becomes optimal, an automaton will not discover this, since it does not play that strategy anymore and a sub-optimal performance will result.

Shenker and Friedman [SF96] attempt to remedy this problem by slightly altering a linear reward-inaction automaton to obtain a “responsive” learning automaton (RLA). By restricting the probability of selecting any strategy to be at least $\alpha / 2$, for some constant $0 < \alpha < 1$, they prevent this automaton from discarding strategies. Since every strategy is now played infinitely often, there is no danger of performing sub-optimally merely because the optimal strategy is no longer available.

The update rules for the RLA are:

$$p_i^{t+1} = p_i^t + \alpha r_i^t \sum_{j \neq i} a_j^t p_j^t$$

$$p_j^{t+1} = p_j^t - \alpha r_i^t a_j^t p_j^t \quad \forall j \neq i$$

where

$$a_j^t = \frac{\min[1, p_j^t - \alpha / 2]}{\alpha p_j^t r_i^t}.$$

If $p_i > \alpha$ for all i , then the update rules for the RLA revert to those stated for the linear reward-inaction automata. The vector of probabilities p^t is said to be valid if $p_i \geq \alpha / 2$ for all i and $\sum_i p_i = 1$. The updating rules maintain the validity of the probability vector.

An automaton in a game with no a priori information must resort to experimentation and in doing so its goals are to optimize (by seeking optimal payoffs and being monotonic with respect to those payoffs), and to adapt to changing environments. The second goal is met by preventing the RLA from discarding strategies. Shenker and Friedman [SF96] show that the optimization goal is also met.

Because it is unreasonable to expect an automaton to optimize in any non-stationary environment, Shenker and Friedman [SF96] consider eventually stationary environments — environments that are independently and identically distributed after some finite time — and prove that the RLA do optimize in such environments. Optimization requires convergence to the optimal strategy, which the linear reward-inaction automata achieves by discarding non-optimal strategies. RLA does not discard non-optimal strategies, but rather spends most of its time playing optimal strategies. Occasionally, however, it does play non-optimal strategies so that it can check if their payoffs have changed, possibly due to a change in the environment. Convergence to an optimal strategy set for RLA thus requires a different definition, which is provided by Shenker and Friedman [SF96].

Definition A discrete time random process x^t parameterized by α

α -converges to 0 if there exist positive constants α_0 , β , b_1 , b_2 , b_3 , and q such that, for any $0 < \alpha < \alpha_0$:

- $\lim_{T \rightarrow \infty} \left(\frac{1}{T} \int_0^T dt P[x^t > \sqrt{\alpha}] \right) < \alpha$
- If T_f is the first time that $x^t \leq \beta\alpha$, then $E[T_f] \leq b_1/\alpha^q$
- If T_r is the first time that $x^t \geq \sqrt{\alpha}$, given that $x^0_\alpha \leq \beta\alpha$, then $E[T_r] \geq b_2 e^{b_3/\sqrt{\alpha}} / \alpha$.

As Shenker and Friedman [SF96] note, α - convergence of x^t is characterized by a rapid (polynomially fast) fall to near zero, followed by a very long (exponentially large) period near zero, before x^t moves away from zero. The exponentially large period during which x^t stays near zero dominates the polynomially large period it takes x^t to arrive near zero, given any average of x^t and any initial condition.

Given the definition of α - convergence, Shenker and Friedman [SF96] prove that the RLA has the two important properties of learning automata: convergence to optimality, and monotonicity with respect to payoffs.

Convergence to optimality for an RLA means that the probability of playing non-optimal strategies α - converges to zero. The mixed strategy that the RLA plays in each period has most of its probability mass concentrated on the optimal strategy, but occasionally non-optimal strategies are selected. α - convergence ensures that the mixed strategy vectors which result in notably sub-optimal payoffs occur very rarely.

Shenker and Friedman [SF96] prove that in any eventually stationary environment, every single play sequence of an RLA results in the long-run payoff approaching the optimal payoff. The authors emphasize that this is a significant improvement over the performance of standard learning automata like the linear reward-inaction automaton. While the latter automaton has close to optimal performance on average, a given individual play sequence can be significantly sub-optimal.

As mentioned, Shenker and Friedman [SF96] also prove that the RLA fulfills the monotonicity condition, i.e., increasing payoffs to certain strategies results in the increased probability that they will be played in the future.

Having proven that the RLA converges in optimality and fulfills the monotonicity condition, Shenker and Friedman [SF96] then focus on the asymptotic and collective behaviour of RLA in a non-cooperative game. Since the results depend

critically upon whether the automata are synchronous or asynchronous, the cases are dealt with separately.

The general model of the non-cooperative game used by Shenker and Friedman [SF96] involves n players. Player A has m_a possible strategies, $\Sigma_a = \{1, 2, \dots, m_a\}$, and the entire strategy space is thus $\Sigma = \Sigma_1 \times \dots \times \Sigma_n$. The notation $\Sigma = \Sigma_a \times \Sigma_{-a}$ is also used. Let s_a^t be player A 's strategy at time t , s_{-a}^t be the respective strategies selected by all other players, and $s^t = (s_a^t, s_{-a}^t)$. Having selected s_a^t at time t , player A receives $\Gamma_a(s^t)$, where $\Gamma : \Sigma \rightarrow \mathbb{R}^n$ is the payoff function of the game. The game Γ is normalized such that $\Gamma(s^t) \in [0, 1]^n$ for all $s^t \in \Sigma$. The players have no a priori information about the game and are allowed to change their strategies during the game. Player A knows only her own strategy set Σ_a and its associated probability distribution, and knows only the value of her own payoff in each round, *not* the entire payoff function Γ_a . As stated above, the eventual outcome of a game depends on whether or not the automata update their strategies at the same time.

Synchronous Automata

Consider a repeated game where there are well-defined rounds of play. A synchronous automaton selects a strategy during every round, receives its payoff from this strategy, and adjusts the probability distribution for its strategies according to this

payoff and the updating rules. The updated probability distribution is used to select the strategy for the next round.

Understanding the behaviour of a synchronous automaton requires that we be familiar with the concept of dominated strategies.

Definition For player A, strategy i **dominates** strategy j with respect to some set

$$S_{-a} \subseteq \Sigma_{-a} \text{ if for all } s_{-a} \in S_{-a}, \Gamma_a(i, s_{-a}) > \Gamma_a(j, s_{-a}).$$

Thus, for player A, strategy i dominates strategy j if the payoff from playing i exceeds that for playing j, given that the other players choose strategies from S_{-a} .

Correspondingly, the set of undominated strategies with respect to $S_{-a} \subseteq \Sigma_{-a}$ can be defined as a mapping $U_a : 2^{\Sigma_{-a}} \rightarrow 2^{\Sigma_a}$ such that

$$U_a(S_{-a}) = \{ s_a \in \Sigma_a \mid s'_a \in \Sigma_a \text{ s.t. for all } s_{-a} \in S_{-a}, \Gamma_a(s'_a, s_{-a}) > \Gamma_a(s_a, s_{-a}) \}.$$

Strategy k belongs to $U_a(S_{-a})$ if there is no strategy i for player A such that strategy i dominates strategy k, given that the other players choose strategies from S_{-a} .

Shenker and Friedman [SF96] prove that in a normalized game Γ , where player A is a synchronous RLA and all other players choose strategies from $S_{-a} \subseteq \Sigma_{-a}$ with probability $1 - \delta$ in each period, then if δ is sufficiently small, the synchronous RLA can eliminate dominated strategies. This result is reassuring, since dominated strategies are typically inferior to undominated ones, but as Shenker and Friedman [SF96] note, it is also surprising. The definition of dominated strategies contains the entire payoff function $\Gamma_a(i, s_{-a})$, which is unknown to player A. Even without such knowledge, synchronous RLA are able to eliminate dominated strategies.

Not only do Shenker and Friedman [SF96] prove that individual synchronous RLA can eliminate dominated strategies, but they further prove that in a game with no a priori information where all players are synchronous RLA, the collective asymptotic play is restricted to the serially undominated set $U^\infty(\Sigma)$ ¹. The set $U^\infty(\Sigma)$ represents the infinite iteration of the set mapping U as defined above, i.e.,

$$U^\infty(\Sigma) = U(U(\dots U(\Sigma)\dots)) = \bigcap_0^\infty U^0(\Sigma).$$

¹ Since all players are RLA in this game, there are n different α 's. A small restriction must be placed on these α 's, such that as the α 's approach zero, $\alpha^p_{\max} < \alpha_{\min}$ for some power of $p > 1$.

To understand the set $U^\infty(\Sigma)$, consider the first iteration

$U(\Sigma) = (U_1(\Sigma), \dots, U_n(\Sigma))$. By definition, each player i eliminates its dominated strategies under the assumption that the other players are selecting strategies from the entire strategy space Σ . In the next iteration, $U^2(\Sigma)$, each player i eliminates its dominated strategies under the assumption that the other players are selecting strategies $U(\Sigma)$. Thus, in iteration k , each player i eliminates its dominated strategies under the assumption that the other players are selecting strategies from the strategy space $U^{k-1}(\Sigma)$.

Milgrom and Roberts [MR91] note that some important learning models in economics converge to the serially undominated set $U^\infty(\Sigma)$. In fact, in three important models, Cournot's duopoly model, Bertrand's oligopoly with differentiated products, and the general equilibrium model with gross substitutes, the serially undominated set $U^\infty(\Sigma)$ is a singleton and thus the resulting equilibrium is unique. Unfortunately, no research has concentrated on the nature of the serially undominated set which results in games we are considering.

Asynchronous Automata

Consider a repeated game where there are no well-defined rounds of play, but rather each automaton independently chooses when to change strategies. Shenker and Friedman [SF96] model this asynchrony by having an RLA average its payoff over some period of time during which its strategy was fixed. To ensure that the time scale used in the averaging is large enough to produce perceptible changes in the probability distribution of the strategies, the time scale must vary inversely with α . Subsequently, as α approaches zero, RLA should use an asymptotically infinite averaging period for their payoff calculations. Recognizing that it is difficult to identify a single ‘correct’ averaging method for payoff determination, Shenker and Friedman [SF96] selected the following means of payoff calculation, which allows for significant variation.

Let $RLA_{\alpha}^{T,G}$ be an RLA which updates its strategy every T/α units of time. Because $RLA_{\alpha}^{T,G}$ wants to base its update on some weighted average of its payoffs in the previous time period, let the reward for playing strategy i in the last period be

$$r_i^t = \frac{1}{T/\alpha} \int_{t-T/\alpha}^t \Gamma_a(s^{t'}) dG\left(\frac{t'}{T/\alpha}\right)$$

where $G(t)$ is a cumulative distribution function and $s_a^{t'} = i$ for all $t' \in [t - T/\alpha, t]$.

Suppose that one of the players in the game is an $RLA_{\alpha}^{T,G}$, and nothing is known about the other players. While $RLA_{\alpha}^{T,G}$ keeps its strategy fixed over the period T/α , the other players can change their strategies, given the asynchrony of the game. As a result, Shenker and Friedman [SF96] point out, it is no longer appropriate to look at the undominated set of strategies, since the definition of a dominated strategy assumes that the other players' strategies are also fixed for the same period.

In response to this problem, Shenker and Friedman [SF96] introduce the concept of an overwhelmed strategy.

Definition For player A, strategy i **overwhelms** strategy j with respect to some set

$$S_{-a} \subseteq \Sigma_{-a} \quad \text{if} \quad \min_{s_{-a} \in S_{-a}} \Gamma_a(i, s_{-a}) > \max_{s_{-a} \in S_{-a}} \Gamma_a(j, s_{-a})$$

Thus, for player A, strategy i overwhelms strategy j if the minimum payoff from playing i against any $s_{-a} \in S_{-a}$ exceeds the maximum payoff from playing j against any $s_{-a} \in S_{-a}$.

