2005

A comparison of reputation-based trust systems

John Folkerts

A Comparison of Reputation-based Trust Systems

Master's Thesis

John B. Folkerts

Department of Computer Science
Rochester Institute of Technology
jbf6340@cs.rit.edu

April 29, 2005

Thesis Committee:

_____
Dr. Hans-Peter Bischof, Chair

_____
Dr. Stanisław P. Radziszowski, Reader

_____
Dr. Leon Reznik, Observer

Table of Contents

Index of Figures

Index of Tables

Acknowledgment

I thank God, who is the source of all wisdom and knowledge, for the abilities He has given me. This work would not have been possible without the loving support and encouragement of my wife, Sharon, over the last two and a half years. Nor would it have been accomplished without the values and high expectations which my parents instilled in me. Additionally, my appreciation extends to Xerox Corporation for supporting my degree financially, and particularly to MJ Hannan, whose support never waned and smoothed out many of the rough spots. I would like to especially thank my committee chair, Dr. Hans-Peter Bischof, for his insight, encouragement, and assistance during the development of this thesis paper, and also Dr. Stanisław Radziszowski and Dr. Leon Reznik for their input and for serving on the committee.

Soli Deo Gloria.

1 Abstract

Recent literature contains many examples of reputation systems which are constructed in an ad hoc way or rely upon some heuristic which has been found to be useful. However, comparisons between these reputation systems has been impossible because there are no established methods of comparing performance. This paper introduces a simulation framework which can be used to perform comparison analysis between reputation models. Two reputation models, one from Abdul-Rahman and Hailes (ARH) [1], and one from Mui, Mohtashemi, and Halberstadt (MMH) [17] are implemented and compared with regard to accuracy, performance and resistance to deception. In order to improve performance in certain cases, MMH is modified to distinguish the concept of "trust" from the concept of "reputation." Additionally we examine the results of shortening the memory of MMH in order to improve results in environments that are constantly changing.

2 Introduction

2.1 Managing Trust with Reputation Systems

Trust is an elusive concept upon which researchers have repeatedly attempted to model in their implementations of security technologies. The global, hierarchical trust model of X.509 certificates [11] has not proved manageable enough to realize its initial vision for an environment as large and dynamic as the Internet. PGP incorporates a concept called the Web of Trust [22], which is more dynamic, but still requires a lot of human interaction and oversight. SPKI/SDSI [7] uses a dynamic model based on authorizations instead of identities, but this has not seen widespread use, presumably because managing authorization is more difficult than managing identities.

The advent of e-commerce and peer-to-peer (P2P) systems such as Gnutella has driven new interest in reputation systems for managing trust between electronic trade partners and agents in a P2P network. This approach is attractive because it models the way that trust often works in the real world. Additionally, the trust model dynamically maintains itself, and it can work on both global and local levels. This is important because, while pre-defined systems of trust (such as PKI) can be provably secure, the effort to administer them is high and they are not as secure as proven when deployed in the real world. Reputation systems do not promise to be perfectly secure, but can be used where there is no central authority to define security.

2.2 Comparing Reputation Systems

Reputation systems have often been defined in an ad hoc manner [18], yet measuring the success of reputation systems is straightforward and offers insight into the validity of the model. In this paper we create a test framework that can be generically applied to many kinds of reputation systems, and we analyze two which are presented in recent literature. The first reputation system is one proposed by Abdul-Rahman and Hailes [1]. This model depicts *direct* trust and *recommender* trust, as distinguished by the source of the reputation information and assessed in a particular context. We will refer to this reputation system as ARH throughout this thesis, and it is further described in section 6.1.

The second reputation system to be examined will be the one presented by Mui, Mohtashemi, and Halberstadt [17] which defines reputation and trust as two distinct concepts and incorporates reciprocity as the key driver for increases in reputation. This reputation system is described further in section 6.2 and is referred to as MMH throughout this thesis.

2.3 Improving Reputation Systems

Some researchers distinguish between the concepts of reputation and trust: reputation is considered to be the perceived likelihood of success in a transaction, while trust weighs the degree of risk that a party is willing to subject itself to. Thus trust might be considered the minimum reputation required before conducting the transaction.

When viewed this way, it is possible that managing an agent's level of trust will enable it to minimize failure in a hostile environment. By way of analogy, it is quite natural for a person who has been unsuccessful in one aspect of their life to retreat and be more cautious in other areas of their life. We propose a Trust Level Management scheme which mimics this real world behavior of "once bitten, twice shy" with the hopes that it will be able to minimize failures in a simulated environment.

Also, reputation systems are by their nature a reflection of historical behavior. However, historical behavior is sometimes a faulty indicator of future behavior. We propose managing the agent's level of experience in order to tune its ability to respond more appropriately to the present situation, and not become mired in an outdated understanding of its environment. These improvements are made to the MMH model and are described in detail in section 9. We refer to the modified model as TMMH throughout this thesis.

3 Background

The study of reputation has its beginnings in the disciplines of sociology, economics, and game theory. Sociologists such as Deutch and Gambetta have provided the formal definitions and theoretical background for understanding abstract concepts such as trust and reputation and how they operate in our world. Without this work it would be difficult to focus efforts on implementing models that mean something to the people who use them.

Economists have applied repeated games such as the Prisoner's Dilemma (see [5] and section 7.4.1) to the idea of reputation to understand how concepts like reputation, trust, and reciprocity work in the real world. Economists such as Dellarocas, Friedman, and Resnick have been among those expanding these efforts to study the effects of reputation in e-commerce environments.

Computer scientists have taken the lead from these disciplines and have developed mathematical models and network protocols to replicate the concepts defined by their economist and sociologist colleagues. This is the level at which this thesis addresses reputation systems.

3.1 Applications

3.1.1 Internet Reputation Systems

One of the most widely known operating reputation systems is the one which enables eBay's on-line auction system (http://www.ebay.com). This system has been studied in [19], and has been found to be successful despite its shortcomings. Participants are incented to build a good reputation, and it has been found that a good reputation as a seller results in higher prices, and that good reputations do seem to lead to higher transaction volume. Other on-line reputation systems of note include product ratings at Amazon (http://www.amazon.com) and buyer-supplier reputation clearinghouses such as Reputation.com.

3.1.2 Peer-to-peer Networking

Reputation models have been proposed as a method of trust for peer-to-peer network environments in a number of papers such as [2] and [12]. Since these environments have no central authorities, the use of reputation and referral as a mechanism of trust is appealing. Additionally, reputation networks need not have a strong concept of identity in order to operate. For instance, a node's identity can be ensured by public/private key pair that is self-generated and validates later transactions as coming from the same party.

3.1.3 Anti-SPAM

Unsolicited commercial e-mail (SPAM) has generated a great deal of effort to respond to the larger and larger volumes of unwanted e-mail. One of the ways in which this is done is via reputation. One commercial example of this is BrightMail (http://www.brightmail.com), which centrally and automatically maintains reputation information on mail hosts throughout the Internet and allows customers to decline e-mail which originates from mail hosts that have a bad reputation.

3.1.4 E-Commerce

Electronic commerce is a term which loosely describes all kinds of automated purchase capabilities that operate over the Internet. One of the visions of e-commerce is automated competition in purchasing. For instance a buyer may have a choice of multiple suppliers of the same commodity item. The supplier with the lowest price can easily be chosen through an automated bidding algorithm. However, reliability, service, and other elements of trust factor into a business exchange, and these are not so easily automated. A reputation system can automate the vendor selection process in this case by ensuring that vendors with good service, and not just low prices, win the bid. Yu and Singh have applied their model to e-commerce in [24].

3.2 Attacks

It is well known that reputation is something that is easily attacked in the real world, and automated reputation systems are no different. The first issue for a reputation system to contend with is identity, since an easy way to subvert the system is to assume the identity of an agent that has a good reputation. In an automated system public-key cryptography can be of assistance here by allowing agents to identify themselves with a unique public key giving a guarantee that others are indeed communicating with the same agent that has built up a good reputation in the system.

However, this does not solve all problems because, in an Internet environment, it is relatively easy for an agent to shed its poor reputation by assuming a different identity. Friedman and Resnick [8] assert that reputation systems which require agents to "pay their dues" is a fundamental aspect of a successful reputation-based economy.

Also, the concept of reputation both stands and falls on the veracity of agents which provide input to the system. Deception of a number of different kinds might be considered: first, reputations may be overrated or underrated, second deception may be amplified by the perception that the liar agent has a high reputation or has a large amount of experience. It is presumed that a reputation system will completely fail given enough deceptive input, but resistance to deception is a desirable trait of a reputation system.

Deception may be a coordinated effort between two or more agents targeted at specific target agents. This type of attack is very difficult to recognize in a complex system with many agents, and the deceptive information will skew all of the results.

Lastly, the "Sybil attack" is one described by Morselli, et. al. [15] in which an agent behaves with a split personality, maintaining two distinct identities in an attempt to game the system and fool other agents into supporting it with good reputation ratings.

4 Definitions

4.1 Reputation

4.1.1 Intuition on Reputation

Reputation is an intuitive way of establishing and quantifying trust, however formalizing our notions of reputation and trust are necessary in order to implement an automated reputation system.

In the real world, finding a reputable mechanic involves obtaining a reference from someone that you trust. However, reputation is contextual. For instance, no matter how much you trust your mechanic to fix your car, this does not automatically translate to trust when it comes to obtaining medical advice. Likewise, automated reputation systems should account for context, and this is the approach taken by most, including [1] and [17].

Real world reputation may also be garnered personally, or transferred based on group membership. Reputation may be derived from a social network; in other words, our real

world reputation with different individuals varies depending on the network of relationships that tie us together. Similarly automated reputation-based systems may take group dynamics into account.

Reputation may be influenced by direct or indirect information. That is, you may evaluate someone's reputation based on direct encounters with them, or based on behavior that others have observed. One of the reputation systems recounted later [1] in this thesis has similar concepts which it labels *direct* and *recommender* reputation. Of course the only reliable reputation information is direct, because it is based on first hand observation and is not subject to exaggeration, deception, or rumors.

Also, a number of additional issues that influence reputation revolve around identity: should it be bad to have an unknown reputation? Or is it simply a neutral situation? The models studied approach this issue with different solutions, each with its advantages and disadvantages.

Lastly, since the system relies on individuals telling the truth about each other, deception is a significant challenge. A society full of compulsive liars will quickly have no utility for reputation. Similarly, automated reputation models must be resistant to deception, and it is anticipated that there is a minimum level of cooperation necessary for a reputation-based trust system to function.

4.1.2 Formal Definition

Many, such as Mui, et. al. [17], have equated reputation and trust, treating them as synonyms in their work. However, not all treatments have done this, so we define trust and reputation separately. Reputation ρ_b^a is defined as the *expectation of success that an agent a has that agent b will act in a beneficial way towards a.*

This definition of reputation follows closely upon the work done by Diego Gambetta, which is also referenced in [14], which defines trust as a probability: "...trust (or symmetrically distrust) is a particular level of the subjective probability with which an agent assesses that another agent or a group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it)" [9].

However, in order to keep the simulation system generic, we will not suppose that ρ_b^a is always a probability with values between 0 and 1. Reputation ρ_b^a may be a member of an ordered set of values V which describes both trust and distrust, or it may be a history of interactions, or a cumulative rating with arbitrary values. However, we will define some functions that must be possible within a given reputation system in order to work in our simulation.

- The function *vouchFor(a,b)* returns $\rho_b^a \in V$ for a given pair of agents *a* and *b*. Each agent must be capable of expressing its expectation of success when queried.

- Reputation systems must implement the operators <, >, and = to allow direct comparison between reputation values.

- Reputation values V must be partially ordered, i.e. they must exist on a continuum between a defined ρ_{min} and ρ_{max} where $\rho_{min} \leq \rho_b^a \leq \rho_{max}$ for all agents a and b. Again, ρ_{min} and ρ_{max} need not be single values, but could be sets that are evaluated such that the comparison operators above are valid.

- If ρ_k^i represents the reputation value that agent *i* assigns to agent *k*, and ρ_k^j represents the reputation value that agent *j* assigns to agent *k*, then the function $combine(\rho_k^i, \rho_k^j)$ must exist within a reputation system to allow a third party to combine reputation ratings from two different agents. Figure 1 depicts the communication that allows agent *a* to combine ratings from agents *i* and *j* to develop an combined reputation about agent *k*.

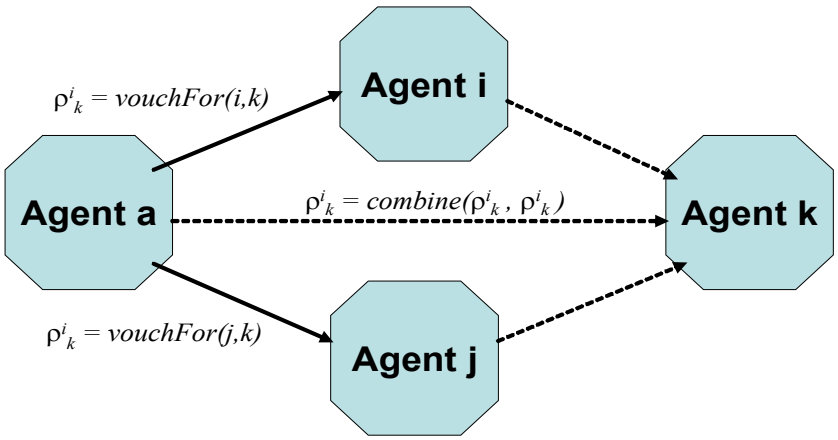

Figure 1: Combining Reputations Ratings from Two Different Agents

While different reputation systems may have their unique characteristics, this simple set of functions and operators is very prevalent in the systems described in the existing literature. We will now go on to describe our notion of trust.

4.2 Trust

4.2.1 Intuition on Trust

Trust is defined in everyday parlance as "Firm reliance on the integrity, ability, or character of a person or thing" [10]. While this definition strikes a hopeful tone, the alternative view is voiced by the security professional's aphorism "trust is for suckers." [6] Either way, we see the elements of a person's inability to fully predict or control the situation into which they enter. Trust is the decision to act despite uncertainty with a hope that the outcome will result in your favor.

In the computer world, a trusted system is one in which verifications have taken place in advance and a decision is made to connect to the system in an unrestricted way. Trusted domain hosts are those hosts for which no further scrutiny is required in advance of future interactions. And trusted certificates are those which have been signed by a known certificate authority and therefore it has been decided that no further proof of authenticity is required.

It is widely believed that cooperation in society is based on mutual trust. A non-trusting world would be one that is paralyzed, unable to move forward when there are no guarantees of behavior on the part of our neighbors. Distrust on the other hand is the inclination to avoid uncertain interactions with the expectation that they would be harmful. Typically trust of a person may be founded on reputation, social mores, legal recourse, or an altruistic belief.

4.2.2 Formal Definition

For a formal definition of trust we follow Marsh [14] in referencing Morton Deutsch, a pioneer in studies of cooperation and competition. Trust for Deutsch occurs when there is an ambiguous set of choices and a good or bad result hangs in the balance and solely depends on another person. Trust is to go down the path expecting a good result. Distrust is avoiding the path. Deutsch also defines the trust dilemma as being weighted by a bad outcome which would be worse than the benefit to be derived.

Marsh formalizes this understanding by establishing upper and lower trust thresholds. If trust is quantified as τ, there is an upper threshold τ_U, above which an agent would be trusting of the interaction. The lower threshold τ_L is the point at which the agent distrusts the interaction with another, and any trust values τ such that $\tau_L < \tau < \tau_U$ are considered ambiguous [14].

First, since Marsh equates trust with reputation, as noted above, we define trust as existing on the same continuum as reputation, so that $\rho_{min} \leq \tau \leq \rho_{max}$ Also, we will simplify Marsh's definition by making $\tau_L = \tau_U$, therefore eliminating ambiguity about behavior and defining trust as the threshold at which a decision can be made concerning an ambiguous choice. In the context of a reputation system, trust τ_a is then *the minimum expectation of success that agent a is willing to accept prior to initiating an interaction with another agent*.

5 Survey of Related Research

5.1 Theoretical Models of Reputation

5.1.1 Marsh

For his doctoral thesis in computer science [14], Stephen Marsh investigates the notions of trust in various contexts and develops a formal description of its use with distributed, intelligent agents. The model he creates is complex, but draws on many relevant real world phenomenon. In doing so he defines trust in three categories:

- *Basic trust* which is considered a value from -1 to 1 that encompasses the agent's experiences and basic disposition to trust. This is called “trust level” later in this thesis.

- *General trust* is the rating from -1 to 1 which indicates a specific agent's trust in another given agent. In this definition 0 represents no trust or distrust, positive values represent trust and negative values represent distrust. Other papers recounted here refer to Marsh's

general trust as reputation, i.e. the expectation of success in a future encounter with another agent.

- *Situational trust* is the measure of how much trust will be conferred given a certain situation. Other models such as [1] refer to this as a "reputation context", and this thesis only examines single context situations in order to reduce complexity.

Trust is gained by knowledge, i.e. what one agent knows about another. The knowledge that agent x has about agent y is defined as $K_x(y)$ and can be used to help x form a trust opinion about y. Knowledge may be boolean (i.e. x knows y or x does not know y) or it may be represented in a way to describe partial knowledge. Other authors refer to this as interaction history.

Marsh also defines a notion of utility within his agent economy, where $U_x(\alpha)$ is the amount of utility that agent x gains from situation α. It is assumed that rational agents seek to maximize utility, as in the real world. Along with utility, Marsh distinguishes importance $I_x(\alpha)$ to indicate the relative importance of situation α to agent x. Marsh maintains that this is different than utility: for instance a lunch has a utility that may be measured in terms of calories, but the importance of the lunch to a person is dependent on the last time they have eaten.

Other aspects that Marsh elaborates on include the following:

- Competence: The level of experience that agent x has with agent y
- Group membership: An agent's level of trust towards agents from the same group
- Agent disposition: An agent's inclination towards cooperation
- Reciprocation: The modification of behavior based on a recent history of cooperation

It is apparent by now that Marsh's model is very complex and therefore is not considered applicable in most real world scenarios. Unfortunately his model displays difficulties at the extreme values of -1 and 1 and also at 0, and struggles to rationalize the separate concepts of distrust and trust. And although the subject of hearsay information does appear, Marsh does not incorporate the ability for agents to share information about each other. However, this early work analyzing the application of reputation to autonomous agent networks is valuable for the definitions presented and its comprehensive synthesis and application of economics, sociology, and game theory to the area of trust models for intelligent agents.

5.1.2 Abdul-Rahman and Hailes

Abdul-Rahman and Hailes proposed in [1] that the concept of trust be extended to incorporate input from surrounding agents. They divide trust into two categories: *direct trust* and *recommender trust* and provide an ad hoc algorithm for combining the two into a single evaluation of trust between two agents for a given context. This algorithm (referred to as ARH) is recounted extensively below.

5.1.3 Mui, Mohtashemi, and Halberstadt

Mui, Mohtashemi, and Halberstadt have proposed a computational model in [17] (referred to later as MMH) which separates the concepts of trust and reputation into the following: reputation is the "perception that an agent creates through past actions about its intentions and norms" and trust is "a subjective expectation an agent has about another's future behavior based on the history of their encounters." Mui, et. al. use a Bayesian approach to calculate probabilities of an agent acting in one way or another and express reputation as a probability of success ranging from 0 to 1. Mui does not examine effects of deception in this model, nor is there any concept of what Marsh refers to as basic trust – the minimum probability of success that an agent is willing to accept for an encounter. This approach is recounted more thoroughly below.

5.1.4 Yu and Singh

Yu and Singh have published extensively on the topic of reputation [23, 24, 25, 26]. Their most recent approach applies Dempster-Shafer theory which has the advantage of explicitly dealing with evidence in terms of pros and cons, and contains a construct for dealing with uncertainty in an interaction.

In Yu and Singh's model, a set of propositions θ are defined such that $\theta=\{T,\neg T\}$ Proposition T is the probability of success, $\neg T$ is the probability of failure, and $\{T,\neg T\}$ represents the level of uncertainty. Using Dempster-Shafer Theory, they define a basic probability assignment (BPA) which is a function m such that

$$m(\{T\})+m(\{\neg T\})+m(\{T,\neg T\})=1$$

Yu and Singh then define their belief function to be m and can use that to gauge the trustworthiness of an agent in the community. This is similar to Marsh's separate concepts of trust and distrust, and like Marsh, Yu and Singh establish both lower and upper thresholds for trust. Each agent also maintains a quality of service (QoS) metric $0\le s_{jk}\le 1$ that equates to an agent j's rating of the last interaction with agent k.

Now in order to use Dempster-Shafer with a network of distributed agents, Yu and Singh make use of the orthogonal sum $\oplus$ as defined in [21] such that $m(\emptyset)=0$ and

$$m(A)=\frac{\sum_{X\cap Y=A} m_1(X)\cdot m_2(Y)}{1-\sum_{X\cap Y=\emptyset} m_1(X)\cdot m_2(Y)}$$

for all non empty $X, Y, A\subset\Theta$ Thus, belief functions m_1 and m_2 can be combined into m with $m = m_1 \oplus m_2$.

An example of combining BPAs is given in Table 1. Given two agents, each communicates a reputation basic probability assignment, m_1 and m_2 respectively. The first agent has a basic probability assignment of 0.4 for success, 0.0 for failure, and 0.6 for uncertainty. The second

agent has a BPA of 0.1 for success, 0.0 for failure, and 0.9 for uncertainty.

	$m_2(\{T\})=0.1$	$m_2(\{\neg T\})=0$	$m_2(\{T,\neg T\})=0.9$
$m_1(\{T\})=0.4$	0.04	0	0.36
$M_1(\{\neg T\})=0$	0	0	0
$M_1(\{T,\neg T\})=0.6$	0.06	0	0.54

Table 1: Combining Belief Functions in Dempster-Shafer Theory

For the specific set of propositions that Yu and Singh have laid out, we can apply the orthogonal sum definition above to the three possible cases of belief and uncertainty:

$$m(\{T\})=\frac{m_1(\{T\})\cdot m_2(\{T\})+m_1(\{T,\neg T\})\cdot m_2(\{T\})+m_1(\{T\})\cdot m_2(\{T,\neg T\})}{1-m_1(\{T\})\cdot m_2(\{\neg T\})}$$

$$m(\{\neg T\})=\frac{m_1(\{\neg T\})\cdot m_2(\{\neg T\})+m_1(\{T,\neg T\})\cdot m_2(\{\neg T\})+m_1(\{\neg T\})\cdot m_2(\{T,\neg T\})}{1-m_1(\{T\})\cdot m_2(\{\neg T\})}$$

$$m(\{T,\neg T\})=\frac{m_1(\{T,\neg T\})\cdot m_2(\{T,\neg T\})}{1-m_1(\{T\})\cdot m_2(\{\neg T\})}$$

Upon combining these belief functions, we get the following results:

$$m(\{T\}) = 0.04 + 0.36 + 0.06 = 0.46$$
$$m(\{\neg T\}) = 0$$
$$m(\{T, \neg T\}) = 0.54$$

We interpret this to mean that, even though our expectation of failure is 0, we cannot anticipate success since uncertainty has the highest resulting value (0.54). Thus, our understanding of the evidence leads us to believe that the outcome is uncertain.

Like ARH and MMH, Yu and Singh distinguish between local history and the information passed by other agents as references, and they construct a TrustNet of limited depth whereby reputation information is compiled. Some limitations to the model exist however: since Dempter-Shafer calculations quickly become computationally intensive, the number of neighbors and the depth of the referral chains must be restricted to small numbers.

Yu and Singh simulate environments with between 100 and 500 agents, limiting agents to 4 neighbors and the number of referrals to a depth of 6. Yu and Singh define the system view of an agent A's reputation to be the cumulative belief function of A's neighbors $a_1 \ldots a_k$ so that

$\beta_A = \tau a_1 \oplus \ldots \oplus \tau a_k$ where τa_i is the individual belief of agent a_i. This is then used to measure the number of interactions required to "bootstrap" or initialize the system, and the performance of the system when some nodes decide to abuse their high reputation.

Additionally, they simulate the effect of grouping to allow the model to scale to large numbers of agents.

In further work Yu and Singh go on to study the effects of deception in reputation systems [25], and apply their reputation model to electronic commerce [24].

5.2 Simulated Models of Reputation

5.2.1 Sen and Sajja

Sen and Sajja [20] present a method for ensuring robustness of a reputation model that is used to select processor resources. This model is unlike others in that it uses service selection as its measure of success instead of the frequently used Prisoner's Dilemma. In this model, an agent selects the service provider that has the highest reputation from a pool. The key innovation described in this system is the dynamic adjustment of the size of the selection pool, and how it can be adjusted based on a threshold that guarantees success probabilistically despite the existence of a set of liars within the population of agents.

Reputation is modeled using a reinforcement learning algorithm such that a reputation estimate $0 \le e_{ij}^{t+1} \le 1$ represent an agents estimate of reputation at interaction *t*. It is initialized at 0.5 and updated after every interaction using the following equations:

$$e_{ij}^{t+1} = (1-\alpha_i)\, e_{ij}^{t} + \alpha_i r_t$$
$$e_{ij}^{t+1} = (1-\alpha_o)\, e_{ij}^{t} + \alpha_o r_t$$

The first equation represents the estimate based on interactions and the second based on observation. α_i and α_o represent the learning rates assigned to direct interaction and simple observation respectively, and it is given that $\alpha_i > \alpha_o$ since they assume direct interaction to be a more reliable measure of reputation.

Given a population of *N* user agents and *P* service providers, Sen and Sajja assume that $l \le N/2$ of the population are liars (agents which always invert the ratings of the other service agents). The algorithm presented minimizes *q* (the size of the selection pool) such that the following equation holds:

$$\sum_{i=max(\lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor + 1)}^{P} \frac{\binom{N-l}{i}\binom{l}{q-i}}{\binom{N}{q}} \;\geq\; g$$

This produces a selection pool size which will guarantee that the desired threshold of truthful agents are queried when selecting service providers.

The algorithm results in good performance when *g* is set to 0.95 and the number of liars in a set of 40 agents remains at 16 or below. Increases in the number of liars beyond this level cause the performance of the entire system to quickly degrade.

5.2.2 Morselli, Katz, and Bhattacharjee

Morselli et. al. [15] provide a game-theoretic framework for assessing the robustness of trust-inference protocols, otherwise known as *trust propagation protocols* or *reputation systems*. This framework is similar to the framework shown in this thesis in that it is intended for objective comparison between reputation systems, was designed to be flexible enough to accommodate a variety of reputation models, and focuses on distributed systems as opposed to those reputation systems which require a trusted, centralized authority.

The framework which Morselli et. al. describe defines the following characteristics:

- Each agent in the framework is known by its *pseudonym*. Pseudonyms are distinct, easy to generate by the agent themselves, and impossible to impersonate. It is assumed that these characteristics would be provided by using public key cryptography.

- The adversary is assumed to have almost complete control over the system. Adversaries are not allowed to interfere with transactions between honest agents, however, they are able to mount coalition attacks (attacks that cooperate with other malicious agents) and Sybil attacks (attacks that allow the adversary to simulate the actions of multiple agents). Likewise, adversaries might lie when it comes time to report reputation information to the rest of the society.

- Agents participate in simultaneous two party games which are modeled after Prisoner's Dilemma [5], with scores assigned as in Table 2. For more background on Prisoner's Dilemma, see section 7.4.1.

	Cooperate	*Defect*
Cooperate	1,1	-1,2
Defect	2,-1	0,0

Table 2: Prisoner's Dilemma Score Table

The work presented in [15] differs from this work in that it requires the agents to broadcast reputation information to each other in order to report reputation information to neighbors. Also, their simulation is modeled after repeated, simultaneous Prisoner's Dilemma, and their focus is more on robustness than on performance of the protocols examined.

5.3 Applied Models of Reputation

5.3.1 NICE

Lee, Sherwood, and Bhattacharjee present a distributed scheme for representing trust called NICE in [13] which is intended to apply to a variety of applications which include media streaming, multiparty conferencing, and other peer-to-peer applications.

NICE applications use must provide local resources in order to gain access to remote resources. Transactions in NICE are represented by secure exchange of certificates which can

then be used to acquire resources in the future. Each agent is identified using a PGP style identifier along with a public key. In that way impersonation is eliminated and the only remaining difficulty is the ability to generate a new identity. Lee, et. al. define non-cooperative agents as those which attempt to get "free" resources by issuing cookies (digital certificates) that they do not intend to redeem. The goal of the system is to identify those non-cooperative agents to avoid being cheated.