Correspondingly, the set of unoverwhelmed strategies with respect to

$S_{-a} \subseteq \Sigma_{-a}$ can be defined as a mapping $O_a: 2^{\Sigma_{-a}} \rightarrow 2^{\Sigma_{-a}}$ such that

$$O_a(S_{-a}) = \{s_a \in \Sigma_a \mid s'_a \in \Sigma_a \text{ such that } \min_{s_{-a} \in S_{-a}} \Gamma_a(s'_a, s_{-a}) > \max_{s_{-a} \in S_{-a}} \Gamma_a(s_a, s_{-a})\}.$$

Strategy k belongs to $O_a(S_{-a})$ if there is no strategy i for player A such that strategy i overwhelms strategy k , given the strategies $S_{-a} \subseteq \Sigma_{-a}$. The set of unoverwhelmed strategies is a superset of the set of undominated strategies, $U_a(S_{-a}) \subseteq O_a(S_{-a})$, since an overwhelmed strategy must also be a dominated strategy, but the converse need not be true.

Shenker and Friedman [SF96] prove that in a normalized game Γ , where player A is an asynchronous RLA and all other players choose strategies from $S_{-a} \subseteq \Sigma_{-a}$ with probability $1 - \delta$ in each period, then if δ is sufficiently small, the asynchronous RLA can eliminate overwhelmed strategies. This result is again reassuring, since overwhelmed strategies are inferior to unoverwhelmed ones, but once more, as Shenker and Friedman [SF96] note, it is surprising. The definition of overwhelmed strategies contains the entire payoff function $\Gamma_a(i, s_{-a})$, which is unknown to player A . Even without such knowledge, asynchronous RLA are able to eliminate overwhelmed strategies.

Not only do Shenker and Friedman [SF96] prove that individual asynchronous RLA can eliminate overwhelmed strategies, but they further prove that in a game with no a priori information where all players are asynchronous RLA, the collective asymptotic play is restricted to the serially unoverwhelmed set $O^\infty(\Sigma)$. The set $O^\infty(\Sigma)$ represents the infinite iteration of the set mapping O as defined above, i.e., $O^\infty(\Sigma) = O(O(O(\dots O(\Sigma) \dots))) = O(\Sigma)$.

The structure of $O^\infty(\Sigma)$ mirrors that of $U^\infty(\Sigma)$. Consider the first iteration $O(\Sigma) = (O_1(\Sigma), \dots, O_n(\Sigma))$. By definition, each player i eliminates its overwhelmed strategies under the assumption that the other players are selecting strategies from the entire strategy space Σ . In the next iteration, $O^2(\Sigma)$, each player i eliminates its overwhelmed strategies under the assumption that the other players are selecting strategies from $O^1(\Sigma)$. Thus, in iteration k , each player i eliminates its overwhelmed strategies under the assumption that the other players are selecting strategies from the strategy space $O^{k-1}(\Sigma)$.

As Shenker and Friedman [SF96] point out, it is important to realize that the specific outcome of a game is dependent on the timing and the averaging method of the different automata. In general, then, $O^\infty(\Sigma)$ is unlikely to be a singleton, so that the above results do not uniquely define an equilibrium outcome. Shenker and Friedman

[SF96] contend that this is an unavoidable difficulty of learning in asynchronized decentralized systems.

Because different averaging methods are available to asynchronous automata, the following rather counter-intuitive result was discovered by Shenker and Friedman [SF96]. Consider a two automata game where the first player, A1, is changing her strategy and thus updating her probability distribution much more often than the second player, A2. If player A1 evaluates her payoff at the end of each of her respective play periods while player A2 averages her payoff over each entire respective play period, then as α_1 (the reinforcement parameter for player A1) approaches zero, the game converges to a Stackelberg equilibrium, with player A2 as the leader and player A1 as the follower. In such an equilibrium, player A1 always plays the strategy which is the best response to player A2's strategy choice, i.e., given A2 plays s_{A2} , then A1 plays s^* such that $\Gamma_{A2}(s^*_{A2}, s_{A2}) > \Gamma_{A1}(s'_{A1}, s_{A2})$ for all $s'_{A1} \neq s^*_{A2}$. Player A2 can then exploit player A1's willingness to respond. This is an unexpected result, as Shenker and Friedman [SF96] point out, since one would expect that more frequent updating would be a beneficial.

Shenker and Friedman [SF96] developed the RLA as an alternative to ϵ -optimal automata and also for use in non-stationary environments. They developed the RLA by revising the updating rules of the reward-inaction learning automaton. How different an automaton would result (versus the RLA) if a more complicated

automaton were substituted for the initial reward-inaction learning automata? The remainder of this thesis explores the automaton that results from revising the updating rules of Thathachar and Sastry's [TS85] automaton in the manner of Shenker and Friedman [SF96].

Chapter 5

An Alternative Reinforcement Scheme for Learning Automata

Thathachar and Sastry [TS85] developed a learning automaton (LATS) which improved upon the slow convergence rate of almost all learning automata known at the time. The general method used by the authors can be extended to handle a hierarchical system of automata, as well as a general n-teacher environment.

In its simplest form, this automaton collects sample information about a game as it plays, exploits this information in its updating rules, and thus achieves a better convergence result than ϵ - optimality. As mentioned earlier, automata which are at best ϵ - optimal have a non-zero probability of converging to a non-optimal strategy.

In LATS, however, for a sufficiently small reinforcement parameter α , the extra information obtained from sampling a game's environment ensures that the probability of playing the optimal strategy converges to 1 in probability. LATS is thus optimal in probability, which is a stronger result than ϵ - optimality.

In the discussion below, the optimal strategy is the one with the greatest mean payoff, although LATS can be easily extended to choose the optimal strategy based on any criterion.

LATS draws samples from a game's environment so that it can estimate the respective mean payoffs of its strategies. When the automaton plays strategy i , it receives the payoff r_i^t , which is a sample point from the payoff population for strategy i . Since the sample mean is a good estimator of a population mean, the sample mean payoff for strategy i is used as an estimate for the population mean payoff for strategy i . LATS essentially draws the requisite samples when it records the payoffs resulting from its chosen strategies during a game. A monotonically increasing function of the corresponding sample means is then used in the updating rules.

The following definitions and equations formally describe the updating rules for LATS:

$\hat{d}_i^t$ sample mean of the payoff for strategy i (i.e., sum of total payoffs received from playing strategy i divided by the number of times played)

$f(\hat{d}_i^t)$ $f : [0, 1] \rightarrow [0, 1]$ is a monotonically increasing function of $\hat{d}_i^t$

$$S_{ij} = 1 \quad \text{if } \hat{d}_i^t > \hat{d}_j^t$$

$$S_{ji} = 1 \quad \text{if } \hat{d}_j^t > \hat{d}_i^t$$

$$p_i^{t+1} = p_i^t + \lambda \sum_{j \neq i} [f(\hat{d}_i^t - \hat{d}_j^t) \cdot \left[S_{ij}^t p_j^t + S_{ji}^t \frac{p_i^t}{m-1} (1 - p_j^t) \right]] \quad \text{when strategy } i \text{ played}$$

$$p_j^{t+1} = p_j^t - \lambda [f(\hat{d}_i^t - \hat{d}_j^t) \cdot \left[S_{ij}^t \cdot p_j^t + S_{ji}^t \cdot \frac{p_i^t}{m-1} (1 - p_j^t) \right]] \quad j \neq i$$

In the linear reward-inaction automata with which Shenker and Friedman [SF96] started, if a strategy i is selected and its payoff r_i^t is received, the updating rules dictate that the probability of selecting strategy i increases and the respective probability of selecting each other strategy decreases. In LATS, however, the sign of the additive term in the updating rules depends on the sign of $\hat{d}_i^t - \hat{d}_j^t$ (since f is monotonically increasing). If $\hat{d}_i^t > \hat{d}_j^t$, such that the sample mean payoff for strategy i exceeds that of strategy j , then an amount proportional to p_j is subtracted from p_j and added to p_i . If $\hat{d}_i^t < \hat{d}_j^t$, such that the sample mean payoff for strategy i is less than

that of strategy j , then an amount proportional to $(p_i/(m-1))(1-p_j)$ is subtracted from p_i and added to p_j . The asymmetry ensures that each $p_k > 0$ and $\sum_k p_k = 1$ throughout the game.

In the original LATS, the probability of playing a particular strategy can be driven to zero. If LATS is to respond effectively to changing environments, it must be slightly altered so that each strategy has a probability of at least $\alpha/2$. Since each strategy would then be played infinitely often, the automaton could discern if the environment has changed and thus select its strategies accordingly. The altered LATS is as follows:

$$p_i^{t+1} = p_i^t + \alpha \sum_{j \neq i, j \text{ such that } S_{ij}=1} \gamma_{ij}^* (\hat{d}_i^t - \hat{d}_j^t) p_j^t S_{ij}^t + \alpha \sum_{j \neq i, j \text{ such that } S_{ij}=1} \gamma_{ij}^- (\hat{d}_i^t - \hat{d}_j^t) \frac{p_i^t}{m-1} (1-p_j^t) S_{ij}^t$$

$$p_j^{t+1} = p_j^t - \alpha \gamma_{ij}^* (\hat{d}_i^t - \hat{d}_j^t) p_j^t S_{ij}^t - \alpha \gamma_{ij}^- (\hat{d}_i^t - \hat{d}_j^t) \frac{p_i^t}{m-1} (1-p_j^t) S_{ij}^t$$

where

$$\gamma_{ij}^* = \min \left[1, \frac{p_j^t - \alpha/2}{\alpha p_j^t (\hat{d}_i^t - \hat{d}_j^t)} \right], \quad \gamma_{ij}^- = \min \left[1, \frac{p_i^t - \alpha/2}{\alpha p_i^t (1-p_j^t) (\hat{d}_j^t - \hat{d}_i^t)} \right].$$

Since the original LATS is optimal in probability and the linear reward-inaction automaton is merely ϵ - optimal, it would be interesting to discover if Shenker and Friedman's [SF96] results hold for the altered or "responsive" LATS (RLATS).

An obvious first step is to check whether the theorems in Shenker and Friedman [SF96] can be proven for RLATS (as opposed to RLA). If so, then the same results hold for RLATS as for RLA in games with no a priori information.

As noted earlier, Shenker and Friedman [SF96] start their analysis of RLA by verifying that it has two important properties: convergence to optimality, and monotonicity. In an *eventually* stationary environment, α -convergence is the appropriate type of convergence, and Theorem 1 explores whether the learning automaton in question meets this criterion.

Theorem 1

Consider a set of strategies $A \subseteq S_a$ and define $p_A^t = \sum p_i^t$. Assume there exists some $\beta > 1$ and $T > 0$ such that $E[r_i^t | h^t] < (1 - m/\beta) E[r_j^t | h^t]$ for all $t > T$, for all $i \in A$, and for all $j \in A$, where $h^t = (r^1, r^2, \dots, r^{t-1})$. Then p_A^t α - converges to 0 from any valid initial condition p^0 .

As in Shenker and Friedman [SF96], Theorem 1 is proven by a series of claims. The first claim states that p_A^t decreases on average when it is sufficiently far away from the boundary $\beta\alpha$.

Claim 1

There exists a constant c_1 such that, for all $\alpha > 0$ sufficiently small, if $p_A^t > \beta\alpha$, then

$$E [p_A^{t+1} | p_A^t] < p_A^t - c_1 \alpha^2 .$$

Proof:

The maximum value of p_i^{t+1} is

$$p_A^{t+1} \leq p_A^t + \alpha \left(\underbrace{\sum_{\substack{j \notin A \\ S_{ij}=1}} \gamma_{ij}^* (\hat{d}_i - \hat{d}_j) p_j}_1 - \underbrace{\sum_{\substack{j \notin A \\ S_{ij}=1}} \gamma_{ij}^- (\hat{d}_j - \hat{d}_i) \frac{p_j}{(m-1)} (1 - p_i)}_2 \right)$$

where term 1 represents the gains from strategies not in A if a strategy in A is played, and term 2 represents the gains from strategies not in A if a strategy not in A is played.

The upper bound for p_A^t is achieved when $\gamma_{ij}^* = \gamma_{ij}^- = 1$ and $\hat{d}_i > \hat{d}_j$, $\forall i \in A, j \notin A$.

The latter implies that $|j \in S_y| = | - A |$.

Thus

$$p_A^{t+1} \leq p_A^t + \alpha \sum_{i \in A} \sum_{j \notin A} \left(\hat{d}_i - \hat{d}_j \right) p_j^t + \left(\hat{d}_i - \hat{d}_j \right) \frac{p_j^t}{(m-1)} (1 - p_i^t)$$

$$\mathbb{E}[p_A^{t+1} | p_A^t] \leq p_A^t + \mathbb{E} \left[\alpha \sum_{i \in A} \sum_{j \notin A} \left(\hat{d}_i - \hat{d}_j \right) \left(p_j^t + \frac{p_j^t}{(m-1)} (1 - p_i^t) \right) \right]$$

Applying the central limit theorem implies that $\mathbb{E}[\hat{d}_i] = \mathbb{E}[r_i]$ asymptotically, so that

asymptotically

$$\mathbb{E}[\hat{d}_i - \hat{d}_j] = \mathbb{E}[r_i] - \mathbb{E}[r_j]$$

and since

$$\mathbb{E}[r_i] < \left(1 - \frac{m}{\beta} \right) \mathbb{E}[r_j] \quad \forall i \in A, j \notin A$$

then

$$\mathbb{E}[r_i] - \mathbb{E}[r_j] \leq \bar{r}_A - \bar{r}_{-A} < \left(-\frac{m}{\beta} \right) \bar{r}_{-A}.$$

This implies that

$$\mathbb{E}[p_A^{t+1} | p_A^t] - p_A^t \leq \left\langle \alpha \left(-\frac{m}{\beta} \bar{r}_{-A} \right) \right\rangle \sum_{i \in A} \sum_{j \notin A} \sum_{S_{ij}=1} p_j \left(\frac{m - p_i}{m - 1} \right).$$

Since the subexpression $\left\langle \alpha \left[-\frac{m}{\beta} \bar{r}_{-A} \right] \right\rangle < 0$, we can replace $p_j \left(\frac{m - p_i}{m - 1} \right)$ with $\frac{p_j p_i}{m}$

and maintain the inequality:

$$\begin{aligned} &\leq \alpha \left(-\frac{m}{\beta} \bar{r}_{-A} \right) \sum_{i \in A} \sum_{j \notin A} \frac{p_j p_i}{m} \\ &\leq \alpha \left(-\frac{\bar{r}_{-A}}{\beta} \right) p_A p_{-A} . \end{aligned}$$

Thus
$$E[p_A^{t+1} | p_A^t] - p_A^t \leq \alpha_{\max} \left(\frac{\bar{r}_{-A}}{\beta} \right) p_A^t p_{-A}^t .$$

Since
$$\max(p_A^t p_{-A}^t) = \frac{1}{4} \Rightarrow \frac{p_A^t p_{-A}^t}{\alpha} > \frac{1}{4}$$

$$E[p_A^{t+1} | p_A^t] - p_A^t \leq \alpha^2 \left(\frac{\bar{r}_{-A}}{4\beta} \right) \leq -c_1 \alpha^2 \text{ where } c_1 = \left(\frac{1}{4\beta} \right) .$$

□

Claim 2 indicates that p_A^t falls rapidly (polynomially quickly) to near zero.

Claim 2

Let τ_f be the first time that $p_A^t < \beta\alpha$. Then

$$E[\tau_f] < 1 / c_1 \alpha^2.$$

Proof:

Define $q^t \equiv p_A^{\min(t, T_F)} + c_1 \alpha^2 \min(t, T_F)$

If $t > T_F$, $q^{t+1} = q^t$.

If $t < T_F$, $q^{t+1} = p_A^{t+1} + c_1 \alpha^2 (t+1)$.

Thus $E[q^{t+1} | q^t] = E[p_A^{t+1} | p_A^t] + c_1 \alpha^2 (t+1)$.

From Claim 1, this implies that

$$\begin{aligned} E[q^{t+1} | q^t] &\leq p_A^t - c_1 \alpha^2 + c_1 \alpha^2 (t+1) \\ &\leq p_A^t + c_1 \alpha^2 t = q^t \quad (A) \\ \therefore E[q^{t+1} | q^t] &\leq q^t \quad \text{supermartingale by (A) and (B)} \end{aligned}$$

As well,

$$\begin{aligned} E[q^t] &\leq p_A^0 \quad \text{since } q^0 = p_A^0 \\ E[q^t] &= E[p_A^t] + c_1 \alpha^2 E[\min(t, T_F)] \leq p_A^0 \quad \text{from the definition of } q^t \\ \Rightarrow c_1 \alpha^2 E[\min(t, T_F)] &\leq p_A^0 \quad \text{since } E[p_A^t] \geq 0 \end{aligned}$$

$$\lim_{t \rightarrow \infty} c_1 \alpha^2 E[\min(t, T_F)] \leq p_A^0 \quad \text{by taking limit as } t \rightarrow \infty.$$

By the monotone convergence theorem, if $X_n(w) \geq 0$ and $X_n(w) \leq X_{n+1}(w)$ for every n , w , then $E(X_n) \rightarrow E(X)$. Thus

$$\lim_{t \rightarrow \infty} c_1 \alpha^2 E[\min(t, T_F)] = E(c_1 \alpha^2 \min(t, T_F)) = c_1 \alpha^2 E[T_F]$$

$$\therefore E[T_F] \leq \frac{p_A^0}{c_1 \alpha^2} \leq \frac{1}{c_1 \alpha^2}.$$

□

Claims 3 and 4 indicate that once $p_A^t < 2\beta\alpha$, then there is a much greater probability of it falling below $\beta\alpha$ before it rises above $k\alpha$.

Claim 3

There exists $c_3 > 0$ such that $e^{c_3 p_A^t / \alpha}$ is a supermartingale.

Proof:

Let $z^t = e^{c p_A^t / \alpha}$ for some $c > 0$

Note that $\hat{p}_A^t : p_A^t$ stops as soon as $p_A^t > k\alpha$ or $p_A^t < \beta\alpha$

Now, $E[z^{t+1} | z^t] \leq z^t$ requires that $\frac{E[z^{t+1} | z^t]}{z^t} \leq 1$.

That is, $E\left[e^{c(\hat{p}_A^{t+1} - \hat{p}_A^t)/\alpha}\right] = \frac{E[z^{t+1} | z^t]}{z^t} \leq 1$ when $\hat{p}_A^t < \beta\alpha$ or $\hat{p}_A^t > k\alpha$,

then $E[z^{t+1} | z^t] = z^t$.

Therefore $z^t = e^{c_3 \hat{p}_A^t / \alpha}$ is a supermartingale when $\hat{p}_A^t < \beta\alpha$ or $\hat{p}_A^t > k\alpha$.

Let $f(c) = E\left[e^{c(\hat{p}_A^{t+1} - \hat{p}_A^t)/\alpha}\right]$

For $\beta\alpha \leq \hat{p}_A^t \leq k\alpha$, $f(0) = 1$

$$f'(c) = E[e^{c(\hat{p}_A^{t+1} - \hat{p}_A^t)/\alpha} (\hat{p}_A^{t+1} - \hat{p}_A^t) / \alpha]$$

$$\therefore f'(0) = E[(\hat{p}_A^{t+1} - \hat{p}_A^t) / \alpha] \leq -c_1\alpha$$

$$\therefore \exists c \text{ such that } f(c) < 1$$

Let the c mentioned above be called c_3 . Therefore, for $c_3 > 0$, z^t is a supermartingale.

□

Claim 4

If $p_A^0 < 2\beta\alpha$, then there exists a constant c_4 such that

$$P[\lim p_A^t > k\alpha] < c_4 e^{-c_3 k}.$$

Proof:

Define $\Pr_k^t \equiv \Pr[\hat{p}_A^t > k\alpha]$ and $\Pr_F^t \equiv \Pr[\hat{p}_A^t < \beta\alpha]$.

Note that $z^t = e^{c_3 \hat{p}_A^t / \alpha}$ is a supermartingale from claim 3.

For every $t > 0$, we can condition the expected value of z^t on three mutually exclusive states. That is, we can examine the behaviour of the supermartingale as

$$\hat{p}'_A < \beta\alpha, \quad \hat{p}'_A > k\alpha \text{ and } \beta\alpha \leq \hat{p}'_A \leq k\alpha.$$

$$E[z^t] = E[z^t \mid z^t < e^{\beta c_3}] \Pr_F^t + E[z^t \mid z^t > e^{k c_3}] \Pr_k^t + E[z^t \mid e^{\beta c_3} \leq z^t \leq e^{k c_3}] (1 - \Pr_F^t - \Pr_k^t)$$

$$\hat{p}'_A \text{ has absorbing states } z^t < e^{\beta c_3} \text{ and } z^t > e^{k c_3}.$$

Therefore, $\Pr_F = \lim_{t \rightarrow \infty} \Pr_F^t$ and $\Pr_k = \lim_{t \rightarrow \infty} \Pr_k^t$ exist and sum to one (with probability 1).

$$\text{That is, in the limit } E[z^t] = E[z^t \mid z^t < e^{\beta c_3}] (1 - \Pr_k) + E[z^t \mid z^t > e^{k c_3}] (1 - \Pr_k).$$

$$\text{Therefore, } \Pr_k = \frac{E[z^t] - E[z^t \mid z^t < e^{\beta c_3}]}{E[z^t \mid z^t > e^{k c_3}] - E[z^t \mid z^t < e^{\beta c_3}]}.$$

Note that:

1. $1 \leq E[z^t] \leq z^0 \leq e^{2\beta c_3}$ z^t is a supermartingale and $\hat{p}_A^0 \leq 2\beta\alpha = k\alpha$
2. $1 \leq E[z^t \mid z^t < e^{\beta c_3}] \leq e^{\beta c_3}$ z^t is a supermartingale
3. $e^{k c_3} \leq E[z^t \mid z^t > e^{k c_3}] \leq e^{(k+1)c_3}$ $\hat{p}'_A$ stops as soon as it is $> k\alpha$

The above statements imply that $e^{\beta c_3} \leq E[z^t \mid e^{\beta c_3} \leq z^t \leq e^{k c_3}] \leq e^{k c_3}$.

Therefore,

$$p_k \leq \frac{e^{2\beta c_3} - 1}{e^{k c_3}} = e^{-k c_3} \frac{e^{2\beta c_3} - 1}{1 - e^{-k c_3}} < e^{-k c_3} \frac{e^{2\beta c_3}}{1 - e^{-(k-1)c_3}}$$

$$\therefore P_k < c_4 e^{-kc_3} \text{ where } c_4 = \frac{e^{2\beta c_3} - 1}{1 - e^{-(k-1)c_3}}.$$

□

Claims 5 and 6 show that p_A^t will stay near zero for a long (exponentially large) period of time.