Trust is defined in terms of a directed trust graph $A_0 \rightarrow A_1 \rightarrow ... \rightarrow A_n$ on *T*, where edge values are weighted based on how much node A_i trusts node A_{i+1}. Agent A's trust of agent B is denoted as $0.0 \leq T_A(B) \leq 1.0$. If $T_A(B) > T_B(A)$, then A will only barter with B if B offers more resources than it gets in return. These resources are represented with cookies that are valued on a scale from 0 to 1. If A wants to infer a trust value for B, it will choose the strongest path along *T* which exists between A and B and select the minimum trust value along that path. This path can be easily calculated using depth-first search.

Lee et. al. show results that indicate that the NICE system is scalable, because the average number of nodes visited remains relatively constant even as system size increases. However, the success ratio and number of nodes visited vary proportionally depending on how many cookies are stored and the degree of neighbors which each node has.

This work is notable for the fact that population sizes up to 2048 nodes were used. Additionally, Lee et.al. show that cooperative agents can establish cooperative groups despite the fact that there may be a large percentage of non-cooperating agents in the environment.

5.3.2 Kamvar, Schlosser, and Garcia-Molina (EigenRep)

EigenRep [12] is a reputation system which makes use of matrices of reputation information which are maintained and stored by agents in their system. The authors explicitly target their system at providing reputation for peer-to-peer systems where malicious peers can generate illegitimate files for sharing and the general population of peers have no way of distinguishing illegitimate files from the legitimate ones.

Kamvar et. al. describe the following design considerations that are relevant to each of the reputation systems recounted here:

- The system should be self-policing
- The system should maintain anonymity
- The system should not assign any profit to newcomers
- The system should have minimal overhead
- The system should be robust in the face of malicious collectives of agents

As an estimate of trustworthiness, each agent *i* rates its peer agent *j* with the formula $s_{ij} = sat(i, j) - unsat(i, j)$ -- the difference between the number of satisfactory interactions *i* has had with *j* and the unsatisfactory interactions. Each agent maintains a matrix C of normalized local reputation values such that C is comprised of elements c_{ij} where

$$c_{ij} = \frac{max(s_{ij}, 0)}{\sum_j max(s_{ij}, 0)}$$

c_{ij} is the normalized reputation that agent *i* has assigned to agent *j*. This method ensures that the reputation matrix is protected against exaggeration and deception because each of agent *i*'s ratings are weighted in terms of the other ratings that agent *i* has provided.

The reputation system is then applied to simulations of Gnutella file distributions. They conclude that their model is able to bound downloads of illegitimate files despite populations of malicious peers that are as high as 70%.

5.3.3 Aberer and Despotovic

Aberer and Despotovic describe a reputation system for Peer-to-peer (P2P) systems in [2] which is intended to meet needs that are left unfulfilled by other reputation systems: scalability to large numbers of nodes, and reduced amounts of required data storage and network communications. In order to meet these goals, they consider the problem in three dimensions:

- Global trust model: how may an agent be described as trustworthy?
- Local algorithm to determine trust: how may an agent locally approximate the global reality of the trust system?
- Data and communications management: how is the data distributed and stored and what kinds of communications must take place in order for the trust model to work?

In order to reduce the amount of data stored and communicated, the model works on a binary rating system – an agent is either considered trustworthy or not. The only data that is exchanged is then reduced to complaints about an agent registered by other agents which have interacted with it. So a complaint *c(q,p)* is a complaint registered by *q* about agent *p*, and an assessment of *p*'s trustworthiness, *T(p)*, is constructed with the following:

$$T(p) = |c(p, q)| \times |c(q, p)|$$

A high value of *T(p)* is an indication that agent *p* can not be trusted.

Complaint data is stored in a distributed data format called a P-Grid [3], which is essentially a distributed binary search tree as in the figure below. At the leaf level of the tree, agents store complaints about and by a certain agent. Agents are mapped into the tree structure based on their agent identifier (an integer). In this way, given *n* agents, comprehensive complaint information about an agent can be obtained in $O(log(n))$ time, and each agent need only store $O(log(n))$ complaint data. This means that this trust model promises to scale well over large numbers of agents. Figure 2 shows an example P-Grid for n=8 agents.

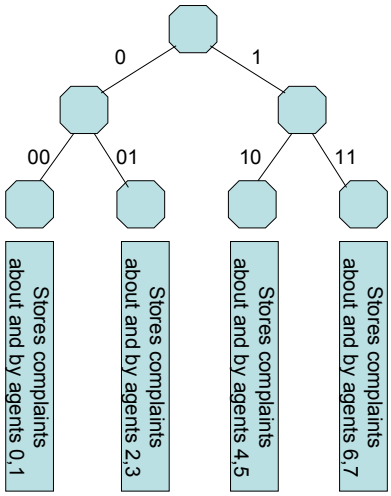

Figure 2: Example P-Grid

Now, this model has the disadvantage that one agent, say agent *a*, carries the entire data set for another, say agent *b*. If agent *a* is malicious, agent *b*'s reputation is ruined forever. So Aberer and Despotovic define a threshold π_{max} for the probability of malicious behavior in the system, and use that threshold to decide how many replicas need to be instantiated to ensure that one agent can not thwart the whole system.

Aberer and Despotovic perform simulations on their model and find that it is robust against "cheater agents," i.e. agents which complain about other agents who have not wronged them with a constant probability. They find their model somewhat less robust against like populations of cheater agents who complain with a variable probability.

6 Implemented Reputation Systems

6.1 Abdul-Rahman and Hailes Reputation Model

Abdul-Rahman and Hailes [1] have proposed a reputation model which draws in part from prior work defining trust in a computational manner [14]. Their main objective is to allow trust decisions based on a variable scale, not the binary decision usually employed by cryptographers, but more a subjective degree of belief by which agents make choices.

Their working definition of reputation relies on the experiences of the agent themselves, part of which may be information gathered from other agents. This allows them to generalize reputation information from the agent and also the agents neighbors.

In their definition, an experience is either information gathered in a personal experience or reliance on a recommendation from another agent. The first is known as *direct trust* while the second is referred to as *recommender trust.* Direct trust is associated with agent's name, a context, and a degree of trust, which is defined as one of the following: Very Trustworthy, Trustworthy, Untrustworthy, or Very Untrustworthy. Evaluation of recommender information is defined in a similar way.

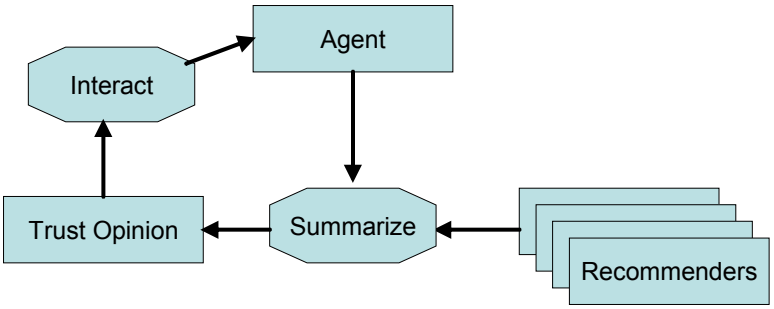

Figure 3: Abdul-Rahman and Hailes' Trust Model

The agent then is able to keep a set *Q* of 4-tuples of information related to their own experiences, where *Q* is a set of cross products of contexts, agents, and subjective reputation ratings. Likewise, the agent keeps a set *R* of information related to recommenders. The algorithm allows the agent to keep adjustment values on recommenders which do not see the world as they do (i.e. the recommender's "very trustworthy" is only a "trustworthy" to the agent). This set R consists of cross products of contexts, agents, and adjusted recommendation information. Information from *Q* and *R* are combined using a statistical formula, and a single evaluation of trust is obtained.

Abdul-Rahman and Hailes account for a number of the design concerns associated with reputation systems. They provide for context, they consider that not all recommender information will be reliable, so they allow adjustments when evaluating it. Also, they allow a means of bootstrapping new agents into the network, by providing them with a number of pre-trusted entities to provide them with recommender information as they build their own set of experiences.

6.2 Mui, Mohtashemi, and Halberstadt's Reputation Model

Mui, Mohtashemi, and Halberstadt [16, 17] provide a rather different notion of reputation than ARH. Their idea is based on the concept of reciprocity and its role in cooperation as described by social scientists. Additionally, they differentiate between trust and reputation, since reputation is one measure whereby trust can be gained, i.e. a good reputation is not always sufficient to inspire trust. The following diagram depicts the relationship that Mui et.al. see between reputation, trust, and reciprocity.

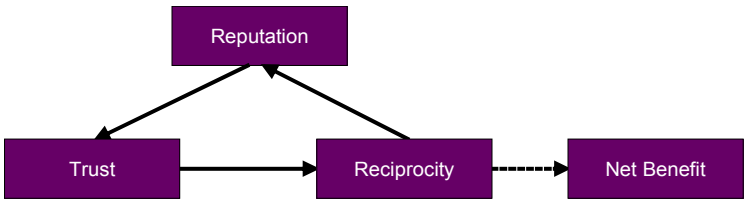

Figure 4: Relationships between Reputation, Trust, and Reciprocity

The following definitions are given in the paper for these three concepts:

- **Reciprocity:** mutual exchange of deeds (such as favor or revenge)

- **Reputation:** perception that an agent creates through past actions about its intentions or norms.
- **Trust:** a subjective expectation an agent has about another's future behavior based on the history of their encounters.

Mathematically, this system is represented using Bayesian probability theory. Each agent keeps track of a set of encounters, along with their results. Reputation is a probability measure from 0 to 1 on the likelihood of another agent to reciprocate. Reputation scores are kept for each pair of agents in the model. Observed encounters count equally with experienced encounters, and are kept in the same set of encounter data.

Trust, τ_{ab}, is defined as an agent *a*'s expectation for reciprocation by a particular agent *b* in a given context. Agent *a* uses its encounter history with agent *b* to calculate a reputation $\hat{\theta}_{ab}$ using a Beta prior distribution. This process is as follows:

The encounter history D_{ab} is the set of individual encounters $\{x_{ab}(1), x_{ab}(2), \ldots, x_{ab}(n)\}$.

The prior probability uses a Beta distribution function, $p(\hat{\theta}) = Beta(c_1, c_2)$ where c_1 and c_2 are parameters determined by prior assumptions such that $0 \le c_1 \le 1$ and $0 \le c_2 \le 1$. An estimator for reputation is the proportion of successful encounters *p* to total encounters *n,* so

$$\hat{\theta}_{ab} = \frac{p}{n}$$

Assuming that the probability of successful encounters is independent, the likelihood of *p* cooperations in *n* attempts (probability of cooperation) can be modeled as

$$L(D_{ab} | \hat{\theta}) = \hat{\theta}^{p} (1 - \hat{\theta})^{(n-p)}$$

After combining the prior and the likelihood, we get a posterior estimate of

$$p(\hat{\theta} | D_{ab}) = Beta(c_1 + p,\ c_2 + n - p)$$

Trust τ_{ab} is defined as the probability of cooperation on encounter *n+1*, or the probability that $x_{ab}(n+1) = 1$. This means that at *a's* next encounter with *b*, agent *a* will estimate the probability of cooperation from agent *b* to be

$$\tau_{ab} = p(x_{ab}(n+1) = 1 | D_{ab}) = \frac{c_1 + p}{c_1 + c_2 + n}$$

Mui's model accounts for different levels of confidence, however, it does not differentiate between observed encounters and real encounters, assuming that all observations are first hand and not transferred by "word of mouth." For the case in which two unknown agents do not have any encounter history, Mui chooses a uniform distribution with $c_1 = 1$ and $c_2 = 1$, so that $\tau_{ab} = 0.5$ initially.

7 Simulation Framework

The objective of the thesis is to implement the ARH and MMH reputation systems, simulate their behavior in a variety of circumstances, and create metrics that can be used to compare the performance of these models. This section identifies the issues in developing a suitable simulation framework and formally defines that framework and its accompanying metrics.

7.1 Issues

The intuitive notions of reputation can be translated into reputation models for network systems. A subset of design issues surrounding these models include the following:

- Will the system store reputation in a central repository or in a distributed database?
- How will reputations be calculated?
- What constitutes positive behavior or negative behavior?
- Is there a concept of redemption once an agent has gained a bad reputation?
- Can we prevent or discourage agents from changing their identity to wipe their bad reputation clean?
- How will the reputation system combat deception?
- How long does the system take to reliably identify disreputable agents?
- What is the performance of requests for reputation information?

Depending on the application, trade offs may be made with these design decisions. One of the advantages of the reputation system is that since it is greatly automated, one can simulate the dynamics of this model as it would be deployed in the real world. Additionally, measuring the model's security can be done by comparing the results of individual queries (e.g. "Is agent X trustworthy?") to global knowledge about agent X. The model also has the advantage of years of research from other disciplines such as economics, game theory, and statistics.

7.2 Design Goals

The goals of the test framework are as follows:

- **Simplicity:** The framework must be simple in that the results should be easily explainable. Simplicity also promotes an objective evaluation and comparison of the reputation models.
- **Realism:** The framework should model real world scenarios as much as possible, without compromising the simplicity of the system.
- **Measurable:** Because this is a distributed system, information is spread throughout a network of independent agents. However the test framework must be able to measure the entire system's results.
- **Re-usable:** The framework's design should be generic enough to allow it to be re-used for simulation of other reputation models.

7.3 Characteristics

The results of empirical comparison of reputation models largely depend on the framework that is used to test the models. If the comparison was to be done given a certain context or application, the test framework should model that application. For instance, we would

assume that in order to compare reputation data for the purpose of spam filtering, one would create a framework similar to what a spam filter would need to do and feed the framework real world data if at all possible.

However, in this case we are taking a more generic approach, so we must design the framework in terms of characteristics which have a real world analog. So, if we assume that spam filters recognize that it is unlikely that spam is generated from your local domain, then a simulation framework must have a concept of a group which can influence the reputation results.

The following things have been outlined as worthwhile characteristics to emulate with the framework:

- **World size:** Ability to measure performance depending on the number of agents in the virtual world.
- **Groups:** The ability to group agents in order to allow cooperation.
- **Agent Strategies:** The results of various strategies should be comparable. By strategy we mean the way an agent behaves in the simulation.
- **Deception:** The ability to inject deceptive agents into the simulation to see what the impact is on the level of success.
- **Agent-to-agent interactions:** Clearly an agent in a distributed reputation model which requests information from neighbors is likely to get a better picture of reputation than one which does not request information.
- **Level of hearsay evidence:** Models will perform differently depending on the amount of information that has been incorporated by hearsay (i.e. indirect experience reported by neighbors) versus individual experience.
- **Level of Trust:** The ability to vary the degree to which an agent will "step out in faith" given a limited amount of experience.

7.4 Protocol

Reputation and trust potentially involves many different variables. In order to simplify simulations and their analysis, researchers have historically turned to game theory to create simple tests of success and failure for their reputation models [e.g. 14, 17, 23, 15], and in particular they have used Repeated Prisoner's Dilemma [5]. The framework presented here creates a different binary game which we refer to as The Resource Game which is described below, but first we will review Repeated Prisoner's Dilemma.

7.4.1 Repeated Prisoner's Dilemma

The Prisoner's Dilemma [5] is a hypothetical situation where two criminals are arrested and put into separate jail cells. The authorities want each prisoner to testify against the other. The dilemma is that if a prisoner cooperates with their partner in crime and says nothing, they are rewarded with a shorter sentence. If the prisoners both defect and try to incriminate the other, they are in a neutral position. If Prisoner A cooperates and keeps quiet, while Prisoner B defects, then this is a bad situation for A, and a very good situation for B since he will receive a shorter sentence in return for his testimony. The following payoff matrix describes

the outcomes of one iteration of Prisoner's Dilemma (the rows are Prisoner A's actions while the columns are Prisoner B's choices):

	Cooperate	*Defect*
Cooperate	b,b	d,a
Defect	a,d	c,c

Table 3: Scoring for Prisoner's Dilemma

Points are assigned to score each interaction such that a > b > c > d. Prisoner's Dilemma is a simple model which describes both the benefit and pain associated with trust and betrayal. Prisoner's dilemma is not without it's critics, however. Marsh quotes Argyle [4] as identifying the following differences between Prisoner's Dilemma (PD) and real life experience:

1. *Play is simultaneous*, which means that each player is required to make a move. In the real world a player can "go home" or not play at all, in which case their score would be an even 0. Since the concept of trust is of primary concern to our study, distrust should be punished by lack of opportunity as it is in the real world, and not rewarded as it is in PD.
2. *The game is too abstract*. A valid concern, but one that is generally overlooked in the generic analysis of reputation models.
3. *The prisoner's dilemma assumes no communication.* However communication increases cooperation, and is certainly characteristic of reality. Reputation models seek to use communication to eliminate uncertainty and increase cooperation.
4. *The players are strangers to each other.* In real life there may exist some prior experiences which will inform behavior. The point of reputation systems is to collect experience and use it to inform behavior, and this runs counter to the assumptions that go into PD.
5. *Social norms are absent.* While automated societies may not have social norms as we think of them humanly, they should bear some resemblance to the norms of the human society which they support. So, for instance, although it is very possible to create malicious ActiveX controls and present them on the Internet for widespread consumption, only a small percentage of actual ActiveX controls found on the Internet are malicious. This is because of the real life consequences of anti-social behavior that exist outside of the game.

To these criticisms, we will add one more. Prisoner's Dilemma assumes that both players have opportunities in the interaction to lose in an equivalent fashion. For instance, if Party A transacts business through Party B's e-commerce site, and Party B does not fulfill the order, Party A is out the amount of money spent, and Party B has acquired the money along with a non-tangible loss reputation. However, in Internet transactions, often one party has nothing to lose except for its reputation, and this loss is not necessarily comparable to the real loss experienced by the other party. In a peer-to-peer network, there is no tangible loss to a user who spreads a virus which masquerades as a legitimate file. The only loss occurs to the unfortunate user who acquires the virus.

7.4.2 The Resource Game

In ordert to counter these shortcomings, our simulation framework is built upon a simple repeated game that reflects a model for resource distribution. Prisoner's Dilemma relies on the idea that the two players are helplessly thrown into the dilemma with each other, which is again, not a reflection of how transactions take place in the real world. The natural use of reputation in a resource-oriented model would be to select a resource provider from a pool of candidates, and not use reputation to inform the strategy to be taken once a candidate is selected for you. In the first scenario, agents are free to use principles of reciprocity to guide way suppliers are selected.

In order to work around the shortcomings of PD, the simulation framework was built on the following game of binary outcomes. In any given encounter there is an *customer* agent and a *supplier* agent. All agents have the capability to be both *customers* and *suppliers*. Each customer desires a particular kind of resource and must find a supplier which can supply that resource. For any given turn, the customer randomly selects *c* choices which could potentially supply the desired resource and requests reputation information from *r* referrers. Then the *customer* uses its reputation model to assess the reputation information and select the *supplier* which it believes is most likely to provide a successful response. Once the *customer* initiates an encounter, the *supplier* responds in one of two ways, either with success or failure, based on a pre-defined strategy. Once the encounter is over and the results are known, the information is assimilated into the customer's local reputation information which will then be available to inform the next interaction or query from another agent.

The object of the game, like Prisoner's Dilemma, is to maximize success while minimizing failure. The following payoff matrix is assumed (the rows are the *customer's* actions while the columns are the *supplier's* choices):

	Success	*Failure*
Interaction	+1,0	-1,0
No Interaction	0,0	0,0

Table 4: Scoring for the Resource Game

Thus, an agent's score for the game can be calculated from successes – failures. The game is not zero-sum, but rather assumes that each agent has unlimited resources to give. Also, all moves in the game happen sequentially – in other words the results of encounter *n* are available to inform choices in encounter *n+1*.

7.4.3 Formal Specification of Simulation Environment

Assume that the simulated environment is made up of *N* agents. This environment is designated as a world *W* and consists of a set of agents a_i so that $W = \{a_1, a_2, ..., a_N\}$. Each agent a_i is assigned a resource which it can provide from a set of resources $R = \{r_1, r_2, ..., r_{max}\}$, where r_{max} is the total number of resources.

We will use the symbol E to refer to encounters which are numbered globally throughout W. The encounters between agents are numbered globally such that $0 \le e \le E_{max}$

Each encounter is numbered globally such that encounter E_j represents the j[th] global interaction with $0 \le j \le E_{max}$ The order of encounters between the agents in this world is scripted according to a simulation script *S,* which defines E_{max} encounters for agents that exist in world *W*. For each encounter, script S pre-determines the customer agent and the resource that agent will be requesting:

$$S \rightarrow \{\{a_e, r_e\} \mid a_e \in W, r_e \in R, \forall\, 0 \le e \le E_{max}\}$$

Since individual agents have no visibility to the global situation, we will use the notation e_k^a to represent the number of encounters agent *a* has had where agent $a \in W$ and $0 \le k \le e_{max}^a$ We define e_{max}^a as the maximum number of encounters agent *a* will have during simulation script *S*.

Each recorded interaction *I* between agents consists of a customer agent, a supplier agent, a resource exchanged, the result of the interaction, and the encounter number. The result *t* of the interaction, is one of the discrete values of set *T = {SUCCESS, FAILURE, NO_INTERACTION}*. So formally,

$$I \rightarrow \{\{c, s, r, t, e\} \mid c, s \in W,\ r \in R,\ t \in T, \text{ and } 0 \le e \le E_{max}\}$$

Each agent *a* maintains its own history of interactions known as H_a, such that $H_a \rightarrow \{I_k \mid 1 \le k \le e_{E_j}^a\}$ where $e_{E_j}^a$ is the number of interactions agent *a* has had by the *j[th]* global encounter. If agent *a* elects to set a window size $\omega < e_{E_j}^a$ for its history, the history then becomes a set $H_a' \rightarrow \{I_k \mid (e_{E_j}^a - \omega) \le k \le e_{E_j}^a\}$

In order to calculate reputation ρ_b^a -- the reputation of agent *b* from agent *a*'s perspective, we assume the reputation system defines a function *f* such that $\rho_b^a = f(b, H_a)$ where $a, b \in W$ Reputation ρ is not assumed to be numeric, but it must be a set of ordered values and be comparable. The reputation system must also define minimums and maximums such that $\rho_{min} \le \rho \le \rho_{max}$ holds.

Trust level τ is a value such that $\rho_{min} \le \tau \le \rho_{max}$ which signifies the minimum level of reputation an agent is willing to accept prior to initiating an interaction with another agent.

7.4.4 Formal Specification of Metrics

In order to evaluate reputation systems against one another, it is necessary to define some standard metrics that will work no matter how the reputation system operates. Below we define some of these generic metrics.

7.4.4.1 Average Reputation

Average reputation is useful in seeing how the system as a whole is categorizing agents with different strategies. The average reputation P_a of an agent *a* is

$$P_a = \frac{\sum_{i=0}^{N} \rho_a^i}{N}$$

7.4.4.2 Accuracy

Accuracy is the percentage of successful predictions of behavior. Given a well-defined population of agents and a standard number of interactions between agents, accuracy is the level of success that is achieved across the simulation environment.

The level of success λ_a for an individual agent *a* is the sum of the interactions which agent *a* has recorded that resulted in successes. Correspondingly, agent *a*'s level of failure $\bar{\lambda}_a$ is the sum of the failures, and agent *a*'s level of non-interaction $\hat{\lambda}_a$ is the sum of the interactions that did not occur (i.e. were recorded as NO_INTERACTION).

The total level of success Λ for agents in simulation *S* is $\Lambda = \sum \lambda_a$ Also, the resulting score of a particular agent can be calculated as $\bar{\Sigma}_a = \lambda_a - \bar{\lambda}_a$ and the total score of a set of agents in simulation *S* is $\bar{\Sigma} = \sum \bar{\Sigma}_a \mid a \in W$

7.4.4.3 Performance

Because the agent interactions are governed by a random distribution and a constant number of interactions and information transfers between agents, performance of the reputation system is solely dependent on how an agent's reputation information is updated (or learned) from the encounters they experience and the referral information they collect. Thus the performance metric equates to a "rate of learning or unlearning."

For performance we set a target success value (e.g. 90% success) for the system and the performance of the reputation system is the number of system wide encounters required before the system reaches that target success value. So, more formally, performance Φ_{target} is the minimum value of *j*, such that $0 \le j \le E_{max}$ and $\Lambda \ge \Lambda_{target}$

7.4.4.4 Resistance to Deception

Resistance to Deception evaluates the accuracy and performance of the reputation system when certain agents are introduced which practice deception. This is described as the difference between system-wide level of success Λ and system-wide score $\bar{\Sigma}$ with and without deception.

7.5 Implementation

The reputation simulation framework is implemented using Sun's Java SDK 1.5. In order to balance the need for flexibility in the model with the generality desired, reputation models are implemented on top of abstract classes. Agents for a given trust model are implemented as

subclasses of an abstract *ReputationAgent* class that defines the basic communications between agents. These agents exist in a class *World* and are manipulated into interactions with each other through the class *Simulation*.

Each simulation is based on a set of steps defined in a simulation script. The interactions defined in the simulation script sequentially direct which customer agent will request which resource. The simulation script is determined by choosing customer agents randomly with an even distribution from the simulated population. Also, the resource which the agent will request is assigned randomly from the pool of resources. Functions such as communication and propagation of reputation information are as common as possible. Also, metric evaluation is done in a common way, but calculation of reputation information is model specific, as is evaluation of trust.

The following terms are defined to describe the simulation framework:

- **World:** This is the simulation's environment. It contains agents, resources, and the means for selecting.
- **Agent:** These are the independent actors (both suppliers and customers) in The Resource Game. They exchange reputation information and interact according to a prescribed strategy.
- **Resources:** Each agent has a resource to offer in unlimited supply. Collecting resources successfully is the object of The Resource Game.
- **Script:** The script dictates which agent will be searching for which resource and in what sequence. The choice of agent and resource is randomly determined in an even distribution across the population of agents and the pool of resources.
- **Interaction:** This is the historical record of one turn at the resource game, and record the customer, supplier, the resource requested and the results of the interaction. We use the term encounter interchangeably with interaction.
- **Reputation:** This is a referrer agent's estimate of the likelihood of success of a future interaction with a given supplier agent.
- **Trust Level:** The level of risk that a customer agent is willing to entertain in order to fulfill its desire for acquiring the resource. Or, in other words, the minimum reputation that a supplier must have in order to be considered for an interaction.

7.5.1 Simulation of Agents

Agents are simulated as Java objects who interact with each other based on the instructions of the simulation script. Agents only have visibility to their own experiences and are able to request reputation information from other agents. They have no way of knowing whether this referrer-provided reputation information is based in fact, but can use it to establish their own evaluations of particular agents. While some simulation models have created graphs to show interrelationships between agents, this model avoids that approach because of the observation that the real world is much more dynamic and that computers on the Internet have a high degree of interconnectivity, and not just a small number of neighbors as some models simulate. Since agents do have full insight into their own experiences, they can use this information to make trust decisions.

Each agent has a defined persona or strategy which directs its actions throughout the simulation. Agents are initialized with a primary strategy and a secondary strategy that are one of GOOD, BAD, or RANDOM. When a customer agent interacts with a supplier with the GOOD strategy the encounter always results in success. Likewise, BAD strategies generate failures for the encounter, and supplier agents with a RANDOM strategy will respond with success with a probability of 0.5. The primary strategy is used initially until a defined threshold WAIT_MOVES is reached. At that point the secondary strategy takes over. Because of the infinite possibilities of mixing agent strategies, a vital few scenarios are chosen which illustrate the effects of increasing deception in the agent network.

7.5.2 Simulation Protocol

The following diagram shows the protocol for communication in this model. First the customer agent (Agent A below) begins with a *pickPreferred(resource)*, which in turn randomly chooses *c* choices from the World which provide the requested resource. Then the agent randomly chooses *r* referrers from the World of agents. The customer requests reputation information from each referrer for each of the *c* choices with *vouchFor(agent)*. Then, the encounter is made by the customer calling the supplier agent's *interact()* function. Lastly the results are used to update the customer's history and reputation information using *addHistory(interaction)*.