Claim 5

Assume that $p_A^0 < 2\beta\alpha$. Let τ_k be the first time that $p_A^t > k\alpha$. Then

$$E[\tau_k] > e^{c_3 k} / 2c_4 \alpha.$$

Proof:

$$E[p_A^{t+1} - p_A^t \mid p_A^t \leq 2\beta\alpha] \leq \alpha p_A + \alpha \text{ from the updating rules and Claim 5a below.}$$

Therefore the expected time from $p_A^t \leq \beta\alpha$ to $p_A^t \geq 2\beta\alpha$ is at least $\frac{\beta\alpha}{\alpha p_A + \alpha}$, since

there are $\beta\alpha$ steps and the speed is at most $\alpha p_A + \alpha$ steps per unit of time. Therefore,

$$\begin{aligned} E[T_k] &\leq E[\text{number of times to } \beta\alpha \text{ before } k\alpha] * E[\text{time from } \beta\alpha \text{ to } k\alpha] \\ &= \frac{P_F}{P_k} \cdot \frac{\beta}{(2\beta\alpha + 1)} > \frac{\beta}{(2\beta\alpha + 1)P_k}. \end{aligned}$$

Hence,

$$E[T_k] \geq \frac{\beta}{(2\beta\alpha + 1)P_k} \geq \frac{\beta e^{kc_3}}{(2\beta\alpha + 1)c_4} \geq \frac{e^{kc_3}}{(2\beta\alpha + 1)c_4} \quad \text{since } \beta > 1.$$

Aside: P_k is the probability that p_A goes to $k\alpha$ first. Now,

$E[\text{the number of times } p_A \text{ goes to } \beta\alpha \text{ before } k\alpha] =$

$$0 \cdot P_k + 1 \cdot P_F + 2 \cdot P_F^2 + \dots = \sum_{i=1}^{\infty} i P_F^i = \frac{P_F}{(1 - P_F)^2} = \frac{P_F}{P_k^2} > \frac{1}{P_k}.$$

Consider $(2\beta\alpha + 1)c_4 \leq (2\beta\alpha + 2)c_4 = 2c_4\alpha(\beta + \frac{1}{\alpha})$.

Let $c_4' = c_4(\beta + \frac{1}{\alpha})$. $c_4' > c_4$

Therefore claim 4 $P_k < c_4 e^{-c_3 k} < c_4' e^{-c_3 k}$ still holds.

Hence, $E[T_k] \geq \frac{e^{kc_3}}{2c_4'\alpha}$.

□

Claim 5a

In the above proof, it was claimed that $E[p_A^{t+1} - p_A^t \mid p_A^t \leq 2\beta\alpha] \leq \alpha p_A + \alpha$.

This is proven in this claim, where $E[p_A^{t+1} - p_A^t] \equiv E[\Delta]$.

$$\begin{aligned}
 E(\Delta) &= \sum_{i \in A} \Delta p_A p_i^* + \sum_{j \notin A} \Delta p_A p_j^{\bullet} = \\
 &\sum_{i \in A} \left[\alpha \sum_{j \notin A, S_j=1} v_{ij}^*(\hat{d}_i - \hat{d}_j) p_j p_i^* + \alpha \sum_{j \notin A, S_j=1} v_{ij}^-(\hat{d}_i - \hat{d}_j) \frac{(p_i^*)^2}{(m-1)} (1 - p_j) \right] \\
 &- \sum_{j \notin A} \left[\alpha \sum_{i \in A} v_{ij}^*(\hat{d}_i - \hat{d}_j) p_j p_i^{\bullet} S_{ji} + \alpha \sum_{i \in A} v_{ij}^-(\hat{d}_i - \hat{d}_j) \frac{(p_i^{\bullet})^2}{(m-1)} (1 - p_j) S_{ji} \right] \\
 &\leq \sum_{i \in A} \left[\alpha \sum_{j \notin A, S_j=1} p_j p_i - \sum_{j \notin A, S_j=1} \frac{(p_i^* - \alpha/2) p_i}{m-1} \right] \\
 &- \sum_{j \notin A} \left[\sum_{i \in A} (p_i^* - \alpha/2) p_j S_{ji} - \alpha \sum_{i \in A} \frac{p_j^2}{m-1} (1 - p_i) S_{ji} \right]
 \end{aligned}$$

$$\begin{aligned}
&\leq \sum_{i \in A} \left[\alpha \sum_{j \in A, S_{ij}=1} p_j p_i - \sum_{j \in A, S_{ji}=1} \frac{(p_i - \alpha/2) p_i}{m-1} \right] \\
&\quad - \sum_{i \in A} \left[\sum_{j \in A, S_{ij}=1} (p_i - \alpha/2) p_j - \alpha \sum_{j \in A, S_{ji}=1} \frac{p_j^2 (1-p_i)}{m-1} \right] \\
&\leq \sum_{i \in A} \left[\alpha \sum_{j \in A, S_{ij}=1} \left(p_i p_j + p_j^2 \frac{(1-p_i)}{m-1} \right) - \sum_{j \in A, S_{ji}=1} \left(\frac{(p_i - \alpha/2) p_i}{m-1} + (p_i - \alpha/2) p_j \right) \right] \\
&\leq \sum_{i \in A} \left[\alpha p_i \sum_{j \in A, S_{ij}=1} p_j + \frac{\alpha(1-p_i)}{m-1} \sum_{j \in A, S_{ij}=1} p_j^2 - \frac{(p_i - \alpha/2) p_i}{m-1} \sum_{j \in A, S_{ji}=1} 1 - (p_i - \alpha/2) \sum_{j \in A, S_{ji}=1} p_j \right].
\end{aligned}$$

(1)
(2)

Let $|j \notin A \text{ such that } S_{ji} = 1| = 1$. Therefore $|j \notin A \text{ such that } S_{ij} = 1| = |A^c| - 1$.

Then, in the equation above (1) = 1, (2) $\leq \alpha/2$.

That is, $P_{A^c, S_{ij}=1} = P_{A^c} - \alpha/2$. Thus $E[\Delta]$

$$\leq \sum_{i \in A} \left[\alpha p_i p_{A^c, S_{ij}=1} + \frac{\alpha(1-p_i)}{m-1} p_{A^c, S_{ij}=1} - \frac{(p_i^2 - p_i \alpha/2)}{m-1} - (p_i - \alpha/2) \alpha/2 \right].$$

The second term in the inequality above is weighted by $p_{A^c, S_{ij}=1}$ since

$$\sum p_i^2 \leq \sum p_i \text{ for } p_i < 1.$$

Now substitute $p_{A^c} - \alpha/2$ for $p_{A^c, S_{ij}=1}$

$$\begin{aligned} &\leq \sum_{i \in A} \left[\alpha p_i (p_{A^c} - \alpha/2) + \frac{\alpha(1-p_i)}{m-1} (p_{A^c} - \alpha/2) - \frac{p_i^2 - p_i \alpha/2}{m-1} - (p_i \alpha/2 - (\alpha/2)^2) \right] \\ &\leq \alpha p_A p_{A^c} - \frac{\alpha^2 p_A}{2} + \alpha \frac{|A|}{(m-1)} p_{A^c} - \frac{\alpha^2 |A|}{2(m-1)} - \frac{\alpha p_A p_{A^c}}{(m-1)} + \frac{\alpha^2 p_A}{2(m-1)} \\ &\quad - \frac{1}{m-1} \sum_{i \in A} p_i^2 + \frac{\alpha p_A}{2(m-1)} - \frac{\alpha p_A}{2} + \frac{\alpha^2}{4} \\ &\leq \alpha p_A - \alpha p_A^2 - \frac{\alpha^2 p_A}{2} + \frac{\alpha |A|}{(m-1)} - \frac{\alpha |A| p_A}{(m-1)} - \frac{\alpha^2 |A|}{2(m-1)} - \frac{\alpha p_A}{m-1} + \frac{\alpha p_A^2}{m-1} \\ &\quad + \frac{\alpha^2 p_A}{2(m-1)} - \frac{1}{m-1} \sum_{i \in A} p_i^2 + \frac{\alpha p_A}{2} + \frac{\alpha^2}{4} \\ &\leq \alpha p_A + \frac{\alpha p_A^2}{(m-1)} + \frac{\alpha^2 p_A}{2(m-1)} + \frac{\alpha p_A}{2(m-1)} + \frac{\alpha^2}{4} \\ &\quad - \left(\alpha p_A^2 + \alpha^2 p_A + \frac{\alpha |A| p_A}{2(m-1)} + \frac{\alpha^2 |A|}{2(m-1)} + \frac{\alpha p_A}{m-1} + \frac{\sum p_i^2}{(m-1)} + \frac{\alpha p_A}{2} \right) \end{aligned}$$

$$\leq \alpha p_A + \frac{\alpha |A|}{(m-1)} - \frac{(m-2)\alpha^2 p_A}{2(m-1)} - \frac{\alpha p_A}{2(m-1)} - \frac{\alpha^2}{4}$$

$$- \frac{\alpha |A| p_A}{m-1} - \frac{\alpha^2 |A|}{2(m-1)} - \frac{\sum p_i^2}{(m-1)} \leq \alpha p_A + \frac{\alpha |A|}{m-1} \leq \alpha p_A + \alpha .$$

□

Claim 6

Assume that $p_A^0 < 2\beta\alpha$. Let τ_r be the first time that $p_A^t > \alpha^{1/2}$. Then

$$E[\tau_r] > e^{c_3/\alpha^{1/2}} / 2c_4 \alpha .$$

Proof:

This follows immediately from choosing $k = 1/\sqrt{\alpha}$ in the preceding claim.

□

Given the claims proven so far, we have shown that p_A^t collapses polynomially quickly to near zero and then moves away from near zero exponentially slowly. As a result, the limiting probability density of p_A^t is concentrated near zero. This allows us to show, in claim 7, that the first condition in the definition of α - convergence is obeyed.

Claim 7

There exists an α_0 such that for all $\alpha < \alpha_0$

$$\lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T dt P[p_A^t > a^{1/2}] < \alpha .$$

Proof:

The proof follows from the proof for claim 9 in Shenker and Friedman [SF96], with c_4' from claim 5 above replacing c_4 .

□

If α and β are chosen according to the preceding claims, and $b_1 = 1 / c_1$, $b_2 = 1 / 2c_4$, and $b_3 = c_3$, then p α - converges to zero and Theorem 1 is proven.

Besides α - convergence, a learning automata should also display monotonicity, i.e., increasing the payoffs for certain strategies increases the probability that they will be selected in the future. Theorem 2 shows that RLATS possess this property.

Theorem 2

Consider an automaton RLATS which is playing against an environment with a set of payoffs r_i^t , and the same automaton playing against a different environment with a set of payoffs r_i^{*t} . Let p_i^t and p_i^{*t} denote the probabilities in the two cases. Let A be any set of strategies. If r_i^{*t} is stochastically greater than or equal to r_i^t for all t and $i \in A$, and if r_i^{*t} is stochastically less than or equal to r_i^t for all t and $i \in A$, then $p_A^{*t} = \sum p_i^{*t}$ stochastically dominates $p_A^t = \sum p_i^t$ for all t .

Proof:

Define $p_A^t = \sum_{i \in A} p_i^t$. Notice that the update rules for p_A^t are, when strategy i is chosen at step t ,

$$p_A^{t+1} = p_A^t + \alpha \sum_{i \in A} \sum_{j \neq i | S_y = 1} \gamma_y^* (\hat{d}_i - \hat{d}_j) p_j^t S_y^t + \alpha \sum_{i \in A} \sum_{j \neq i | S_y = 1} \gamma_y^* (\hat{d}_i - \hat{d}_j) \frac{p_i^t}{(m-1)} (1 - p_j^t) S_y^t$$

Thus, p_A^{t+1} is monotonically increasing in p_A^t , monotonically increasing in r_i^t with $i \in A$, and monotonically decreasing in r_i^t with $i \notin A$. Over any sample path

$$\hat{p}_A \geq p_A.$$

□

Once the properties of convergence to optimality (from Theorem 1) and monotonicity (from Theorem 2) have been established, the synchronous and asynchronous cases for RLATS should be considered separately, given that Shenker and Friedman's results for RLA depend on the synchronicity of the automata.

Synchronous RLATS

Consider again a repeated game where there are well-defined rounds of play. A synchronous automaton selects a strategy during every round, receives its payoff from this strategy, and adjusts the probability distribution for its strategies according to this payoff and the updating rules. The updated probability distribution is used to select the strategy for the next round.

Using the earlier results of Shenker and Friedman [SF96], the following theorems can be shown to hold for RLATS.

Theorem 3

Consider a normalized game Γ and a player a whose strategies are chosen by a synchronous RLATS. Assume that the other players choose strategies from S_a with probability greater than $1 - \delta$ in each period. Then, for sufficiently small δ ,

$$p'_D \equiv \sum_{i \in U(S_{-a})} p'_i$$

α -converges to 0 for any valid initial condition p^0 .

If all the players are synchronous RLATS, then their play will again converge to the serially undominated set $U(\Sigma)$, as proven below.

Theorem 4

For any group of n ($n > 1$) synchronous responsive learning automata RLA playing a normalized game, and for any $p > 1$, for any automaton

$$p'_D \equiv \sum_{i \in U_a^p(\Sigma)} p'_i$$

α -converges to zero, where α -convergence is defined as all α_a 's converge to zero satisfying $\alpha_{\max}^p < \alpha_{\min}$.

Asynchronous Automata

The asynchronous case is likely to be more applicable in our case. Consider a repeated game where there are no well-defined rounds of play, but rather each

automaton independently chooses when to change strategies. The method of calculating the average payoff is the same as that described for the RLA.

Shenker and Friedman's [SF96] results can be used to prove that an asynchronous RLATS will not play overwhelmed strategies.

Theorem 5

Consider a normalized game Γ and a player a whose strategies are chosen by an asynchronous RLATS. Now assume that the other players choose strategies from S_{-a} with probability greater than $1 - \delta$ in each period. Then, for δ sufficiently small,

$$p_D^t \equiv \sum_{i \in O(S_{-a})} p_i^t$$

α -converges to zero for any valid initial condition.

The collective asymptotic behaviour of a group of asynchronous RLATS in a game can also be proven following the results in Shenker and Friedman [SF96].

Theorem 6

For any group of n ($n > 1$) asynchronous responsive learning automata RLATS playing a normalized game, and for any $p > 1$, for any automaton

$$P_D^t \equiv \sum_{i \in O_a^p(\Sigma)} P_i^t$$

α -converges to zero, where α -convergence is defined as all α_a 's converge to zero while satisfying $\alpha_{\max}^p < \alpha_{\min}$.

As mentioned earlier, this result may not provide much information about asymptotic play since $O(\Sigma)$ is most likely not a singleton, and thus the outcome is not uniquely defined. In such cases, it is again possible that a Stackelberg equilibrium occurs. As mentioned earlier, a Stackelberg equilibrium is characterized by one player being the leader and the other player being the follower, i.e., the second player always responds to the actions of the first.

Theorem 7

In the two player game there exist RLATS₁ and RLATS₂ such that player 1 converges to the Stackelberg leader and player 2 to the follower.

Chapter 6

Network Game Experiments

Shenker and Friedman [SF96] prove that, in a game with synchronous RLA, the strategy space will converge to the serially undominated set asymptotically. The purpose of the following experiments is investigate how quickly the convergence to the serially undominated set occurs, especially in the face of changing environments. Although RLA can detect changes in a game environment, such detection will not be useful if the time necessary for convergence greatly exceeds the time between changes in the environment. If the time necessary for convergence exceeds the time between changes in the environment, the strategy space of the game will not converge and will always contain dominated strategies with relatively large probabilities associated with them. The game selected for the experiments was developed by Lazar et al. [LOP95].

It is a network bandwidth allocation game, and thus it is likely that players of the game would have no a priori information about the game. As well, Lazar et al. [LOP95] proved that this game has a unique Nash equilibrium and also proved that the game converges to this equilibrium. Since the game is well-behaved under complete information yet is more likely to be played in the absence of any a priori information, it is an appropriate game for our investigation.

Description of the Network Game

Lazar et al. [LOP95] study a game in which a multi-user network is shared by non-cooperative users. The set of n users shares a resource of total capacity (bandwidth) of B units. Each user reserves some of the resource capacity in order to establish a virtual path for its incoming calls. Calls that are blocked at the virtual path level, because the corresponding virtual path is full upon arrival, are assumed to be lost such that blocking at the virtual path level leads to performance degradation.

The users are non-cooperative since each seeks to optimize its own performance. A given user optimizes performance by minimizing a cost function. The cost function accounts for the following tradeoff. A user will try to minimize the blocking probability of its incoming calls at the virtual path level, which is a decreasing function of the user's reserved capacity. Reserving capacity, however, becomes more difficult and more expensive as the system's resources become less available.

Lazar et al. [LOP95] prove that when certain restrictions are imposed on the users' cost functions, the network game has a unique Nash equilibrium that possesses a particular fairness property. The fairness property is such that if a given user's quality of service requirements make it more sensitive to call loss then, at the Nash equilibrium, such a user will get more capacity vis-a-vis a user with less call loss sensitivity.

Lazar et al. [LOP95] formally describe the game as follows:

C_i is the strategy of user i , ie., the amount of capacity that user reserves

$C_i \in [0, B]$, where B is the total available bandwidth

$C = (C_1, C_2, \dots, C_n)$ is the game strategy vector

$\Psi = \{ C \mid \forall i \ 0 \leq C_i \leq B \}$ is the game strategy space

C also denotes the total amount of reserved capacity, $C = \sum_i C_i$

$J_i(C) = J_i(C_i, C) = F_i(C_i, C) + G_i(C_i)$ is the cost function for user i , where F_i accounts for the availability of resources, as perceived by the i -th user and function G_i accounts for the performance of that user.

In their implementation of the game, Lazar et al [LOP 95] define $F_i(C_i, C)$ as follows:

$$F_i(C, C_i) = C_i \lambda_1 + \frac{\lambda_2}{\left(1 - \frac{C}{B}\right)^\eta}$$

where λ_1 , λ_2 , and η are positive real numbers. λ_1 represents a fixed cost per unit of bandwidth and λ_2 represents a unit cost attributed to 'congestion'. Lazar et al. [LOP95] define congestion as a situation in which total reserved capacity C approaches the link bandwidth B . The η parameter determines how early congestion is detected and is called the congestion avoidance parameter.

The unit cost of capacity, $F_i(C_i, C)$, has two regions of interest. If the total reserved capacity is a small proportion of the total available bandwidth, then $F_i(C_i, C) \cong (\lambda_1 + \lambda_2)$ is such that the unit cost of capacity is almost constant. As reserved capacity approaches total available bandwidth, however, the unit cost of capacity will increase rapidly with total reserved capacity. Lazar et al. [LOP95] call the regions, respectively, the low utilization region and the congestion region. They point out that when users operate in the first region, their strategies are decoupled and the cost of capacity is independent of other users' actions. In the congestion region, however, small changes in capacity reserved by one user may dramatically affect the capacity cost of other users.

The performance function, $G_i(C_i)$, is implemented as

$$G_i(C_i) = \begin{cases} \frac{1}{\kappa_i - E(\omega_i, C_i)} & \text{if } \kappa_i > E(\omega_i, C_i) \\ \infty & \text{otherwise} \end{cases}$$

The call arrival process of each user is assumed to be Poisson with rate ω_i , and $E(\omega_i, C_i)$ is the Erlang-B loss function corresponding to user i . The upper bound on user i 's call blocking probability, as determined by its quality of service requirements, is denoted by κ_i .

Lazar et al. [LOP95] further prove that the above game converges to the unique Nash equilibrium under both the Gauss-Seidel and the Jacobi schemes of computation. Since this game has a proven unique, desirable, and achievable equilibrium when certain restrictions are imposed on the cost function, such a game will form the basis for the game simulated in this thesis.

Experiments with the Game

In defining a game to be simulated, the following variables are assigned values:

n number of players

m_i number of strategies for player i , $i = \{1, \dots, n\}$

B total bandwidth = 100 Mbps

cost parameters (one set per player):

unit cost of bandwidth (.01 - .03)

unit cost of congestion (.001 - .003)

congestion avoidance parameter (1 - 9)

performance parameters (one set per player):

upper bound on blocking probability (.5 - 1.0)

arrival rate (10 - 50 erlangs)

Once the parameters of the game have been set, the strategy sets are calculated for each player. Since no player has any a priori information about the game, the strategy set spans the range from 25 - 100, e.g. if a player has four strategies, then her strategy set is {25, 50, 75, 100}, meaning that she will request either 25, 50, 75, or 100 units of bandwidth.

After calculating the strategy sets, the initial probability distributions are calculated. Again, the absence of any a priori information about the game means that each player's initial probability distribution y is uniform over that player's strategy set.

Before the game can begin, the appropriate α values are assigned or calculated. For example, in an LA_{R-I} , the alpha values (one per automaton) are chosen by the implementor of the LA, while in RLA and RLATS, the alpha values (again one per automaton) are calculated to be 1.5 times the minimum probability value in the distribution.

After the above parameters have been assigned values, then the simulation of the game can begin.

In each round of the game, player i will:

1. Player i selects a random number from the interval $[0, 1]$.
2. Player i uses the random number from step 1 to select a strategy according to the probability distribution over player i 's strategy set.
3. Each automaton plays the strategy selected in step 2.

4. The sum of the n players' selected strategies (from step 3) is the total bandwidth requested. If the total bandwidth requested is less than or equal to the total amount available, then each player is allocated the amount of bandwidth that they requested. If, however, the total bandwidth requested exceeds the total amount available, then each player is allocated bandwidth according to their proportion of the total request. For example, if a player request comprised 50 units of the total 250 unit request, then the player would receive $50/250$ of the available 100 units, or 20 units.
5. After the available bandwidth has been allocated, the payoffs to each player can be calculated. The cost and performance functions are calculated so that payoff can be determined for each player. The payoffs are normalized so that they are in the interval $[0,1]$.
6. When a player receives her payoff for playing the strategy selected in step 2, she updates the probability distribution over her strategy set according to the learning algorithm for the automata.
7. A round of the game is thus completed, and the game begins again from step 1.

The games used to explore the RLA and the RLATS need only differ in the learning algorithm applied in step 6.

In the experiments, the learning automata were allowed to converge in the environment described above as game 1, and then the environment was changed to that described above as game 2. Let this composite game be called game 3. The goal was to determine how quickly initial convergence to the serially undominated set

occurred, and how quickly the RLA and RLATS respectively responded to the change in environment.

The initial and second environments were chosen such that the respective serially undominated sets would provide a single strategy for each player. Since the results of the experiments were quite similar across simulations of the game, the discussion of the results will concentrate on single yet very typical simulation of the game.

In game 3, the serially undominated set for the first environment restricts each player to the strategy of making the largest bandwidth request possible. In the second environment, however, the serially undominated set has each player selecting the strategy which makes the smallest non-zero bandwidth request possible, given the strategy set. Note that the strategy set may be such that the smallest possible bandwidth request is still significantly more than zero.

In the initial environment, the RLATS converges faster than both the LA_{R-I} and the RLA. The LA_{R-I} , in turn, converges more quickly than RLA. The RLA also has more variation in its path to the optimal strategy for the initial environment, relative to LA_{R-I} and RLATS. These results for the entire game 3 are shown in Figure 1, while Figure 2 show the results for the initial environment only.