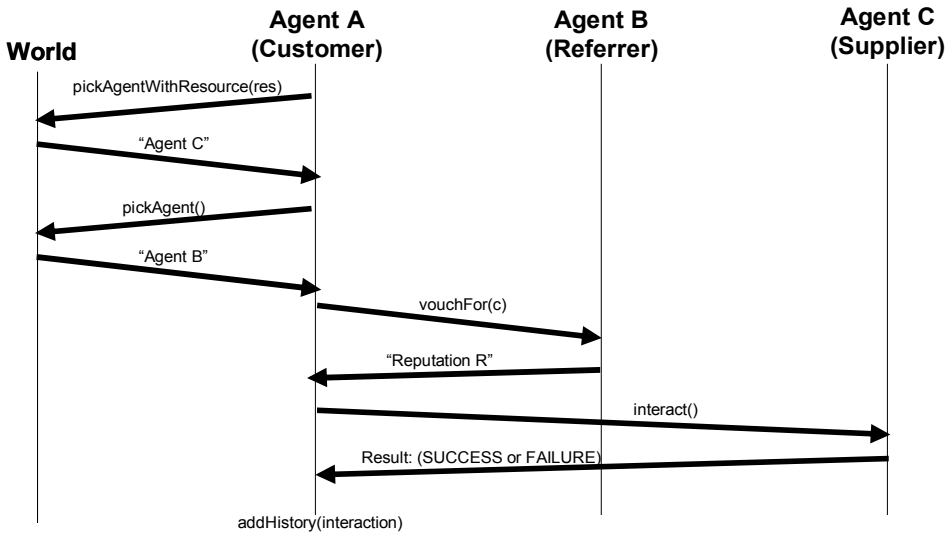

Figure 5: Simulation Protocol with One Choice and One Referrer

The protocol is intentionally simplistic to enable an accurate accounting of all of the reputation models and their behavior. Since each encounter requires a constant number of network requests, it allows us to focus analysis the mathematical representation of reputation itself, not the limitations or advantages of the network protocol that may be paired with that reputation system.

Another possible approach would be to allow variations in the amount of communications allowed between agents. Also, the protocol could be extended by allowing chained referrals as some models have done [17, 23]. These would both be a worthy areas for future work.

7.5.3 Simulation Variables

There are potentially a large number of variables that could be introduced into simulations of reputation. The intention of the simulations is to bound these variables in a reasonable way so that the impact of changes in the environment or the way reputations were calculated could be easily seen and applied in order to make improvements of the model for a given application.

The following table summarizes the key parameters used in simulations:

Parameter	***Description***
WORLD_SIZE	The number of agents with which the simulated world is populated
SCRIPT_LENGTH	The total (global) number of encounters to be simulated in a given script
NUM_RESOURCES	The number of resources to make available in the simulated world
MAX_GROUP_RATIO	Used to determine the level of grouping in the simulated world. The number of groups G is calculated by $G=N/(NG_r)$ where N is the world size and G_r is MAX_GROUP_RATIO. This functionality is disabled by default.
BAD_GUY_RATIO	Varied between 0 and 1 and is used to populate the world with a given percentage of agents with a BAD strategy. Similar ratios exist for other defined strategies, and any agents that are not included in the total of these ratios are assigned the default GOOD strategy.
RANDOM_GUY_RATIO	Varied between 0 and 1 and is used to populate the world with a given percentage of agents with a RANDOM strategy.
GOOD_BAD_GUY_RATIO	Varied between 0 and 1 and is used to populate the world with a given percentage of agents with a GOOD_BAD strategy (primary strategy is GOOD but turns to BAD at a given interval).
BAD_GOOD_GUY_RATIO	Varied between 0 and 1 and is used to populate the world with a given percentage of agents with a BAD_GOOD strategy (primary strategy is BAD but turns to GOOD at a given interval).
DECEPTIVE_RATIO	Varies between 0 and 1 to determine what ratio of the world will exercise deception when reporting reputation information.
TARGET_RATIO	The ratio of the simulated population which are targets of deception. By default, GOOD agents will be assigned as targets first, followed by agents with RANDOM and BAD strategies.
DECEPT_TYPE	Deception Type. 0 = No Deception, 1=Complement, 2=Exaggerated Positive, 3=Exaggerated Negative
EXAG_REP	A coefficient of exaggeration for reputation values
EXAG_EXP	A coefficient of exaggeration for experience values
NUM_CHOICES	The maximum number of supplier agents that a customer will consider for any given interaction
NUM_REFERRALS	The maximum number of references a customer agent will seek for any given interaction

Parameter	*Description*
REPORT_GRANULARITY	This dictates how many global interactions take place between report summaries (of total system successes, failures, average reputation by strategy, etc...)
REPORT_WINDOW_SIZE	This adjusts the level of smoothing applied to the reported results, allowing for easier comparison between simulations.
WAIT_MOVES	The number of individual agent encounters prior to a change in strategy (to the secondary strategy for instance).
OSCILLATE	Boolean value. If true, all agents will oscillate between their primary and secondary strategies.
TRUST_THRESHOLD	A constant value that corresponds to the rate of change above which an agent will become "shy"
TRUST_HISTORY	The number of encounters an individual agent remembers and applies to its calculation of trust level.
TRUST_MODEL	Selects the type of decay function that will be used for trust level management. 0= No TLM, 1=Linear, 2=Squared, 3=SQRT
TRUST_TRIGGER	Selects the type of trigger function that will be used for trust level management. 0= No TLM, 10=EQN10, 11=EQN11, 12=EQN13.
TRUST_DECAY	The number of encounters an agent delays before returning its trust level to the minimum value.
REP_WINDOW_SIZE	The number of individual encounters that an agent remembers for recalculating reputations over time (Experience Window).
REP_WINDOW	Binary switch to turn on the use of the Experience Window (values of true or false).
REP_RECALC	The number of individual interactions that must reached prior to sliding the Experience Window forward and recalculating an agents reputation information.
REP_WINDOW	Boolean value. "true" turns on experience windows and "false" turns them off.
HEARSAY	Only applicable to the MMH model, but the fraction at which reference reputations are incorporated into the prior probability of success with a given supplier agent.
TRUST_FLOOR	A constant which dictates a system-wide minimum acceptable reputation.
MMH_THRESHOLD	A variable defined in MMH which indicates how referrals should be weighted for inclusion into the prior probability based on the amount of experience that a reference claims.
MMH_UPDATE	A variable to direct how frequently (in terms of individual agent encounters) MMH agents should adjust their prior probabilities.

Table 5: Simulation Parameters

8 Initial Results

8.1 Description

Initial experimentation was conducted in order to get a feel for how the simulation framework works. First, we want to compare performance of reputation systems using some simple simulation parameters so that we can understand how the simulation environment effects the success of the reputation system to predict behavior. Also, there are a number of simulation parameters that we have discussed which may have impacts on the overall results, so it is necessary to understand those impacts. The following test cases were defined and run in the simulation environment:

1. Ratio of Good to Bad Agents
2. Random Agents
3. Effects of Varying Hearsay
4. Effects of Varying World Size
5. Effects of Varying Number of Choices
6. Effects of Varying Number of References
7. Reversing Learned Behavior

8.2 Test Cases

8.2.1 Test Case: Ratio of Good to Bad Agents

8.2.1.1 Description

In this test case we construct a simple scenario with agents that have two fundamentally different strategies. The GOOD agents always respond to an interaction with a SUCCESS and the BAD agents always respond with a FAILURE. The assumption is that after a learning phase, the reputation model will be able to distinguish GOOD agents from BAD. Relevant questions include: How many encounters will be required to achieve a certain percentage of successful interactions? How does the total number of successes possible in a given number of encounters vary depending on the percentage of BAD agents?

It should be noted that there is a theoretical upper bound to the amount of success the system has depending on the percentage of BAD agents. Since we restrict the number of choices that a customer agent can select from, and those choices are selected at random in an even distribution, then the probability of selecting at least one GOOD agent in an encounter is

$$\Lambda_{\max} = \mathrm{P}\,(\text{good agents} \geq 1) = 1 - (\mathrm{R_b})^{\mathrm{c}}$$

where R_b is the percentage of BAD agents, and c is the number of choices to select from. So, for R_b=0.1 and c=5, customer agents will have at least one GOOD agent to choose from approximately 99.999% of the time. However, for R_b = 0.8, this value would drop to 67.2% and for R_b = 0.9 it drops further to approximately 41%. An ideal reputation system would therefore be able to perform at a level equal to $\Lambda_{\max}$ We will compare our reputation systems to this ideal in the experiment below.

8.2.1.2 Results for MMH Model

Both GOOD and BAD agents begin with a reputation of 0.5 (equivalent to a 50% probability of success). Good agents are very quickly sorted from the BAD agents, and it is interesting to note that no matter what the level of BAD agents, the system reaches $\Lambda_{\max}$ within approximately 6000 encounters.

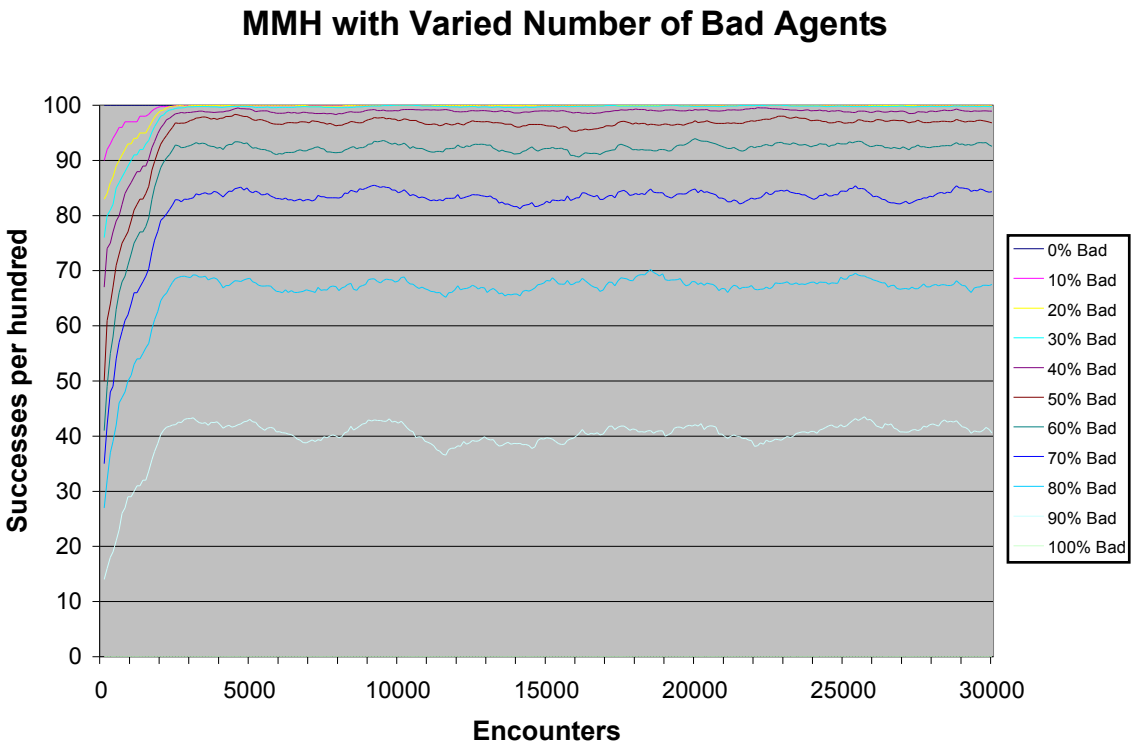

Figure 6: MMH Model with Varied Number of Bad Agents

8.2.1.3 Results for ARH Model

We have arbitrarily set both GOOD and BAD agents to begin with a default rating of 2 (equivalent to Good) in the ARH Model. Good agents quickly rise to 3 (Very Good) rating, while BAD agents quickly drop to a 0 (Very Bad) rating. As the percentage of BAD agents increases, the eventual percentage of successes that ARH can achieve is limited by $\Lambda_{\max}$ as described above. ARH reaches this level at different times depending on what the percentage of BAD agents is. So, for instance, it takes approximately 10,000 encounters for 90% Bad Agents to reach $\Lambda_{\max}$, while it takes as many as 16,000 encounters for 70% BAD agents to reach $\Lambda_{\max}$. Additionally, when we consider the case with 10% BAD agents, it appears that $\Lambda_{\max}$ has not been reached even at the end of the 30,000 encounters simulated. At the 30,000th encounter, MMH is reporting 100% success, while ARH is still reporting 99.8% success. This indicates that ARH may not be suitable for applications which have expectations for a low number of malicious agents, but require tight margins of error.

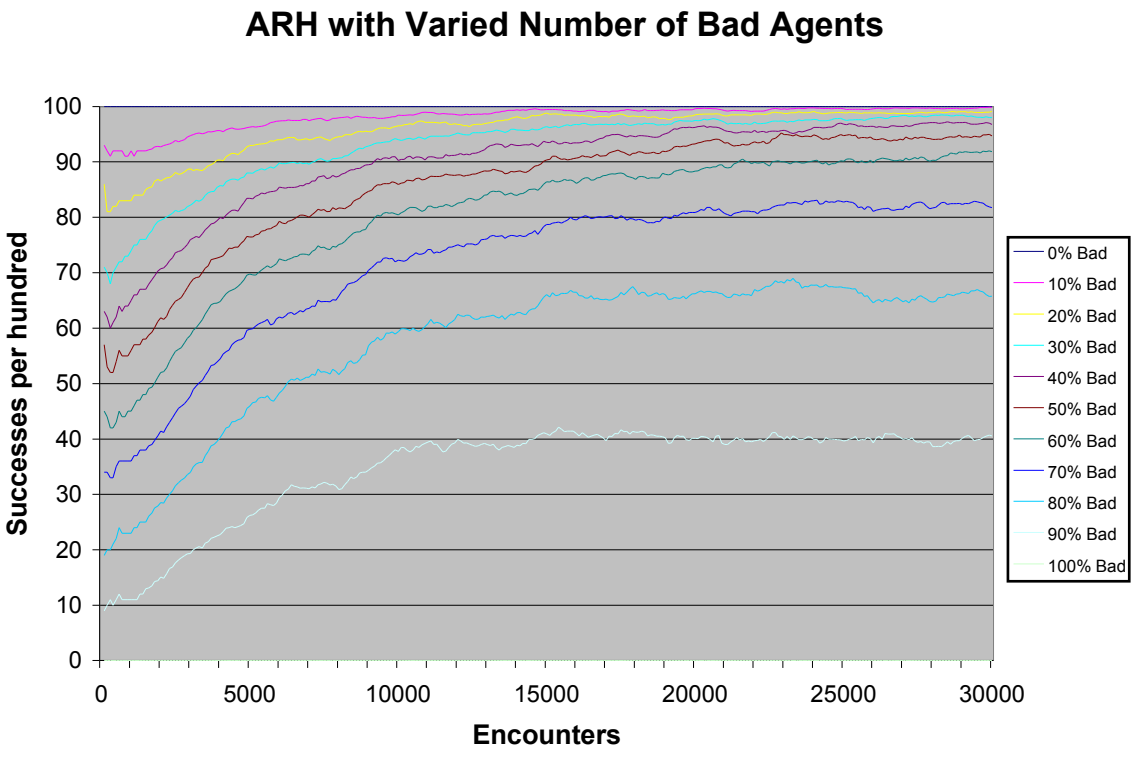

Figure 7: ARH Model with Varied Number of Bad Agents

8.2.1.4 Discussion

Separating GOOD from BAD agents is a key test because the rate in which a reputation system can distinguish good behavior from bad behavior will limit its suitability for certain applications. MMH is very sensitive to successful interactions, quickly sorting the GOOD agents from the BAD agents. It is therefore probable that MMH would work well maintaining tight margins of error in an environment with only a few hostile agents. ARH, on the other hand, takes a long time to reach $\Lambda_{\max}$ and this means that it is likely to have a greater margin of error, at least initially when the system is still learning.

8.2.2 Test Case: Performance of Random Agents

8.2.2.1 Description

Theoretically, agents which respond randomly to interaction requests (i.e. respond with a SUCCESS or FAILURE each with 50% probability) should receive an average reputation P_{random} that reflects this 50/50 chance of success. The population of this test is comprised of one third GOOD agents, one third BAD agents, and one third RANDOM agents. After a given number of encounters, the average reputations of the RANDOM agents will be compared with their theoretical values.

8.2.2.2 Results for MMH Model

MMH regards reputation as a degree of probability for success, and so agents which deliver success with 50% probability should have an average reputation rating of 0.5, i.e. $P_{\text{RANDOM}} = 0.5$. We find that after 30,000 encounters, the average reputation of a RANDOM agent is 0.5 just as expected. The reputation of GOOD agents rises much more steeply than BAD agents decreases. This is due to the fact that GOOD agents quickly begin to have more interactions than BAD agents and thus increase reputation faster. When the number of choices in the game is reduced from 5 to 1, the rise and fall of GOOD and BAD agents happens evenly.

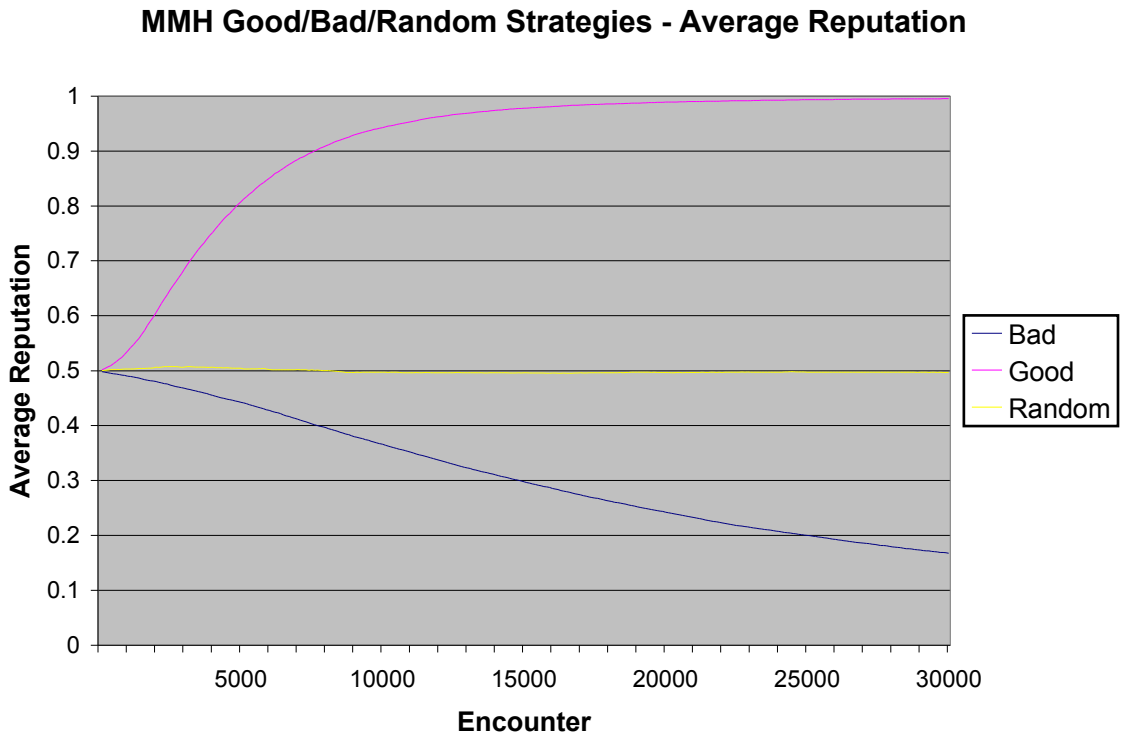

Figure 8: Average Reputation for MMH with Good/Bad/Random Strategies

8.2.2.3 Results for ARH Model

Although ARH reputations are discrete numbers, since we are using a 0 .. 3 sliding scale to represent ARH "Very Untrustworthy", "Untrustworthy", "Trustworthy", and "Very Trustworthy", the half-way point between Good and Bad reputations can be represented by 1.5. While there is no mathematical basis on which to presume that results for ARH would be similar to MMH, one might guess that in this scale of 0.0 - 3.0 P_{random} would approach a value of 1.5. However in practice the average rating converges to 0.95 after 30,000 encounters. One could read this result as indicating that the ARH model interprets a 50/50 chance of success as "Untrustworthy," but there is every indication that this value would continue to decrease with more encounters.

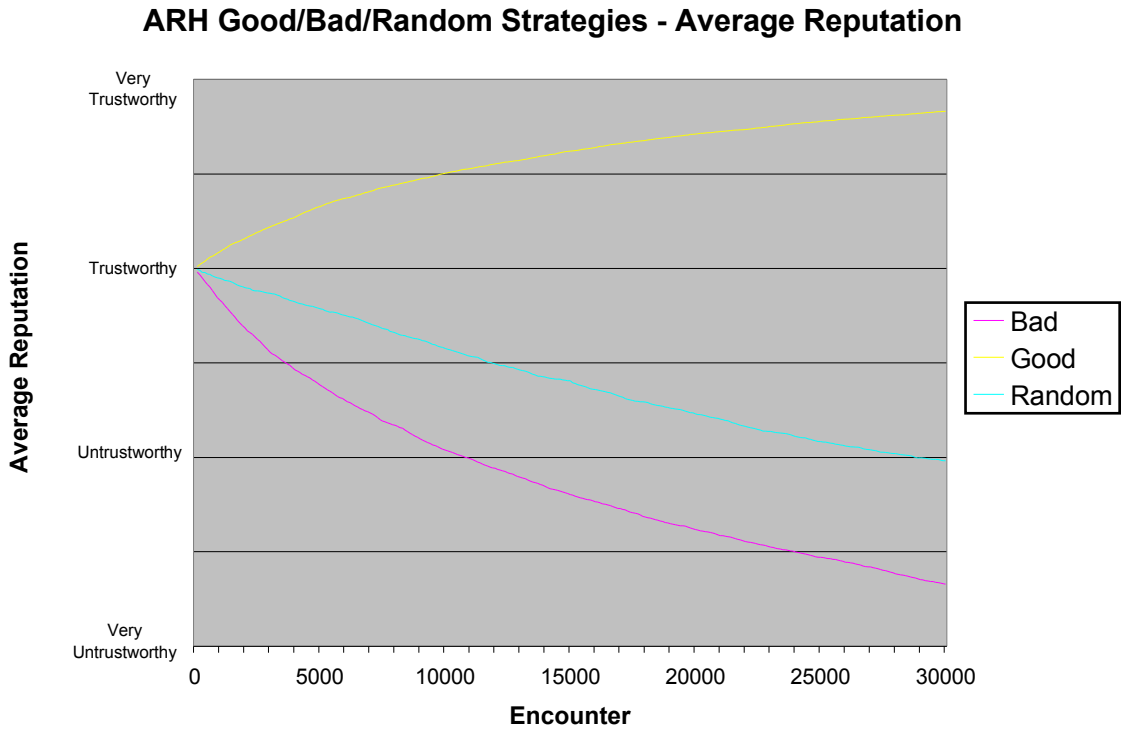

Figure 9: Average Reputation for ARH with Good/Bad/Random Strategies

8.2.2.4 Discussion

Average reputation is not that relevant to actual performance, but it does give some indication of what a reputation means. For MMH, reputation is a probability, and this is evident from its results. For ARH, reputation seems more comparative than absolute. RANDOM agents are not consistent in their responses, so their reputations are continually decreasing, just as a consistently BAD agent would have a continually decreasing reputation.

8.2.3 Test Case: Effects of Hearsay Evidence

8.2.3.1 Description

Hearsay evidence is evidence which an agent incorporates into its reputation history which it then uses itself to make judgments or would pass on to others. The danger of hearsay evidence is that it could lead to rumors, in other words inaccurate information could be passed to other agents and poison the entire system's reputation information. On the other hand, hearsay allows agents in the system to learn more quickly.

This test only applies to MMH, since ARH explicitly directs how hearsay information is to be incorporated into an agent's reputation tables. For MMH we define the HEARSAY parameter as the fraction at which reference reputations are incorporated into the prior probability of success with a given supplier agent. So if HEARSAY=1.0, this means that each reputation value passed to an agent in a referral is counted equivalent to the agent's own personal experience when updating the prior probability. If HEARSAY=0.5, then the value of referral reputations is discounted to half of the agent's own experience, and if HEARSAY=0, then this is equivalent to no hearsay evidence included at all.

8.2.3.2 Results for MMH Model

The test is a mixture of 50% GOOD and 50% BAD agents. The results are pictured in Figure 10. Clearly with no HEARSAY, the system requires slightly more time to learn, requiring 3800 encounters to reach its maximum level of success $\Lambda_{\max} \approx 0.97$ Slight improvements occur however, when HEARSAY is increased to 0.6, and maximum level of success value $\Lambda_{\max}$ is reached within 2400 encounters. We don't see any further improvements as HEARSAY is increased up to 1.0, so it seems the law of diminishing returns begins taking effect. Other experiments recounted in this paper generally use HEARSAY=0.4, since not much improvement is noted at larger values.

8.2.3.3 Discussion

In the quest to eliminate the negative effects that deception can have on reputation systems, some researchers have eliminated hearsay from their models. However, it is an important part of reputation, and this test shows that it can effect the learning rate.

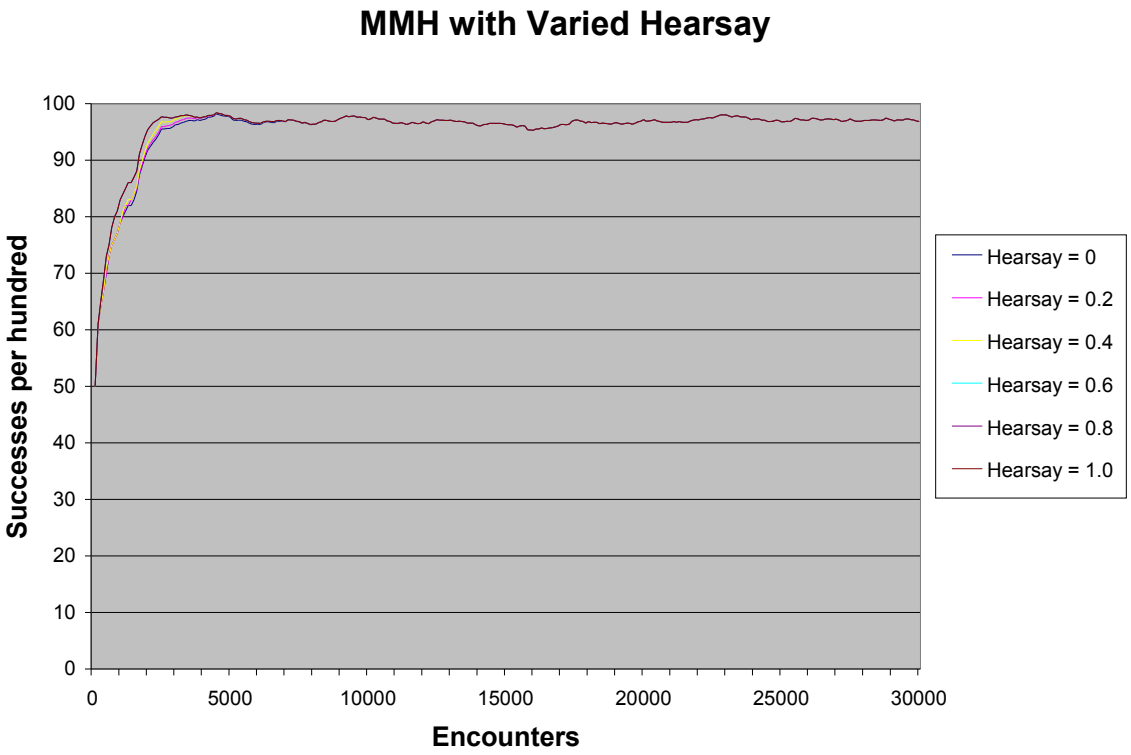

Figure 10: Effects of Varied Hearsay with MMH

8.2.4 Test Case: Effects of World Size

8.2.4.1 Description

This test traces the effects of world size on the ability for the reputation system to accurately sort GOOD agents from BAD. This test is run with a 50/50% mix of GOOD to BAD agents, and is run over the course of 100,000 total encounters. The test is run for populations of 100, 400, and 700 agents.

8.2.4.2 Results for MMH Model

The chart below depicts the trajectory of MMH towards $\Lambda_{\max}$. Smaller world sizes such as 100 agents only takes approximately 1900 encounters to reach a 90% success rate. Larger world sizes such as 400 and 700 agents appear that they will not reach a 90% success rate until encounters 10,700 and 29,500 respectively. It is evident from the data that for large populations, reputation systems must use other methods to enable agents to get adequate reputation information efficiently.

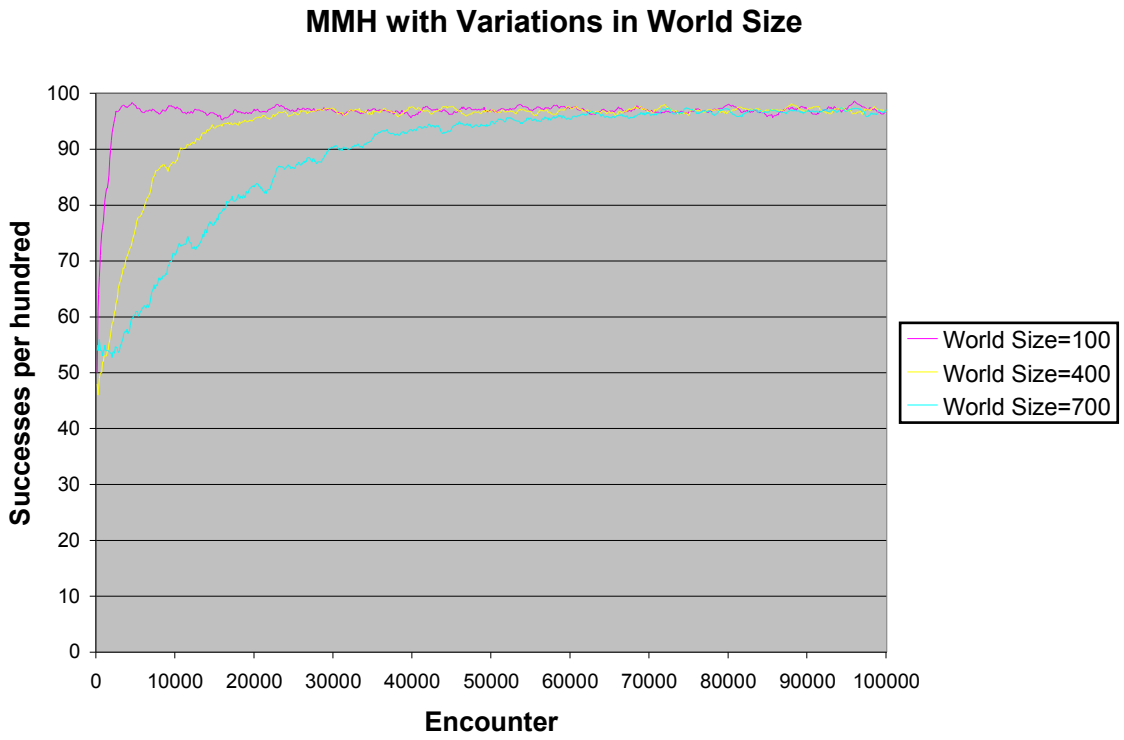

Figure 11: MMH Reputation System with Varied World Size

8.2.4.3 Results for ARH Model

ARH requires a lot of storage, so this test was one of the most memory intensive tests run. For a population of 700 agents, this test required a JVM which had 1Gb of memory allocated to it. The chart below shows that ARH's performance for larger world sizes is significantly worse than MMH. Smaller world sizes such as 100 agents take approximately 15,000

encounters to reach a 90% success rate. Larger world sizes such as 400 and 700 agents appear that they will not reach a 90% success rate until encounters 120,000 and 200,000 respectively.

Clearly the ARH reputation system is very sensitive to world size and thus should employ group reputations or other techniques in order to be successful tracking the reputations of a large number of agents.

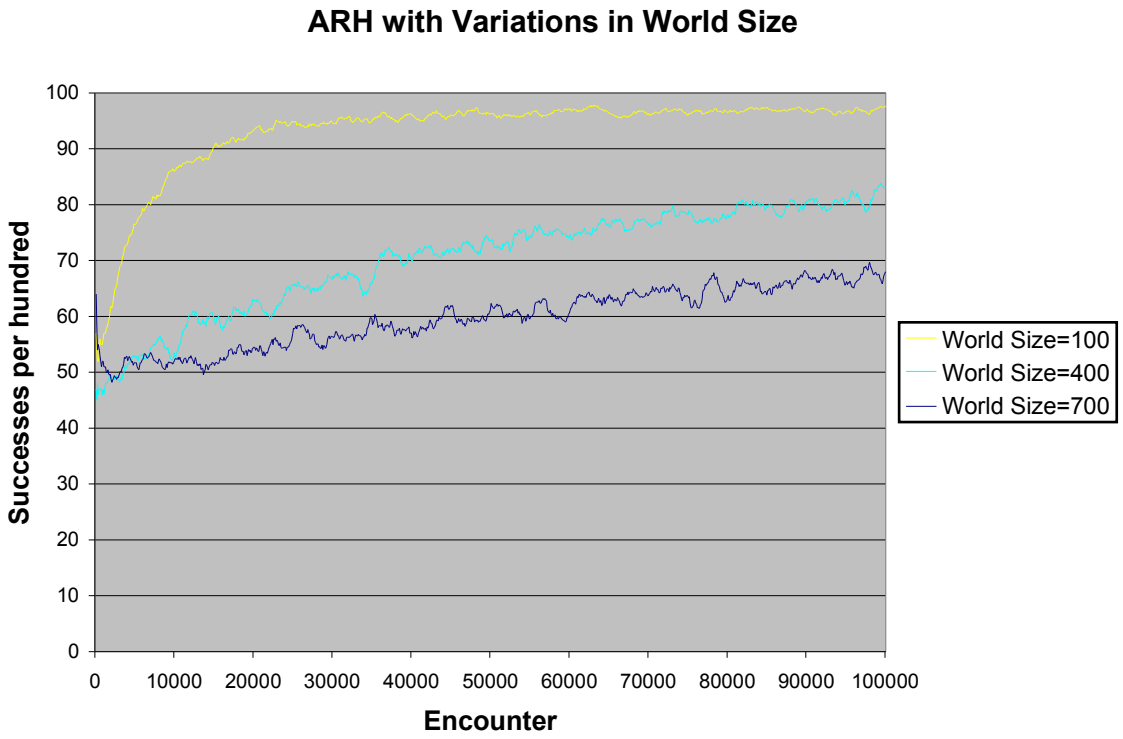

Figure 12: ARH Reputation System with Varied World Size

8.2.4.4 Discussion

World size may be the biggest obstacle to using reputation systems successfully in the real world. Both MMH and ARH were slow to sort through larger numbers of agents, and we clearly would not be able to extend this experiment to a real world number, like 10,000 agents. Here are some alternative ways of coping with large populations that could be considered for future work:

- Instead of expecting referrer agents to have information about just any supplier randomly chosen, ask them to specifically point out which agents on which they have collected good reputation information that will meet the need.
- Increase the number of referrers and/or the number of choices proportionate to the world size.
- Use group reputation to communicate reputation more efficiently through a large population.

8.2.5 Test Case: Effects of Varying Number of Choices

8.2.5.1 Description

Our simulation protocol calls for each agent to select *c* agents at random as potential supplier agents for its next interaction. The agent evaluates the reputations of each of these suppliers and then selects the one with the best reputation. It stands to reason that an agent would be able to improve its level of success by selecting more suppliers to choose from. This test varies the number of choices, *c*, and tracks the resulting level of success for each of our reputation systems.

We recall that the maximum level of success is $\Lambda_{\max} = P(\text{good agents} \geq 1) = 1 - (R_b)^c$ and varying c will impact the results the simulation is able to attain. For these simulations we set $R_b = 0.5$, so $\Lambda_{\max}$ will have the following values for each value of *c* tested.

c	Λ_{max}
2	0.75000
3	0.87500
4	0.93750
5	0.96875
6	0.98438
7	0.99219
8	0.99609
9	0.99805
10	0.99902

Table 6: Λ_{max} Values for Choices c=2..10

8.2.5.2 Results for MMH Model

Figure 13 shows the results of varying choices *c* from 1..10. It is clear that MMH very quickly rises to the $\Lambda_{\max}$ value, and no matter what the value of c is, MMH reaches $\Lambda_{\max}$ after approximately the same number of encounters every time.

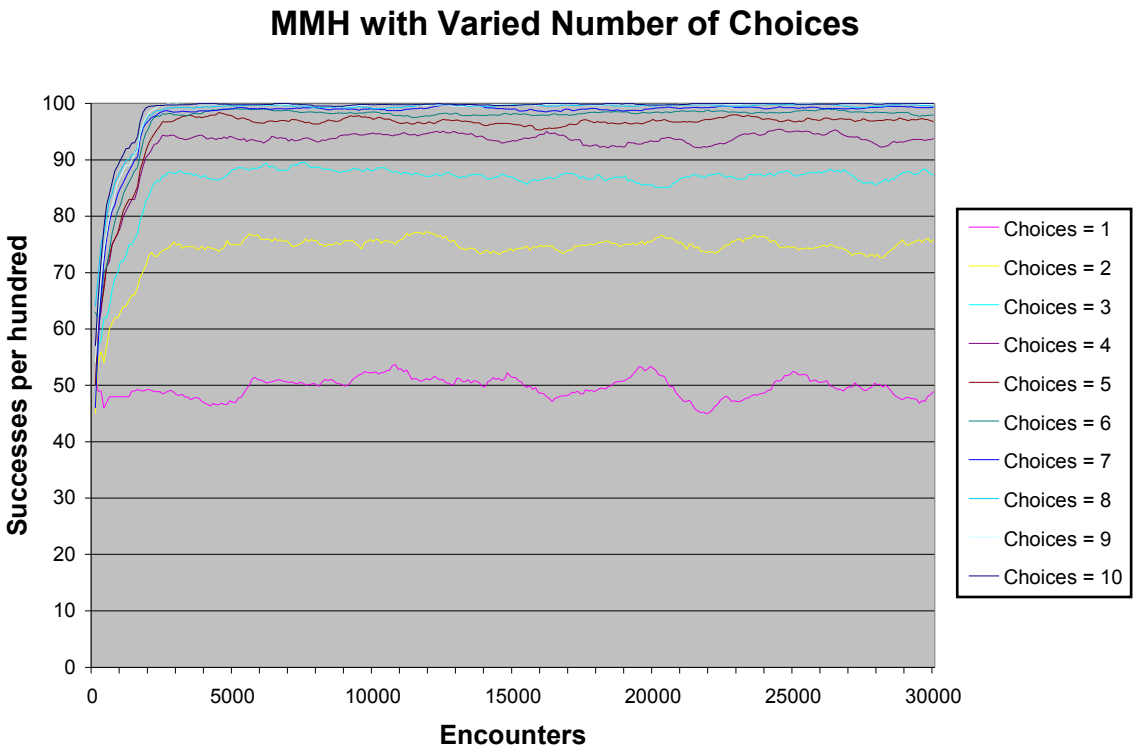

Figure 13: MMH with Varied Number of Choices

8.2.5.3 Results for ARH Model

Figure 14 shows the results for the ARH reputation system. ARH takes significantly longer to reach $\Lambda_{\max}$ --13,500 encounters for *c*=2, and 26,700 encounters for c=3. As *c* increases, the number of encounters required to reach $\Lambda_{\max}$ continues to increases dramatically. Over the course of the 30,000 encounters run in this test, the highest level of success attained by ARH was 0.976 for c=10 choices. This is well short of the maximum level of success, and it is again apparent that it will require a very large number of training encounters to eliminate the top 2% of failures possible using the ARH model.

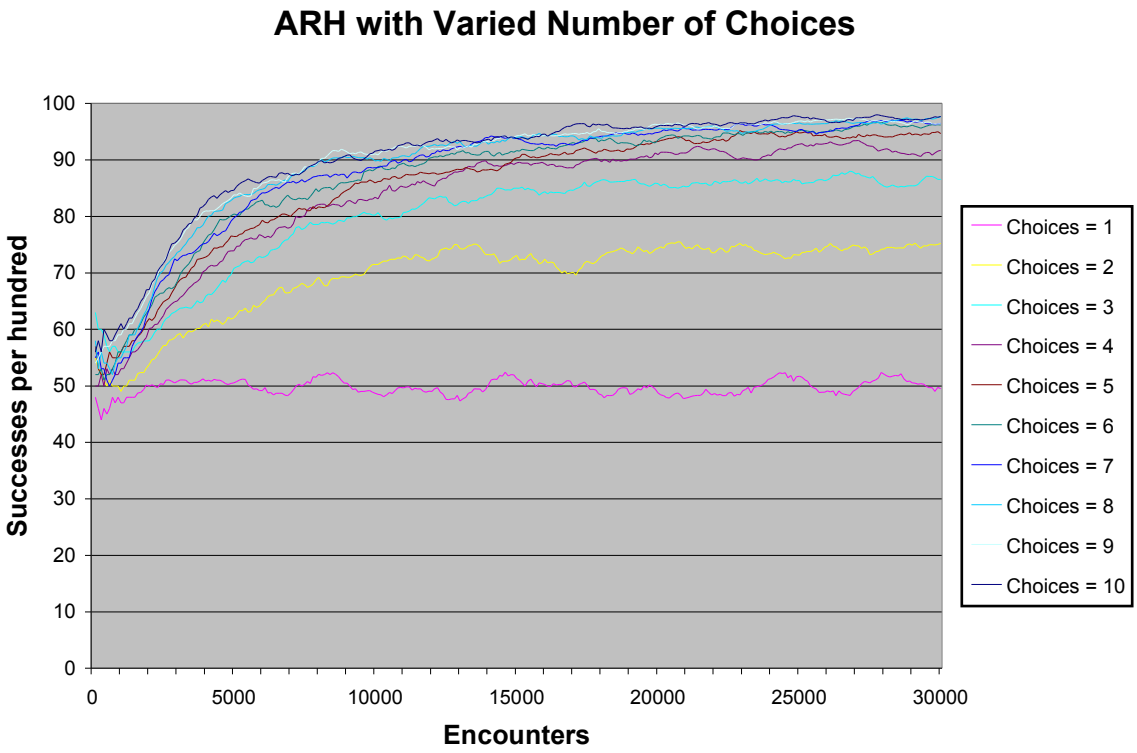

Figure 14: ARH with Varied Number of Choices

8.2.5.4 Discussion

Our predictions on $\Lambda_{\max}$ were realized in this test, and we see once again how MMH learns much faster than ARH, no matter how few or how many choices are assigned. At this world size there seems to be diminishing returns once *c* is set to 5 or higher. Since this parameter corresponds directly to the amount of communication that must take place per encounter, it is important to discover at what point additional choices become irrelevant.

8.2.6 Test Case: Effects of Number of Referrals

8.2.6.1 Description

As described earlier, the Resource Game requires each agent to select c supplier candidates from the simulated world. The agent then requests reputation information on these supplier candidates from a number of referral agents. We presume that getting referrals from more agents will increase the accuracy of the reputation information that is received. This in turn should increase the rate at which the simulation becomes successful.

Also, since each of the r referral agents is queried once for each of c choices, there are $c \times r$ communications which must take place prior to every interaction. It is therefore helpful to know how useful these communications are to the overall success of the agent in obtaining useful reputation data.

8.2.6.2 Results for MMH Model

Figure 15 shows MMH results after varying the number of referral agents used to make a decision on choosing a supplier. Once again, this simulation uses a 50/50 mix of GOOD and BAD agents. We see that the overall level of success is increased by adding referral agents, but not as dramatically as it was increased as in the last test case by adding choices. Also, there are diminishing returns once $r=8$. The improvement seen with r set at greater values than 8 are negligible.

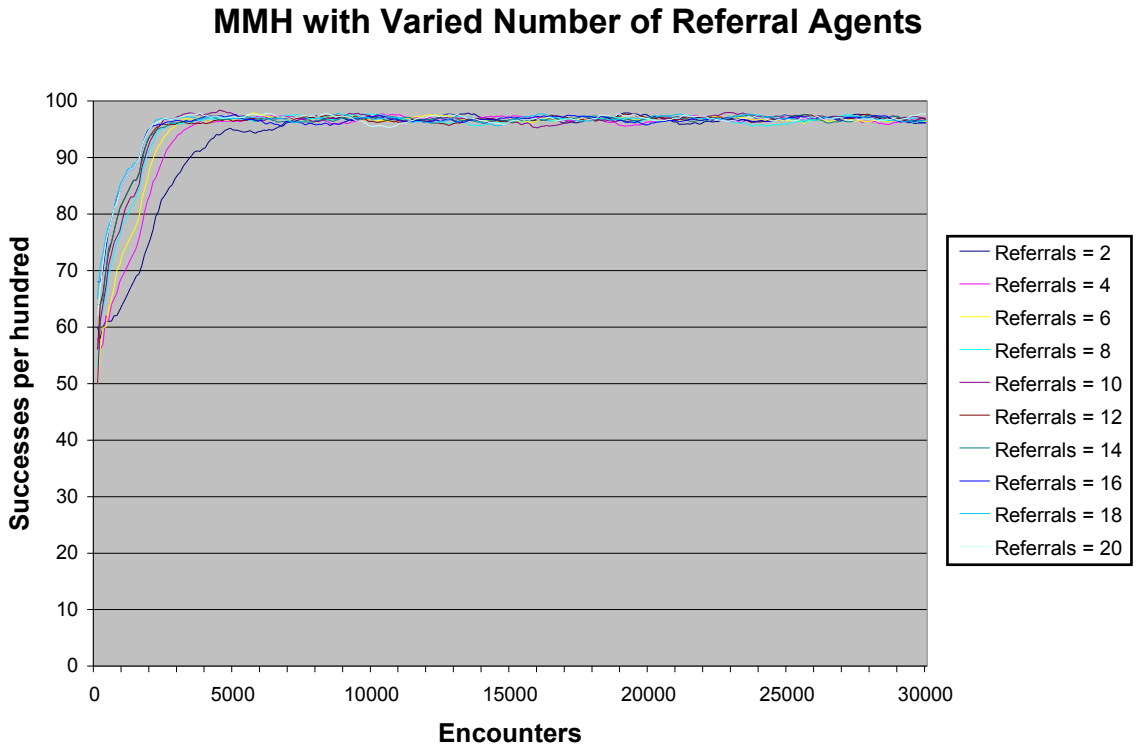

Figure 15: MMH with Varied Number of Referrals

8.2.6.3 Results for ARH Model

The results for ARH are interesting in that they do not reflect our postulate that increases in referrals should increase the overall success of the reputation system. Success levels increase at generally the same pace no matter what *r* is set at.

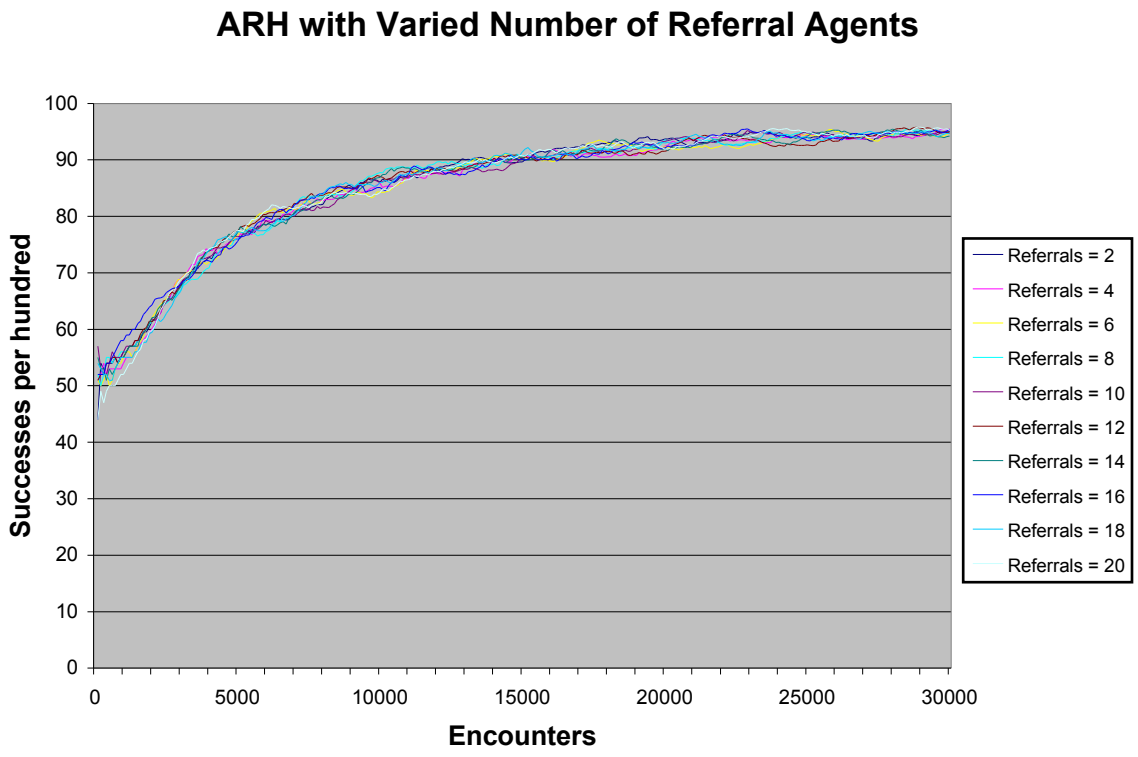

Figure 16: ARH with Varied Number of Referrals

8.2.6.4 Discussion

It is apparent that the number of referrals is not a large factor in the rate of learning. By default we have chosen to perform most of the simulations documented here with a default number of referrals set to 10. By fixing the referrals and choices during simulations, we ensure that we are comparing reputation systems alone, not differences that may be introduced by variations in the amount of communication.

8.2.7 Test Case: Reversal of Behavior

8.2.7.1 Description

Up until now, we have focused on the ability of a reputation system to quickly learn about its environment and apply it within the context of the Resource Game. However, a reputation system must not only be able to quickly acquire the information necessary to distinguish between cooperative and non-cooperative agents, but it must be able to react appropriately when agents act in a way that goes against their reputation. This test looks at the ability to reverse learned reputations.

As a fairly extreme example, we will take 10 agents, half of which are GOOD and half of which are BAD. In the simulation the GOOD and BAD agents suddenly switch strategies so that the GOOD agents behave badly and the BAD agents behave cooperatively. The question is then how long it will take for the reputation systems to reflect the switch.

8.2.7.2 Results for MMH Model

The MMH model which performs so well in terms of speed reaching the maximum level of success, is very challenged by this simulation. Figure 17 shows that it becomes immediately paralyzed once the change in strategy occurs and does not begin to recover until encounter 64,500. At that point it quickly rises to the maximum level of success, but by this time it has lost significant ground to ARH.

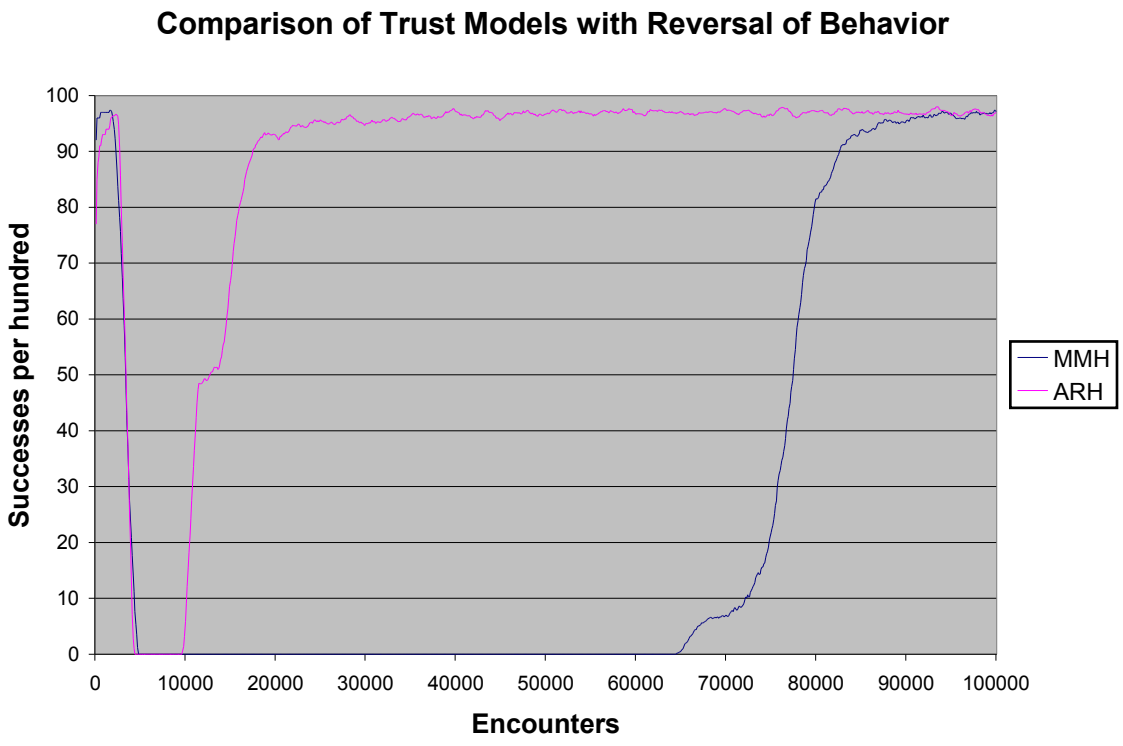

Figure 17: Comparison of Trust Models with Reversal of Behavior

8.2.7.3 Results for ARH Model

ARH, which generally has a much slower way of attaining success compared to MMH, shines in this example by quickly recovering once the change in strategy occurs. It begins to have successful interactions by encounter 9700, and rather quickly works its way up to the maximum level of success. The difference is apparent in Figure 17, but is also highlighted by the fact that in this window of 100,000 encounters, ARH has more than 3 times the successes compared to MMH.

8.2.7.4 Discussion

Both ARH and MMH have trouble adjusting to a dramatic change in environment. Despite the fact that they are intended to change when the environment changes, they find it hard to do so in this extreme example. At no time during this test is it impossible to have successful interactions – as soon as the group of GOOD agents turns hostile, the group of BAD agents turns GOOD. However it is necessary that the reputation systems experience a lot of failure to counter the successes that they initially achieved. The next section will go into depth about ways in which we can improve the MMH reputation system to perform better under this circumstance of reversal of behavior.

9 Improvements to Reputation Systems

Given the definitions that we have proposed for defining and simulating a reputation system, we attempt to improve on these systems by manipulating two aspects: the agent's level of trust and the agent's level of experience.

9.1 Trust Level Management

9.1.1 Description

We have defined reputation ρ_b^a earlier to be the *expectation of success that an agent a has that agent b will act in a beneficial way towards a.* Also, we have defined trust level τ_a to be the *minimum perceived expectation of success that an agent a is willing to accept prior to initiating an interaction*. Since $\rho_{min} \le \tau_a \le \rho_{max}$ it is possible we may gain improvements on the overall success of an agent by manipulating the trust level dependent on the level of success an individual agent has.

This situation is analogous to the difference between a person who has been successful in the marketplace versus a person who is continually unsuccessful. The successful person will have high expectations for the reputations of the people they transact business with. Likewise, if the same person were placed in a hostile environment where they suddenly began experiencing a high degree of failures, high expectations for reputation would prevent some failed transactions, but they would have to relax their reputation requirement in order to do any business at all.

We propose that one way of maximizing success and minimizing failure in a hostile or rapidly changing environment is to perform trust level management. For our Trust Level Management method we propose that agents mimic the real-world behavior of "once bitten, twice shy." In other words, agents become "shy" of future encounters if they have experienced too much failure in the past. This is similar to natural defense mechanisms seen repeatedly in nature. After a period of time, the agent loses its shyness and re-engages other agents. It is our expectation that this would have a positive effect on the overall success of agents in a hostile environment.

We define Trust Level Management as an operation of two functions, a trigger function $f_{trigger}$, and a decay function f_{decay}. So for an agent a, its trust level is $\tau_a \leftarrow g(f_{trigger}, f_{decay})$ Agents are able to use the following data points as input to the trigger function:

- The number of failures the agent has recorded locally ($t_{failure}$)
- The number of successes the agent has recorded locally ($t_{success}$)
- The rate at which failures and successes are increasing or decreasing

Later on we will describe experiments with various equations for the trigger and decay functions and see what effect they have in different simulated situations.