The faster convergence of the RLATS was expected, as the original learning automaton of Thathachar and Sastry [TS85] has better performance than the LA_{R-I} . The smoothness of its convergence relative to that of the RLA stems from the added information of the sample mean payoffs. An increase in strategy's probability is determined both by the payoff of the given strategy (when it has been played), and the difference between the sample mean payoffs of all the strategies. Thus, while a large but errant payoff from strategy j will increase the probability of strategy j under the RLA, the effect on the probability of strategy j under the RLATS will depend on the current sample mean payoffs, which are less and less affected by one large payoff as the number of rounds played increases.

When the second environment is imposed in the 15,000th round of game 3, the learning automata react quite differently. Of course, in the initial environment, the LA_{R-I} discarded all but one strategy. Since the LA_{R-I} is ϵ -optimal, there is a non-zero possibility that the remaining strategy is non-optimal. In the game discussed here, the LA_{R-I} converged to the appropriate strategy. In the second environment, however, this strategy was non-optimal and the LA_{R-I} was forced to play this strategy for the remainder of the simulation.

The RLA responds to the second environment by converging to the appropriate strategy but the convergence, as expected, is significantly slower in the second environment. This is because a player had a uniform probability distribution over her strategy set at the beginning of the initial environment. As the second environment begins, however, the player has a very skewed distribution, as the dominated strategies have α -converged to zero. What is surprising is that the convergence of the RLA in the second environment appears to be smoother than that observed in the initial environment.

Figure 3 presents the results of play in the second environment. RLATS recognizes that the environment has changed, but it does not converge to the serially undominated set in the time that the RLA converges. In fact, the sum of squared errors for the RLATS is falling quite slowly. The use of the sample mean payoffs to provide more information to the RLATS will slow down the convergence in the second environment. Since optimal strategy for the second environment has a very small probability at the beginning of that environment, it will be played infrequently at first (just as in the RLA.) The effect of the higher payoff to the optimal strategy, however, will only occur through the change in the sample mean payoffs. If the change of environment occurs after a large number of rounds, the impact of the now higher payoff for a given strategy will be felt slowly, as it will take many rounds of play for the sample mean payoffs to reflect the change.

Even though the RLA is the automaton with the fastest convergence to the serially undominated set in second environment, this convergence requires more than 10,000 rounds of play in the second environment. The question arises as to whether an environment in the network game will persist for that length of time. If not, then the RLA will not have converged to the serially undominated set for a new environment before the serially undominated set is rendered inappropriate by yet another environment change. If the RLA does converge to the serially undominated set between environment changes, then there will always be dominated strategies that have large probabilities of being played.

In an attempt to improve the behaviour of the RLATS in the second and subsequent environments, the sample size used to calculate the sample mean payoff was restricted. Sample sizes of 1000, 500, and 100 were tried. If the RLATS used a constant sample size of c , then the c most recent payoffs for a given strategy were used to calculate that strategy's sample mean payoff. Unfortunately, a sample size of 1000 did not appreciably improve the behaviour of RLATS in second or subsequent environments.

As figure 4 indicates, however, a sample size of 100 did allow the RLATS to converge much more rapidly than in the case where the sample was ever increasing.

The RLATS does not always outperform the RLA however. Consider the second and third environments of Figure 4. In the second environment, the RLATS is closer to the optimal strategy set than is the RLA. When the environment changes, however, the RLATS is initially further away from the optimal set than is RLA. In some cases, being further away from the optimal strategy set more than offsets the faster convergence of the RLATS, and the RLA is closer to converging before the environment again changes.

While Figure 4 indicates that the performance of RLATS under subsequent environment changes is improved when the sample size is kept constant, it also indicates that when the time between environment changes is less than the time to convergence, then both RLA and RLATS will have significant non-zero probabilities assigned to dominated strategies.

Chapter 7

Conclusions

The work undertaken in Chapter 5 of this thesis proves that RLATS, a learning automaton based on the automaton of Thathachar and Sastry [TS85], has the same asymptotic properties proven for the RLA of Shenker and Friedman [SF95]. Most important among these properties are the results of games between such automata. As this thesis proves, if a group of RLATS are playing a synchronous game, then the strategy space for the game will converge to the serially undominated set. Similarly, this thesis proves that in an asynchronous game, the strategy space will converge to the serially unoverwhelmed set, which is a superset of the serially undominated set. As well, this thesis proves that RLATS in a game will be able to detect when the game's environment has changed and they will adapt to this new environment. That is, in the

synchronous game, the strategy space of the game will converge to the serially undominated set for the new environment. In the asynchronous game, the strategy space of the game will converge to the serially unoverwhelmed set for the new environment.

The experimental results of Chapter 6 in this thesis, however, indicate the RLATS responds very slowly to any change in the environment, relative to the response speed of the RLA. Because the probability distribution over the strategy set for a RLATS changes only as the sample mean payoffs change, the larger the sample (number of rounds played), the slower the change. While some investigation of the effect of keeping the sample size constant indicated that the performance of the RLATS could be improved in subsequent environments, the overall performance still depends on whether the RLATS has time to converge before the next environment change occurs subsequent.

While RLA has a faster rate of convergence versus the RLATS with an ever-increasing sample size in some subsequent environments, it still required at least 10,000 rounds of play in a given environment in order for convergence to occur. To the extent that network games have environments with short horizons, since network users are frequently joining or leaving the network, or changing their strategy sets, the RLA may be unable to converge to the serially undominated set of given environment

before that environment is replaced. If the RLA are unable to converge to the serially undominated set between environment changes, then the probability distribution over the RLA's strategy set will be assigning large probabilities to dominated strategies, and the performance of the RLA will suffer.

Another problem which plagues the RLA and the RLATS is that the serially undominated set and the serially unoverwhelmed set respectively will not necessarily contain only one strategy per player. To the extent that the serially undominated set or the serially unoverwhelmed set is not significantly smaller than the whole strategy space, the convergence of the RLA (and RLATS respectively) to the serially undominated or unoverwhelmed set will not benefit the performance of the RLA (and RLATS respectively).

Finally, just as the choice of α is important when using the LA_{R-I} , it is equally important when using the RLA and the RLATS. Because the RLA is based on the LA_{R-I} , the choice of α in the RLA involves a tradeoff between the speed convergence and the probability of converging to a non-optimal strategy, just as it does in the LA_{R-I} . The larger the value of α , the faster the speed of convergence but the greater the probability that the automaton will converge to a non-optimal strategy. This tradeoff must be recognized when choosing to use RLA. In the case of the RLATS, the choice

of α will merely affect the speed of convergence, since it is optimal in probability, not merely ε -optimal.

In conclusion, it appears that despite their desirable asymptotic properties, the RLA and RLATS may be of limited use in games where the environments have short horizons. It is likely that the RLA and RLATS will not be able to converge to the serially undominated set nor the serially unoverwhelmed set between environments changes. Thus both the RLA and the RLATS will have dominated strategies in their strategy sets which will maintain a high probability of being played.

An interesting extension of this thesis would involve quantifying the bias introduced into the RLATS mean payoff estimation when the sample size is restricted. The central limit theorem used in the proof of Theorem 1 assumes that the sample size approaches infinity. Once the sample size is kept constant, the mean payoff estimator is biased and that bias is a function of the sample size. It would be interesting to discover the magnitude of the bias in the mean payoff estimation, and its subsequent effect on the performance of the RLATS in the network game.

Appendix A

Graph of Results

In the experiments the learning automata were allowed to converge to equilibrium under various environments which are described in Chapter 6. Figure 1 shows the results for the composite game. Figure 2 shows the results for game 1 and Figure 3 shows the results for game 2. Figure shows the results for the RLATS with a constant sample size of 100 observations. Figure 4 shows the results when the RLATS has a constant sample size of 100 observations

Figure 1

Figure 2

Figure 3

Figure 4

Appendix B

Source Code

The code for the experiments was written in Matlab. This package is well suited for modeling and simulations.

Worktrial.m

%Script file for testing the setup function

```
n = 4;% maximum number of automata is 4

numStrategies = [4 4 4 4];

strategyMatrix = [25 46 68 90 ; 25 46 68 90 ; 25 46 68 90 ;25 46 68 90];

targetValues = [0 0 0 1;0 0 0 1;0 0 0 1;0 0 0 1];

alpha = .05;

iterations = 30000;

change = 1;

ttlBWD = 100;% total bandwidth available

useableBWD = floor(.9 * ttlBWD);

costParameters = [.01,.001,1; .01,.001,1;.01,.001,1;.01,.001,1]; % 4 x n matrix --
unitCostBWD, unitCostCongestion, cAP

performanceParameters = [1.0, 10.0, 3.0; 1.0, 10.0, 3.0; 1.0, 20.0, 3.0; 1.0, 30.0, 3.0]; %
n x 4 matrix -- kappa, lambda, averageDuration

%y = setup(n, numStrategies, strategyMatrix, costParameters, performanceParameters);

%rlats = playRLATSGame(n, numStrategies, strategyMatrix, costParameters,
performanceParameters,targetValues);

%rlaAlpha = playRLAGameALPHA(n, numStrategies, strategyMatrix,
costParameters, performanceParameters,targetValues, alpha);

lari = playLARIasTrial(n, numStrategies, strategyMatrix,
costParameters,performanceParameters,alpha,targetValues,iterations, change);

rla = playRLAGameALPHA(n, numStrategies, strategyMatrix, costParameters,
performanceParameters, alpha, targetValues, iterations, change);
```

```
rlats = playRLATSGame(n, numStrategies, strategyMatrix, costParameters,  
    performanceParameters, alpha, targetValues, iterations, change);
```

playlariastrial.m

% Script for playing a specified game

```
function y = playLARlAsTrial(n, numStrategies, strategyMatrix, costParameters,  
    performanceParameters, alpha, targetValues, iterations, change)
```

```
if(iterations > 10000)  
    arrayLength = iterations/10;  
else  
    arrayLength = iterations;  
end
```

```
totalBWD = 100;  
useableBWD = 90;  
sseArray = zeros(1, arrayLength);
```

```
chosenStrategies = zeros(1, n);  
chosenIndices = zeros(1,n);
```

```
erlangBValues = zeros(1, n);  
performanceValues = zeros(1, n);  
costValues = zeros(1, n);  
payoffValues = zeros(1, n);  
tempsse = zeros(1,n);
```

% Calculate initial probability distributions over the respective strategy sets.

```
probabilityCell = calculateProbabilityCell(n,numStrategies); %(checked)
```

```
flag = 0;  
count = 1;
```

```
while(flag == 0)
```

```

if(change == 1)
    if(count > (iterations / 2))
        costParameters = [.01 .2 15;.01 .2 15;.03 .2 15;.03 .2 15];
        performanceParameters = [.9 10 2;.9 10 2;.9 20 2;.9 30 2];
        targetValues = [1 0 0 0; 1 0 0 0;1 0 0 0;1 0 0 0];
    end
end

% Each automaton chooses a strategy

chosenRandNum = rand(1,n);

for automaton = 1:n

    probFlag = 0;
    cumProb = calculateCumProbabilityDistri(probabilityCell{1,automaton});
    index = 1;
    while(probFlag == 0)
        if(index == numStrategies(1,automaton))
            chosenStrategies(1, automaton) = strategyMatrix(automaton,
                index);
            chosenIndices(1, automaton) = index;
            probFlag = 1;
        else
            if(chosenRandNum(1,automaton) <= cumProb(1, index))
                chosenStrategies(1, automaton) = strategyMatrix(automaton,
                    index);
                chosenIndices(1, automaton) = index;
                probFlag = 1;
            end
        end
        index = index + 1;
    end
end

% Each automaton is allocated BWD.

totalRequestedBWD = sum(chosenStrategies);