9.1.2 Definitions

First it should be noted that we assume that agents only have information from their own histories available for managing trust. The simulation environment does not provide the capability for agents to query each other in order to make trust level decisions. An agent's history is also limited to reduce storage requirements.

With this as a starting point agent *a* has the ability to calculate a few useful metrics. Agent *a*'s failure rate is calculated by dividing *a*'s level of failure within the window ω of its history *H'*, or $\bar{\lambda}_a/\omega$ Likewise, agent *a*'s success rate is calculated as λ_a/ω

Agent *a* can also estimate the rate of change of the number of failures or successes by dividing the agent's history window in half and calculating the slope. If λ_a^1 is the level of success in the first half of the history window, and λ_a^2 is the level of success in the second half, the rate of change for successes is $(\lambda_a^1 - \lambda_a^2)/(\omega/2)$ and the rate of change for failures is $(\bar{\lambda^1}_a - \bar{\lambda^2}_a)/(\omega/2)$

9.1.3 Test Cases

9.1.3.1 EQN10: Triggering based on Failure Rate

In developing test cases, we presume that a good trigger function will be one which triggers when the agent finds itself in a sufficiently bad environment. Therefore we will define four different $f_{trigger}$ functions. The first function, referred to as EQN10, triggers shyness when the failure rate is greater than a certain value TRUST_THRESHOLD. This follows the inference that an agent should stop attempting to interact when the environment has become too hostile.

Our initial test case will be using the 50/50 mix of GOOD and BAD agents, and the results are shown in Figure 18. The decay function which was used is linear over the a number of interactions equal to the parameter TRUST_DECAY. TRUST_THRESHOLD was set to a constant value of 0.8. Figure 18 shows that there is a slight decrease in the rate of growth as TRUST_DECAY increases, and in each case adding Trust Level Management results in fewer successes overall.

Next we examine the results using the reversed learning test which has a mix of 5 GOOD and 5 BAD agents which switch strategies after 500 encounters. As shown earlier, MMH performed very poorly on this test, and we would like to see if we can improve things by managing trust levels. Figure 19 shows that Trust Level Management has significantly lengthened the time required to recover in the reversal of behavior test defined in section 8.2.7. Also, increasing TRUST_DECAY, only worsens the effect. This is because unlearning requires failure, and trust level management is avoidance of failure.

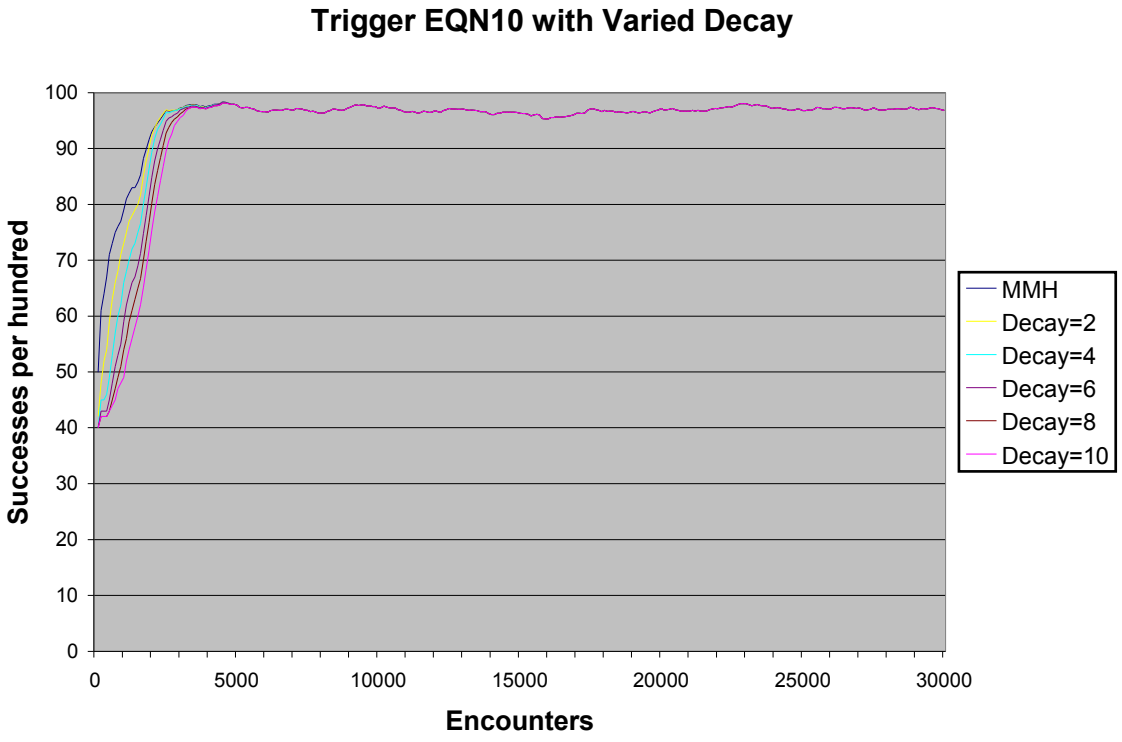

Figure 18: 50/50 Good/Bad Mix with EQN10 Trigger Function

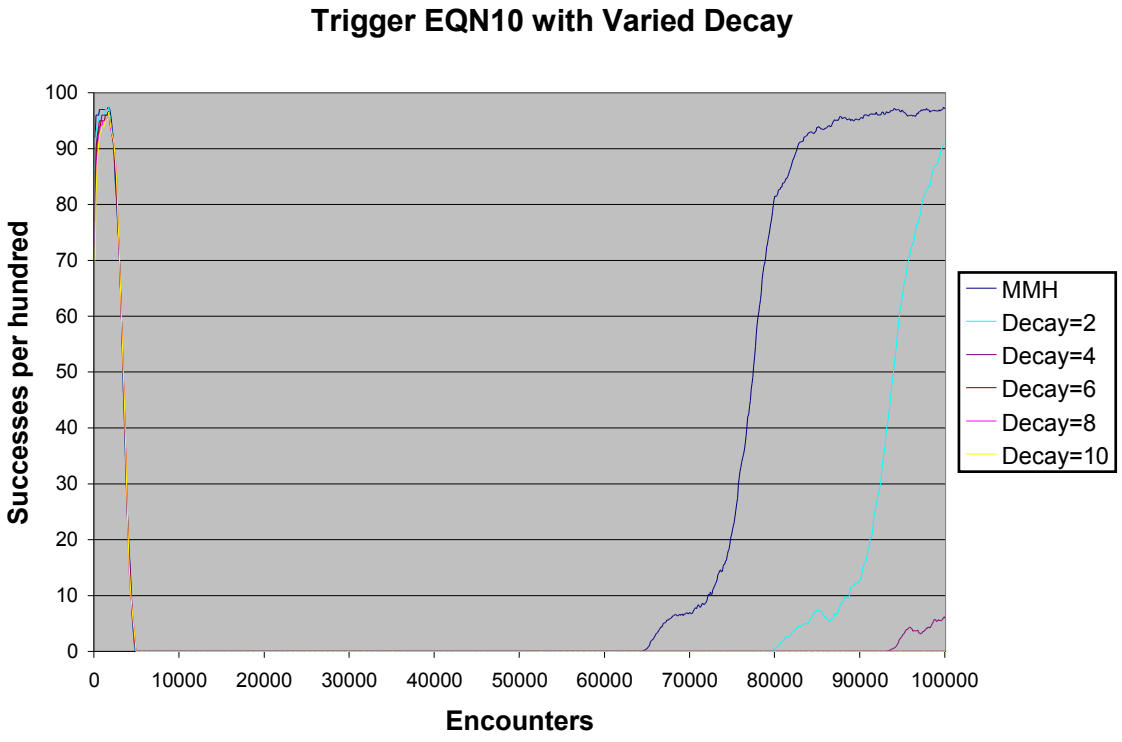

Figure 19: Reversal of Behavior Test with EQN10 Trigger Function

While the number of successes decreased is apparent in Figure 19, unseen is the number of times there was no interaction because Trust Level Management assigned a trust level high enough to eliminate all of the choices from an encounter. The results can be seen if we calculate the score $\bar{\Sigma}$ of each trial run, as defined in section 7.4.4.2. Figure 20 shows the scores obtained from different values of TRUST_DECAY applied to EQN10. Only one value, TRUST_DECAY=8, results in a score that is slightly less than MMH without Trust Level Management. The remainder are significantly worse than MMH.

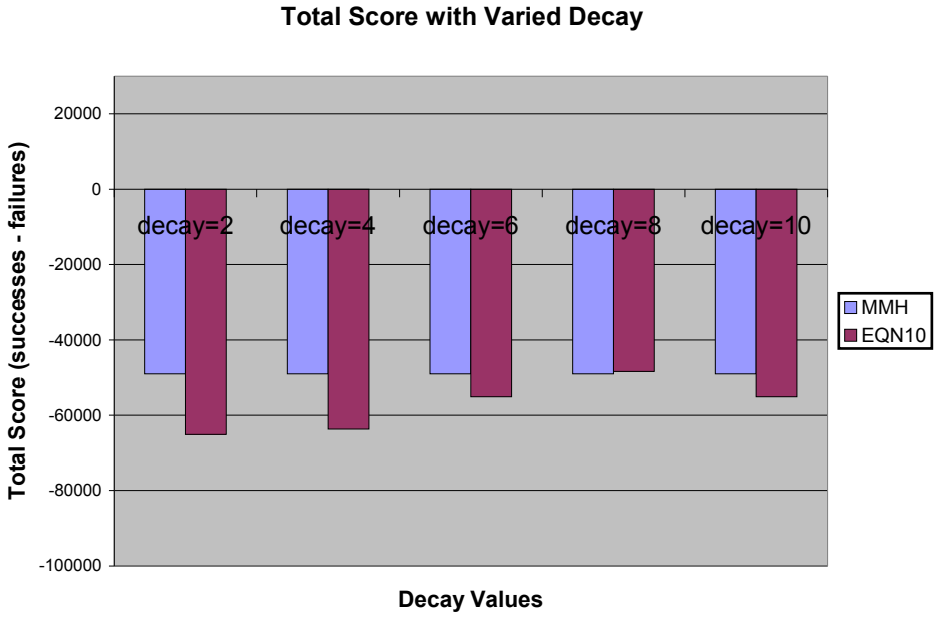

Figure 20: Scores from Reversal of Behavior Test with EQN10 Trigger Function

By examining the results of the tests with EQN10, we can draw a few conclusions. First, increasing the decay time almost always results in fewer successes overall. This finding makes sense in light of the game and the MMH reputation model. The key factor for doing well on the 50/50 Good-Bad Test is to learn reputation quickly. Increasing trust level slows down this process, and increased decay times will only slow the process down more.

Secondly, attempts to reduce failures by using a trigger function that is keyed off of failure rate seems to result in very slow recovery time in in the Reversal of Behavior test. Again, efforts to manage trust level reduce the number of interactions that an agent will have, both successful and unsuccessful. This slows down the re-learning process necessary to succeed in this test. Lastly, the fact that the overall scores achieved with EQN10 are generally lower than the score of MMH only shows that the interactions that our agents avoided would have been successful at least as frequently as they would have been unsuccessful.

Lastly, triggering with failure rate does not even increase the overall score that a model achieves in the simulation.

9.1.3.2 EQN11: Triggering based on Success Rate

If triggering on failure rate doesn't give good results, perhaps triggering on success rate will give good results. The challenge in the Reversal of Behavior Test is reducing the reputation of the agents which initially gave good responses down to the reputation values of the agents which initially gave bad responses (the "un-learning phase"). Once this happens, agents have the opportunity to re-sort the population based on their current behavior.

The second function we examine, EQN11, triggers when the success rate is greater than TRUST_THRESHOLD. This test was run in a similar fashion as the previous one, with TRUST_THRESHOLD set at a constant value of 0.8. Figure 21 shows the results obtained for the 50/50 Good/Bad Test.

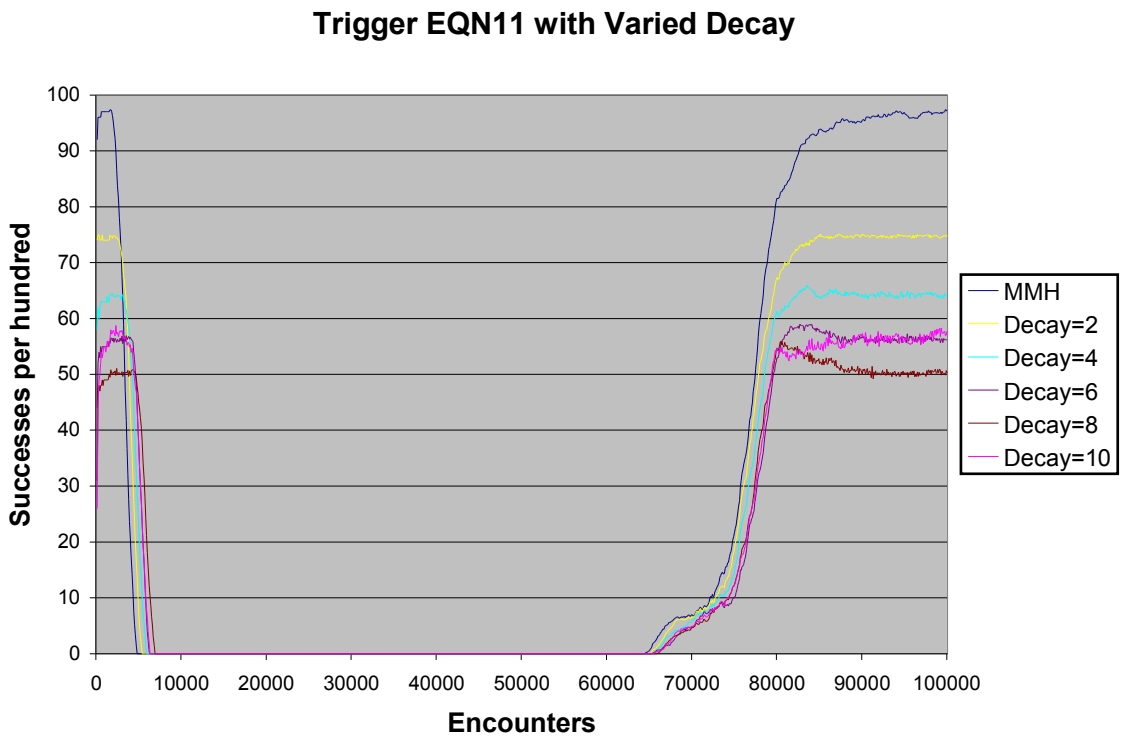

Figure 21: Reversal of Behavior with EQN11 Trigger Function

It is apparent from Figure 21 that, although triggering on success rate prevents a steep rise initially in successes (and the steep rise in average reputation that accompanies that), it also prevents the simulation from reaching its potential later on, once the un-learning phase has completed. Also, despite the fact that so many success opportunities were passed by in the first 5000 encounters, this did not seem to result in a shortening of the un-learning phase.

9.1.3.3 EQN12 and EQN13: Triggering based on Rate of Change

Given the lessons learned from trigger functions EQN10 and EQN11, we refine our trigger function by defining it based on rates of change. We speculate that our opportunity to improve in the Reversal of Behavior test is by flattening the initial wave of successes so that the un-learning phase will be shorter and overall score will be improved.

EQN12 triggers when the rate of change for successes exceeds a constant parameter TRUST_THRESHOLD. In order to cover all bases, we also define EQN13 which triggers when *either* the rate of change for successes *or* the rate of change for failures exceeds TRUST_THRESHOLD.

Figures 22 and 23 show the results for EQN12. As predicted, the initial learning phase is slowed by managing trust based on rates of change. However, in Figure 22, this does not impede the eventual growth to Λ_{max} When we examine the Reversal of Behavior test (Figure 23), we see that EQN12 gives us a slight improvement (about 1%) because the "un-learning phase" starts as much as 700 encounters earlier than MMH. When score $\bar{\Sigma}$ is considered, the results are improved about 0.6%.

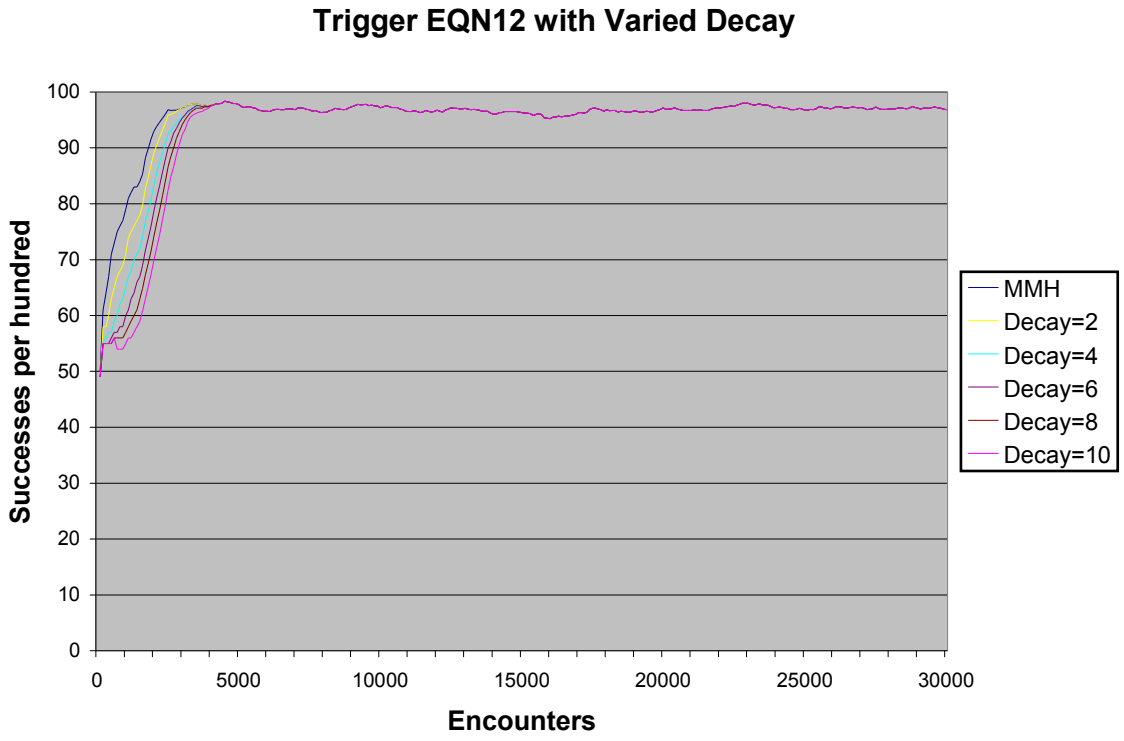

Figure 22: 50/50 Good/Bad Mix with EQN12 Trigger Function

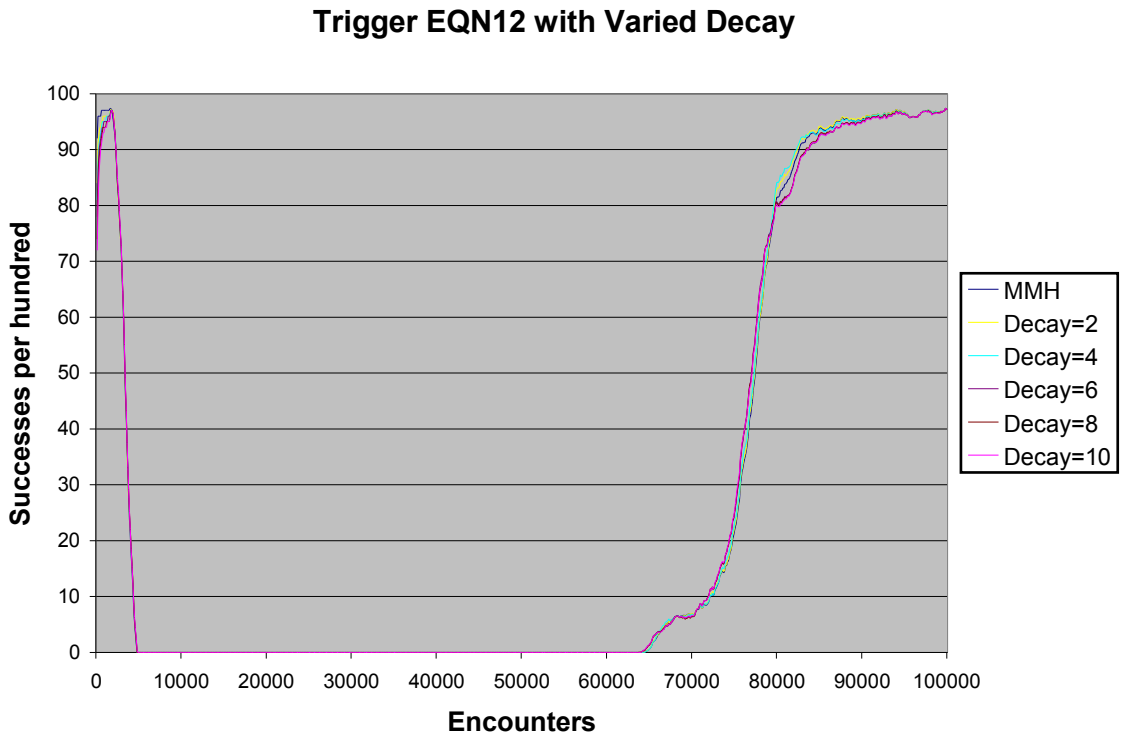

Figure 23: Reversal of Behavior Test with EQN12 Trigger Function

Given the fact that EQN13 triggers on either the rate of change for successes or the rate of change for failures, we would hope to get better results with this trigger function. In fact, EQN13 does improve slightly on EQN12, but not by much. Figures 24 and 25 show that EQN13 does in fact have a slower learning rate than EQN12. When applied to the Reversal of Behavior test, however, the start of the "un-learning" phase begins at the same time as EQN12 (1% better than MMH), and the score creeps up to 0.7% improvement over MMH.

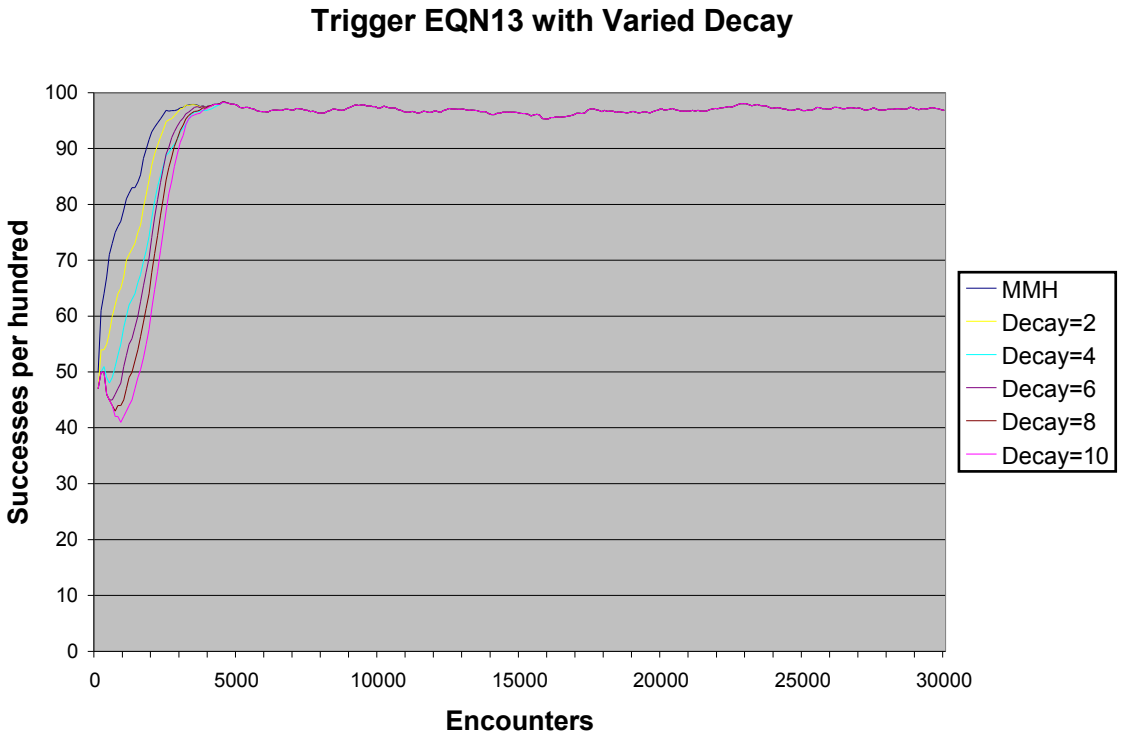

Figure 24: 50/50 Good/Bad Mix with EQN13 Trigger Function

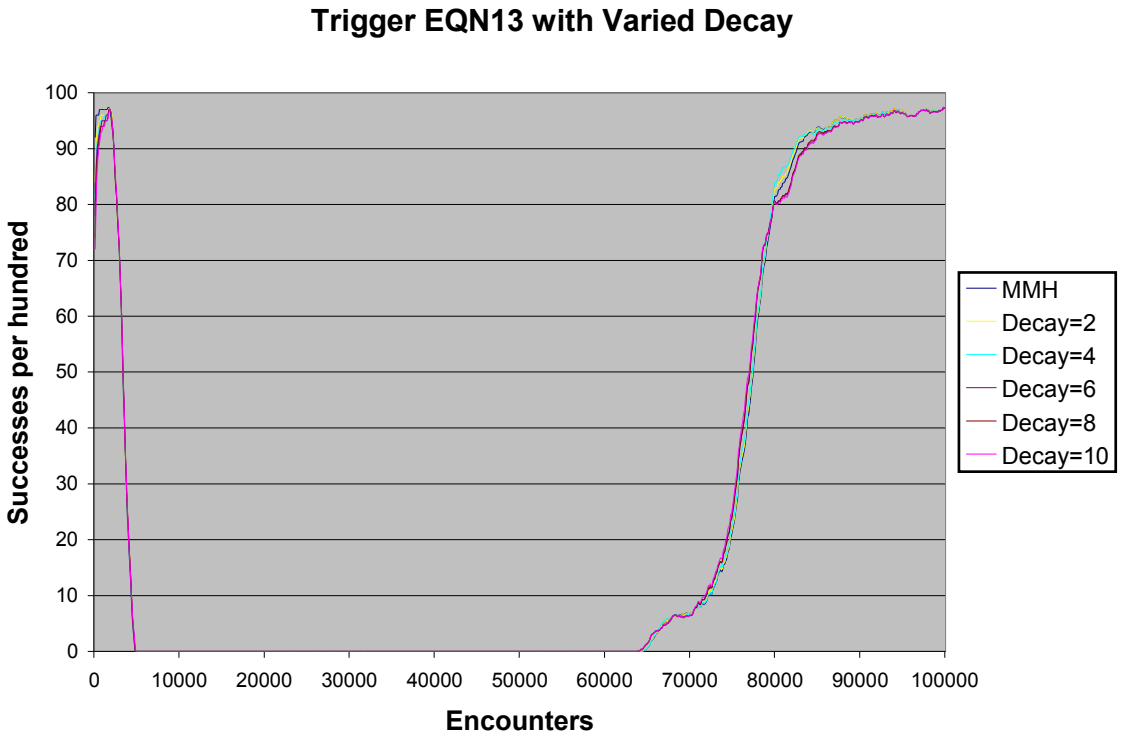

Figure 25: Reversal of Behavior Test with EQN13 Trigger Function

9.1.3.4 Variations of the Trust Threshold

Now that we have arrived at some trigger functions that show that Trust Level Management can provide minor improvement over MMH, we will attempt to tune these trigger functions for our test cases by examining the effects of varying the TRUST_THRESHOLD parameter. In prior experiments we have arbitrarily set TRUST_THRESHOLD=0.8. We presume that larger values will decrease the rate of success on the initial phase of the Reversal of Behavior test, and smaller values will increase the initial rate of success, making the results look more like MMH without Trust Level Management applied.

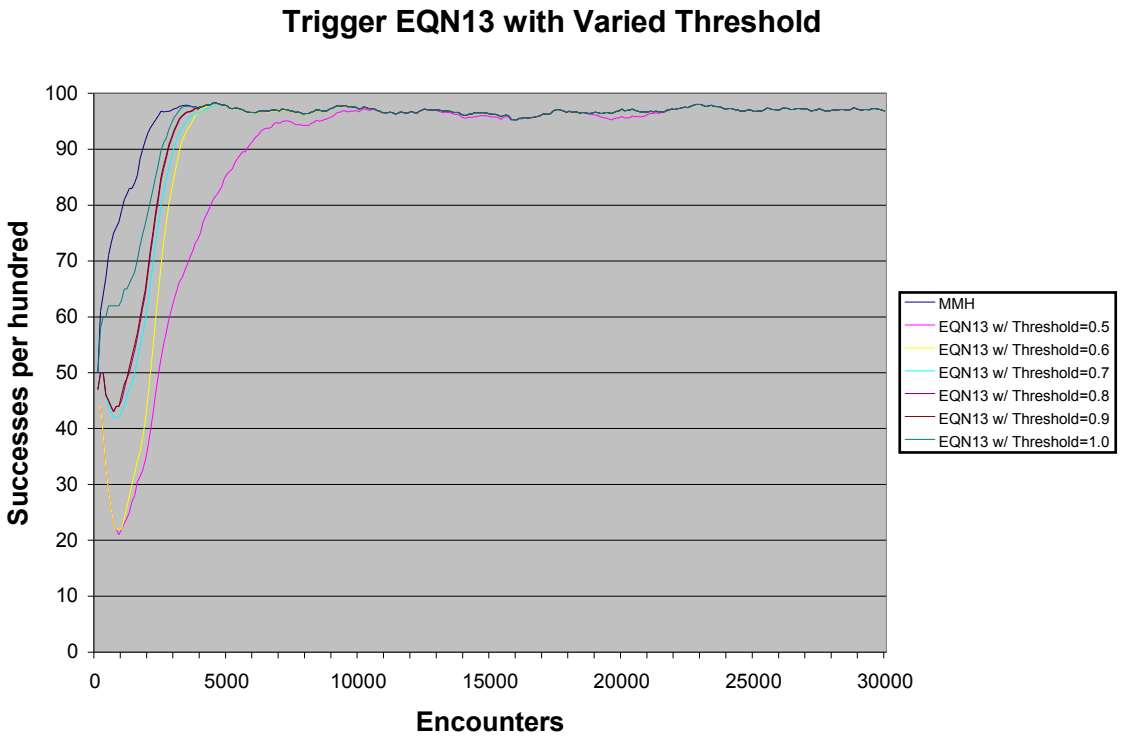

Figure 26: 50/50 Good/Bad Mix with EQN13 Trigger and Varying Threshold

Judging the results of the 50/50 Good/Bad Mix test in Figure 26, our conclusion is correct that smaller values of TRUST_THRESHOLD yield a slower learning rate at the beginning of the simulation. Larger values such as 1.0 are still slower than MMH without Trust Level Management. Now we compare these results to those in Figure 27, the Reversal of Behavior test, and we see that at TRUST_THRESHOLD value 1.0 the slight advantage that TLM has over the plain MMH system has disappeared. Thus, the best balance appears to be setting the threshold to a value of about 0.8.

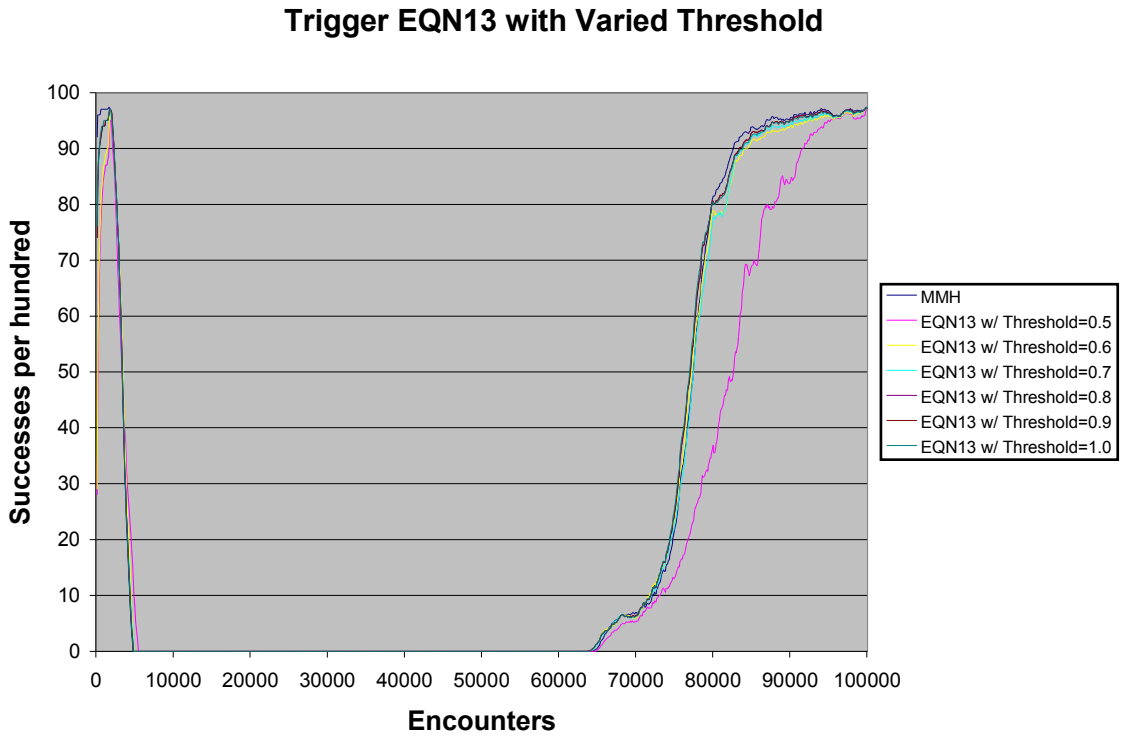

Figure 27: Reversal of Behavior Test with EQN13 Trigger and Varying Threshold

9.1.3.5 Variations of the Decay Function

All of the tests run so far have only manipulated the trigger function or the duration of the decay. Now we will focus on fine tuning the decay function itself, which until now has been linear. Assuming the trust level is immediately set to the maximum reputation value (1.0 in MMH) when "triggered", decay tells us at what rate the trust level decreases, intersecting with 0.0 once the decay duration has been reached. So, for a decay duration of 4, the resulting trust level values of three decay functions are seen in Table 6.

Time	Trust Level		
	Linear	SQRT	No Decay
0	1	1	1
1	0.75	0.5	0
2	0.5	0.29	0
3	0.25	0.13	0
4	0	0	0

Table 7: Trust Level Values for Various Decay Functions

If time is represented by t and the decay duration is d, we define Linear as $\tau = 1-(t/d)$ SQRT as $\tau = 1-(\sqrt{t}/\sqrt{d})$ and No Decay as $\tau = 1 \; if \; t = 0 \; ; 0 \; otherwise$ Figures 28 and 29 show the results of these functions given a decay duration of 4 encounters.

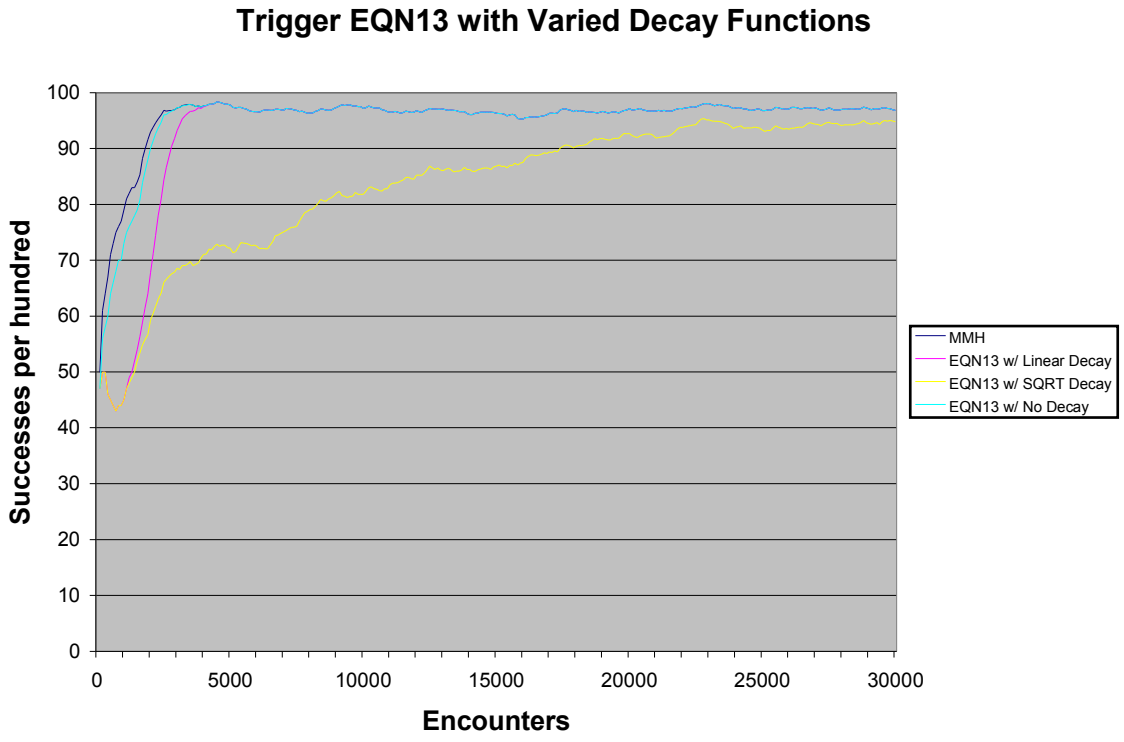

Figure 28: 50/50 Good/Bad Mix with EQN13 Trigger and Various Decay Functions

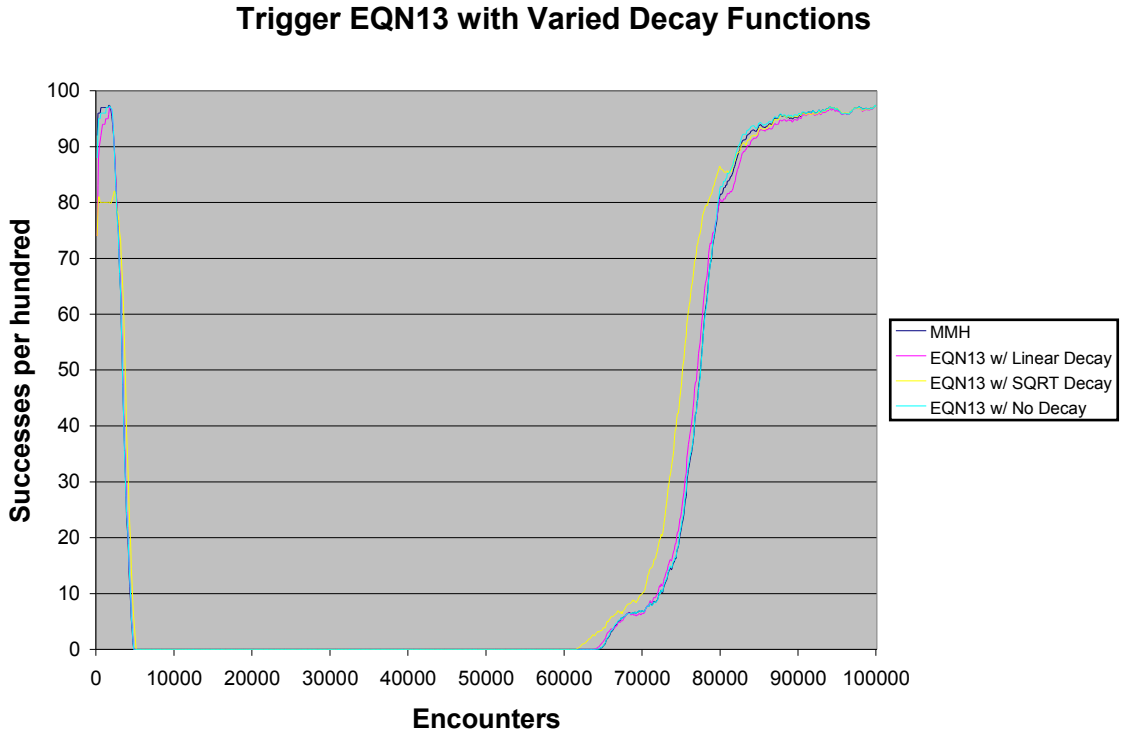

Figure 29: Reversal of Behavior Test with EQN13 Trigger and Various Decay Functions

The first thing to note from these results is the significant delay that the SQRT decay function puts into the 50/50 Good/Bad Mix test. However, the upside of SQRT is that this delay helps the reputation system un-learn faster in the Reversal of Behavior test – shortening the unlearning phase by 3300 encounters, and improvement of about 5.5%. As expected, the No Decay function (essentially a decay duration of 1) maps closest to the original MMH results, but does deliver a slight shortening of the unlearning phase by 200 encounters.

If we compare the score metrics $\bar{\Sigma}$ for these decay functions (Tables 7 and 8) we see that no matter which decay function is used, any improvement in the Reversal of Behavior test seems to be offset by a corresponding decrease in the 50/50 Good/Bad Mix.

	MMH	Linear	SQRT	No Decay
Score	27384	26808	23932	27307
Difference	0.0%	-2.1%	-12.6%	-0.3%

Table 8: Comparison of Decay Functions in 50/50 Good/Bad Mix

	MMH	Linear	SQRT	No Decay
Score	-48993	-48730	-45013	-48801
Difference	0.0%	0.5%	8.1%	0.4%

Table 9: Comparison of Decay Functions in Reversal of Behavior Test

9.1.4 Conclusion

After much experimentation, we conclude that there is no significant advantage to using trust level management for improving the accuracy or performance of the base reputation system. Trust level management can be used to improve relearning rates, but these improvements come at the cost of decreases in the rate of initial learning. Also, the concept of "shyness", while seeming good to protect against negative encounters in a hostile environment, has the disadvantage of not imparting any information about changes in the environment. So, placing our agents in a hostile environment, the more encounters they avoid, the less opportunity they will have to take advantage of successful encounters later.

We now propose addressing MMH's re-learning challenges by limiting the amount of encounters that are used to calculate reputation by creating an experience window, i.e. only counting the last n encounters when calculating reputation.

9.2 Experience Windows

9.2.1 Description

Experience windows are the second means we propose of improving MMH. By this we mean varying the amount of history an agent keeps for calculating reputation. In situations where the environment is changing rapidly, agents will have difficulty "unlearning" the behaviors they expect from their neighbors, and experience windows are intended to speed this process.

Unfortunately, experience windows come with a disadvantage as well: while an experience window speeds the "unlearning" process, it also removes information which might prove useful. Perhaps an agent has a distant memory of an interaction with a certain agent who behaved badly. An experience window eventually erases this memory leaving the agent vulnerable to making the same mistake with the badly behaving agent.

For our purposes we define an *experience window size* ω as the number of interactions remembered, and a *reputation re-calculation parameter* as the number of interactions between recalculations of the experience window. Reputation is re-calculated from agent history using the rules from the reputation system as if the previous encounters had never occurred. So, for example, if our experience window size is 200 encounters and our reputation re-calculation parameter is 300 encounters, then at encounters <300, all history would be used for reputation calculations, but at encounter 300 the first 100 encounters are removed from a reputation window and decisions are now based on the more recent encounters 101-300.

9.2.2 Test Cases

In order to test reputation windows, we will define TMMH as MMH using EQN13 trigger function, the SQRT decay function, and decay value of 4. Additionally, TMMH uses reputation windows, and we will test multiple reputation window sizes.

9.2.2.1 50/50 Good/Bad Mix

Because agents don't change strategies mid-course in this test, using a reputation window makes no difference on this test, as long as the size exceeds the number of agents in the simulation. Known GOOD agents always receive better reputation scores than known BAD agents, and it only takes one encounter with an agent to know its inclinations. Simulations performed on an environment with 100 agents returned exactly the same results as MMH when reputation windows of 400 to 2000 encounters were used.

9.2.2.2 Reversal of Behavior

Reversal of Behavior is the test on which we would like to see improvement. As predicted, adding a reputation window to MMH dramatically improves the re-learning time for the Reversal of Behavior test. Results for reputation windows ranging from 400 to 2000 encounters are seen in Figure 30.

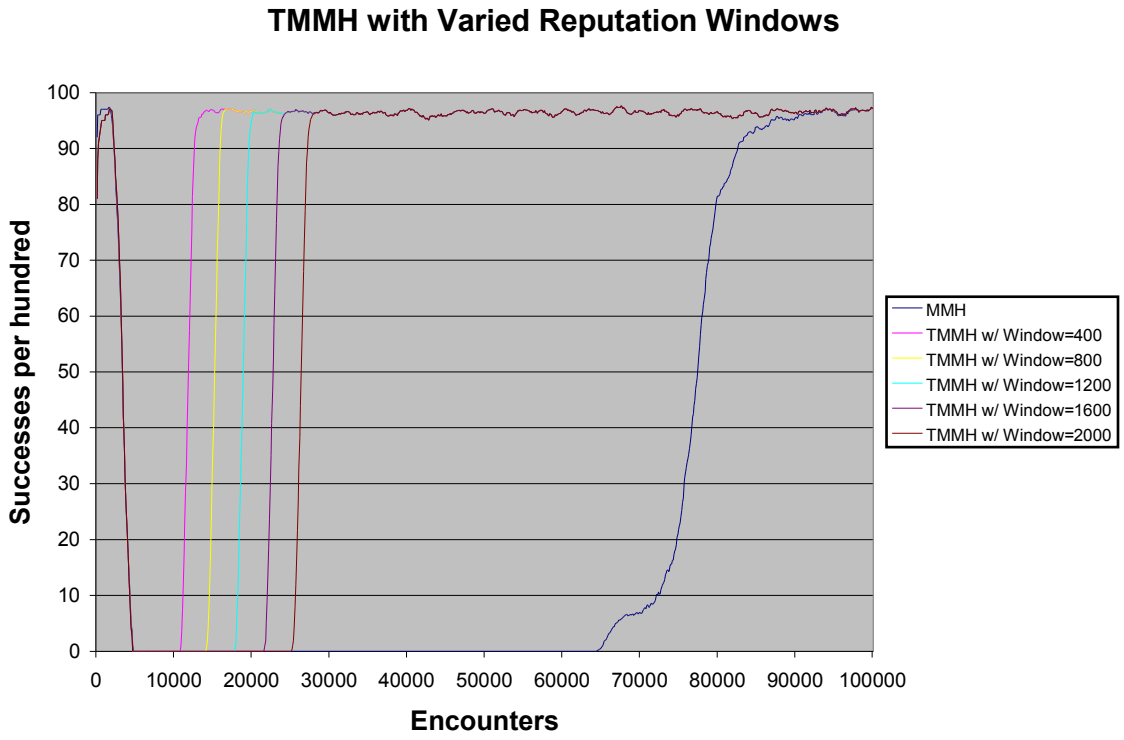

Figure 30: Reversal of Behavior Test with TMMH and Various Experience Window Sizes

9.2.2.3 Average Reputation

Figure 31 shows average reputation for a mix of one third GOOD agents, one third BAD agents, and one third RANDOM agents as in Section 8.2.2. Because TMMH is set to "shift" the reputation window every 100 encounters (as counted per agent), there is a marked change in reputation values at every 10,000 encounters (as counted globally) that is visible. This is because information is lost as the reputation window is shifted. The result is lower calculated probability of success (for the GOOD agents) and a higher calculated probability of success (for the BAD agents). This is only a problem in the current type of simulation, as long as the reputation window exceeds the number of agents in the simulated environment. However, for some applications this could be problematic.

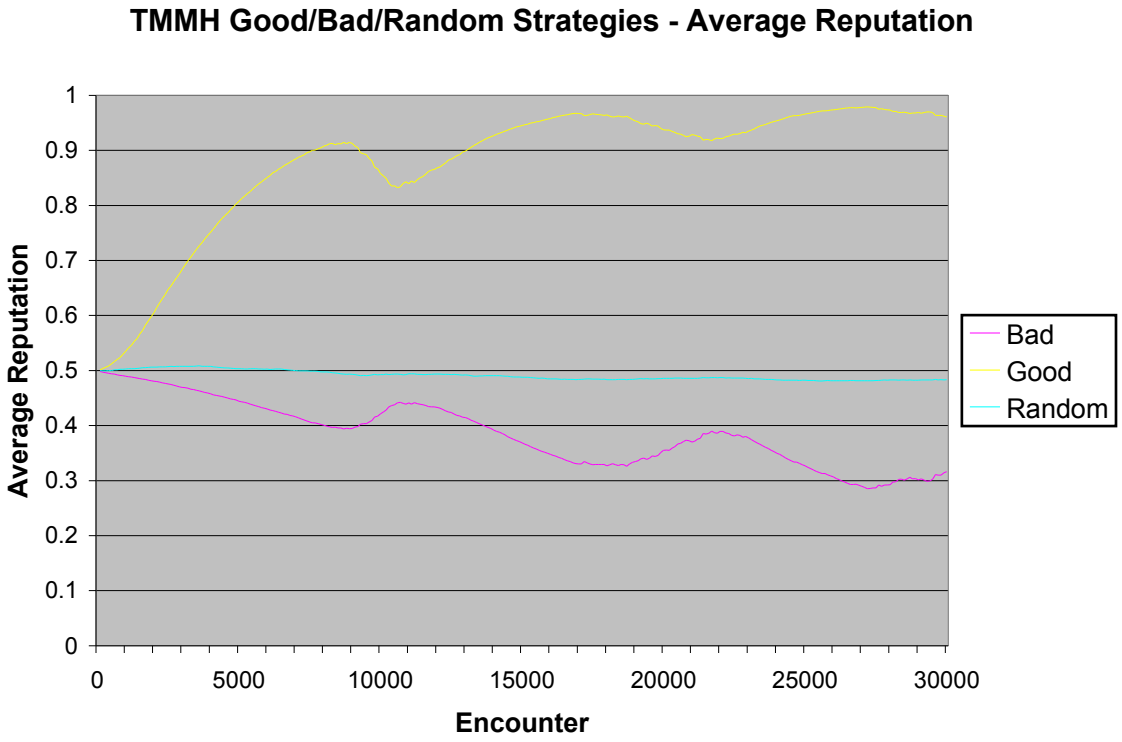

Figure 31: Average Reputation for Good/Bad/Random Agents with TMMH

9.2.3 Conclusion

Using an experience window is a useful modification to MMH, since without it a change in behavior takes too long to be reflected in the agent's reputation. There are some caveats to its use, however. Experience windows do lose information, and in an environment where interactions are not determined by an even random distribution there will be results from agent interactions lost as time progresses. Not every application will be able to tolerate that loss of information, particularly those which have a low tolerance for failure.

Also, when an experience window is recalculated, MMH reputations are recalculated back towards the initial neutral value of 0.5, and this goes against MMH's careful definition of reputation as a probability of success. Instead we are making a judgment that recent behavior is more likely to re-occur than more distant behavior, and this may not always be the case in every application.

10 Further Results

10.1 Description

In this section we will analyze all three reputation systems, MMH, ARH, and MMH with Trust Level Management (TMMH), in more complex scenarios. The first of these scenarios is where agents behave with strategies that oscillate between GOOD and bad behavior at some interval. The second scenario introduces deception, where a certain population of the simulated world lies about the reputation of their neighbors when acting as referrers for other agents.

10.2 Oscillating Strategies

In order to present a more complex simulation case, we have implemented the capability for agents to oscillate between GOOD and bad strategies at a pre-defined frequency. This extends the analysis done in the Reversal of Behavior test (see section 8.2.7) by examining changes in reputation when behavior changes frequently.

10.2.1 Test Case: Reputation of Agents with Oscillating Strategies

10.2.1.1 Description

Initially we would like to see how well each of the models performs under a more complex scenario which includes agents with different strategies. This simulation is populated with a mixture of agents, 20% which have a GOOD strategy, 20% which have a BAD strategy, 20% which have a RANDOM strategy, 20% oscillate between GOOD and BAD behavior every 50 encounters (GOOD_BAD), and 20% also oscillate but start with BAD behavior and change to GOOD every 50 encounters (BAD_GOOD).

10.2.1.2 Results for MMH Model

The MMH reputation system shows in Figure 32 that it treats BAD_GOOD agents in almost the same way as BAD agents, assuming that because they started out with BAD behavior that they will not improve. The reputation score decreases so quickly that oscillating BAD_GOOD agents have almost no opportunity to compete with RANDOM and GOOD_BAD agents for selection as a resource supplier. GOOD_BAD agents fare better in this scenario. Since their reputation initially increase, they are more likely to be selected and their reputation scores reflect the oscillating behavior to a very slight degree. The oscillating behavior is more visible in the changes in level of success which decreases whenever the GOOD_BAD agents turn to BAD behavior.

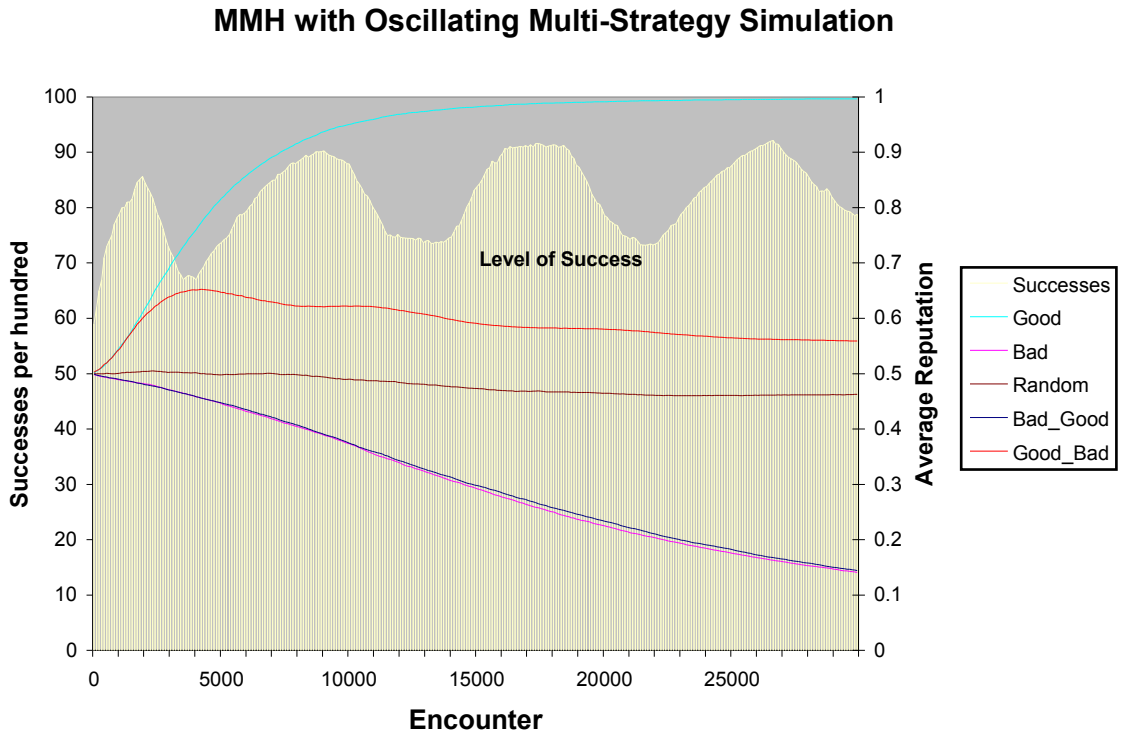

Figure 32: Average Reputations with Oscillating Behavior (MMH)

10.2.1.3 Results for ARH Model

Figure 33 shows the results of this simulation with the ARH model. Because ARH reputation values increase and decrease less quickly in response to success and failure than MMH, both GOOD_BAD and BAD_GOOD agents have opportunity to be selected, and the resulting reputation values therefore reflect the oscillating behavior quite clearly. Unfortunately, despite the fact that ARH reputation more accurately represents the agents' oscillation, the ARH reputation model still can not anticipate the changes that oscillation introduces, and fares worse overall than MMH does with regard to level of success.

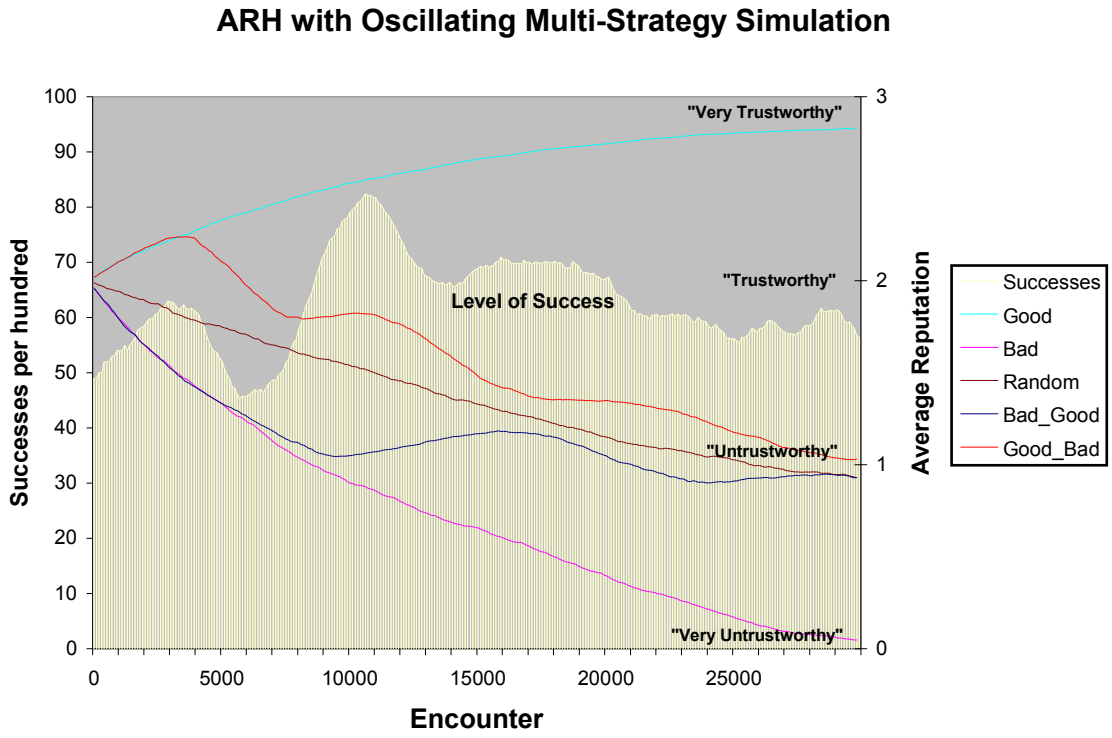

Figure 33: Average Reputations with Oscillating Behavior (ARH)

10.2.1.4 Results for MMH Model with TLM

When we add trust level management and reputation windows to MMH (referred to as TMMH from now on), the results are similar to MMH. However, the impact of the reputation window is seen clearly in the average reputation of the GOOD, BAD, and BAD_GOOD agents. Presumably this is much more visible with these strategies because because the rates of change of reputation are much higher with these strategies as compared to RANDOM and GOOD_BAD. Also, the peaks and valleys shown in TMMH's level of success are generally wider than MMH's level of success. This is probably due to the slightly slower learning rate that TMMH exhibits compared to MMH.

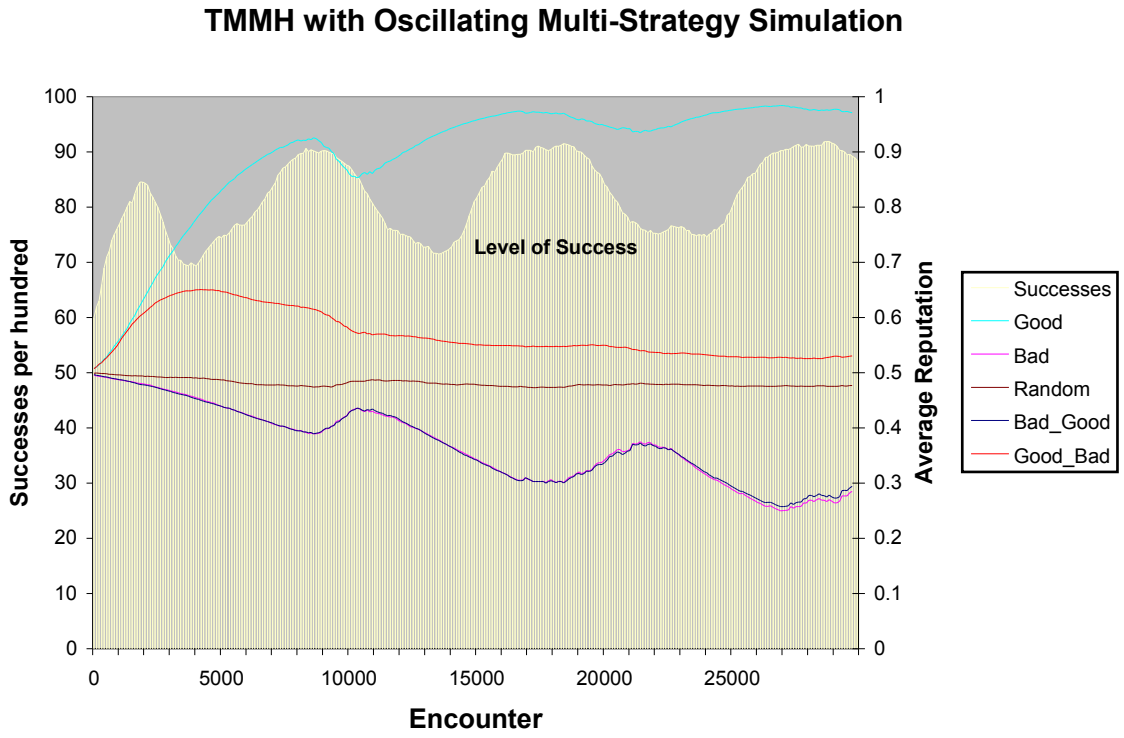

Figure 34: Average Reputations with Oscillating Behavior (TMMH)

10.2.1.5 Discussion

Oscillation is difficult for any reputation system to deal with, since reputation is by definition a reflection of past behavior and oscillation prescribes a behavior that is diametrically opposed to past behavior. Peak reputation values shown in this test consistently trail the associated peak in level of success by 1,000 - 3,000 encounters which shows that our reputation systems are slow to reflect the real situation. This might lead to the conclusion that the best approach for handling oscillation would be to design a model which drops an agent's reputation back to neutral at the first or second indication of a change in behavior. This would likely be vulnerable to other environmental challenges, however, and we leave this as an open question to be examined in future work.

10.2.2 Test Case: Effects of Varying Frequencies of Oscillation

10.2.2.1 Description

Like the previous test case, this one uses a mixture of agents with different strategies. The simulation is populated with 20% GOOD agents, 20% BAD agents, 20% RANDOM agents, 20% oscillating (first GOOD then BAD), and 20% oscillating (first BAD then GOOD). Oscillating agents change strategies every 25, 50, 75, and 100 individual encounters.

10.2.2.2 Results for MMH Model

MMH performed the best of the three in terms of overall level of success. From Figure 35 it is clear that the level of success follows the pattern of oscillation exhibited by the set of agents with the GOOD_BAD strategy. While the swings in the graph are sized proportionally with the frequency as expected, it is interesting to note that the size of swings does not decrease with time. This seems to indicate that MMH has not learned more about the environment at 30,000 encounters than it knew at 10,000 encounters.

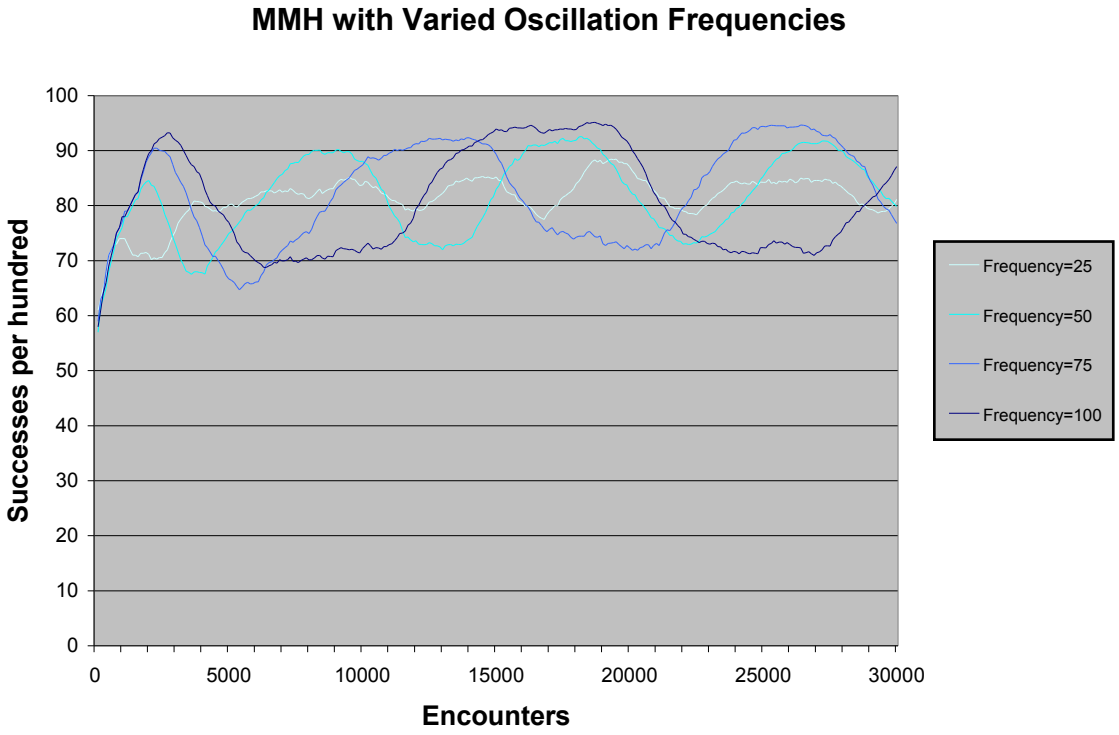

Figure 35: MMH with Varied Frequency of Oscillation

10.2.2.3 Results for ARH Model

The ARH model exhibits the same oscillating level of success in tune with the frequency of change by the GOOD_BAD agents. Compared to MMH, the highest levels of success are approximately 80% instead of 90-95%. However, one interesting note is that, unlike MMH, the oscillations begin to dampen as time goes on, indicating that the model reacts to lessons learned and there is less of a swing every time the agents oscillate behavior. Unfortunately,

along with the smaller swing, the level of success seems to trend downward as behavior is learned. This is especially evident with the lower frequency values.

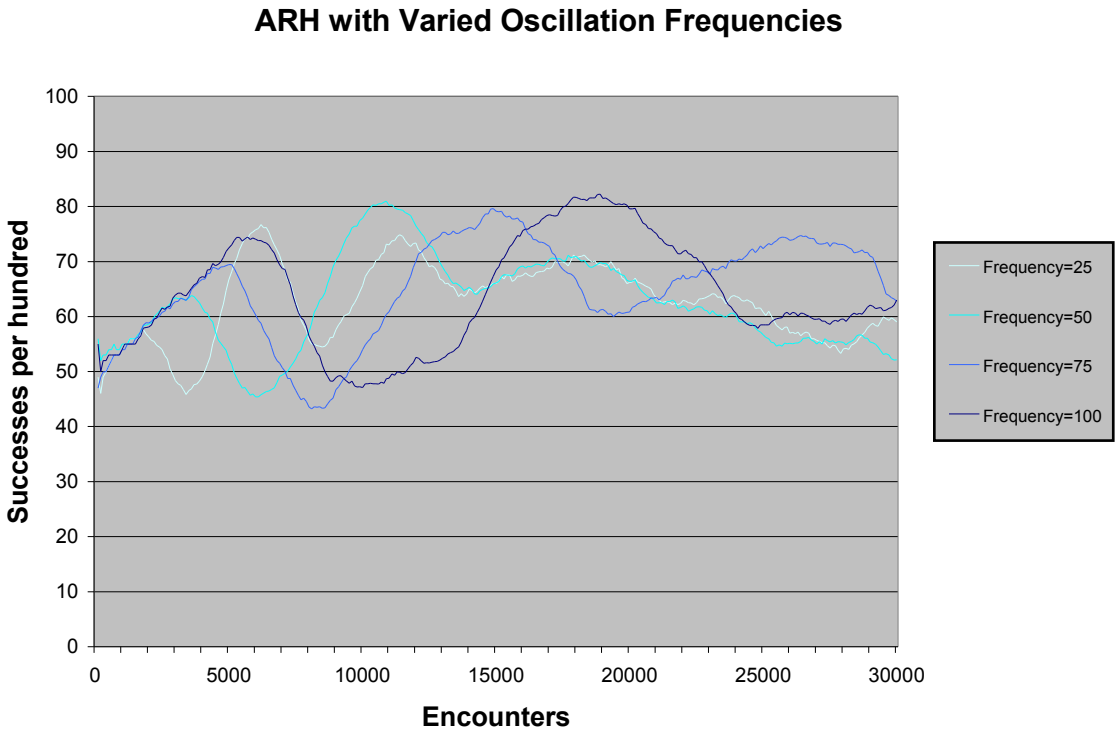

Figure 36: ARH with Varied Frequency of Oscillation

10.2.2.4 Results for MMH Model with TLM

The TMMH results, shown in Figure 37, exhibit similar shape and size oscillation as compared to MMH. However, the level of success is significantly smaller with maximums at around 80%. While TMMH has significantly fewer successes, in terms of score, TMMH performed on a par with MMH (see Figure 38). This is yet another indication that maximizing success requires taking risks, risks that TMMH with its trust level management scheme is not as willing to take as MMH.

10.2.2.5 Discussion

Increasing the frequency of oscillating behavior does not significantly change the overall level of success or score that any of these models can achieve. While level of success oscillates with the changes in strategy of the GOOD_BAD agents, only ARH seems to react negatively by giving GOOD_BAD agents progressively lower reputation ratings over time. This results in ARH agents occasionally selecting RANDOM and BAD agents instead of the oscillating agents and this contributes to an overall decline in level of success.

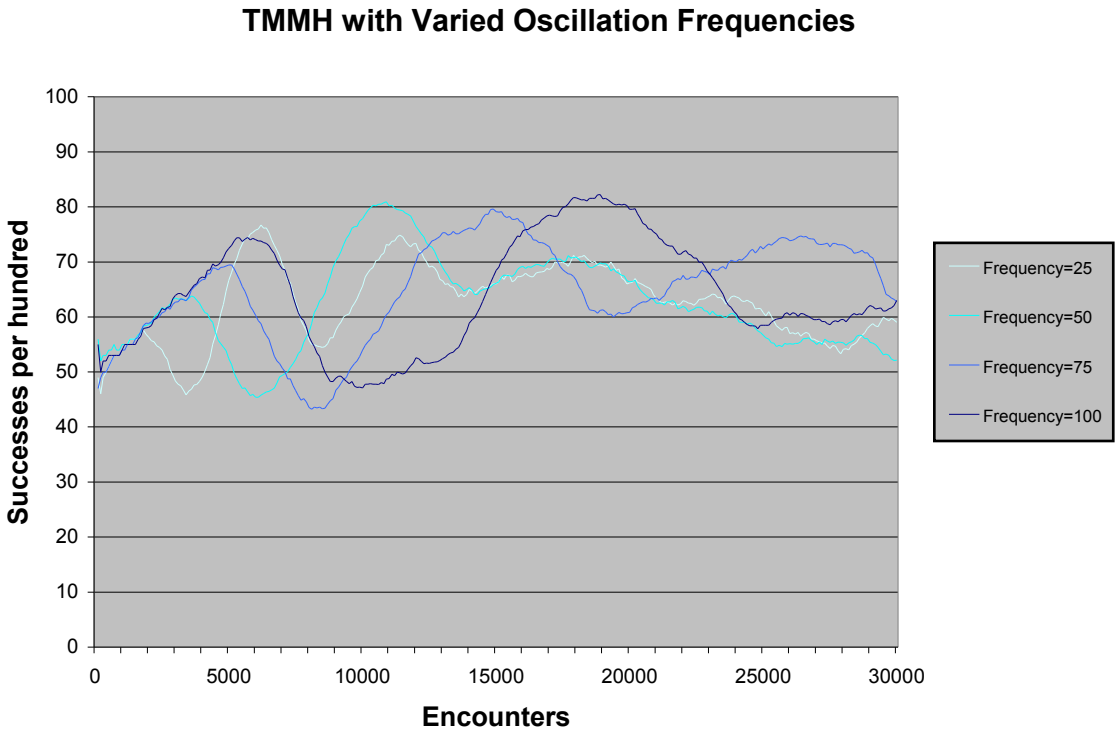

Figure 37: TMMH with Varied Frequency of Oscillation

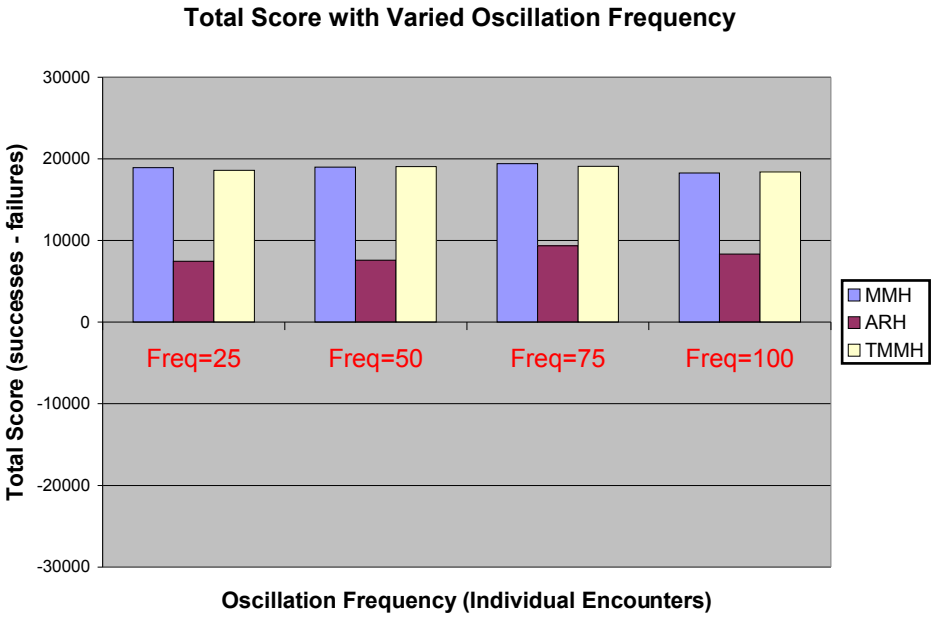

Figure 38: MMH, ARH, and TMMH Scores with Varied Frequency of Oscillation

10.3 Effects of Deception

One of the problems with using reputation in the real world is that opinions vary and reports of an individual's reputation are exaggerated because of the subjectivity that is involved with reputation. It is clear that automated reputation systems must be somewhat vulnerable to attacks involving deception. In this suite of tests we examine which types of deception are most effective in thwarting each of the reputation systems and what level of tolerance each system has to deceptive agents.

10.3.1 Test Case: Types of Deception

10.3.1.1 Description

In order to test deception we define a certain subset of agents as deceptive using the DECEPTIVE_RATIO parameter. We define deception as a modification of reputation ρ that is made whenever reputation is requested from an outside agent. Naturally, we make an assumption that agents calculate reputation correctly for their own interactions. After Yu and Singh [25] we define three different types of deception: complementary, exaggerated positive, and exaggerated negative. In each, deception is an operation which returns a fictional reputation ρ' which is based on the agent's real reputation value ρ, the maximum reputation value P_{max}, and an exaggeration coefficient α such that $0<\alpha<1$ The deception types are as follows:

- Complementary: $\rho' = P_{max} - \rho$
- Exaggerated positive: $\rho' = \alpha + \rho - \alpha\rho$
- Exaggerated negative: $\rho' = \rho - \alpha\rho / (P_{max} - \alpha)$

While it may be a large assumption that deceptive agents will always behave badly in the simulation, we have elected to first assign the role of deceiver to agents which have a BAD strategy, then select RANDOM and GOOD agents once we have run out of BAD agents. This was done to give ARH the benefit of the doubt, since ARH takes agent reputation into account when weighting referral information in its calculations.

Also, deceptive agents do not always return deceptive results, but select certain targets about which to lie. These targets are selected from the pool of GOOD agents first, then RANDOM, then BAD, in an effort to highlight the impacts of deception, and the number of targets is set by the parameter TARGET_RATIO. After the simulation, we will examine the level of success for the entire simulation environment over time and see how this is effected by the introduction of deception.

10.3.1.2 Results for MMH Model

The results of MMH are shown in Figure 39 which gives us the first indications of how deception affects reputation results. This simulation was performed with 20% deceptive agents which were targeting 60% of the total agents, and the exaggeration coefficient was set to 0.5. Exaggerated negative seems to have little effect on overall success, and since GOOD agents are assigned as targets, it seems that exaggerated positive deception results in reaching

the maximum level of success very quickly. Complementary deception results in the most visible negative impact to overall success.

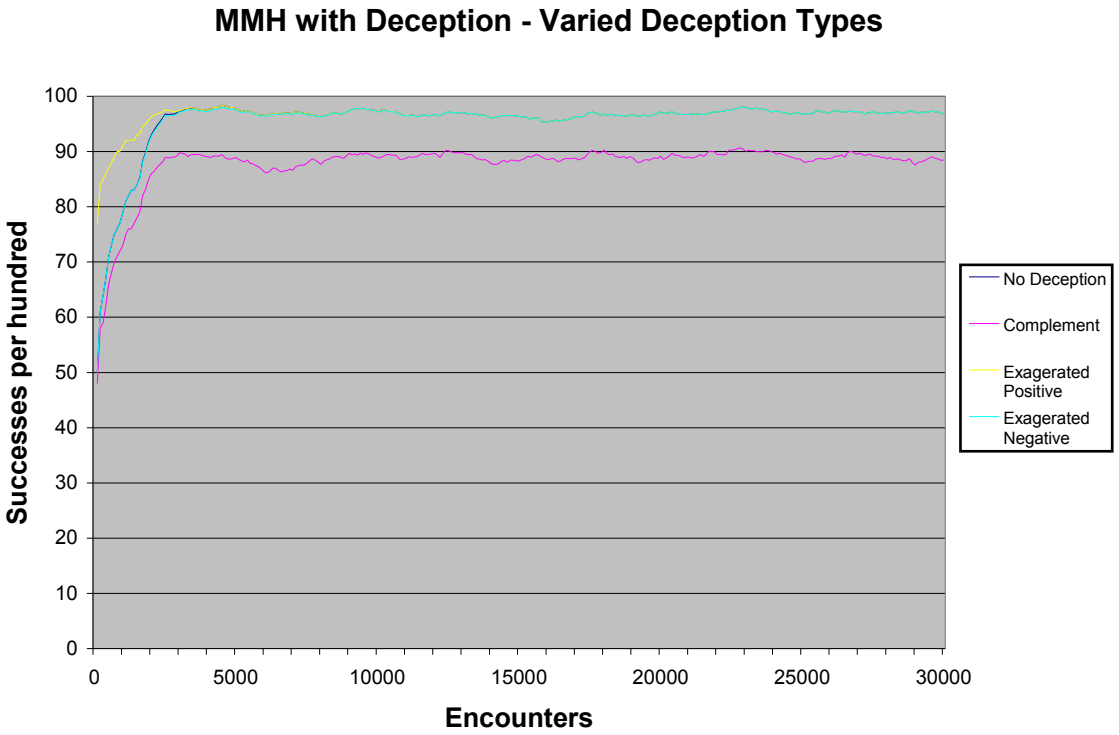

Figure 39: Results of MMH with Varied Deception Types

10.3.1.3 Results for ARH Model

ARH reacts somewhat differently to the different deception types in Figure 40. Here exaggerated positive is the least harmful, but still not as beneficial as with MMH. Exaggerated negative has a much more pronounced effect on the results than in MMH, and while ARH uses the semantic distance method of recalibrating the results of consistent under-estimators, it seems not to have effected the results for exaggerated negative deception very much. Lastly, complementary deception is once again proven to be the most damaging to overall success.

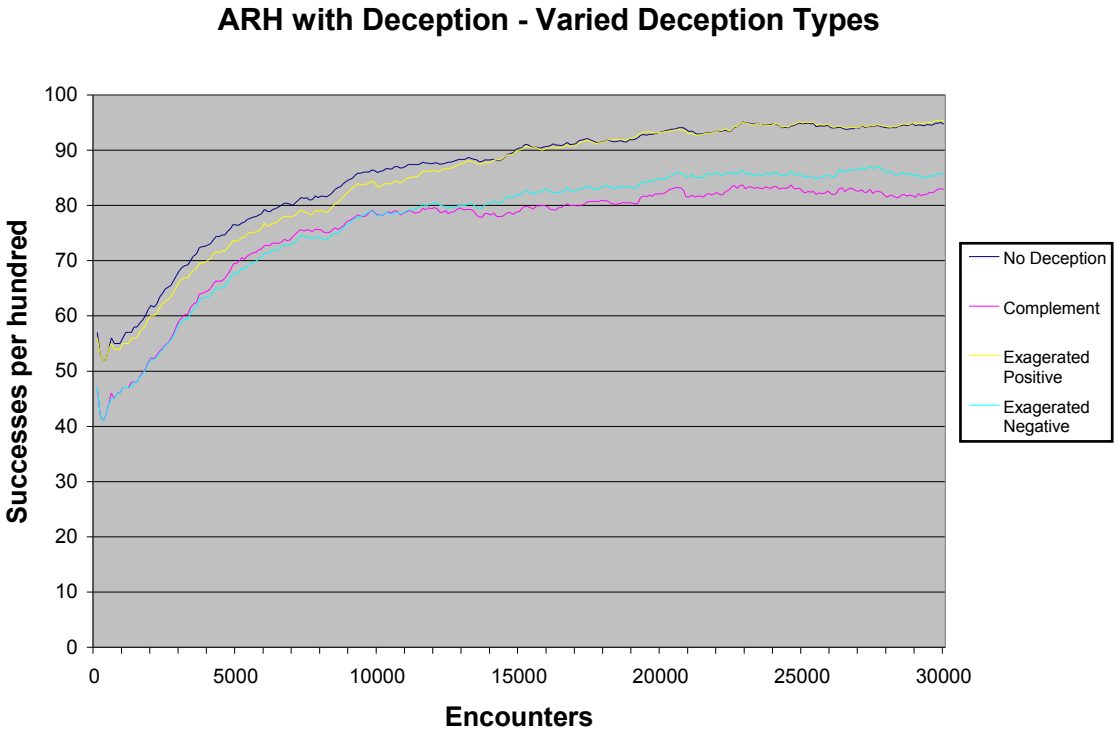

Figure 40: Results of ARH with Varied Deception Types

10.3.1.4 Results for MMH Model with TLM

Trust Level Management does not make significant differences on how MMH responds to deception, but careful examination of Figure 41 will show a slight depression in the level of success at approximately 10,000 and 20,000 encounters, particularly for complementary deception. This is due to the limited reputation window, and the effects of this window will become more evident in future tests.

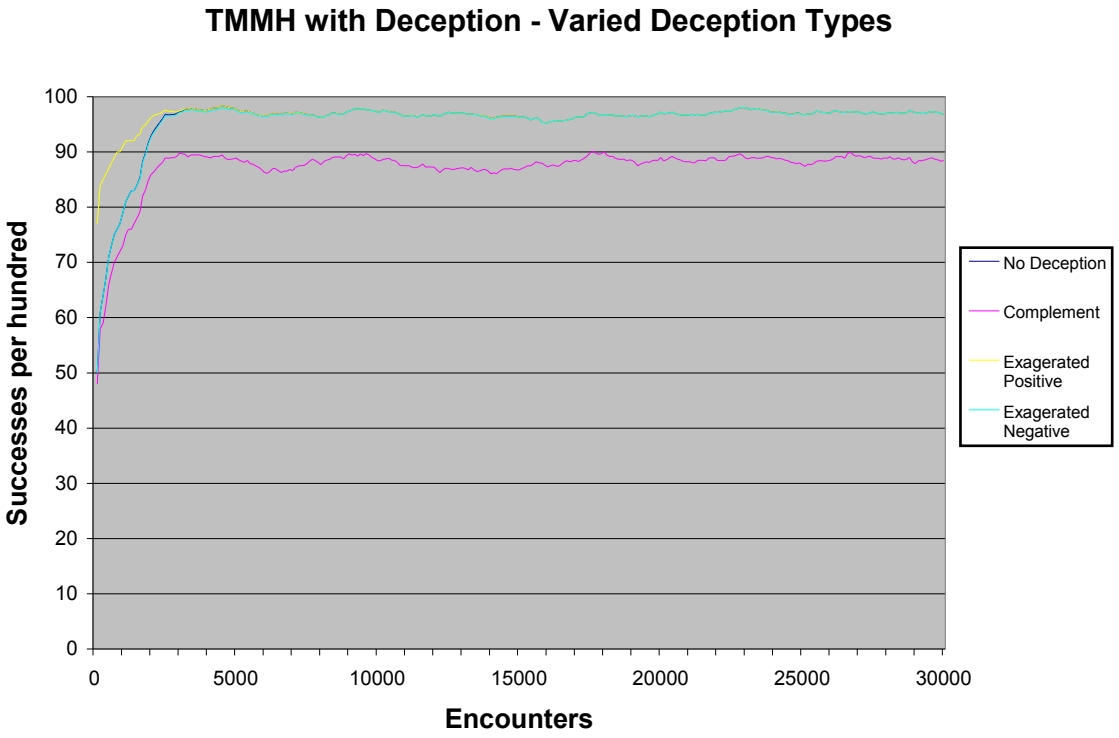

Figure 41: Results of TMMH with Varied Deception Types

10.3.1.5 Discussion

It is clear from the data that a 20% population of liars can have a significant impact on the level of success enjoyed by the entire population of agents. While exaggerated positive and negative deception types seem capable of reaching the maximum level of success Λ_{max}, complementary deception runs into a barrier in all three reputation systems. For the remainder of the tests we will focus on complementary deception in order to determine the tolerance that each system has against deception.

10.3.2 Test Case: Number of Deceptive Agents

10.3.2.1 Description

As in [25], we are interested in finding out what level of tolerance a reputation system has for deception. We predict that as the number of liars increase in the agent population, overall success levels will be adversely impacted. In the worst case scenario, perhaps deception would make the level of success dip below the 50% mark, which in the case of a 50/50 mix of GOOD and BAD agents would be worse than choosing randomly without the assistance of a reputation system. In this test we examine the effects of increasing the number of deceptive agents from 0 to 100%. Like the previous test, we choose first BAD agents to be deceptive, then GOOD agents once we have exhausted the population of BAD agents. In all of these tests we limit the target agents to only 20% of the population.

10.3.2.2 Results for MMH Model

In Figure 42 we see the unusual results of increasing the number of deceptive agents with the MMH reputation model. Initially increasing the number of deceptive agents steadily decreases the maximum level of success that MMH can achieve, up until 50%. Once the level of deceptive agents exceeds 50%, the level of success starts to increase once again, until with 100% deceptive agents, MMH is enjoying a maximum level of success equal to the simulation which did not have any deception. This behavior makes sense when we consider that increasing deception to 100% of the population evenly distributes the lies and allows MMH to learn behavior more easily.

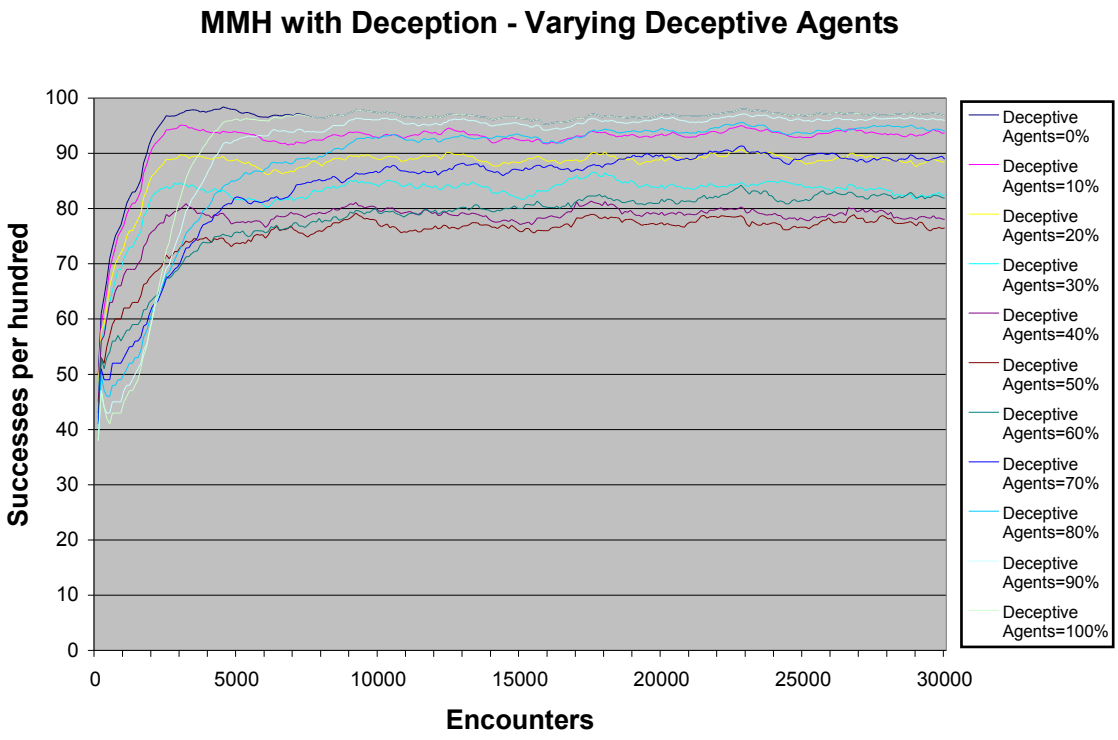

Figure 42: Results of MMH with Varied Number of Deceptive Agents

10.3.2.3 Results for ARH Model

ARH demonstrates a much more even distribution of performance with an increase in deceptive agents resulting in decreases in overall performance that are very proportional to the number of deceptive agents. Note that as deceptive agents approaches 100%, ARH can only attain a 50% success level – equal to randomly selecting agents with which to interact.

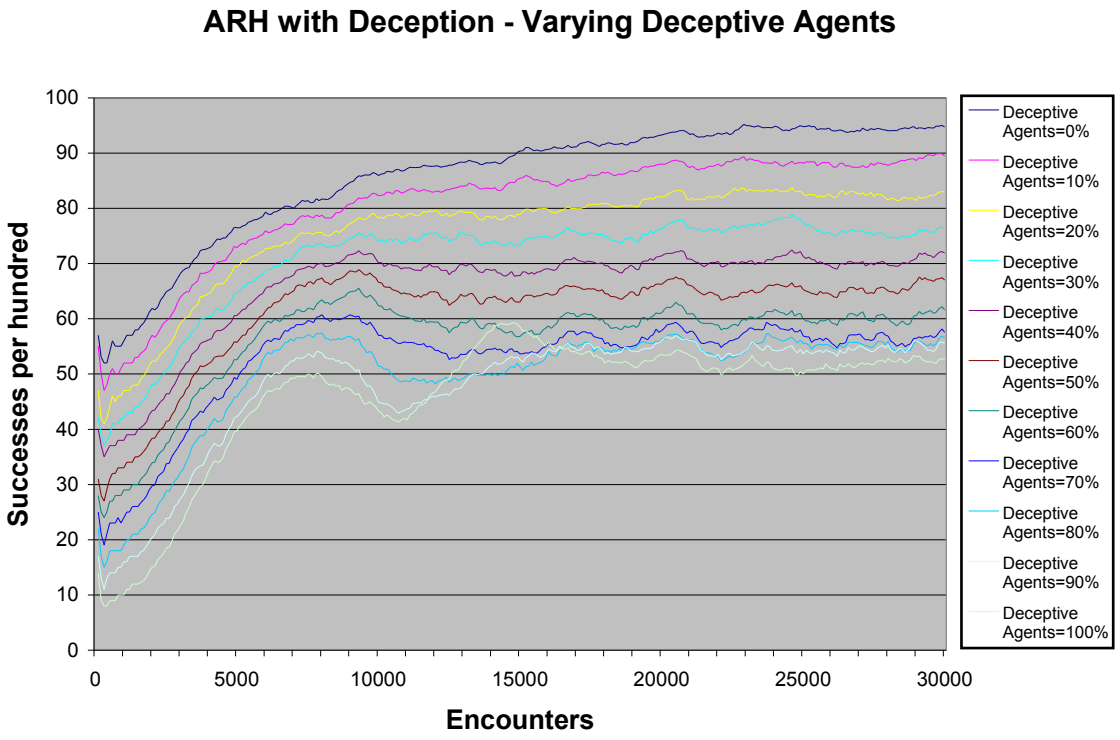

Figure 43: Results of ARH with Varied Number of Deceptive Agents

10.3.2.4 Results for MMH Model with TLM

When we add Trust Level Management to MMH, we immediately see the impact of reputation windows on our results (Figure 44). At encounters 10,000 and 20,000, the approximate times when agents would be trimming back their reputation windows, we see a large dip in level of success for the simulations of 50% deception and greater. This is because success at these levels of deception requires a great deal of prior knowledge about the environment. Once the reputation windows is trimmed back, this learning process must take place all over again.

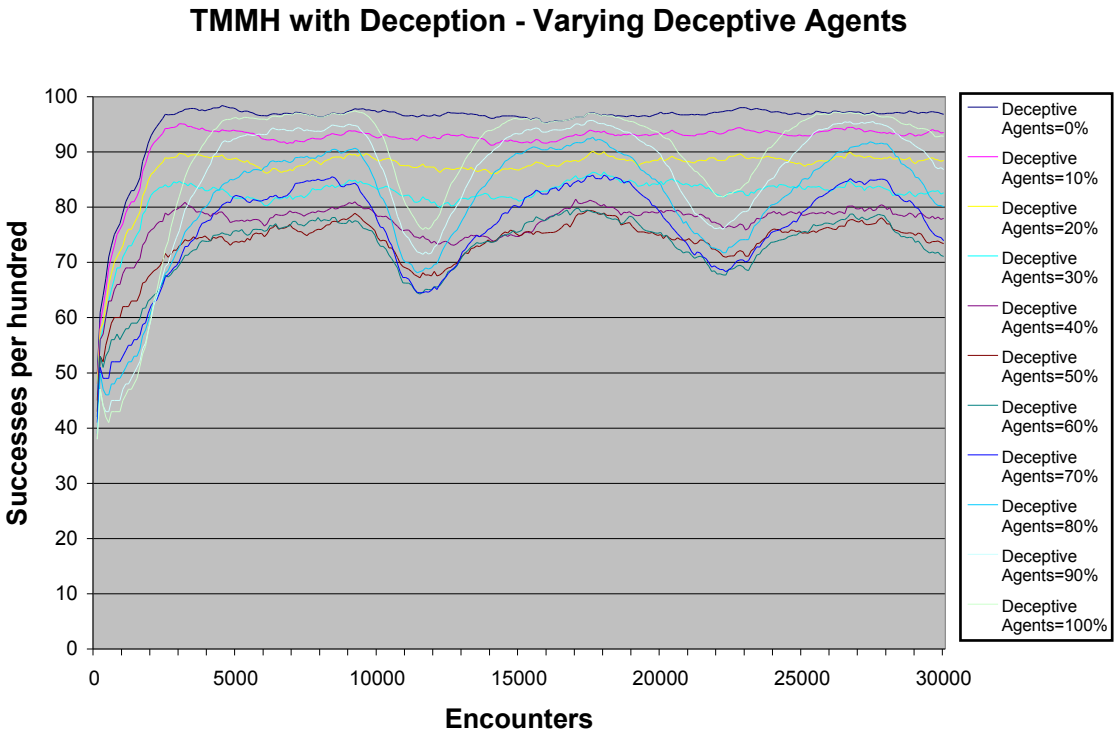

Figure 44: Results of TMMH with Varied Number of Deceptive Agents

10.3.2.5 Discussion

In comparison, MMH performs significantly better than ARH with regard to deception, even compensating for deception once lying becomes rampant in the environment. This is somewhat surprising since ARH is designed to be resilient to references from agents which are not in line with real experiences, and the designers of MMH have not explicitly made resistance to deception one of their design goals. However, neither of the models holds up well to large numbers of deceptive agents. Also TMMH shows the weakness of retaining limited amounts of history in this test as each agent must relearn its environment whenever the reputation window is recalculated.

10.3.3 Test Case: Number of Deception Targets

10.3.3.1 Description

While it is apparent that an overall increase in deception will decrease the effectiveness of a reputation system, it is possible that the reputation system will be able to recognize deception which comes from a certain population of agents and accommodate by discounting the opinions of those agents. In this test case we fix the number of deceptive agents at 20% and vary the amount of lies they tell, by varying the TARGET_RATIO.

10.3.3.2 Results for MMH Model

The results for MMH, shown in Figure 45, indicate that level of success is not impacted until the TARGET_RATIO exceeds 50%. Remembering that the first 50% of targets will be GOOD agents, we can conclude MMH experiences no detrimental effects until the deceptive agents begin to overrate BAD agents. At the point that BAD agents begin to receive complementary reputation ratings, MMH starts to have difficulty distinguishing between the agents with GOOD and BAD strategies.

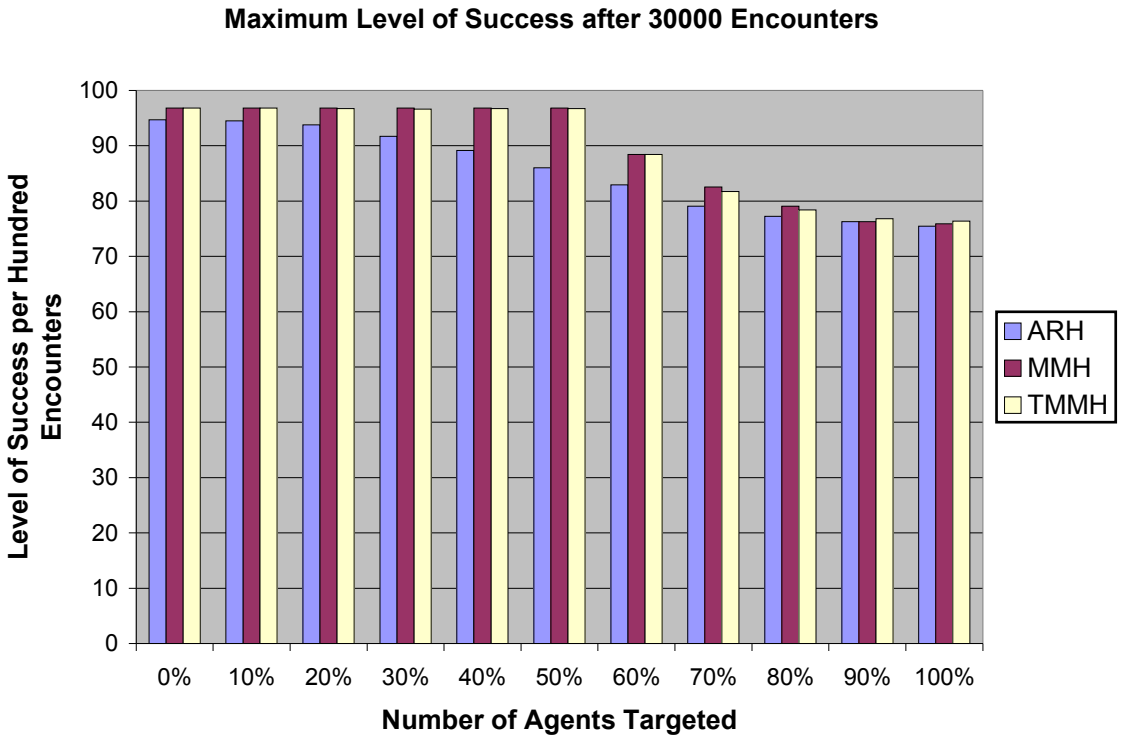

Figure 45: Maximum Level of Success for Varied Number of Targeted Agents

10.3.3.3 Results for ARH Model

ARH shows a much more measured decline in level of success, almost appearing proportional with the increase in targeted agents. Eventually it levels off at approximately 75% success.

10.3.3.4 Results for MMH Model with TLM

The TMMH reputation system fared about the same as MMH throughout this test, only outperforming MMH when the number of targets exceeded 90% of the total population. The improvement at this high rate of deception is likely due to the fact that information is being lost over the course of the simulation because of the sliding reputation window. At target levels of 90% and greater, a large portion of the information collected is actually *misinformation*. This means that with a smaller memory for interactions, TMMH has an advantage over MMH because it remembers less of the incorrect information.

10.3.3.5 Discussion

All three of the models presented in this section are somewhat vulnerable to deception and have decreasing level of success as the number of lies told increases. MMH has better resistance to deception overall than ARH or TMMH in this particular circumstance (using complementary deception), but this advantage could quickly fade given the right kind of lie.

As a general rule, the resistance to deception can likely be improved by tuning the degree to which "hearsay" evidence is included into the reputation model. Naturally this has a disadvantage of negatively impacting the rate at which the agent learns its environment. However, tuning the degree to which hearsay evidence is introduced seems to be a valid approach in the real world where statistics about deception can be observed over time.

Figure 46 shows an overall picture of resistance to deception, which we have defined earlier in Section 7.4.4.4.

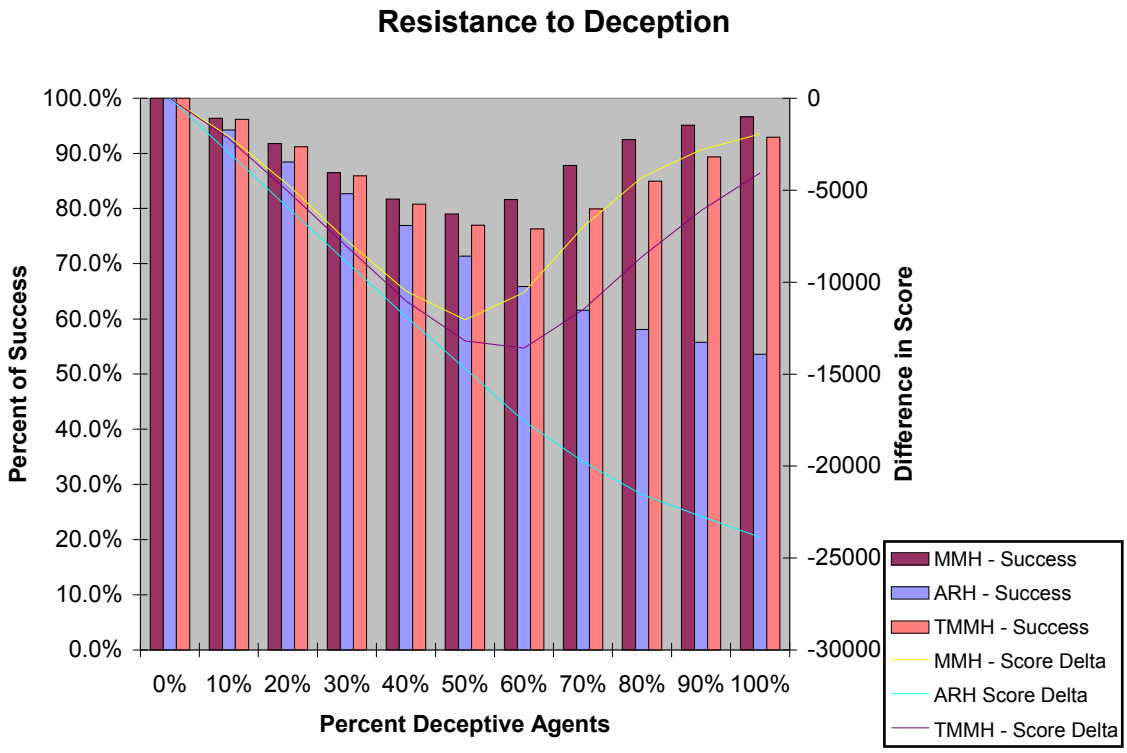

Figure 46: MMH, ARH, and TMMH Resistance to Deception

11 Analysis and Conclusion

11.1 Summarized Comparison of Reputation Systems

In this thesis we have compared two reputation systems, ARH and MMH, and proposed a modified version of MMH which employs trust level management and reputation windows to enhance its performance under quickly changing conditions. These reputation systems have been compared in terms of accuracy, performance, and resistance to deception.

11.1.1 Accuracy

Accuracy is defined as the percentage of successful predictions of behavior, or level of success. In comparison with our ideal level of success Λ_{max}, MMH did better than ARH in approaching levels close to Λ_{max} for the majority of simulations performed. ARH was able to eventually reach the level of success that MMH did in some of the simplest simulations, but as complexity increased, ARH fell further behind.

11.1.2 Performance

Performance and accuracy can be increased by tuning certain parameters; in order to get an effective reputation system, the environment must be well understood so that these parameters can be engineered correctly. In every simulation MMH was able to outperform ARH and TMMH. However, since it was the fastest learner, it also had difficulty in unlearning its environment. MMH's capabilities were most significantly undercut by the Reversal of Behavior test. In this test ARH was able to unlearn behavior much faster than MMH. The modifications proposed to MMH in the form of TMMH show that it is possible to compensate for this disadvantage.

11.1.3 Resistance to Deception

In all of the models examined, it seems that the impacts of deception are felt substantially once the number of liars exceeds 20% of the population. This is similar to the results found by Yu and Singh in [25]. Increasing the number of liars, the number of lies they tell, and the size of the lies are effective in reducing the accuracy and performance of the reputation systems reviewed here. One way to counter the effects of deception are to weight each agent's personal experience higher than hearsay evidence. For this reason MMH, ARH, and many of the reputation systems surveyed in this paper include methods of weighting hearsay evidence.

11.2 Contributions

This thesis represents a beginning for objectively comparing reputation systems. We propose that reputation and trust are different concepts, and, along with Aberer and Despotovic [2] that a reputation system is comprised of three components: a mathematical representation of reputation, a decisioning or trust function, and a communications protocol whereby reputation information is shared. This thesis focuses on assessing the first two components while defining a static communications protocol to ensure that each model is able to obtain the same amount of information and therefore ensure fairness in comparison tests.

In order to ascertain the utility of a particular mathematical model to represent reputation, we have created a simulation environment has a theoretical upper limit for performance that an ideal model would be able attain. The simulated models were then compared to this ideal model. Simulations were then built from a variety of conditions including dramatically changing environments and the use of deception.

We have proposed two methods of improving reputation systems to better handle changing environments called Trust Level Management and Experience Windows. In doing this we have shown that learning and unlearning rates are key to the overall performance of a reputation system, and that these goals are diametrically opposed. Also, we demonstrate that the process of learning and unlearning requires failure. This runs counter to the goal of Trust Level Management, which is to minimize failure. Experience Windows are useful in handling dramatic changes to the environment. However, eliminating information from the reputation model can be a disadvantage if the window is too small. Overall, Trust Level Management and Experience Windows are useful for tuning the performance of a reputation model for certain environments.

11.3 Future Work

Given the generic simulation model develop here, there are many possibilities for possible experimentation. Some of these include the following:

- Introducing groups and trust within groups to allow the experiment to scale to more than 1000 agents.
- Modeling the behavior of communities instead of using even distributions of population and interaction.
- Enabling chained references in order to accommodate a wider variety of simulation models.
- Enabling agents to autonomously change strategies in order to achieve greater benefit.
- Allowing some kind of collaboration between deceptive agents.

Once potential reputation system candidates are identified for a given application, these systems should be tested against real-world data to ensure that they are suitable for the actual conditions in which they would be placed.

11.4 Conclusion

In conclusion, we have found that reputation systems are useful for predicting success of interactions between two known or unknown parties. However, a primary limitation of reputation systems remains deception, and these models can only tolerate small amounts of deception. Additionally, when reputation systems are placed into environments which change dramatically, they must unlearn prior behavior. This unlearning process requires a significant amount of failure before successes can be achieved in the new environment. Instructing reputation agents to be shy when faced with a great deal of failure only increases the time required to restore the system to success. There may be no ideal mathematical model for reputation, but through simulation we have shown ways in which these models can be engineered and tuned to work well given a particular environment.

12 Bibliography

[1] Alfarez Abdul-Rahman and Stephen Hailes. Supporting Trust in Virtual Communities. *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Maui, Hawaii, January 2000.

[2] Karl Aberer and Zoran Despotovic. Managing Trust in a Peer-2-Peer Information System. *Proceedings of the Tenth International Conference on Information and Knowledge Management*, Atlanta, GA, November 2001.

[3] Karl Aberer. P-Grid: A self-organizing access structure for P2P information systems. *Proceedings of the Ninth International Conference on Cooperative Information Systems*, 2001.

[4] Michael Argyle. Cooperation: The Basis of Sociability. Routledge, London, 1991.

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

[6] Randy Barrett, quoting Laura Koetzle in "Threats to Network often come from next cubicle." *Government Security News*. August 2004. http://www.gsnmagazine.com/aug_04/inside_insider.html Retrieved as of 3/26/05.

[7] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. *IETF RFC 2693,* September 1999. http://www.ietf.org/rfc/rfc2693.txt. Retrieved as of 3/26/05.

[8] Eric J. Friedman and Paul Resnick. The Social Cost of Cheap Pseudonyms. *Telecommunications Policy Research Conference*, Washington, DC, October 1998.

[9] Diego Gambetta. "Can We Trust Trust?" in *Trust*. Diego Gambetta, ed. Blackwell Publishing, pp. 213-237. 1990.

[10] The American Heritage® Dictionary of the English Language, Fourth Edition, Houghton Mifflin Company, 2000.

[11] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. *IETF RFC 3280,* April 2002. http://www.ietf.org/rfc/rfc3280.txt. Retrieved as of 3/26/05.

[12] S. Kamvar, M. Schlosser, and H. Garcia-Molina. EigenRep: Reputation Management in P2P Networks. *In Twelfth International World Wide Web Conference*, 2003.

[13] Seungjoon Lee, Rob Sherwood, Bobby Bhattacharjee. Cooperative Peer Groups in NICE. *IEEE Infocom*, April 2003.

[14] S. Marsh. Formalizing Trust as a Computational Concept. PhD Thesis, Department of Computing Science and Mathematics, University of Sterling, April 1994.

[15] Ruggero Morselli, Jonathan Katz, and Bobby Bhattacharjee. A Game-Theoretic Framework for Analyzing Trust-Inference Protocols. *Second Workshop on the Economics of Peer-to-Peer Systems,* Cambridge, MA, June 2004.

[16] L. Mui, M. Mohtashemi, C. Ang, P. Szolovits, and A. Halberstadt. Ratings in Distributed Systems: A Bayesian Approach. *Proceedings of the 11th Workshop on Information Technologies and Systems,* New Orleans, LA, December 2001.

[17] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. A Computational Model of Trust and Reputation. *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, Maui, Hawaii. January, 2002.

[18] L. Mui, A. Halberstadt, and M. Mohtashemi. Notions of Reputation in Multi-Agents Systems: A Review. *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002.

[19] P. Resnick and R. Zeckhauser. Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System. *Advances in Applied Microeconomics, Volume 11,* Amsterdam, Elsevier Science. 2001.

[20] Sandip Sen and Neelima Sajja. Robustness of Reputation-based Trust: Boolean Case. *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002.

[21] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[22] W. Stallings. The PGP Web of Trust. *BYTE*, Feb 1995. http://www.byte.com/art/9502/sec13/art4.htm. Retrieved as of 3/26/05.

[23] Bin Yu and Munindar P. Singh. An Evidential Model of Distributed Reputation Management. *Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002.

[24] Bin Yu and Munindar P. Singh. Distributed Reputation Management for Electronic Commerce, *Computational Intelligence*, Volume 18, Issue 4, pages 535-549, 2002.

[25] Bin Yu and Munindar P. Singh. Detecting Deception in Reputation Management. *Proceedings of Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Melbourne, Australia, July 2003.

[26] Bin Yu, Munindar P. Singh, and Katia Sycara. Developing Trust in Large-Scale Peer-to-Peer Systems. *Proceedings of First IEEE Symposium on Multi-Agent Security and Survivability*, Philadelphia, PA, August 2004.