```



```

if(totalRequestedBWD > useableBWD)
    chosenStrategies = floor((chosenStrategies/totalRequestedBWD) *
    useableBWD);
end

p = zeros(1,1);
t = zeros(1,1);

for automaton = 1:n
    erlangBValues(1,automaton) = erlangB(chosenStrategies(1, automaton),
    performanceParameters(automaton, 2),performanceParameters(automaton, 3));

    performanceValues(1, automaton) = performance(erlangBValues(1, automaton),
    performanceParameters(automaton, 1));

    costValues(1, automaton) = cost(useableBWD, chosenStrategies(1, automaton),
    costParameters(automaton, 1), costParameters(automaton, 2),
    costParameters(automaton, 3));

    payoffValues(1, automaton) = (performanceValues(1, automaton) + costValues(1,
    automaton))^-1;

    probabilityCell{1,automaton} = updateProbDistriLARI(chosenIndices(1, automaton),
    numStrategies(1, automaton), probabilityCell{1,automaton}, payoffValues(1,
    automaton), alpha);

    p = [p probabilityCell{1,automaton}];

    t = [t targetValues(automaton,:)];
end

if(iterations > 10000)
    if(rem(count, 10) == 0)
        item = count/10;
        sseArray(1,item) = sse(p,t);
    end
end

```

```
if(iterations <= 10000)
    sseArray(1,count)= sse(p,t);
end
```

```
count = count + 1;
```

```
if(count > iterations)
    flag = 1;
end
```

```
end
```

```
y = sseArray;
```

playrlatsgame.m

% Script for playing a specified game

```
function y = playRLATSGame(n, numStrategies, strategyMatrix, costParameters,  
    performanceParameters, alpha, targetValues,iterations,change)
```

```
if(iterations > 10000)
    arrayLength = iterations/10;
else
    arrayLength = iterations;
end
```

```
totalBWD = 100;  
useableBWD = 90;  
sseArray = zeros(1,arrayLength);
```

```

erlangBValues = zeros(1, n);
performanceValues = zeros(1, n);
costValues = zeros(1, n);
payoffValues = zeros(1, n);
tempsse = zeros(1,n);

% Calculate initial probability distributions over the respective strategy sets.

probabilityCell = calculateProbabilityCell(n,numStrategies); %(checked)

for index = 1:n
    sampleMeanPayoffCell{1,index} = zeros(numStrategies(1,index),2);
end

flag = 0;
count = 1;
chosenStrategies = zeros(1, n);
chosenIndices = zeros(1,n);

while(flag == 0)

if(change == 1)
    if(count > (iterations /2))
        costParameters = [.01 .2 15;.01 .2 15;.03 .2 15;.03 .2 15];
        performanceParameters = [.9 10 2; .9 10 2; .9 20 2; .9 30 2];
        targetValues = [1 0 0 0; 1 0 0 0; 1 0 0 0; 1 0 0 0];
    end
end

% Each automaton chooses a strategy

chosenRandNum = rand(1,n);

```

```

for automaton = 1:n

    probFlag = 0;
    cumProb = calculateCumProbabilityDistri(probabilityCell{1,automaton});
    index = 1;
    while(probFlag == 0)
        if(index == numStrategies(1,automaton))
            chosenStrategies(1, automaton) = strategyMatrix(automaton, index);
            chosenIndices(1, automaton) = index;
            probFlag = 1;
        else
            if(chosenRandNum(1,automaton) <= cumProb(1, index))
                chosenStrategies(1, automaton) = strategyMatrix(automaton, index);
                chosenIndices(1, automaton) = index;
                probFlag = 1;
            end
        end
        index = index + 1;
    end
end

% Each automaton is allocated BWD.

totalRequestedBWD = sum(chosenStrategies);

if(totalRequestedBWD > useableBWD)

    chosenStrategies = floor((chosenStrategies/totalRequestedBWD) * useableBWD);
end

p = zeros(1,1);
t = zeros(1,1);

for automaton = 1:n

    erlangBValues(1,automaton) = erlangB(chosenStrategies(1, automaton),
        performanceParameters(automaton, 2),performanceParameters(automaton, 3));

    performanceValues(1, automaton) = performance(erlangBValues(1, automaton),
        performanceParameters(automaton, 1));

```

```

costValues(1, automaton) = cost(useableBWD, chosenStrategies(1, automaton),
costParameters(automaton, 1), costParameters(automaton, 2),
costParameters(automaton, 3));

payoffValues(1, automaton) = (performanceValues(1, automaton) + costValues(1,
automaton))^-1;

newCell = updateRLATS(numStrategies(1,automaton), chosenIndices(1,
automaton),payoffValues(1, automaton), alpha, probabilityCell{1,automaton},
sampleMeanPayoffCell{1,automaton});

probabilityCell{1,automaton} = newCell{1,1};
sampleMeanPayoffCell{1,automaton} = newCell{1,2};
p = [p probabilityCell{1,automaton}];
t = [t targetValues(automaton,:)];
end

if(iterations > 10000)
    if(rem(count,10) == 0)
        item = count/10;
        sseArray(1,item) = sse(p,t);
    end
else
    sseArray(1,count)= sse(p,t);
end

count = count + 1;

if(count > iterations)
    flag = 1;
end

end

y = sseArray;

```

playrlgame.m

% Script for playing a specified game

function y = playRLGameALPHA(n, numStrategies, strategyMatrix, costParameters,
performanceParameters, alpha, targetValues, iterations, change)

```
if(iterations > 10000)
    arrayLength = iterations/10;
else
    arrayLength = iterations;
end
```

```
totalBWD = 100;
useableBWD = 90;
sseArray = zeros(1, arrayLength);
```

```
chosenStrategies = zeros(1, n);
chosenIndices = zeros(1,n);
```

```
erlangBValues = zeros(1, n);
performanceValues = zeros(1, n);
costValues = zeros(1, n);
payoffValues = zeros(1, n);
tempsse = zeros(1,n);
```

% Calculate initial probability distributions over the respective strategy sets.

```
probabilityCell = calculateProbabilityCell(n,numStrategies); %(checked)
```

```
for index = 1:n
    probabilityCell{1,n}
end
```

```
flag = 0;
count = 1;
```

```

while(flag == 0)

    if(change == 1)
        if(count > (iterations / 2))
            costParameters = [.01 .2 15;.01 .2 15;.03 .2 15;.03 .2 15];
            performanceParameters = [.9 10 2; .9 10 2; .9 20 2; .9 30 2];
            targetValues = [1 0 0 0; 1 0 0 0; 1 0 0 0; 1 0 0 0];
        end
    end

    % Each automaton chooses a strategy

    chosenRandNum = rand(1,n);

    for automaton = 1:n

        probFlag = 0;
        cumProb = calculateCumProbabilityDistri(probabilityCell{1,automaton});
        index = 1;
        while(probFlag == 0)
            if(index == numStrategies(1,automaton))
                chosenStrategies(1, automaton) = strategyMatrix(automaton, index);
                chosenIndices(1, automaton) = index;
                probFlag = 1;
            else
                if(chosenRandNum(1,automaton) <= cumProb(1, index))
                    chosenStrategies(1, automaton) = strategyMatrix(automaton, index);
                    chosenIndices(1, automaton) = index;
                    probFlag = 1;
                end
            end
            index = index + 1;
        end
    end

    % Each automaton is allocated BWD.

    totalRequestedBWD = sum(chosenStrategies);

```

```

if(totalRequestedBWD > useableBWD)

    chosenStrategies = floor((chosenStrategies/totalRequestedBWD) * useableBWD);
end

p = zeros(1,1);
t = zeros(1,1);

for automaton = 1:n
    erlangBValues(1,automaton) = erlangB(chosenStrategies(1, automaton),
    performanceParameters(automaton, 2),performanceParameters(automaton, 3));

    performanceValues(1, automaton) = performance(erlangBValues(1, automaton),
    performanceParameters(automaton, 1));

    costValues(1, automaton) = cost(useableBWD, chosenStrategies(1, automaton),
    costParameters(automaton, 1), costParameters(automaton, 2),
    costParameters(automaton, 3));

    payoffValues(1, automaton) = (performanceValues(1, automaton) + costValues(1,
    automaton))^-1;

    probabilityCell{1,automaton} = updateProbDistriRLA(chosenIndices(1, automaton),
    numStrategies(1, automaton), probabilityCell{1,automaton}, payoffValues(1,
    automaton), alpha);

    p = [p probabilityCell{1,automaton}];
    t = [t targetValues(automaton,:)];

end

if(iterations > 10000)
    if(rem(count,10) == 0)
        item = count /10;
        sseArray(1,item) = sse(p,t);
    end
else
    sseArray(1,count) = sse(p,t);
end

```



```
count = count + 1;
```

```
if(count > iterations)
```

```
    flag = 1;
```

```
end
```

```
end
```

```
y = sseArray;
```

updateProbLARI.m

```
% mfile for recalculating the probability matrix according to Shenker and  
% Friedman's RLA
```

```
function y = updateProbDistriLARI(playedIndex, m, probVector, payoff, alpha)
```

```
decrementSum = 0;
```

```
for index = 1:m
```

```
    change = probVector(1,index)*alpha*payoff;
```

```
    decrementSum = decrementSum + change;
```

```
    probVector(1, index) = probVector(1, index) - change;
```

```
end
```

```
probVector(1, playedIndex) = probVector(1, playedIndex) + decrementSum;
```

```
y = probVector;
```

updateProbDistRLA.m

```
% mfile for recalculating the probability matrix according to Shenker and  
% Friedman's RLA
```

```
function y = updateProbDistriRLA(playedIndex, m, probVector, payoff, alpha)
```

```

decrementSum = 0;

for index = 1:m
    weightFactor = (probVector(1, index) - (alpha / 2)) / (alpha * payoff *
probVector(1, index));
    if(weightFactor > 1)
        weightFactor = 1;
    end
    decrement = payoff*weightFactor*probVector(1,index)*alpha;
    if(index == playedIndex)
        decrement = 0;
    end

    probVector(1,index) = probVector(1, index) - decrement;
    decrementSum = decrementSum + decrement;
end

probVector(1, playedIndex) = probVector(1, playedIndex) + decrementSum;

y = probVector;

updateRLATS.m

% Function for updating the RLATS

function y = updateRLATS(m, actionIndex, thePayoff, alpha, probabilityDistri,
    sampleMeanPayoffMatrix)

newProbabilityDistribution = zeros(1, m);

% update the sample mean payoff for the action just played

oldSampleTotal = sampleMeanPayoffMatrix(actionIndex, 1)*
sampleMeanPayoffMatrix(actionIndex, 2);

newSampleTotal = oldSampleTotal + thePayoff;

```

```
sampleMeanPayoffMatrix(actionIndex, 2) = sampleMeanPayoffMatrix(actionIndex, 2) +
    1;
```

```
sampleMeanPayoffMatrix(actionIndex, 1) = newSampleTotal /
sampleMeanPayoffMatrix(actionIndex, 2);
```

```
% calculate the difference between the sample mean payoff just updated and the other
sample mean payoffs
```

```
diffSampleMeanPayoff = -(sampleMeanPayoffMatrix(:,1) - thePayoff)';
```

```
% calculate the Sij values
```

```
Sij = sampleMeanPayoffMatrix(:,1) < thePayoff;
Sij = Sij';
Sij(1, actionIndex) = 0;
```

```
%calculate the Sji values
```

```
Sji = sampleMeanPayoffMatrix(:,1) > thePayoff;
Sji = Sji';
Sji(1, actionIndex) = 0;
```

```
% calculate gammaStar . . . used if Sij = 1 thus diffSampleMeanPayoff j > 0
```

```
gammaStar = zeros(1,m);
```

```
for index = 1:m
```

```
if(diffSampleMeanPayoff(1, index) != 0)
```

```
    term = (probabilityDistri(1, index) - (alpha/2))/ (alpha * probabilityDistri(1,index) *
diffSampleMeanPayoff(1, index));
```

```
else
```

```
    term = 1;
```

```
end
```

```
term = term * Sij(1, index);
```

```
term = min(1, term);
```

```
gammaStar(1,index) = term;
```

```
end
```

```

% calculate gammaTilde . . . used if Sji = 1 thus diffSampleMeanPayoffj < 0

gammaTilde = zeros(1,m);

for index = 1:m

if(diffSampleMeanPayoff(1, index) != 0)
    term = (probabilityDistri(1, actionIndex) - (alpha/2))/ (alpha *
probabilityDistri(1,actionIndex)*(1 - probabilityDistri(1,index)) *
diffSampleMeanPayoff(1, index)*-1);
else
    term = 1;
end
term = term * Sji(1, index);
term = min(1, term);
gammaTilde(1,index) = term;
end

% calculate the update probabilities for the actions not played this round

term3 = (alpha * gammaStar .* diffSampleMeanPayoff .* probabilityDistri .* Sij);

term4 = (alpha * gammaTilde .* diffSampleMeanPayoff * (probabilityDistri(1,
    actionIndex) / (m - 1)) .* ((-1) * probabilityDistri) + 1)).*Sji;

newProbabilityDistribution = probabilityDistri - term3 - term4;

% calculate the updated probability for the action

newProbabilityDistribution(1, actionIndex) = probabilityDistri(1, actionIndex) +
    sum(term3) + sum(term4);

returnCell{1,1} = newProbabilityDistribution;
returnCell{1,2} = sampleMeanPayoffMatrix;

y = returnCell;

```

cost.m

% Script for calculating the cost term

```
function y = cost( totalBWD, allocatedBWD, unitFixedCostBWD,  
unitCongestionCost, congestionAP)
```

```
y = allocatedBWD * (unitFixedCostBWD + (unitCongestionCost /  
((1 - (allocatedBWD / totalBWD))^ congestionAP)));
```

erlangB.m

% Script for calculating Erlang B values

```
function y = erlangB(allocatedBWD, lambda, averageDuration)
```

```
term1ln = allocatedBWD * log(lambda * averageDuration);
```

```
term2ln = gammaln(allocatedBWD + 1);
```

```
numeratorln = term1ln - term2ln;
```

```
index = linspace(0, allocatedBWD, allocatedBWD + 1);
```

```
indexPlusOne = index + 1;
```

```
term3ln = index * log(lambda * averageDuration) -  
gammaln(indexPlusOne);
```

```
term3 = exp(term3ln);
```

```
denominator = sum(term3);
```

```
y = exp(numeratorln) / denominator;
```

performance.m

% Script for calculating performance

function y = performance(erlangBValue, kappa)

if kappa > erlangBValue

 y = 1.0 / (kappa - erlangBValue);

 return

end

y = 1.0 / eps;

References

- [BeP93] Ben-Porath, Elchanan, "Repeated Games with Finite Automata," *Journal of Economic Theory*, Vol. 59, 1993, pp 17-32.
- [BG92] Bertsekas, Dimitri and Robert Gallager, *Data Networks*, 2nd edition, Prentice Hall, Englewood Cliffs, New Jersey, 1992.
- [Bro65] Brownlee, K.A., *Statistical Theory and Methodology in Science and Engineering*, 2nd edition, John Wiley and Sons, New York, 1965.
- [CM84] Carter, M. and R. Maddox, *Rational Expectations*, MacMillan, London, 1984.
- [Con89] Condon, Anne, *Computational Models of Games*, MIT Press, Cambridge, 1989.
- [CSEZ93] Cocchi, R., S. Shenker, D. Estrin, and L. Zhang, " Pricing in computer Networks: Motivation, Formulation, and Examples," *IEEE/ACM Transactions on Networking*, Vol. 1, pp 614-627, December 1993.
- [Cou38] Cournot, A. *Recherches sur les Principes Mathématiques de la Théorie des Richesses*, 1838, translated by N. T. Bacon as *Researches into the Mathematical Principles of the Theory of Wealth*, Hafner, London, 1960.
- [DM92] Dougligeris, Christos and Ravi Mazumdar, "A Game Theoretic Perspective to Flow Control in Telecommunication Networks," *Journal of the Franklin Institute*, Vol. 329, No. 2, pp 383-402, 1992.

-
- [Dub86] Dubey, Pradeep, "Inefficiency of Nash Equilibria," *Mathematics of Operations Research*, Vol. 11, No. 1, pp 1-8, February 1986.
- [Eic93] Eichberger, Jürgen, *Game Theory for Economists*, Academic Press, San Diego, 1993.
- [ES91] Economides, A. A., and J.A. Silvester, "Multi-Objective Routing in Integrated Services Networks: A Game Theory Approach," in *Proceedings of INFO-COM'91*, pp 1220-1225, 1991.
- [Fri86] Friedman, James W., *Game Theory with Applications to Economics*, Oxford University Press, New York, 1986.
- [FK95] Fudenberg, Drew and David Kreps, "Learning in Extensive-Form Games I. Self-Confirming Equilibria," *Games and Economic Behaviour*, Vol. 8, pp 20-55, 1995.
- [Gib92] Gibbons, Robert, *Game Theory for Applied Economists*, Princeton University Press, Princeton, 1992.
- [Gib96] Gibbons, Robert, "An Introduction to Applicable Game Theory," *National Bureau of Economic Research*, Technical Working Paper 199, August 1996.
- [GS83] Grimmett, G.R. and D.R. Stirzaker, *Probability and Random Processes*, Oxford University Press, New York, 1983.
- [HPS72] Hoel, Paul G., Sidney C. Port, and Charles J. Stone, *Introduction to Stochastic Processes*, Houghton Mifflin Company, Boston, 1972.

-
- [HL91] Hsiao, Man-Tung T. and Aurel A. Lazar, "Optimal decentralized flow control of Markovian queueing networks with multiple controllers," *Performance Evaluation*, Vol. 13, pp 181-204, 1991.
- [Jai91] Jain, Raj, *The Art of Computer Systems Performance Analysis*, John Wiley and Sons, New York, 1991.
- [Jor91] Jordon, J., "Bayesian Learning in Normal Form Games," *Games Econ. Behaviour*, Vol. 3, pp 60-81, 1991.
- [KL93] Kalai, E. and E. Lehrer, "Rational Learning leads to Nash Equilibria," *Econometrica*, Vol. 61, No. 8, pp 1019-1045, 1993.
- [KL094] Korilis, Yannis A., Aurel A. Lazar and Ariel Orda, "Architecting Noncooperative Networks," Columbia University, *Center for Telecommunications Research*, Technical Report TR-385-94-32.
- [KLO95a] Korilis, Yannis A., Aurel A. Lazar and Ariel Orda, "Achieving Network Optima Using Stackelberg Routing Strategies," Columbia University, *Center for Telecommunications Research*, Technical Report: TR-384-94-31 (revised July 9, 1995).
- [KLO95b] Korilis, Yannis A., Aurel A. Lazar and Ariel Orda, "Capacity Allocation under Noncooperative Routing," Columbia University, *Center for Telecommunications Research*, Technical Report: TR-372-94-19 (revised December 9, 1995).
- [KLO96] Korilis, Yannis A., Aurel A. Lazar and Ariel Orda, "The Role of the Manager in a Noncooperative Network," *IEEE Infocom '96*, pp 1285-1293, 1996.

- [KL94] Korilis, Yannis A. and Aurel A. Lazar, "Why is Flow Control Hard: Optimality, Fairness, Partial and Delay Information," unpublished manuscript, presented in part at the 2nd ORSA Conference on Telecommunications, Boca, Raton, FL, March 1992. This version dated March 8, 1994.
- [KL95] Korilis, Yannis A. and Aurel A. Lazar, "On the Existence of Equilibria in Noncooperative Optimal Flow Control," *Journal of the ACM*, pp 210-225, May 1995.
- [Kri84] Krishnan, Venkatarama, *Nonlinear Filtering and Smoothing: An Introduction to Martingales, Stochastic Integrals and Estimation*, John Wiley and Sons, New York, 1984.
- [LH64] Lemke, C. E. and J. T. Howson, "Equilibrium Points of Bi-Matrix Games," *SIAM J. Applied Mathematics*, Vol. 12, pp 412-423, 1964.
- [LOP95] Lazar, Aurel A., Ariel Orda and Dimitrios E. Pendarakis, "Virtual Path Bandwidth Allocation in Multi-User Networks (revised version)," *Proceedings of INFOCOM '95*, Boston Massachusetts, pp 313-320, April 1995.
- [MR91] Milgrom, Paul and John Roberts, "Adaptive and Sophisticated Learning in Normal Form Games," *Games and Economic Behaviour*, Vol. 3, pp 82-100, 1991.
- [Nas50] Nash, John, "Equilibrium Points in n-Person Games," *Proceedings of the National Academy of Sciences*, Vol 36, pp 48-49, 1950.
- [NT89] Narendra, Kumpati S., and Mandayam A. L. Thathachar, *Learning Automata, An Introduction*, Prentice Hall, Englewood Cliffs, New Jersey, 1989.
- [Owe82] Owen, Guillermo, *Game Theory Second Edition*, Academic Press, New York, 1982.

-
- [ORS93] Orda, A, R. Rom, and N. Shimkin, "Competitive Routing in Multiuser Communication Networks," *IEEE/ICM Trans. Net.*, Vol. 1, pp 510-521, October 1993.
- [PTVF92] Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical Recipes in C* second edition, Cambridge University Press, New York, 1992.
- [Ros65] Rosen, J. B., "Existence and Uniqueness of Equilibrium Points for Concave N-Person Games," *Econometrica*, Vol. 33, No. 3, pp 520-534, July 1985.
- [Sam96] Samuelson, Larry, "Bounded Rationality and Game Theory," *The Quarterly Review of Economics and Finance*, Vol. 36, Special Issue, pp 17-35, 1995.
- [SF96] Shenker, S. and E. Friedman, "Synchronous and Asynchronous Learning by Responsive Learning Automata," unpublished manuscript, November 1996.
- [Shu84] Shubik, Martin, *Game Theory in the Social Sciences, Concept and Solutions*, MIT Press, Cambridge, Massachusetts, 1982.
- [TM93] Tang, C.K.K. and P. Mars, "Games of Stochastic Learning Automata and Adaptive Signal Processing," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 3, pp 851-856, May/June 1993.
- [TS85] Thathachar, M.A.L. and P.S. Sastry, "A New Approach to the Design of Reinforcement Schemes for Learning Automata," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-1, No. 1, pp 168-175, January/February 1985.
- [VN28] Von Neuman, John, "Zur Theorie der Gesellschaftsspiele," *Math. Annalen*, Vol. 100 pp 295-320. The proof is reproduced in [VNM40].
- [VNM44] Von Neuman, John and Oskar Morgenstern, *Theory of Games and Economic Behaviour*, Princeton University Press, Princeton, 1944.

-
- [Web81] Weber, Robert James, "Noncooperative Games," in *Game Theory and Its Applications*, editor William F. Lucas, Proceedings of Symposia in Applied Mathematics, Volume 24, American Mathematical Society, Providence, Rhode Island, 1981.
- [Zag84] Zagare, Frank, C., *Game Theory, Concepts and Applications*, Sage Publications, Berverly Hills, 1984.
- [ZD92] Zhang, Zhensheng and Chistos Doulgeris, "Convergence of Synchronous and Asynchronous Greedy Algorithms in a Multiclass Telecommunications Environments," *IEEE Transactions on Communications*, Vol. 40, No. 8, August 1992, pp 1277-1281.

IMAGE EVALUATION TEST TARGET (QA-3)

APPLIED IMAGE, Inc.
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved