6-10-2019

The Trust-Based Interactive Partially Observable Markov Decision Process

Richard S. Seymour

The Trust-based Interactive Partially Observable Markov Decision Process

THESIS

Richard Seymour, Captain, USAF

AFIT/GCS/ENG/09-09

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GCS/ENG/09-09

The Trust-based Interactive Partially Observable Markov Decision Process

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Richard Seymour, B.S.O.R.

Captain, USAF

June 2009

AFIT/GCS/ENG/09-09

The Trust-based Interactive Partially Observable Markov Decision Process

Richard Seymour, B.S.O.R.
Captain, USAF

Approved:

/signed/	10 June 2009
Gilbert L. Peterson, PhD	date
/signed/	10 June 2009
Gary B. Lamont, PhD	date
/signed/	10 June 2009
LtCol. Brett J. Borghetti	date
/signed/	10 June 2009
Maj. Michael J. Mendenhall	date

Abstract

Cooperative agent and robot systems are designed so that each is working toward the same common good. The problem is that the software systems are extremely complex and can be subverted by an adversary to either break the system or potentially worse, create sneaky agents who are willing to cooperate when the stakes are low and take selfish, greedy actions when the rewards rise. This research focuses on the ability of a group of agents to reason about the trustworthiness of each other and make decisions about whether to cooperate. A trust-based interactive partially observable Markov decision process (TI-POMDP) is developed to model the trust interactions between agents, enabling the agents to select the best course of action from the current state. The TI-POMDP is a novel approach to multiagent cooperation based on an interactive partially observable Markov decision process (I-POMDP) augmented with trust relationships.

Experiments using the Defender simulation demonstrate the TI-POMDP's ability to accurately track the trust levels of agents with hidden agendas The TI-POMDP provides agents with the information needed to make decisions based on their level of trust and model of the environment. Testing demonstrates that agents quickly identify the hidden trust levels and mitigate the impact of a deceitful agent in comparison with a trust vector model. Agents using the TI-POMDP model achieved 3.8 times the average reward of agents using a trust vector model.

Acknowledgements

I would like to thank my family for their support and my committee for their guidance.

Richard Seymour

Table of Contents

Page

List of Figures

List of Tables

Table Page

List of Symbols

Symbol		Page
S	State	7
A	Action	7
T	Transition Function	7
R	Reward Function	7
Ω	Observation Set	7
O	Observation Function	7
b	Agent Belief	8
β	Normalizing Constant	8
π	Decision Policy	8
γ	Decay Factor	8
U	Expected Reward	8
I	Number of Agents	10
h	Finite Horizon	10
IS	Interactive State	11
M	Model	11
f	Agent's History and Observation Function	11
h	Agent's History	11
θ	Agent Type	12
l	Strategy Level	13
τ	Trust Model	22

List of Abbreviations

Abbreviation		Page
TI-POMDP	Trust-based Interactive Partially Observable Markov Decision Process	iv
I-POMDP	Interactive Partially Observable Markov Decision Process	iv
MDP	Markov Decision Process	7
POMDP	Partially Observable Markov Decision Process	7
DEC-POMDP	Decentralized Partially Observable Markov Decision Process	10
ANOVA	Analysis of Variance	47
CTG	Cooperative Tiger Game	51
SSTR	Sum of Squares Due to Treatment	53
SSE	Sum of Squares Due to Error	54
MSTR	Mean Square Due to Treatment	54
MSE	Mean Square Due to Error	54

The Trust-based Interactive Partially Observable Markov Decision Process

I. Introduction

In a cooperative system, groups agents accomplish tasks by working together and leveraging the skills and abilities of every agent. While the decisions of individual agents may not maximize their personal gain, they attempt to maximize the overall reward for the group through these cooperative actions [3]. These cooperative actions leverage the individual capabilities of several agents to perform complex tasks beyond the ability of a single agent.

The complex programming required to generate cooperation and allow coordination among agents presents a target for adversaries trying to reduce the effectiveness of the system. Cooperative systems are based on an underlying level of trust (either implicit or explicit) among the group of agents, and the potential exists for "sneaky" agents to exploit this trust by cooperating when the stakes are low and taking selfish, greedy actions when the rewards rise. Ultimately, these actions can degrade the performance of the system leading to a complete lack of cooperation.

The concept of trust is central to agent interactions [10] in much the same way as human interactions. Just as a person refuses to buy a car from a salesman he does not trust, an autonomous agent refuses to cooperate with an agent it does not trust. Trust can be thought of as the fundamental difference between a cooperative and a competitive environment. In a completely cooperative environment, the agents trust and rely on one another to accomplish their goals. In a competitive environment, agent a believes that agent b will act in its own best interests to the detriment of agent a. In between lies a gray area where agents must choose whether to cooperate based on their belief in the trustworthiness of others.

Typical trust modeling treats trust as a "hidden" rating [18, 22, 25]. Once an agent identifies the appropriate rating of another agent, it uses that rating to determine whether or not to interact with the other agent. This method is similar to the eBay™ user rating system. An eBay buyer looks at the ratings of a seller before deciding to purchase an item. If the seller has a positive score, the buyer can purchase with confidence. An occasional pitfall with this system is a deceitful seller looking to cash out. The seller builds a large positive rating before selling several high priced items that he never intends to deliver. Buyers pay for the items and the seller vanishes with the money.

A similar scenario can play out in a multi-agent environment for a variety of reasons. A sneaky agent can act trustworthy for a period of time to build trust until it decides to betray the other agents. A "hacker" can alter an agent's programming causing it to compete instead of cooperate. A random bit flip could corrupt an agent causing it to behave sporadically. In a world where trusting the wrong agent once can lead to the loss of time, money, and system capabilities, a reliable autonomous agent requires a robust ability to determine the true intentions of the other agents it interacts with. This paper takes a step down the path to robust agent reasoning through a multi-agent environment based on trust modeling. This chapter defines the problem, describes the goal of this research, presents the methodology, examines the significance and limitations of the study, and outlines the paper format.

1.1 Problem

This research focuses on the ability of a group of agents to reason about the trustworthiness of each other and make optimal decisions about whether to cooperate. The agents attempt to maximize the common reward of the system while still protecting their individual interests. When possible, the agents attempt to cooperate with one another with the expectation of receiving some benefit from the cooperation.

To successfully cooperate, the agents must be able to identify the appropriate level of trust to place in one another and identify when that trust level changes [10].

Failing to identify the appropriate level of trust opens an agent to exploitation and increases the potential for an agent to miss an opportunity for cooperation.

While trust modeling usually focuses on an individual agent's level of trust in the other entities it interactions with [18,25], this research attempts to create a multi-agent model that entails the trust models and decision processes of all of the agents in the environment. This level of modeling captures the impact of individual actions and trust fluctuations on the entire collective as the agents to manage domain uncertainty.

1.2 Research Goal

The goal of this research is to merge trust modeling and multi-agent decision making so that agents can use their perception of the world (including their estimate of the trustworthiness of others) to select an appropriate course of action when the other agents in the collective become corrupted. The specific objectives of this research are:

- Develop a trust-base multi-agent framework that allows agents to make decisions according to their beliefs about the current state of the world and the trustworthiness of the other agents.
- Implement the trust-based multi-agent framework in a simulation to demonstrate cooperation as an emerging behavior.
- Test the trust-based multi-agent framework against a trust vector model [17] to demonstrate the utility of the framework.

The simulation and testing must answer the question of whether the addition of trust modeling results in better agent decisions. The framework must be adaptable, support a wide range of individual trust models, and include the ability of individual agents to use unique models within the same environment.

1.3 Methodology

This thesis describes the TI-POMDP, a multi-agent framework that takes the novel approach of combining a multi-agent decision process with trust modeling. This

framework expands trust modeling to demonstrate the impact of trust on a group of agents, rather than just an individual agent. The framework is based on a modified I-POMDP [7] with trust modeling added into the decision making process. The trust model is updated with the observations from the I-POMDP portion of the framework. The updated trust model then aids the state belief update of the I-POMDP portion. While an experience-based trust model [2] is used in the demonstration of this framework, any trust model could be used. Because the framework is based on the I-POMDP, an optimal solution to the framework requires up to double exponential time [21].

A two agent game and a multi-agent simulation are used to evaluate the performance of the framework, specifically looking at the amount of cooperation between agents and rewards achieved by the agents. The game is a modified version of the Tiger Game [7, 13]. The multi-agent simulation models a group of agents attempting to cooperate in an environment with fluctuating trust. Testing includes an analysis of the simulation settings that affect the framework results, a demonstration of the framework execution time, and a comparison between the framework and a trust vector model [17].

This research assumes that all environment agents are operating within the framework and that groups of agents can not collude to subvert the ability of the framework. While the simulation entails factors in addition to trust modeling, the testing is designed to explore the specific effects of the trust modeling. The testing does not include communications between the agents. This eliminates the usage of trust models that are based on reputation networks [22, 24].

1.4 Significance

This research merges the high level decision reasoning of a multi-agent environment with the individual oversight of trust modeling. The end result is a decision framework that allows unique agents to reason about the trustworthiness of one another. The agents determine whether to work with other agents that may have

different capabilities and different trust models. In the simulation, the agents using the TI-POMDP accomplish tasks 36.2 percent faster and receive 3.8 times the average reward of agents using a trust vector model [17].

1.5 Outline

This thesis is divided into six Chapters including this introduction. Chapter II provides a literature review of the current work in solving multi-agent environments and trust modeling. Chapter III describes the framework modeling a multi-agent environment incorporating trust into the agent decision process. Chapter IV presents a simulation based on the framework from Chapter III. In addition, Chapter IV outlines the testing methodology used to validate the simulation. Chapter V presents the results and analysis of the testing. Chapter VI provides the detailed conclusion of the trust-based multi-agent environment and outlines future areas of research. This thesis is designed for readers with a working knowledge of I-POMDPs, but related work is referenced where applicable.

II. Literature Review

Similar to heavy objects needing two or more people to lift, many problem domains require multiple agents to satisfy all domain requirements. A given task may require specialized skills from two or more agents, or the task could require multiple agents in separate locations at the same time. Either way, the overall problem complexity increases as coordination and cooperation components are added to each individual agent. This chapter includes a brief description of a state within an environment, examines existing multi-agent environment models, and presents a wide range of trust modeling techniques.

2.1 States

In an environment, the state is the current setting of all variables within the environment. Some settings may be known (location, time of day, etc.) while other may not be observable (that agent's intentions, the man behind the curtain, etc.). As forces act upon the environment items, the settings with respect to those items change, resulting in a different state. Ultimately, the current state is the result of the initial settings of the environment plus all of the actions that altered those settings.

While tracking the initial state and every action taken provides a precise estimate of the current state, it represents a very long path. Estimating my current state in this manner would require examining every step I have taken, every class I have attended, and all of the other numerous factors throughout my life. This same path exists for a simple agent operating in an environment. Its current state is based on its initial position plus all of the actions it has taken, but trying to track all of the calculations and operations a computer makes for even a minute is a very large task.

Fortunately, that long train of calculations and operations can be reduced and still provide an accurate estimate of the current state. For instance, a college student trying to find his roommate does not need to retrace all of the steps the roommate has taken in life, the student only needs to remember that the roommate said he was hungry fifteen minutes ago to realize the roommate is at the cafeteria. Similarly, an

agent may know that another agent just completed a task in a given location and assume that agent is still in the vicinity of the task. This is an example of the Markov assumption which says that the current state only depends on a finite history of the previous states [20]. All of the multi-agent environments reviewed in the following section utilize the Markov assumption to eliminate the need to maintain a complete list of every previous environment state.

2.2 Multi-agent Environments

Single agent domains can be modeled as a Markov decision process (MDP) [4]. An MDP is defined by the tuple $\langle$S,A,T,R$\rangle$, where

- S is the set containing the finite number of environment states.
- A is the set of actions an agent can take.
- T is the transition model $S \times A \rightarrow S'$ that defines the probability that an agent's actions in an initial state s will change to state s'.
- R is the expected reward an agent receives from taking action a in state s and reaching state s'.

The MDP selects the course of action from a given state with the highest expected reward. At each time step, an agent in state s performs action a attempting to reach state s' which holds the highest expected reward r. Transition distribution t maintains the probability of going from s to s' by performing a.

An MDP requires that an agent is operating in a fully observable environment, meaning the current state of the agent is known. The agent is not capable of dealing with uncertainty about its current state or partial observability.

A partially observable Markov decision process (POMDP) [13] allows a single agent to cope with uncertainty about its current state while operating in a stochastic environment. A POMDP is defined by the tuple $\langle S, A, T, \Omega$,O$, R\rangle$, where

- S, A, T, and R are unchanged from the MDP.

- Ω is the finite set of observations an agent can make about its environment.
- $O : S$ x $A \rightarrow \prod(\Omega)$ is the observation function which returns the probability of making observation o given that an agent took action a to get to state s'.

The POMDP uses memory about previous actions and observations to determine states. An agent's state belief is a distribution over S. The state belief is defined by the equation:

$$b_i^t(s^t) = \beta O_i(o_i^t, s^t, a_i^{t-1}) \sum_{s^{t-1} \epsilon S} b_i^{t-1}(s^{t-1}) T_i(s^t, a_i^t, s^{t-1}) \tag{2.1}$$

where:

- $\mathrm{b}_i^t(s^t)$ is agent i's belief b at time t that the state is s.
- β is a normalizing constant.

The belief, b_i^t, in the current state being s^t encompasses the changes in the initial belief, b_i^{t-1}, as a result of taking action, a_i^{t-1}, at time, $t-1$, resulting in the current set of observations, o_i^t.

A decision policy describes an agent's behavior by mapping what action an agent takes in each state and is defined as $S \rightarrow A$.

The policy that includes the decision for every state is defined by the equation:

$$\pi_t^\infty(s) = \underset{a}{^{argmax}} \left[R(s,a) + \gamma \sum_{s' \epsilon S} T(s,a,s') U_{t-1}^\infty(s') \right] \tag{2.2}$$

where:

- π is the decision policy $S \rightarrow A$ specifying the action a to be taken in state s.
- γ is a decay factor.
- U is the expected future reward from state s.

Solving for the optimal policy requires finding the action for each state with the highest expected reward after an infinite number of decisions. While learning the optimal policy is undecidable for an infinite horizon, approximation techniques can be used to determine a reasonable policy.

Value iteration [13] finds the optimal policy to a horizon of t by iterating through all permutations of state-action transitions to determine the path with the highest expected reward. Instead of predefining t, the iteration continues until the expected reward difference between consecutive levels (t and $t-1$) is less than some small ϵ. Policy iteration [20] evaluates the utility of every state using the current policy and then calculates a new policy based on the maximum expected utility of the subsequent state. This process repeats until the current policy and the new policy converge.

The computational complexity of the decision policy can be further reduced using other techniques. Behavioral equivalence [16] is used to collapse large numbers of states into a manageable space. Particle filtering [8] uses particles to represent possible states and carries a subset of particles forward in time. Alpha-beta minimax [20], modified A^* search [23] and dynamic programming [12] can be used to quickly prune dominated branches of the search tree. Additionally, the problem domain dimensionality can be reduced using principal component analysis [19].

Given a decision policy, the an agent's expected reward for a finite horizon of t from state s is calculated inductively by:

$$U_{\pi,t}(s) = R(s, \pi_t(s)) + \gamma \sum_{s' \epsilon S} T(s, \pi_t(s), s') U_{\pi,t-1}(s') \tag{2.3}$$

where:

- π_t is a non-stationary decision policy that is executed for t time steps.

To find the optimal solution to the expected reward for a given state, all possible future states s' must be considered as well as the probability of their occurrence

$(T(s, \pi_t(s), s'))$ and their expected reward value. The future states s' are dependent on their future states s'' in the same manner.

The finite horizon assumes the decision policy changes with each time step because the agent is reaching the endgame. In a given state, an agent might make a greedy decision if it knows there are no long term consequences because this is its final decision. Conversely, the agent chooses to maximize future reward by selecting a different action from the same state earlier in its decision tree. Using an infinite horizon, the agent's decision policy does not change with time because the agent never approaches the endgame. The expected reward for an infinite horizon is given by the recursive equation:

$$U_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \epsilon S} T(s, \pi(s), s') U_\pi(s') \tag{2.4}$$

Unfortunately, the traditional POMDP can not handle a multi-agent environment, but it can be expanded. The decentralized-POMDP (DEC-POMDP) [3] is defined by the tuple $\langle I, S, A, T, R, \Omega, O, h \rangle$

where:

- I is the number of agents in the environment.
- h is a positive integer representing the finite horizon.

In the DEC-POMDP, the state transitions T and rewards R are dependent on the actions of all the agents in the environment. The POMDP is a special case of the DEC-POMDP with only 1 agent in the environment. Ultimately the single group reward for all of the agents works well in a cooperative environment.

While the expected reward calculations do not change from the POMDP (Equations 2.3 and 2.4), the policy function does change.

$$U_{\pi}^{T}(s) = \sum_{\langle \bar{o}_1, \bar{o}_2 \rangle} \sum_{q \epsilon S} \sum_{s' \epsilon S} \bar{T}_{\pi}(s, \bar{o}_1, \bar{o}_2, q) T(q, \pi_1(\bar{o}_1), \pi_2(\bar{o}_2), s') R(q, \pi_1(\bar{o}_1), \pi_2(\bar{o}_2), s') \tag{2.5}$$

In the DEC-POMDP, the policy function depends on the policies and states of all of the agents in the environment. Equation 2.5 is the policy function for a DEC-POMDP with two agents.

The I-POMDP [7] builds further on the framework expanding the concept of states to include the beliefs an agent has about other agents (agent states, intentions, and beliefs). An I-POMDP, consists of the tuple $\langle \mathrm{IS}_i, A, T_i, \Omega_i, O_i, R_i \rangle$ for each agent i within the environment, where

- IS_i is the set of interactive states $S \times \mathrm{M}_j$, with S being the set of environment states, and M_j is the set of models of agent j. Each model m_j consists of the pair $\langle \mathrm{f}_j, \mathrm{h}_j \rangle$ where f_j is a function that maps the possible histories of j's observations to its actions and h_j is one of the possible histories.
- A is the set $A_i \times A_j$ of joint actions of all agents.
- T_i is $S \times A \times S'$ which is the transition model that defines the probability that an agent's actions will change the state.
- Ω_i is the set of observations an agent can make.
- O_i is $S \times A \times \Omega_i$ which is the probability that agent taking action a in state s will make observations Ω.
- R_i is $IS_i \times A \rightarrow R$ which is the expected reward agent i receives from taking action a in states is.

Building off the POMDP belief update, Equation 2.1, the I-POMDP belief update is

$$b_i^t(is^t) = \beta \sum_{is^{t-1}:\hat{m}_j^{t-1}=\hat{\theta}_j^t} b_i^{t-1}(is^{t-1}) \sum_{a_j^{t-1}} Pr(a_j^{t-1}|\theta_j^{t-1}) O_i(s^t, a^{t-1}, o_i^t) \cdot T_i(s^{t-1}, a^{t-1}, s^t) \sum_{o_j^t} \tau_{\theta_j^t}(b_j^{t-1}, a_j^{t-1}, o_j^t, b_j^t) O_j(s^t, a^{t-1}, o_j^t) \quad (2.6)$$

where:

- $\sum_{is^{t-1}:\hat{m}_j^{t-1}=\hat{\theta}_j^t}$ is the summation over all is where agent j's frame is $\hat{\theta}_j^t$
- θ is an agent's type.
- b_j is agent j's belief elements of type θ_j.
- $Pr(a_j^{t-1}|\theta_j^{t-1})$ is the probability that agent j took action a at the last time step given its type θ_j.
- O_j is agent j's observation function.
- $\tau_{\theta_j^t}(b_j^{t-1}, a_j^{t-1}, o_j^t, b_j^t)$ is agent j's belief update.

The policy function for the I-POMDP is

$$\pi_t^\infty(is) = \underset{a}{^{argmax}} \left[R(is, a) + \gamma \sum_{is' \epsilon IS} T(is, a, is') U_{t-1}^\infty(is') \right] \quad (2.7)$$

The policy function solves for the maximum expected reward of each interactive state. The maximum expected reward for a finite horizon is

$$U(\theta_i) = \underset{a_i \epsilon A_i}{^{max}} \left\{ \sum_{is} ER_i(is_i, a_i) b_i(is) + \gamma \sum_{o_i \epsilon \Omega_i} Pr(o_i|a_i, b_i) \cdot U(\langle SE_{\theta_i}(b_i, a_i, o_i), \hat{\theta}_i \rangle) \right\} \quad (2.8)$$

where:

- $ER_i(is, a_i)$ is the expected reward for taking action a_i in interactive state is. $ER_i(is, a_i) = \sum_{aj} R_i(is, a_i, a_j) Pr(a_j|m_j)$.
- SE_{θ_i} is the state estimation, an abbreviation of the belief update.

The maximum expected reward for an infinite horizon is

$$OPT(\theta_i) = {}^{argmax}_{a_i \epsilon A_i} \Bigg\{ \sum_{is} ER_i(is_i, a_i) b_i(is) + \gamma \sum_{o_i \epsilon \Omega_i} Pr(o_i|a_i, b_i) \cdot U(\langle SE_{\theta_i}(b_i, a_i, o_i), \hat{\theta}_i \rangle) \Bigg\} \tag{2.9}$$

The I-POMDP represents a series of individual POMDPs. In addition to the computational complexity of the POMDP (multiplied by the number of agents in the environment), the I-POMDP has additional nesting within the belief model of the agents. Each agent possesses beliefs about the other agents in the environment, and these beliefs include the other agent's beliefs about all of the other agents. This nesting continues infinitely as the agent tries to reason about the other agent's models as depicted in Figure 2.1. The typical solution to this infinite nesting limits the trace to a strategy level, l, usually the first belief model of the other agents, using the Markov assumption [21]. At the 0-th level of nesting, the other agents actions are added to the noise of the T, O, and R functions. The first level beliefs become a probability distribution over S and the 0-th level models of the other agents. The first level beliefs are solved as a series of POMDPs of the 0-th level models. The individual POMDPs are solved using the policy function and reward function from the POMDP discussion. According to Seuken [21], an I-POMDP is PSPACE-hard for finite-time horizons, undecidable for an infinite horizon, and an optimal I-POMDP algorithm takes up to double exponential time.

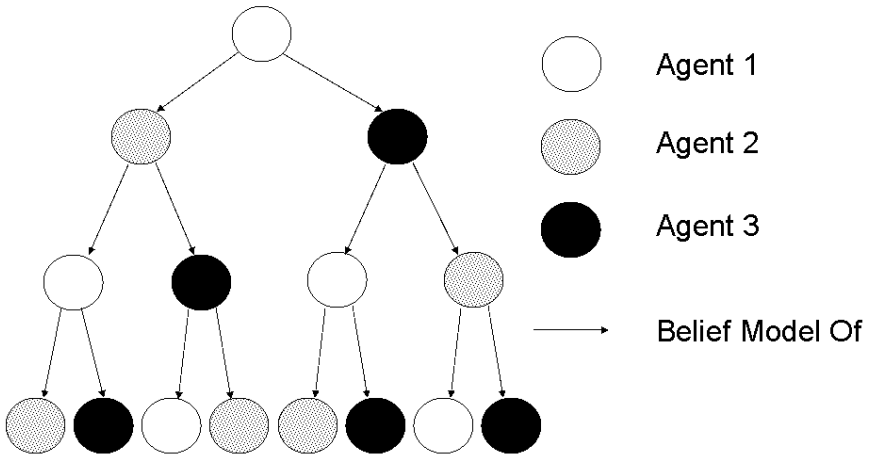

Figure 2.1: Recursive belief nesting of an agent's model of the other agents.

2.3 Trust

One of the underlying assumptions with many of the multi-agent environments is that the agents are cooperating to achieve the highest reward possible (DEC-POMDP is one example), but that is not always the case. Many problems require the agent domain to mimic the human world where adversaries are working against one another. The problem can be further compounded when an adversary acts like a genuine ally to gain trust that can be exploited later. If one component of an automated network defense suite is compromised, the other components still trust the compromised component. This trust allows the other components to be circumvented or corrupted themselves.

The main hurdle with a trust based environment is defining and learning the trust values for each agent. There are several proposed methods for generating the initial trust model.

- Experience-Based

  Experience based models [2] rely on past interactions. The outcomes of previous interactions form the agent's trust rating for future interactions. This type of model is useful in domains that allow repeated interactions with the same agents.

- Attitude-Based

  An extension of the experience-based model is a game theoretic model where an agent selects it willingness to sacrifice its own reward to benefit its opponents reward [6]. The agent learns its opponent's "attitude" through repeated interactions. The agent adjusts its own "attitude" based on whether it can benefit from helping its opponent. The combination of these "attitudes" plus an expected reciprocation threshold guides an agent's decision to cooperate.

- Reinforcement Learning

  One of the basic ways is to build a trust estimate is through reinforcement learning. Reinforcement learning allows an agent to generate its model of the world (in this case the trust rating of another agent) through interactions with the world [20]. As an agent interacts with its surroundings, it receives positive rewards reinforcing its current behavior or negative rewards causing it to change behavior. If agent a trusts agent b to perform a task and b fails to accomplish its mission, then a does not receive its expected reward. Agent a can determine that agent b is untrustworthy and refuse to cooperate with b in the future.

- Network

  A common approach builds a network of trusted agents [22, 24], also called reputation modeling. An agent polls its network to get recommendations about an unknown agent, and the agents in its network return their recommendations which are then combined. If one of polled agents does not have a recommendation about the unknown agent, it will poll its own trust network for recommendations. This method is useful in larger multi-agent environments where an agent is not constantly interacting with the same agent. The network approach allows agents to pass information back and forth, quickly propagating the outcomes of past interactions. This method does not work for domains with only a few agents because there is no network to build.

- Preliminary Interactions

A drawback to reinforcement learning is that agents may only get 1 opportunity to trust another agent. In the worst case, trusting a deceitful agent can lead to the destruction of the agent. A proposed solution to this problem is to allow initial interaction between agents prior to a decision about whether or not to cooperate [18]. The agents discuss their intentions and the responses given are classified into clusters representing the honesty makeup of the population. An agent decides whether to cooperate based on the classification of the other agent. The concept is designed to mimic the human ability to determine whether or not to trust someone based on an initial feeling, but the preliminary interaction may not always be feasible in a multi-agent environment.

- Trust Vectors

  The concept of differing levels of trust led to the creation of trust vectors [17]. Each level or parameter of trust is given a numeric value. The numeric values are stored in a single vector that is normalized to give a trust rating at a particular time. Trust vectors allow trust modeling to extend to multidimensional domains where an agent is trustworthy in some aspects and deceitful in others.

- Adaptive Trust

  Adaptive trust modeling [10] dynamically combines reputation-based models and experience-based models. Reputation systems suffer when reputations are inaccurate. Experience systems have difficulty forming initial trust ratings and suffer in environments that do not allow repeated interactions. Leveraging both models allows an agent to overcome the drawbacks of the individual models.

- Fuzzy Sets

  Trust ratings based on fuzzy sets [1] use a series of overlapping categories to determine the trust rating of an agent. An agent's trust rating is based on the aggregate of the probabilities that the agent belongs to each of the individual categories. Once again, a time decay function can be used to reduce the impact of less recent actions.

Once an agent knows whom he trusts, the agent simply chooses to work with the trusted agents and isolates the untrusted agents. The established trust levels must be updated as agents come and go within the environment. If a trusted agent leaves for a period of time and reemerges, its identity must be authenticated to mitigate the risk of interacting with an agent spoofing the trusted agent. To combat this problem, an agent can be the responsibility of the entity that placed the agent within the domain. The entity can encode information in the agent known only to the entity that other agents would use to authenticate the true identity of the agent [25].

Additionally, agents must continuously reevaluate their trust rankings of the other agents. Reevaluation is used to capture changing trust levels and can be applied to sneaky agents that cooperate when the stakes are low to build trust before betraying to achieve a large reward.

2.3.1 Trust Objectives. Karen Fullam, et al. developed a comprehensive list of objectives for trust-based research [11] to help guide research and aid in comparison testing. The objectives include building trust models that are adaptive, accurate, quickly converging, multidimensional, and efficient. Once these basic requirements are met, agents must be able to identify and isolate untrustworthy agents, evaluate the utility of an interaction, and ultimately decide whether and with whom to interact.

Several specific areas exist within trust research. Determining the best way to initialize a trust model is difficult, especially for experience-based systems. For any trust system, setting the threshold to revisit trusted or untrusted neighbors is a balance between returning too rarely which misses changes in trust and returning too often which wastes computation cycles. The time decay of past interactions tries to maintain past histories of betrayal while not making it impossible for an agent to redeem itself. This is critical for experience-based and trust vector models that rely heavily on past interactions. Reputation-based and similar models are influenced by the neighbor nodes that relay information to an agent. If the agent does not trust any of its neighbors, that agent becomes isolated from the rest of the environment. This

research focuses on the model representation used and the revisit rate of relearning agent trust levels.

2.4 Trust in Multi-agent Environments

The addition of trust into a system reduces the options for formulating the coordination problem. By definition, an MDP and a POMDP can not fully capture the domain because they are designed for single agents. An environment with trust modeling possesses cooperative agents working together and competitive agents attempting to achieve higher individual rewards. The different motives creates the need for individual reward functions that model these motives. The DEC-POMDP is not designed to handle the individual reward functions. That leaves the I-POMDP framework as a starting point for our trust environment. Within this framework, trust may be modeled in multiple dimensions where an agent can be trustworthy in some aspects, but prone to lying in others.

2.5 Summary

Agent models are often dictated by the complexity of the tasks the agents are designed to perform and the environment they operate in. Single agent models are typically defined by the amount of certainty the agent has about it environment. As agent certainty increases, model complexity can decrease. Multi-agent models tie multiple instantiations of the single agent model together allowing agents to cooperate with one another while still operating independently. Among the multi-agent models, the I-POMDP allows the largest amount of individual behavior within the framework. Each agent has its own reward function and attempts to determine the beliefs of every other agent within the environment.

The individual reward functions and belief models allow the I-POMDP to incorporate trust representations. Trust representations are used to determine whether an agent will cooperate or compete with another agent. The individual reward functions embody an agent's trust level. A competitive agent values actions that hinder the

other agents while a cooperative agent values actions that benefit the collective. The individual belief models are used to determine the trust level of another agent.

III. Trust-based I-POMDP

The basic goal of the multi-agent environments and trust models described in Chapter II is to create agents that make decisions based on their interpretation of the world around them. In the multi-agent environments, agents try to determine the current state of the world and select the most beneficial action based on a future expected reward. The trust models analyze the agents in the world to select the most beneficial cooperative action based on the probability of its success. The similarly aligned goals create a natural merger between the two mechanisms to create a complex multi-agent environment where agents reason about trust and the environment state to aid their decision making.

This complex multi-agent environment is the TI-POMDP framework described in this chapter. The TI-POMDP incorporates trust modeling to determine which agents are willing to cooperate with one another while using the decision making process of the I-POMDP [7]. The TI-POMDP mirrors the I-POMDP framework with additional trust components implemented as needed. The trust components directly impact several pieces of the traditional I-POMDP framework without altering the mechanisms used to analyze and reason about the multi-agent environment. The trust implementation is not constrained to a specific trust model (i.e. experience-based [2], reputation-based [22], or trust vector [17]) which allows the framework to be easily modified for a variety of domains.

3.1 Framework Development

The TI-POMDP is an extension of the I-POMDP [7]. The TI-POMDP maintains the basic components of the I-POMDP while incorporating trust modeling as a primary decision factor for the agents. In addition to the state belief model (an agent's estimate of the current environment), an agent maintains and updates a trust model (a rating of the trustworthiness of the other agents) for the environment. This trust model contains an agent's level of trust in the other agents. This level of trust

helps the agent decide whether or not to cooperate with another agent on a given task.

Modifying the I-POMDP framework (Section 2.2) requires identifying the components of the I-POMDP tuple, $\langle IS_i, A, T_i, \Omega_i, O_i, R_i \rangle$, that are directly affected by trust modeling. While almost every component is affected to some extent, most components only receive an increased set or function size.

In the I-POMDP, the interactive state (IS_i) is the cross product of the state(s) and the model (an estimate of the other agent's state belief based on their past actions) of the other agents in the environment. Adding the trust model to the I-POMDP combines an agent's state belief with its current trust model. An agent's current state has an additional focus on which agents it can cooperate with and which agents it should betray.

The set of actions (A) an agent can take does not directly change. The actions an agent takes can now result in cooperation, betrayal, or some other interaction with the other agents in the environment.

The transition function (T_i) becomes more complex as the agent's trust model is updated during state transitions. The transition function covers a larger range of inputs and outputs as the number of states increase.

The set of observations (Ω_i) increases as an agent must have observations that distinguish between cooperative and corruptive actions. In order to reason about the intentions and trustworthiness of another agent. The observation function (O_i) covers a larger range of outputs due to the increased observation set.

The reward function (R_i) is dependent on an agent's trust model which increases the size of the function. The size increase is due to the reward function handling both trustworthy and deceitful agents. A trustworthy agent values states that allow cooperation with other trusted agents while the untrustworthy agent values states that allow exploitation of the other agents [15]. As an agent's trust model shifts, its reward function must account for the shift. In a given state where all other agents

are trustworthy, agent a's decision is based on its expected reward and its expected reward is based on its reward function. If agent a is trustworthy, its reward function associates high reward values with cooperative tasks and low reward values with betraying tasks, but if agent a is untrustworthy, its reward function associates low reward values with cooperative tasks and high reward values with betraying tasks.

The trust model is an agent's current assessment of the level of trust it can place in the other agents. Each time the agent observes the actions of another, it adjusts its trust level of the other agent accordingly.

3.2 TI-POMDP Framework

The trust belief model used in the TI-POMDP is defined by τ. Agent i's trust belief model includes the true trust level of i and i's estimate of the trust level for every other agent j in the environment. In addition, i must also estimate every other agent j's trust level for every other agent j' in the environment. This includes agent j's trust level of agent i. If agents i, j, and k are all assigned to a task and agent i believes all three agents are trustworthy, agent i may still avoid cooperating on the task if it believes that agent j does not trust agent k. The reward function R examines τ_i to determine the expected reward for a given state. If agent j does not trust agent k according to τ_i, then agent i's expected reward for working with agents j and k decreases because agent i does not believe that agent j will work with agent k resulting in agent j not fully cooperating on the task.

Table 3.1 illustrates a simple binary trust model for Agent 1. Each column represents Agent 1's belief about a given agent's trust model. The Agent 1 column shows Agent 1's the actual trust value and its ratings of Agents 2 and 3. The ratings of Agents 2 and 3 are the primary influences on whether Agent 1 is willing to cooperate with the other agents. The Agent 2 column shows Agent 1's estimate of Agent 2's ratings. Agent 2's rating of itself is always identical to Agent 1's rating of Agent 2 since both ratings are based on Agent 1's beliefs. The Agent 3 column has the estimate of Agent 3's ratings.

Table 3.1: An agent's trust model τ of the other agents in the environment. The trust model depicted is a binary model.

Level of Trust in Agent	Agent		
	1	2	3
1	1	0	1
2	1	1	0
3	1	1	1

In this trust model, Agent 1 trusts Agents 2 and 3, but does not believe that Agent 2 trust it or that Agent 3 trusts Agent 2. Based on this trust model, Agent 1 does not believe that Agent 2 is willing to cooperate with it. In addition, if Agents 2 and 3 are trying to work together, Agent 1 believes that Agent 2 tries to cooperate but Agent 3 does not based on its rating.

While the binary trust system is illustrated here, other trust models can be used. Each rating in the table represents a complete trust model system for that rating. If trust vectors [17] are used as the trust model, each rating becomes an individual trust vector with Agent 1's estimates of which actions the other agents observed and how they analyzed those actions. If a reputation network [24] is the trust model, each rating is Agent 1's estimate of the information the other agents received from the network.

After adding trust to the I-POMDP framework, the TI-POMDP tuple remains $\langle IS_i, A, T_i, \Omega_i, O_i, R_i, \rangle$, where

- IS_i for agent i is the set of interactive states $S \times M_j$ where S is the set of environment states, and M_j is a model of agent j, $\forall j \neq i$. Each state s includes a trust belief model τ_i. Each model m_j consists of the tuple $\langle f_j, h_j, \tau_{i,j} \rangle$ where f_j is a function that maps the possible histories of j's observations and i's trust belief model of j to j's actions, h_j is one of the possible observation histories,

and $\tau_{i,j}$ is i's trust rating of j. Agent i uses m_j to predict agent j's actions. Agent i bases its action decision in part on the prediction of agent j's action.

- A is the set $A_i \times A_j$ of joint actions of all agents.
- T_i is $S \times A \times S'$ which is the transition model that defines the probability that an agent's actions will change the state. The change in state includes the change τ_i.
- Ω_i is the set of observations an agent can make.
- O_i is $S \times A \times \Omega_i$ which is the probability that agent taking action a in state s will make observations Ω.
- R_i is $IS_i \times A \to R$ which is the expected reward agent i receives from taking action a in state is.

Building on the I-POMDP belief update, Equation 2.6, the TI-POMDP belief update is

$$\begin{aligned} b_i^t(\tau_i, is^t) =& \beta \sum_{is^{t-1}:\hat{m}_j^{t-1}=\hat{\theta}_j^t} b_i^{t-1}(\tau_i, is^{t-1}) \sum_{a_j^{t-1}} Pr(a_j^{t-1}|\theta_j^{t-1}) O_i(s^t, a^{t-1}, o_i^t) \\ & \cdot T_i(s^{t-1}, a^{t-1}, s^t) \sum_{o_j^t} \tau_{\theta_j^t}(b_j^{t-1}, a_j^{t-1}, o_j^t, b_j^t) O_j(s^t, a^{t-1}, o_j^t) \end{aligned} \tag{3.1}$$

where:

- τ_i is agent i's trust model.
- $b_i^t(\tau_i, is^t)$ is agent i's belief b at time t that the interactive state is is.
- β is a normalizing constant.
- $\sum_{is^{t-1}:\hat{m}_j^{t-1}=\hat{\theta}_j^t}$ is the summation over all is where agent j's frame is $\hat{\theta}_j^t$
- θ is an agent's type.
- b_j is agent j's belief elements of type θ_j.

- $Pr(a_j^{t-1}|\theta_j^{t-1})$ is the probability that agent j took action a at the last time step given its type θ_j.
- O_j is agent j's observation function.
- $\tau_{\theta_j^t}(b_j^{t-1}, a_j^{t-1}, o_j^t, b_j^t)$ is agent j's belief update.

The state belief update still requires a decision policy as outlined in the POMDP discussion in Section 2.2. The same approximation techniques used for solving other POMDP instances (Value Iteration [13], Behavioral Equivalence [16], etc.) are used to reduce the search space and solve the TI-POMDP.

While an agent does not directly alter another agent's belief model, an agent's actions affect the current state which does change the other agent's current observations. The other agent attempts to reconcile its current observations with its expected observations by adjusting its belief model including the models of all of the agents in the environment. Adjusting the models of all of the agents includes adjusting the estimated trust levels of each of the agents.

Each belief state has an associated maximum expected reward as defined by

$$U(\theta_i) = \max_{a_i \epsilon A_i} \Bigg\{ \sum_{is} ER_i(\tau_i, is_i, a_i) b_i(\tau_i, is) + \gamma \sum_{o_i \epsilon \Omega_i} Pr(o_i|a_i, b_i) \cdot U(\langle SE_{\theta_i}(b_i, a_i, o_i), \hat{\theta}_i \rangle) \Bigg\} \tag{3.2}$$

where:

- $U(\theta_i)$ is the expected future reward for an agent of type θ.
- γ is a decay factor.
- $ER_i(\tau_i, is_i, a_i)$ is the expected reward for taking action a_i in interactive state is_i with trust model τ_i. $ER_i(\tau_i, is_i, a_i) = \sum_{aj} R_i(\tau_i, is, a_i, a_j) Pr(a_j|m_j)$.
- SE_{θ_i} is the state estimation, an abbreviation of the belief update.

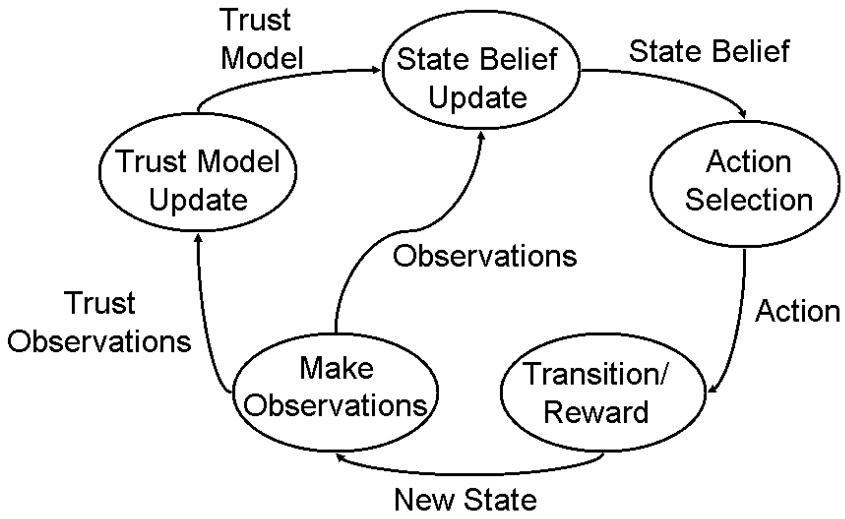

Figure 3.1: The event cycle for the TI-POMDP.

3.3 Model Updates

Each time step, an agent attempts to determine the optimal action to take given the true state which is based on the true trust model of the environment. In a fully observable world, the agent would know where it is at and who it could trust. In a partially observable world, the agent must estimate these values.

At a given time step, an agent calculates the expected reward for each of its potential actions from each of its possible states. The agent selects the action with the highest estimated reward. After taking the selected action, the agent observes the changes in the environment and the actions of the other agents when possible. If the actions of the other agents provide indications of their trust level, the agent updates its trust model accordingly. Finally, after observing the changed environment, the agent updates its state belief model based on its current trust model before attempting to decide on its next action. Figure 3.1 illustrates the cycle of events during the TI-POMDP execution.

The trust model, τ, is a component of the state belief and is updated based on the observations prior to the state belief update. The agent updates its trust model and then updates its state belief distribution.

In the trust model update, if agent a observes agent b commit an untrustworthy act, agent a reduces its trust rating of agent b based on the rules of the trust modeling representation used (ie. trust vectors [17] or reputation based [24]). Using the binary trust model, agent a reduces agent b's trust rating from 1 to 0. Additionally, if agent a believes agent c also observed agent b's action, agent a lowers its estimate of agent c's trust rating of b from 1 to 0. The overall effect is that agent a places less trust in agent b and agent a believes that agent c also lowers its trust in agent b. Agent a uses this trust model in future interactions to decide whether to interact with agent b and to estimate how agent c interacts with agent b. The new trust model and the observations about the new state are required for the state belief update which then restarts the process.

The primary reason to separate the trust model update is to increase the flexibility of the trust model. Handling the trust model update separately allows the underlying trust model to change without impacting the rest of the TI-POMDP framework. The trust model update must provide the agent's current trust ratings of itself, its ratings of the other agents, and the other agent's ratings of all of the agents. A vector trust model [17], a reputation based trust model [22, 24], a multidimensional trust model [18], or another trust model can be implemented to return the appropriate ratings as needed. This allows model selection based on applicability to the domain.

The ability to use different trust models creates the problem of selecting the appropriate model for a given domain. Comparison testing can determine which model performs best in a specific domain. A more general solution is to implement separate models in parallel and use a decision process to dynamically choose which model to use at a given time [26].

The second reason to update the trust model separately from the state belief update is to reduce the combinatorics of the state belief update. Including the trust model as a complete component of the state multiplies the number of states by the total number of possible trust models an agent can have. In a basic two agent envi-

ronment where agents are either completely trustworthy or completely untrustworthy, that state space is multiplied by an agent's potential ratings of itself, potential ratings of the other agents, and potential ratings of the other agent's ratings of every other agent. In this case, the state space is needlessly multiplied by a factor of eight. The agent knows its true trust rating and the other trust ratings are based on prior knowledge and experience. Instead of trying to estimate those values, an agent can use its knowledge to focus on making the best possible decision at the current time.

The final reason to separate the trust model update is to eliminate fluctuations in an agent's trust model that can lead to a breakdown of trust within the system. If an agent's trust model is a component of its state belief probability distribution, the agent can become extremely unpredictable or uncooperative. The purpose of the trust model is to help the agent choose the most beneficial action for the given state. If the trust model is a component of the state belief, the agent must find the most beneficial action for each possible trust state, calculate the expected reward for each action and then select the action that leads to the highest probable reward. For instance, a trustworthy agent believes with 0.9 probability that the other agent is trustworthy in a binary trust domain where successfully cooperating yields a reward of 10, working alone has an associated reward of 1, and the reward for being betrayed is -100. The agents expected reward for cooperating is $0.9*10+0.1*-100 = -1$ while the expected reward for working alone is 1. Given this situation, the agent always chooses to work alone because of all the potential states, this one has the highest expected reward.

The state belief update (Equation 3.1) requires the agent incorporate its current observations into its previous state belief in an effort to determine its current state. The agent calculates the likelihood of making its current observations in each of the possible states it may have reached given the distribution over the prior state(s) and the action(s) taken. The previous state belief is then updated based on the observation likelihoods for each state. Once a group of agents updates their state beliefs, they can select and execute their next action. The action execution causes a state transition and the agents receive rewards based on their new state.

3.4 Impact of Trust

The addition of trust modeling into the I-POMDP framework results in an increase in the overall complexity of the framework. The I-POMDP is already N-EXP complete [21] without the trust model. Adding the trust model multiplies the state space by the trust model space. The size of the trust model space is based on the number of unique trust levels an agent can have and the number of agents in the domain. A single agent must rate every agent in the environment including itself (n ratings). It must also account for the other agent's ($n-1$ agents) ratings of every other agent ($n-1$ agents). Agent i does not have to account for agent j's rating of agent j since it would already match agent i's rating of agent j. In all, a single agent's trust model contains $n + (n-1)^2$ ratings. Since each individual rating can take one of the m possible trust levels, there are a total of $m^{n+(n-1)^2}$ possible permutations of an agent's trust model.

This complexity increase is one reason for handling the trust model updates separately from the state belief updates as described in Section 3.3. Instead of computing the probability distribution for the number of potential states multiplied $m^{n+(n-1)^2}$, we can run a single linear update to the trust model and then compute the probability distribution for just the number of states.

3.5 Summary

The TI-POMDP framework is introduced, creating a multi-agent environment where an agent's actions are influenced by its perception of the trustworthiness of the other agents. The framework is an extension of an I-POMDP with the same basic components (State, Action, Transition, Observations, and Rewards) plus the trust model. The TI-POMDP updates the trust models based on current observations and then updates the state beliefs based on the trust models and the observations. Although the components of the TI-POMDP require exponential time to solve optimally, the framework provides the benefit of incorporating another agent's intentions

into an agent's decision model. This opens a wider range of agent interactions giving agents the ability to perform more complex tasks.

IV. TI-POMDP Testing

The TI-POMDP framework described in Chapter III defines a multi-agent environment that models agent interactions based on trust. This chapter describes the simulation and the experiments used for testing the TI-POMDP. The simulation represents a base defense scenario with agents cooperating to destroy attackers. While a full TI-POMDP framework is used to model the agents in the environment, several assumptions are made to reduce computational complexity of the agents and limit testing to just the TI-POMDP reasoning component. The testing on the simulation includes an analysis to identify factors affecting algorithm output, algorithm performance testing that measures total execution time, and comparison testing with another trust model.

4.1 Scenario

The TI-POMDP framework can be applied to a wide range of multi-agent problems. Agents can use the framework to decide whether to infiltrate a building based on their level of trust in the reconnaissance agents relaying guard locations. An automated call center decides to reroute its call load to an idle center that it trusts to handle the calls. A robotic lawn mower may use the framework to decide when to help another robot mow its area based on whether its trusts the other robot to reciprocate. The TI-POMDP is applicable in environments where task completion can benefit from agent cooperation.

The TI-POMDP Simulation Defender focuses on a group of agents working together to defend an installation from attackers. When an attacker is identified, a subset of the agents are tasked to destroy the attacker. Each agent must decide whether to cooperate with the other tasked agents based on its individual goals (maximize reward) and the amount of trust it has in the other agents. The defense scenario is selected because it provides an environment capable of utilizing multiple agent types, a wide range of actions, and a variety of observation and reward functions. While

these aspects only have a few settings currently, future work can utilize this large domain.

After each task is completed, the agents try to identify the motives and allegiances of the other agents to help refine their trust models of the other agents. The refined trust models are used in future decisions.

4.2 Implementation

The framework can be implemented on robotic hardware or simulation software and is not language dependent. A Java-based simulation is used to allow portability between systems. The simulation is a visual depiction of the agents operating in the environment. The environment consists of a simple building with two doors as shown in Figure 4.1.

Figure 4.2 depicts the major components of the simulation. The Environment Controller is an "Eye in the Sky" overseeing the creation and execution of the simulation, but not actively visible in the environment. Attackers and agents move through the environment. A TI-POMDP is used to model the agents, govern their action decisions, and distribute rewards. The primary functions and responsibilities of each component are as follows:

4.2.1 Environment Controller. An Environment Controller initially creates and randomly distributes a set number of Agents in the space. During the simulation, the Environment Controller creates and randomly distributes a set number of Attackers along the perimeter of the simulation space. The number of active Attackers at any given time is limited by a threshold set prior to running the simulation. When the number of Attackers reaches the threshold limit, the Environment Controller waits until an Attacker is destroyed before it creates a new Attacker. When it creates an Attacker, the Environment Controller creates a task and assigns the task to a random group of agents. This allows the simulation to focus on the interactions between the Agents, not the sensor capabilities of the Agents and the task distribution process.

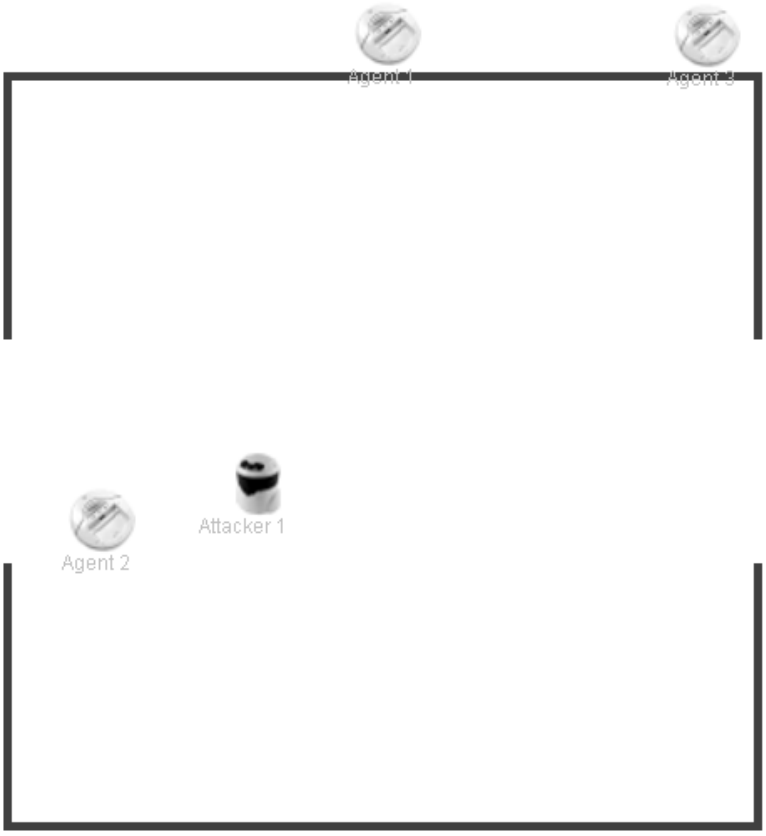

Figure 4.1: An image of the Defender Simulation with three Agents and one Attacker.

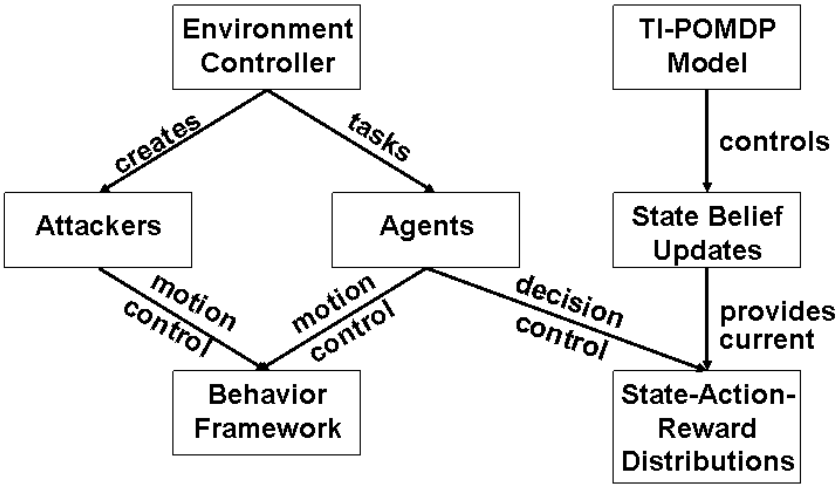

Figure 4.2: The major components in the trust simulation.

This also eliminates a potential trust exploitation where an Agent assigns false tasks to trustworthy Agents. The number of Agents assigned to each Attacker is random. Agents without prior task commitments are assigned first. If necessary, Agents with prior task commitments are assigned. Agents complete the tasks in the order they arrive to eliminate starvation.

The Environment Controller maintains the current reward level for the active tasks. The reward level based on whether previous actions were successfully completed via cooperation. The number of reward levels is equal to the number of Agent trust levels (set prior to simulation) minus one. If the simulation has three trust levels, then each action can have either level 1 or level 2 rewards. If Agents successfully cooperate with a level 1 reward, the next action has a level 2 reward. Higher reward levels do not change the number of assigned Agents.

The Environment Controller does not maintain trust ratings on the Agents, assign Agents based on their past performance, or attempt to maximize expected utility of the task. This forces the individual Agents to track and reason about the trustworthiness of the other Agents in the environment. If the Environment Controller tracked the trust ratings of the Agents, it would not assign untrustworthy agents to tasks, resulting in their isolation. While this action is desired in most scenarios, it reduces the simulation's testing ability of the TI-POMDP framework. When known untrustworthy Agents are assigned to a task, the other Agents tailor their actions to isolate the untrustworthy Agents and mitigate the damage they cause.

4.2.2 Attackers. Attackers are created at the edge of the environment and attempt to move toward the center. Since Attackers are not modeled by the TI-POMDP, they do not receive rewards for reaching the center. The Attackers are enemies with an initial strength of 100. Once an Attacker's strength is depleted it is rendered useless, removed from the environment, and the task of the Agent's assigned to defeat the Attacker is complete.

4.2.3 Agents. Agents have two mutually exclusive tasks, patrolling the environment and defeating Attackers. The patrolling behavior consists of randomly wandering the environment. Once assigned to a defense task, Agents engage and destroy the Attacker before returning to their patrol duties.

The motion of individual Agents and Attackers is controlled by a behavior-based architecture [5]. An Agent's behavior set consists of random walking and going to a target. The specific behavior is determined by the task assigned to the Agent. All tasks include the wall following and obstacle avoidance behaviors to help maneuver through the environment.

Only Agents assigned to a particular task are able to affect that task. If an unassigned Agent is in the vicinity of the Attacker, it will not engage the Attacker. This focuses the trust analysis just on the team of Agents assigned to the task. The Agents do not have to worry about an outside Agent hindering their ability to complete the task. Unassigned Agents may observe the actions of assigned Agents and update their trust model accordingly.

An Agent decreases the Attacker's strength by 25 points every time it chooses to attack the Attacker. The attack power of 25 points is used to ensure groups of one or two Agents must engage the Attacker multiple times since it has a strength of 100. A larger group of Agents (four or more) can eliminate the Attacker after a single engagement. Each action is a single attack by each of Agents assigned to the task. When an Agent decides to betray another Agent, it adds 15 points to the Attacker's strength. The addition to the Attacker's strength prolongs the engagement but does not completely counteract another Agent's action.

Depending on the number of Agents involved and their actions, each engagement has a unique duration. In the case where two cooperating Agents engage a single attacker, each Agent depletes 25 points from the Attacker. After the two Agents engage the Attacker a second time, the Attacker is eliminated with each Agent responsible for 50 points versus a single Agent taking 4 rounds and 100 points. If a cooperating

Agent and a betraying Agent engage an Attacker, the cooperating Agent depletes 25 points from the Attacker, and the betraying Agent adds 15 points for a net change of −10. Two betraying Agents add 30 to the Attackers strength.

4.2.4 TI-POMDP. The TI-POMDP framework is updated when a task is first identified and Agents are assigned to it. Each Agent uses the decision policy to decide which action to take. After action execution, the TI-POMDP framework is updated based on the observations. Figure 3.1 illustrates the TI-POMDP decision cycle. The TI-POMDP consists of the interactive states, actions, transition function, observation function, and reward function.

The domain's interactive state consists of the Agent's trust model,the task level, and which Agents are assigned to a task. This is the smallest possible state for this domain (as opposed to including Agent locations and any other environmental factor). This size reduction allows the simulation and testing to focus on the uncertainty of the trust model rather than the uncertainty created by the environment.

The Agent's action set includes "cooperating," "working alone," "betraying," "concealing," or "redeeming" actions. Their choice of action depends on their trust model. An untrustworthy Agent chooses to "betray" or "conceal" while a trustworthy Agent chooses to "cooperate," "work alone," or "redeem."

- Trustworthy Agents choosing to "cooperate" diminish an Attacker's strength by 25 points. "Cooperating" Agents assume that the other assigned Agents will diminish an Attacker's strength by their share.
- Trustworthy Agents choosing to "work alone" also diminish an Attacker's strength by 25 points. Agents "working alone" assume that the other assigned Agents will not diminish an Attacker's strength and may try to increase the Attacker's strength. They expect the engagement to last longer.
- Untrustworthy Agents choosing to "betray" always add 15 points to the Attacker's strength.

- Untrustworthy Agents choosing to "conceal" always diminish an Attacker's strength by 0.1 points. This action allows a "concealing" Agent to act like it is helping so unassigned Agents cannot determine an accurate trust rating. The 0.1 decrease is used to prevent deadlock if a group of "concealing" Agents are assigned to an Attacker since they eventually reduce the Attacker's strength to 0.
- Trustworthy Agents choosing to "redeem" have the same effect as cooperating Agents, but are making a conscientious decision to work with Agents that do not trust them. This exposes the Agent to potential betrayal as the Agent selects a course of action that has a lower immediate expected reward (versus working alone). "Redeeming" Agents are formerly untrustworthy Agents trying to reestablish their reputation among the other Agents.

The transition function maps one state to the next based on the Agent's actions. The transition function first updates the individual Agent trust models based on the their actions. At this point, each Agent has a probability of being corrupted or redeemed based on their actual trust level. The probability of an Agent's trust level changing is governed by the corruption and redemption rates given at the start of the simulation. The rates do not change during the simulation execution. Both rates can range from 0.0 (never corrupted or redeemed) to 1.0 (always corrupted or redeemed) and the two rates can be adjusted independently of one another. If the corruption rate is 0.1 and the redemption rate is 0.3 a trustworthy Agent has a 10 percent chance of becoming more untrustworthy while an untrustworthy Agent has a 30 percent chance of becoming more trustworthy. If an Agent is somewhere between trustworthy and untrustworthy, there is a 0.05 probability $(0.5 * 0.1)$ of becoming less trustworthy and a 0.15 probability $(0.5 * 0.3)$ of becoming more trustworthy. An Agent knows when its trust rating changes, but the other Agents are not aware of the change. The final step of the transition function adds new tasks, removes completed tasks, and sets the reward level for the next action.

The observation function is a probabilistic model of what actions a given Agent sees within the environment and is tied to the actions Agents take on a task. Agents assigned to a task are guaranteed to observe the actions of the other Agents assigned to the same task. Unassigned Agents may observe "cooperating," "betraying," and "redeeming" actions according to a set probability (0.5), but they can not observe the difference between "working alone" and "concealing" actions. Unassigned Agent observations are based on probability instead of location with respect to the task so the unassigned Agents are not rewarded (by receiving better observations) for neglecting their patrol duties.

The reward function is a distribution based on the collective actions of all Agents assigned to a specific task. Each Agent receives its reward based on its individual action with respect to the actions of the team that assigned to the task. The reward function used in the simulation is based on the Tiger Game reward function [7, 13]. The reward function is designed to value successful "cooperation" and "betrayal", penalize unsuccessful "cooperation" and "betrayal", and allow the safe options of "work alone" and "conceal."

- Agents that "work alone" or "conceal" always receive a reward of -1 regardless of the other Agent's action choices.
- If all Agents "cooperate," "work alone," or "redeem," the reward for the Agents that "cooperate" and "redeem" is $10 * t^2$, where t is the current reward level as set by the Environment Controller.
- If all Agents "cooperate" or "redeem" except for a single "betray" Agent, the "betray" Agent receives a reward of $10 * t^2$ while the other Agents receive a reward of -100.
- If multiple Agents "betray," all "cooperate," "betray," and "redeem" Agents receive a reward of -100. The "betray" Agents receive the negative reward because they betrayed each other.

- If a "betray" Agent is not paired with at least one "cooperate" Agent, the "betray" Agent receives a reward of -100.

The reward level effects the potential reward of a successful cooperation or betrayal on the task. An Agent's trust model is composed of the highest reward level that it believes every other Agent is willing to "cooperate" at. If Agent a's trust model indicates that Agent b is trustworthy at a reward level of 3, then Agent a trusts Agent b when the reward level is 3 or less and does not trust Agent b at higher levels.

Figure 4.3 shows the changes in trusted reward levels as Agents cooperate and compete with each other. Agent 2's betrayal level limits the amount of cooperation between the Agents and indirectly causes Agent 1's trust level of Agent 2 to change. Agent 2's betrayal level changes due to random corruption or redemption caused by the environment. Initially, Agent 1 trusts Agent 2 up to level 10, but Agent 2 is actually only trustworthy up to level 6. Once Agent 2 "betrays" Agent 1 (Action 6), Agent 1's trust level changes to level 6. Agent 1 is unaware that Agent 2's betrayal level changed after Action 6, allowing the "betrayal" at Action 9 which results in Agent 1 lowering its trust level of Agent 2 to level 3. Since Agent 1 no longer trusts Agent 2, it "works alone" (Action 12) the next time it encounters an action with a reward level of 3 which resets the reward level to 0. Agent 2 "redeems" itself (Action 15) after its betrayal level changed. Agent 1 revises its trust level in response to the redemption.

Assuming the Agents only look at the next action, their decision process focuses on maximizing the reward for the next action. To decide whether to cooperate, each Agent reviews its trust model of the other Agents assigned to the task as well as it own trust rating. If the Agent is trustworthy at the current reward level and trusts the other Agents at the current reward level, it chooses to "cooperate." If a trustworthy Agent does not trust another assigned Agent or does not believe that another Agent trusts one of the assigned Agents, it chooses to "work alone." An untrustworthy Agent attempts to "betray" the other Agents if it believes they are all trustworthy

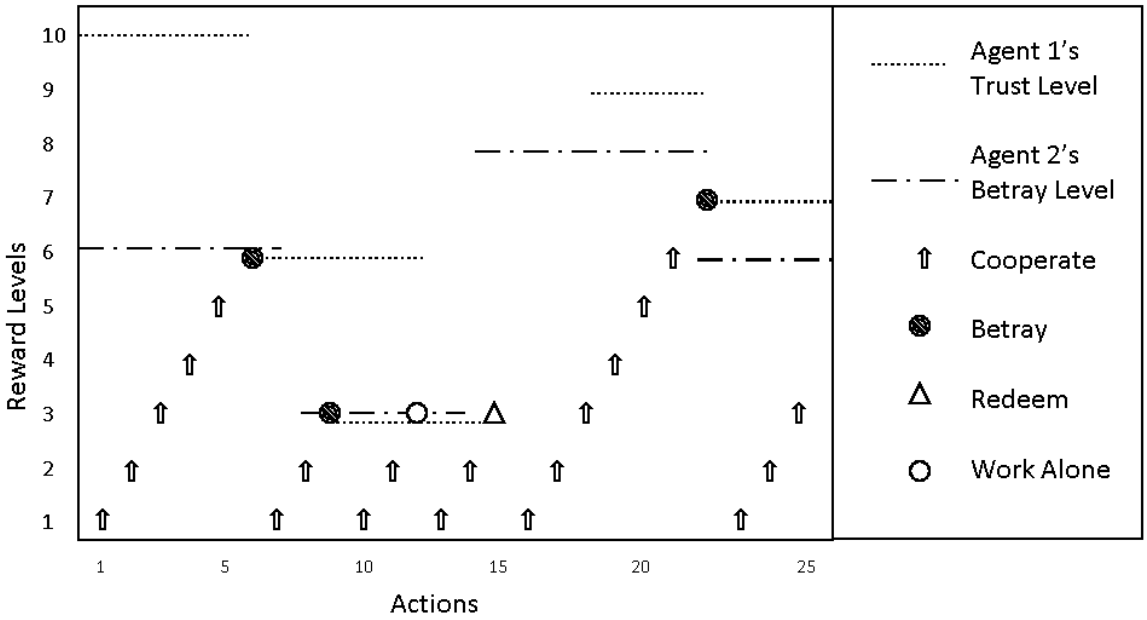

Figure 4.3: Trust interactions between agents.

and that they trust it. Once the Agent "betrays" (Actions 6, 9, and 22), the other Agents update their trust models by reducing their level of trust in the "betraying" Agent. Otherwise, an untrustworthy Agent will "conceal" its actions. If the Agent chooses to "conceal" its actions, it does not help with the task, but it does not hinder it either. The "conceal" action is not shown in this figure because only two Agents are in the domain which eliminates the utility of the action. Agents can benefit from "concealing" in larger domains because the other Agents can not differentiate between "conceal" and "working alone" allowing the untrustworthy Agent to potentially betray in the future.

There is an additional case where a trustworthy Agent does not believe the other Agents trust it. This Agent will choose to "redeem" itself (Action 15) and cooperate on the task, knowing that it is likely to sustain a penalty since it is doubtful that the other Agents are planning on cooperating with it. This sacrificial action "redeems" the Agent, causing the other Agents to update their trust models by increasing their level of trust in the "redeemed" Agent and deciding to "cooperate" with it on future tasks.

After the Agents act, all Agents within the environment update their individual trust models based on their observations of the task. The Agents directly involved with the task are guaranteed to observe the actions (other than the difference between "work alone" and "conceal") of the other task participants. Model updates are more complex for Agents not assigned to the task. If an unassigned Agent does not observe the task at all, it can not determine which Agent took what action. If an unassigned Agent does observe the task, it will know if another Agent "cooperates," "betrays," or "redeems" and updates its model accordingly. Assigned Agents update their trust models of the unassigned Agents based on their probability of observing the task. This can lead to situations where one Agent incorrectly believes another Agent observed a betrayal which skews the first Agent's future expectation.

4.2.5 Decision Policy Calculation. Prior to execution, the TI-POMDP requires a decision policy as defined in Section 2.2. The decision policy defines what action to take in a given state. Since decision policies are domain specific, solving the optimal or approximately optimal policy for a given domain is ideal. The decision policy for the Defender simulation is discovered through policy iteration [20] with the aid of behavioral equivalence [16] and alpah-beta minimax pruning [20].

The first step in the decision policy discovery is using behavioral equivalence to reduce the number of different states. Behavioral equivalence maps similar states onto one another. Rather than finding a decision for every one of these states, a single decision can be used for all of the similar states. The simplest method of behavioral equivalence in the Defender simulation is to collapse the states into eight states. The eight states are as follows:

- Trustworthy Agent that believes the other assigned Agents are trustworthy and they trust each other.
- Trustworthy Agent that believes it is not trusted by at least one other assigned Agent.

- Trustworthy Agent that believes the other assigned Agents are trustworthy but do not trust each other.
- Trustworthy Agent that does not trust at least one other assigned Agent.
- Untrustworthy Agent that believes the other assigned Agents are trustworthy and they trust each other.
- Untrustworthy Agent that believes it is not trusted by at least one other assigned Agent.
- Untrustworthy Agent that believes the other assigned Agents are trustworthy but do not trust each other.
- Untrustworthy Agent that does not trust at least one other assigned Agent.

These eight states are multiplied by the number of reward levels since the decision policy may be different depending on the reward level. In a domain with two reward levels, every state fits into at least one of the 16 equivalence states. The equivalence states are listed in order of precedence. As an example, a trustworthy Agent may believe that it is not trusted by another Agent (equivalence state 2) and not trust at least one other assigned Agent (equivalence state 4). In this case the state is grouped into equivalence state 2.

Once the equivalence states are determined the policy iteration begins. The policy iteration conducts a series of runs of the simulation starting from one of the equivalence states. Each run consists of five action decisions with the cumulative reward after the fifth decision tracked. A five percent decay rate is used in the future reward calculation.

Initially, each equivalence state is run twenty times for each of the allowable actions for that state. If an untrustworthy Agent is not allowed to take a specific action (an untrustworthy Agent cannot "cooperate," "work alone," or "redeem"), the equivalence states with an untrustworthy Agent do not select that action. The other Agents in the environment select a random action from the allowable actions in the

current state. After the state transition and belief update, the simulation conducts four additional decisions and updates. The additional decisions are all random.

After twenty runs of the same initial decision, the average cumulative reward is stored as the expected reward for that initial decision. Once all of the possible initial decisions are run for an equivalence state, the initial decision with the highest expected reward is stored as the desired decision for that state. The initial decision policy consists the desired decisions for every equivalence state.

The process is repeated using the initial decision policy. For each equivalence state twenty runs are conducted using the initial decision policy and ten runs are conducted for each allowable action from the equivalence state with a random decision policy for the four additional decisions. An equivalence state with two possible actions requires 40 runs during this step (20 with the decision policy, 10 with the first action and a randomized policy and 10 with the second action and randomized policy). The runs with the randomized policy are used to explore the reward space. To reduce the required computation, alpha-beta minimax is used to prune dominated runs that use the randomized policy. If a run using the randomized policy cannot achieve the expected reward of the decision policy after 2, 3, or 4 decisions the run is stopped instead of continuing to a suboptimal end. The maximum possible reward (the highest reward it could achieve by successfully cooperating for the remainder of the run) is used in the average expected reward calculation.

At the end of the runs, the initial decision (either from the decision policy or the randomized runs) with the highest expected reward becomes the desired decision for the next decision policy. If the desired decision for an equivalence state does not change from one iteration to the next, the number of additional runs using the randomized policy is reduced by one half during the next iteration of that equivalence state. Once the number of additional runs using the randomized policy is less than 2, the decision for that equivalence state is complete. This decision becomes the decision policy for that equivalence state.

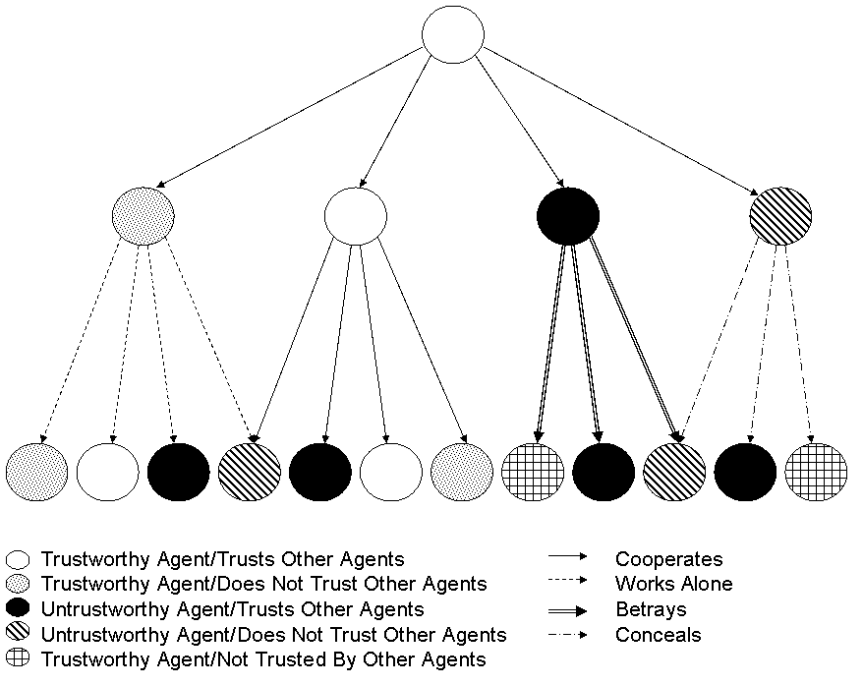

Figure 4.4: An Agent's decision transitions.

Figure 4.4 shows the potential progression of an Agent's trust model (circles) and actions (arrows) through a series of time steps. The actions also serve as transitions between trust models which are probabilistic as outlined in Table 4.1. The probabilities are based on the corruption/redemption rate of the environment and the progression of Agent actions. In an environment with a corruption/redemption rate of 0.1, a Trustworthy/Trusting Agent has a 10 percent chance of becoming untrustworthy (transitioning to model 3 or 4) and a 90 percent chance of remaining trustworthy (transitioning to model 1 or 2). The probability of actually transitioning to 1 versus transitioning to 2 is based on the Agent actions. In an environment with a low corruption/redemption rate, the Agents are more likely to remaining trustworthy and trusting of others (1 to 1 transition). As the corruption/redemption rate increases, the Agents are more likely to not trust each other (1 to 2 transition). A 3 to 3 transition is always unlikely because once an Agent "betrays," it usually can not benefit from repeating that action.

Table 4.1: Probability of trust model transitions based on corruption/redemption rate. Model 1 is Trustworthy/Trusting. Model 2 is Trustworthy/Not Trusting. Model 3 is Untrustworthy/Trusting. Model 4 is Untrustworthy/Not Trusting. Model 5 is Trustworthy/Not Trusted.

Transitions	Corruption/ Redemption Rate		
	0.1	0.3	0.5
1 to 1	0.72	0.28	0.05
1 to 2	0.18	0.42	0.45
1 to 3	0.08	0.12	0.05
1 to 4	0.02	0.18	0.45
2 to 1	0.036	0.042	0.045
2 to 2	0.864	0.658	0.455
2 to 3	0.004	0.018	0.045
2 to 4	0.096	0.282	0.455
3 to 3	0.009	0.007	0.005
3 to 4	0.891	0.693	0.495
3 to 5	0.10	0.30	0.50
4 to 3	0.009	0.042	0.045
4 to 4	0.891	0.658	0.455
4 to 5	0.10	0.30	0.50

The Agent is initially trustworthy and trusts the other Agents so it decides to "cooperate." Depending on the results of its action, the Agent can reach one of four new trust models. If the Agent was "betrayed" by another during the last task, the Agent remains trustworthy, but no longer trusts at least one other Agent. The Agent transitions to the leftmost node of the second row and chooses to "work alone" on the next task involving the untrusted Agents. This transition occurs with probability of 0.18, 0.18, and 0.45 in environments with corruption rates of 0.1, 0.3, and 0.5 respectively. If the previous task was successful, the Agent's trust model stays the same which leads the Agent to "cooperate" on future tasks. This trust model is

represented by the left-center node of the second row. This transition occurs with probability of 0.72, 0.12, and 0.05 in environments with corruption rates of 0.1, 0.3, and 0.5 respectively.

There is also a probability that the Agent becomes corrupted after an action. If the Agent was "betrayed" by another during the last task and the Agent is corrupted, the Agent transitions to being untrustworthy and does not trust at least one other Agent (rightmost node of the second row). The Agent's action choice for this trust model is to "conceal." This transition occurs with probability of 0.02, 0.42, and 0.45 in environments with corruption rates of 0.1, 0.3, and 0.5 respectively. If the last task was successful and the Agent is corrupted, the Agent transitions to being untrustworthy while still trusting the other Agents (right-center node of the second row). The Agent chooses to "betray" for this trust model. This transition occurs with probability of 0.08, 0.28, and 0.05 in environments with corruption rates of 0.1, 0.3, and 0.5 respectively.

These same transitions occur from each of the four trust models with one exception. If an untrustworthy Agent (either trusting of others or not) is redeemed, that Agent transitions to a model where it is trustworthy but not trusted by the other Agents. This model resembles the initial model where the Agent is trustworthy, but the Agent chooses to "redeem" itself regardless of whether it trusts the other Agents in the environment. The probability of transitioning to this trust model is equal to the redemption rate of the environment.

In the simulation Agents look further ahead than just the next action and attempt to maximize their expected reward for a certain number of actions into the future. The simulation explores five actions ahead to limit the search space and reduce the time required to create the model. This look ahead can cause an untrustworthy Agent to "conceal" in an effort to increase the reward level for a future betrayal. On the other hand, a trustworthy Agent that currently trusts the other assigned Agents

may decide to "work alone" simply because the model has determined that betrayals are likely to occur at the current reward level.

4.3 Design of Experiments

Three tests are used to evaluate the performance of the TI-POMDP framework. The first test focuses on the impact of the various simulation settings (number of Agents, number of reward levels, etc.) on the response variables within the framework. The second test measures the change simulation run time as the number of agents involved increases. The final test compares the TI-POMDP framework with the trust vector [17] modeling approach.

The tests are conducted in two separate environments. In the first environment, a single agent can overcome a betrayal and still eventually complete the task. This allows the test to proceed. In the second environment agents can not directly overcome betrayal. As soon as the Agents discover a betrayal, they call for help from other trusted agents to deal with the attacker and complete the task. The call for help is a call to the Environment Controller to add an unassigned agent to the task. The second environment demonstrates an Agent's ability to use its trust model to overcome "betrayals" by adding additional Agents to the task.

4.3.1 Factorial Test. The first test is designed to identify which simulation settings have the largest impact on the output of the simulation. Each setting that can be changed in the simulation is called a factor. As the settings are adjusted the change in output values of the simulation is measured. Significant changes in output values indicate that the factor has a significant affect on the simulation.

The test is based on a factorial design with mixed level factors [14]. This means that each factor tested has two or more possible settings. An Analysis of Variance (ANOVA) is used to determine the level of contribution each factor has on the performance of the simulation and define a regression model for predicting the response value for various factor settings.

The first design step is to identify factors, their ranges, and specific levels. The second step is to identify the desired response variable. The final design step is creating the specific experiment design.

The simulation factors include:

- Number of Agents: the number of Agents operating within the environment. This test uses 3, 4, and 5.
- Mix of Agent: the simulation has three separate Agent mixes. In the first, all Agents have the same probability of corruption and redemption. In the second mix, the Agents are split into two groups where the second group is 10 percent more likely to be corrupted or redeemed than the first. In the final mix, there are three groups of Agents. The second group is 10 percent more likely to be corrupted or redeemed and the third group of the Agents is 20 percent more likely to be corrupted or redeemed than the first group. This test uses all three settings.
- Number of Reward Levels: the number of different reward levels an action can have. This test uses 2, 3, and 4.
- Probability of Corruption: the probability that a given Agent becomes more corrupt every time the transition function occurs. This test uses settings between 0.05 and 0.5 at increments of 0.05.
- Probability of Redemption: the probability that a given Agent becomes less corrupt every time the transition function occurs. This test uses settings between 0.05 and 0.5 at increments of 0.05.

Each test run sets each factor to a specific level (for instance 3 Agents, Agent complexity at 2, 4 reward levels, corruption rate at 0.2, and redemption rate at 0.45). These levels do not change for the entire test run and all combinations of levels are tested in a single replication. The factor levels are designed to fully evaluate the factor impact within the specific range while still providing insight into the impact of

factor settings outside of the tested range (7 Agents or 10 reward levels). Limiting the number of Agents to 5 avoids the time required to calculate the decision policy for 6 or more agents. Limiting the corruption and redemption rates to 0.5 avoids the highly fluctuating environments where Agents cannot predict future trust values.

The experiment limits Probability of Corruption and Probability of Redemption to the lower end of their potential values because higher values result in rapid changes in Agent trustworthiness. These rapid changes greatly degrade system performance, resulting in a situation where Agents no longer trust one another.

The potential response variables include

- Reward Value: the average reward achieved by each Agent.
- Number of "Cooperates": the number of times an Agent chooses to "cooperate."
- Number of Successful "Cooperates": the number times an Agent chooses to "cooperate" and receives its expected reward versus the total number of times an Agent chooses to "cooperate."

Reward value is the primary response variable tracked since it measures the overall Agent performance. The two percentages are also tracked since they give insight into an Agent's decision process. "Betray," "redeem," "work alone," and "conceal" can be substituted for "cooperate" in the two percentages as well.

Based on 3 settings of number of Agents, 3 reward levels, 3 Agents mixes, 10 corruption rates, and 10 redemption rates, a full factorial matrix includes 2700 separate test cases. The two test environments (Agents can overcome "betrayal" and Agents must call for help) result in 5400 unique test cases. Running each case 2 times results in a total of 10800 individual test runs. Each test run for this experiment consists of the group of Agents completing 50 tasks.

According to Montgomery's sample size determination method [14], this size of test has an Type I Error of 0.05 and a Type II Error of 0.01 assuming a response variable standard deviation to noise ratio of 2:1 or less. This means there is a probability

of 0.05 that this test will fail to identify an influential factor and a probability of 0.01 that this test will falsely identify a non influential factor as influential.

Statistica™is used to conduct the ANOVA. ANOVA compares the sum of squares of the factors with the sum of squares of the error to determine which factors influence the response variable. The ultimate goal of the test is to be able to predict how Agents perform in a given environment.

4.3.2 Performance Test. The performance test measures simulation execution time as the number of Agents increases. The amount of time required by the simulation increases as the number of Agents increases (the domain becomes more complex). This increase is the result of a more complex decision policy model and a larger number of Agents making decisions.

This test monitors the time required for a group of Agents to complete 50 tasks. Group size will be the only variable between test runs with groups ranging from 3 to 10 Agents. Each group size is run 10 times to determine an average time required to complete 50 tasks.

4.3.3 Comparison Test. The final set of tests is a direct comparison of action success rates between the TI-POMDP framework and a trust vector modeling approach [17]. Two types of comparison tests are used. The first test is a modified version of the Tiger Game introduced by Kaelbling [13] and expanded into a multi-agent game by Doshi [7] while the second test uses the Defender simulation.

In the multi-agent Tiger Game, two agents must choose which of two doors to open. Opening one door provides a reward while the other frees a tiger that penalizes the agent. Agents may open the left door, open the right door, or listen. Listening has a probability (0.85) of correctly hearing which door hides the tiger. Opening a door resets the location of the tiger and the reward, starts a new game, and results in a squeak that lets the other agent know the game was reset.

The multi-agent Tiger Game is modified into the Cooperative Tiger Game (CTG). In the CTG both agents must cooperate to open the door with the reward. Every time the agents cooperate, the reward value doubles for the next game. One agent can betray the other by opening the tiger door when the other agent tries to open the reward door. The betraying agent receives double the reward value while the betrayed agent is penalized. An agent that believes the other agent is going to betray him can reset the reward back to its original level. Agent trust levels can fluctuate which changes the probability that one agent will betray another. Agents communicate prior to each turn to reach a non-binding agreement on which action to take. The CTG test uses a 0.05 probability of an agent becoming more corrupt or more trustworthy.

Figure 4.5 illustrates the state transition process an agent undergoes. It does not include the belief model update the agent uses to transition between Trusting and Not Trusting beliefs. It is important to note that the Not Trusting belief applies if the agent has been corrupted or the agent believes the other agent has been corrupted. In either situation, the agent chooses to open door with the tiger to maximize its reward.

The Defender simulation comparison testing has a group of Agents complete 50 tasks using either the TI-POMDP framework or a trust vector model as its action decision process. In addition, the TI-POMDP framework is tested with a 5 action horizon and a next action horizon.

The Defender simulation tests are conducted with three number of Agents settings (3, 4, and 5), three reward level settings (2, 3, and 4), the three mix of Agents settings, and three corruption/redemption rates (0.1, 0.3, and 0.5). There are two complete replications of these setting combinations. The average reward, number of cooperations, success rate of cooperations, number of betrayals, and success rate of betrayals are used to compare the models.

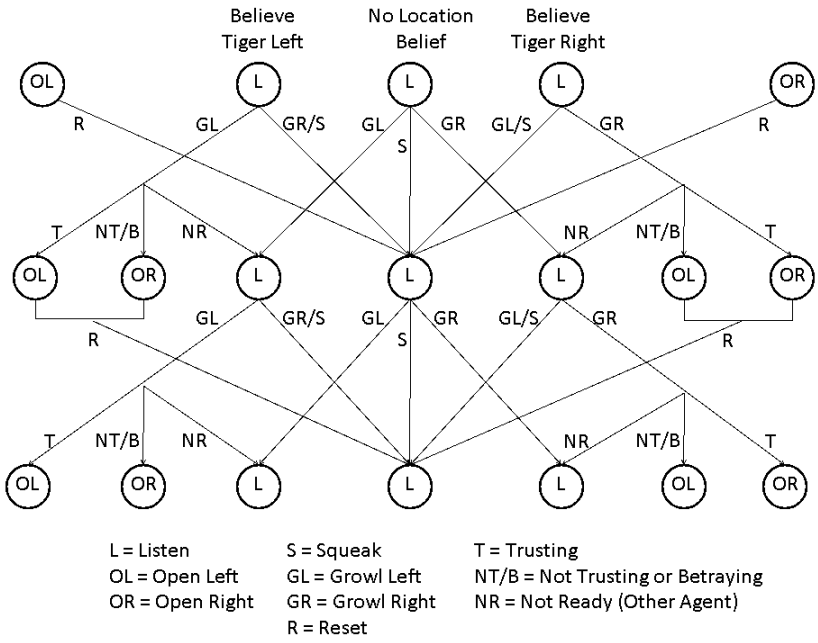

Figure 4.5: The state-action-observation transitions for the Cooperative Tiger Game.

4.4 Summary

Agents use individual trust models to decide how to interact with each other. When possible Agents attempt to benefit from cooperative actions while minimizing their vulnerability to betrayal. The Defender Simulation represents a trust-based multi-agent environment. The simulation provides a tailorable environment that can be applied to wide range of problems. The outlined testing for this simulation examines the impact of the simulation settings, the performance of the TI-POMDP framework, and the compares the TI-POMDP framework to the trust vector model.

V. Results and Analysis

The testing outlined in Section 4.3 is designed to evaluate the performance and capabilities of the Defender Simulation and the TI-POMDP framework. The testing includes an analysis of the simulation settings that affect the TI-POMDP results, an estimate of the TI-POMDP performance, and a comparison of the TI-POMDP with a trust vector model. This chapter presents the results of those tests and an analysis of the associated capabilities of the TI-POMDP.

5.1 Factorial Test Results

The factorial test is designed to illustrate how five separate factors (number of agents, types of agents, number of trust levels, probability of corrupting an agent, and probability of redeeming an agent) impact the five response variables (average reward, number of successful "cooperates," total number of "cooperates," number of successful "betrays," and "total number of "betrays"). The test includes two replications of a full factorial test design [14] with a randomized test run order. The full factorial test is conducted in both the basic and complex task environments. The results for the factorial test are divided by response variables.

ANOVA is the primary analytical tool for this test. ANOVA relies on the Sum of Squares to calculate the F statistic for the level of contribution of each factor. For instance, in this test the Number of Agents has three different settings (3, 4, and 5 agents) and each test run uses one of these settings. The Sum of Squares due to the treatment (SSTR) is

$$n \sum_{i=1}^{a} (\bar{y}_{i\cdot} - \bar{y}_{\cdot\cdot})^2, \tag{5.1}$$

where

- a is the number of factor levels, in this case 3.
- n is the number of samples taken at each factor level.

- $\bar{y}_{i\cdot}$ is the mean of all samples with the same factor setting (when $a = 1$, this is all samples with 3 agents).
- $\bar{y}_{\cdot\cdot}$ is the grand mean of all samples.

The SSTR for the Number of Agents is compared to the unexplained variance or error associated with the data. The error is the variation of the individual data points around $\bar{y}_{i\cdot}$. The Sum of Squares estimate of the total error (SSE) is

$$\sum_{i=1}^{a}\sum_{j=1}^{n}(y_{ij} - \bar{y}_{i\cdot})^2, \tag{5.2}$$

where

- a, n, and $\bar{y}_{i\cdot}$ are the same as Equation 5.1.
- y_{ij} is an individual data point.

The Mean Square Due to Treatment (MSTR) is

$$\frac{SSTR}{a-1}, \tag{5.3}$$

and the Mean Square Due to Error (MSE) is

$$\frac{SSE}{N-a}, \tag{5.4}$$

where N is the total number of samples (usually $a * n$).

The $a-1$ in MSTR and the $N-a$ in MSE are the number of degrees of freedom required to calculate those statistics. The F statistic for a treatment is the ratio

$$F_0 = \frac{MSTR}{MSE}. \tag{5.5}$$

As the ratio approaches 1, the MSTR becomes analogous to MSE which means the treatment has no discernible affect on the outcome data. If the MSTR is signifi-

cantly larger than the MSE, the treatment is considered to impact the outcome data. Significance is determined by the F statistic. Figure 5.1 shows the various components used to calculate the Sum of Squares. In this figure, the factor settings are the different levels of a given factor (for instance number of Agents at 3, 4, 5, and 6), the grand mean is the average of all of the individual samples, and the group means are the averages of the individual samples at a specific factor setting. The variance due to treatment is the difference between the grand mean and each of the group means. This represents the model variance attributed to the factor (number of Agents). The variance of the individual samples within a group is the unexplained variance which is either caused by another factor or is error associated with the process.

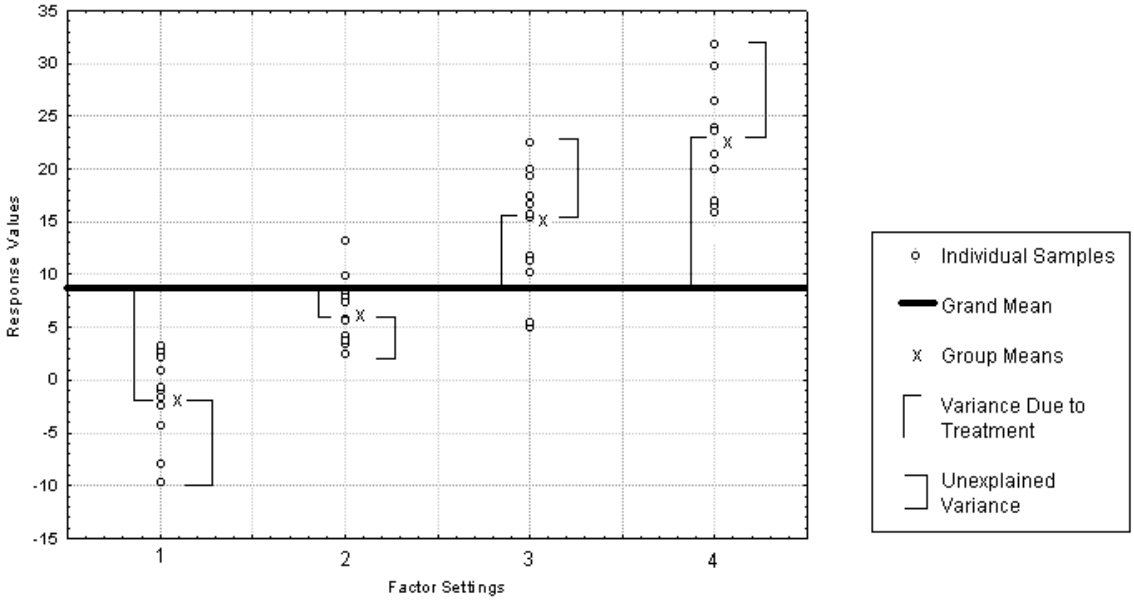

Figure 5.1: Sum of Squares Analysis components. This figure shows the components used in a notional Sum of Squares analysis. The analysis measures the effect of a factor with four different settings on the output response value.

5.1.1 Average Reward. Running ANOVA with average reward as the response variable returns the Table 5.1. The chart has two entries for every factor, a linear component and a quadratic component. In addition, the entries "Number of Agents by Reward Levels," "Number of Agents by Agent Mix," etc. are interactions between two factors. These interactions are the effect specific settings of two factors,

such as Number of Agents and Reward Levels, have on the response variable. It can be the case that a specific setting (say 5 agents and 4 reward levels) results in a large change to the response variable in comparison with 3 or 4 agents and 4 reward levels or 5 agents and 2 or 3 reward levels. The columns of the chart are the Sum of Squares, the degrees of freedom, the Mean Square, the F statistic, and the significance value p of the F statistic. The p value is the probability that a particular component does not have an effect on the response variable, with values less than 0.05 signifying an active factor.

The p values in Table 5.1 indicate that every factor and interaction is significant except for the linear Agent Mix, the quadratic Agent Mix, and four interactions involving Agent Mix.

5.1.1.1 Result Impact on ANOVA Assumptions. The large number of active factors indicates that this data set may have an assumption violation. There are four assumptions made by ANOVA (independent samples, a completely defined model, normal distribution of residuals, and equal variance within cells). The independent samples assumption is satisfied because the runs are collected in a randomized fashion and each run is separate eliminating the possibility of a timing trend. The test design ensures model completeness because only the five identified factors change from run to run, there is not a concern that weather, time of day, or some other factor can influence the model.

Validating the other two assumptions comes from analyzing the data. Residuals are the difference between an expected value and the actual value. ANOVA generates a model that predicts the response variable for a specific group of factor settings. The difference between the predicted value and the value from the actual data points is the residual. When plotted, normally distributed residuals look like random noise with no discernible pattern. Figure 5.2 shows the actual residuals for each of the predicted values from the model. The definite funneling on the left side of the graph is common when cell variances are not equal.

Table 5.1: Complete ANOVA with average reward as response variable.

Factor	SS	df	MS	F	p
Number of Agents(L)	$5.28 * 10^{12}$	1	$5.28 * 10^{12}$	883.01	0.00
Number of Agents(Q)	$1.12 * 10^{11}$	1	$1.12 * 10^{11}$	18.71	0.00
Reward Levels(L)	$1.11 * 10^{14}$	1	$1.11 * 10^{14}$	18504.04	0.00
Reward Levels(Q)	$2.58 * 10^{13}$	1	$2.58 * 10^{13}$	4312.49	0.00
Agent Mix(L)	$6.11 * 10^{9}$	1	$6.11 * 10^{9}$	1.02	0.31
Agent Mix(Q)	$8.26 * 10^{9}$	1	$8.26 * 10^{9}$	1.38	0.24
Corruption Rate(L)	$2.92 * 10^{13}$	1	$2.92 * 10^{13}$	4886.83	0.00
Corruption Rate(Q)	$3.17 * 10^{12}$	1	$3.17 * 10^{12}$	530.20	0.00
Redemption Rate(L)	$6.72 * 10^{12}$	1	$6.72 * 10^{12}$	1123.79	0.00
Redemption Rate(Q)	$5.52 * 10^{11}$	1	$5.52 * 10^{11}$	92.32	0.00
Number of Agents(L) by Reward Levels(L)	$6.32 * 10^{12}$	1	$6.32 * 10^{12}$	1055.57	0.00
Number of Agents(L) by Agent Mix(L)	$7.19 * 10^{9}$	1	$7.19 * 10^{9}$	1.20	0.27
Number of Agents(L) by Corruption Rate(L)	$4.41 * 10^{10}$	1	$4.41 * 10^{10}$	7.38	0.01
Number of Agents(L) by Redemption Rate(L)	$1.48 * 10^{11}$	1	$1.48 * 10^{11}$	24.72	0.00
Reward Levels(L) by Agent Mix(L)	$9.55 * 10^{9}$	1	$9.55 * 10^{9}$	1.60	0.21
Reward Levels(L) by Corruption Rate(L)	$3.53 * 10^{13}$	1	$3.53 * 10^{13}$	5904.70	0.00
Reward Levels(L) by Redemption Rate(L)	$8.27 * 10^{12}$	1	$8.27 * 10^{12}$	1381.49	0.00
Agent Mix(L) by Corruption Rate(L)	$1.00 * 10^{10}$	1	$1.00 * 10^{10}$	1.67	0.20
Agent Mix(L) by Redemption Rate(L)	$8.46 * 10^{9}$	1	$8.46 * 10^{9}$	1.41	0.23
Corruption Rate(L) by Redemption Rate(L)	$5.04 * 10^{11}$	1	$5.04 * 10^{11}$	84.29	0.00
Error	$3.22 * 10^{13}$	5379	$5.98 * 10^{9}$		
Total SS	$2.64 * 10^{14}$	5399			

The issue with unequal variance is that ANOVA incorrectly estimates the mean of cells with large variances. The MSE for these cells becomes falsely small and the mean for the cell appears to be an actual trend influencing the response variable.

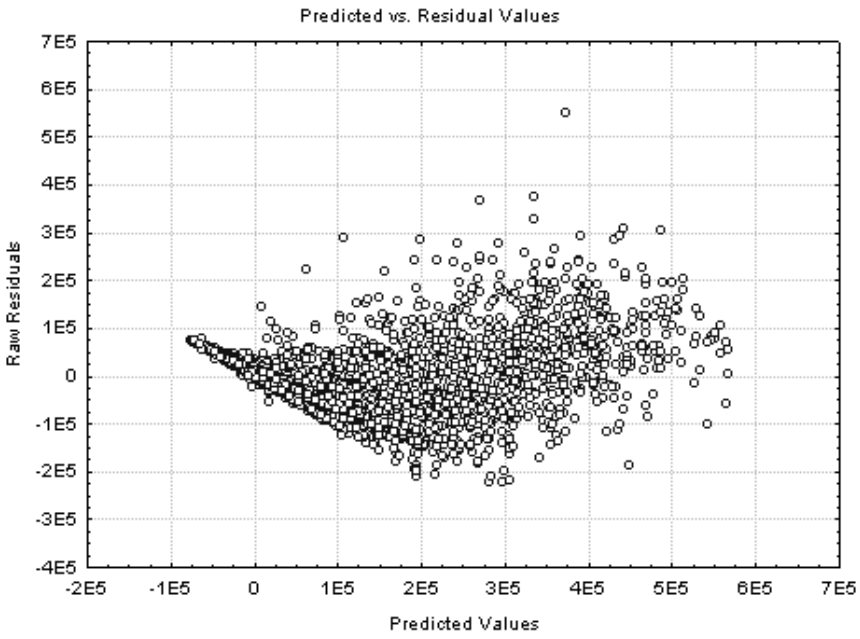

Figure 5.2: Predicted Values vs Residual Values.

Using a contractive transform on the response variable values can pull the data together and reduce the variance. Contractive transforms include square root and log functions. Applying contractive transforms to the data set does not reduce the unequal variances to an acceptable level for analysis.

Although the assumptions violation (unequal variances) can not be remedied, analysis using resampling does not require the assumption of equal variance. Resampling [9] draws repeated samples from our data set. These samples are used describe the entire population. The only assumptions made by resampling are the model is completely defined and the data represents the distribution of population values.

The resampling technique used in this paper breaks the factorial data set into several subsets according to the factor settings. For instance, one subset contains all of the test runs with 3 Agents, while another contains all the 4 Agent runs and a third has the 5 Agent runs. Each factor is broken down in this manner with the subsets of number of Agents, reward levels, and Agent complexity having 1800 runs each while the corruption and redemption rate subsets have 540 runs each.

Using the Resampling Toolkit add-in for Excel™, 200 instances of the response variable (in this case average reward for the group of Agents) are randomly selected from the 3 Agents data subset. The selections are done with replacement, so the same data point may be used more than once. The average of the 200 data points is collected as an estimate of the overall population mean for a test run with 3 Agents in the environment (regardless of the other factor settings). This process is repeated 10,000 times for the 3 Agent subset as well as every other subset.

The average of the 10,000 population mean estimates becomes the actual resampled mean estimate with standard deviation also based on the 10,000 resampled points. Figure 5.3 shows the mean average reward for each Agent size with 95 percent confidence intervals. Average reward is not bounded. The graph indicates that the number of Agents in the environment has a significant effect on the average reward of the Agents. As the number of Agents increases, the average reward decreases. The change in reward is attributed to the higher probability that one or more Agents is corrupt in an environment with more Agents. This trend indicates that the TI-POMDP results are influenced by the number of Agents in the environment. As the size of the environment increases the average individual reward decreases due to Agents participating in fewer tasks and tasks with more assigned Agents have a higher probability of "betrayal" attempts which reduces the overall reward for the task.

Figure 5.4 shows the mean average reward for each reward level with 95 percent confidence intervals using resampling. Average reward is not bounded. The graph shows a significant quadratic increase in average reward as the reward level increases. This increase is expected since the reward is based on the square of the reward level. This trend indicates that the TI-POMDP results are influenced by the number of Agents in the environment. As the reward structure of the environment changes to benefit the Agents, the average individual reward increases because the TI-POMDP is able to leverage the higher rewards.

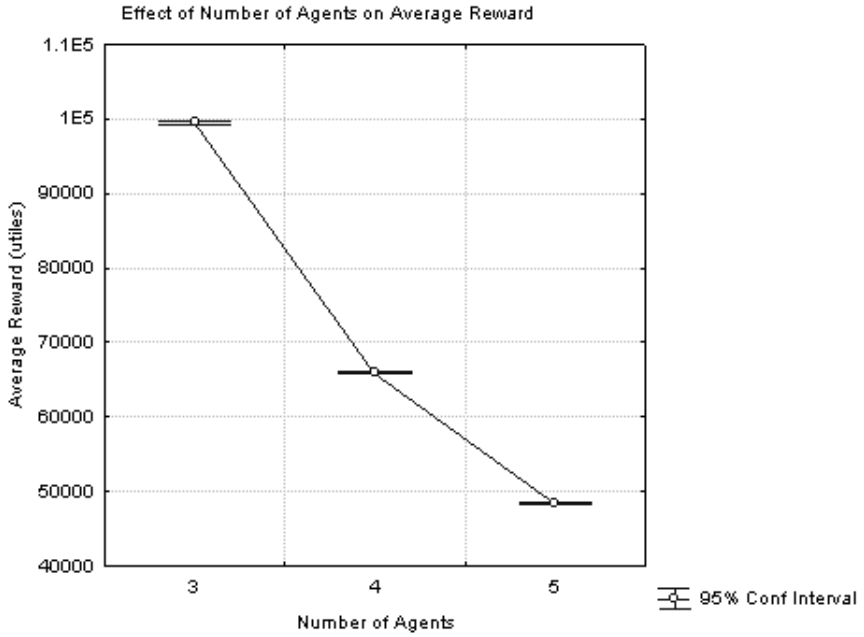

Figure 5.3: Average reward based on the number of Agents in the Environment.

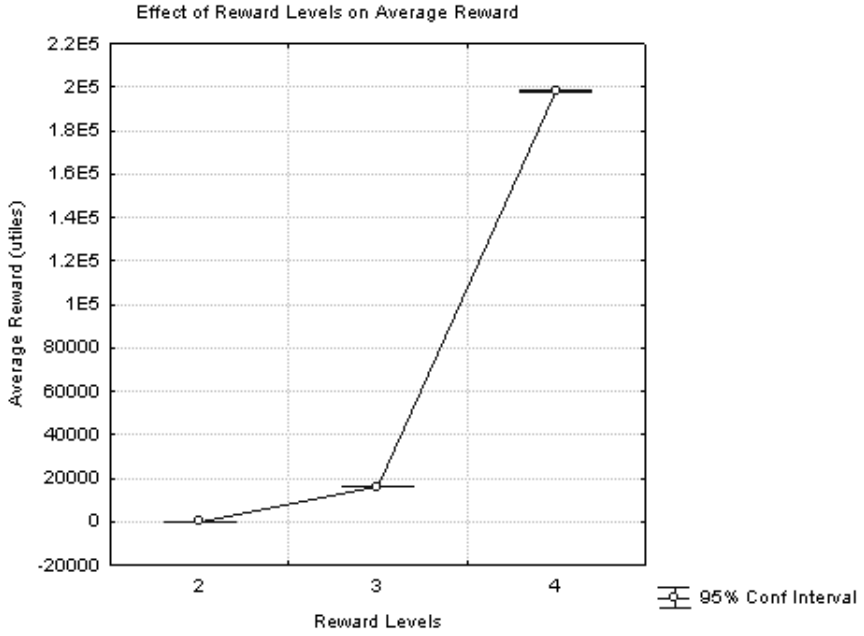

Figure 5.4: Average reward based on the number of reward levels.

Figure 5.5 shows the mean average reward for each Agent mix with 95 percent confidence intervals using resampling. Average reward is not bounded. While there is

a difference between the average reward of mix 1 (all Agents have the same corruption rate) and mix 2 (2 groups of Agents, where one group has a 10 percent higher probability of being corrupted than the other group), there is not a difference between mix 2 and mix 3 (3 groups of Agents, where one group has a 10 percent higher probability of being corrupted and a second group has a 20 percent higher probability than the baseline group). This indicates that a single group operating with a higher corruption rate can achieve higher rewards than a uniform group, but additional groups do not achieve a similar benefit. Since the difference in means between mix 1 and mix 2 is only 1600 reward points (2.2 percent of average reward), the effect of agent mix is negligible. The lack of effect indicates that the TI-POMDP is not heavily influenced by heterogeneous Agents.

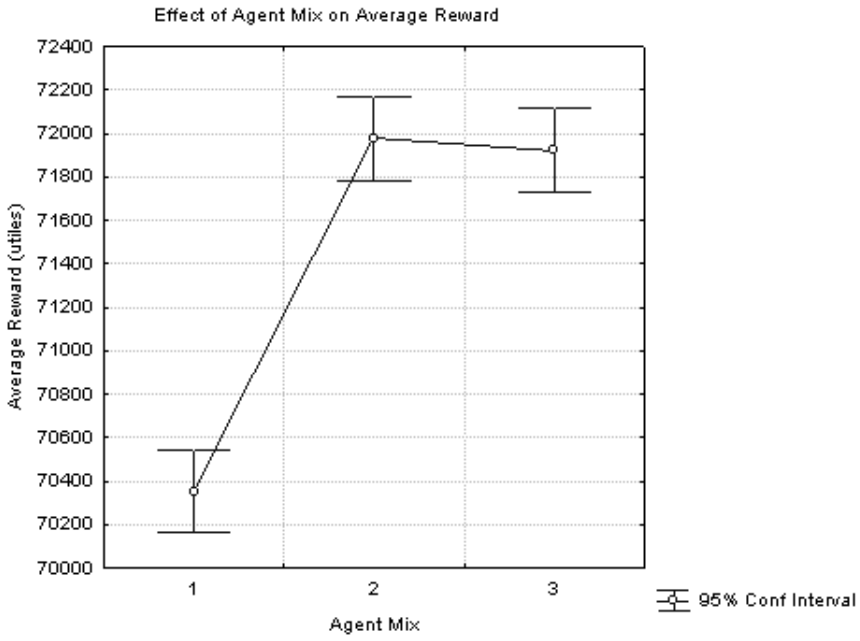

Figure 5.5: Average reward based on the mix of the Agents in the Environment.

Figure 5.6 shows the mean average reward for each corruption rate with 95 percent confidence intervals using resampling. Average reward is not bounded. The graph shows a significant exponential decrease in average reward as the corruption rate increases. This increase is attributed to Agents not "cooperating" while "be-

trays" become ineffective with the increasing corruption rate. This trend indicates that the trustworthiness of the environment has a large impact on the TI-POMDP. In low corruption environments, the TI-POMDP is able to "cooperate" extensively. The TI-POMDP isolates ("works alone" or "conceals") in high corruption environments. In between, the TI-POMDP has a steady decrease in rewards as it balances "cooperating" and isolating actions.

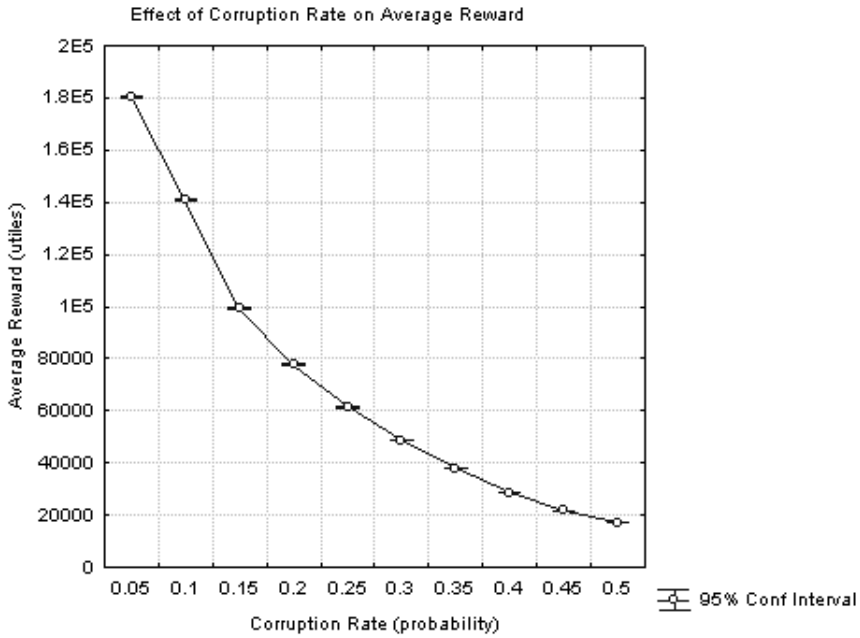

Figure 5.6: Average reward based on the probability of Agent corruption in the Environment.

Figure 5.7 shows the mean average reward for each redemption rate with 95 percent confidence intervals using resampling. Average reward is not bounded. The graph shows a significant exponential increase in average reward as the redemption rate increases. The higher redemption rate increases the probability that an Agent is redeemed before it can "betray," allowing the Agent to continue "cooperating" and increasing the reward for each action. The TI-POMDP is affected by the environment's redemption rate, but not to the same extent as the corruption rate. Higher redemption rates reduce the potential for corruption because Agents may be redeemed before they can "betray."

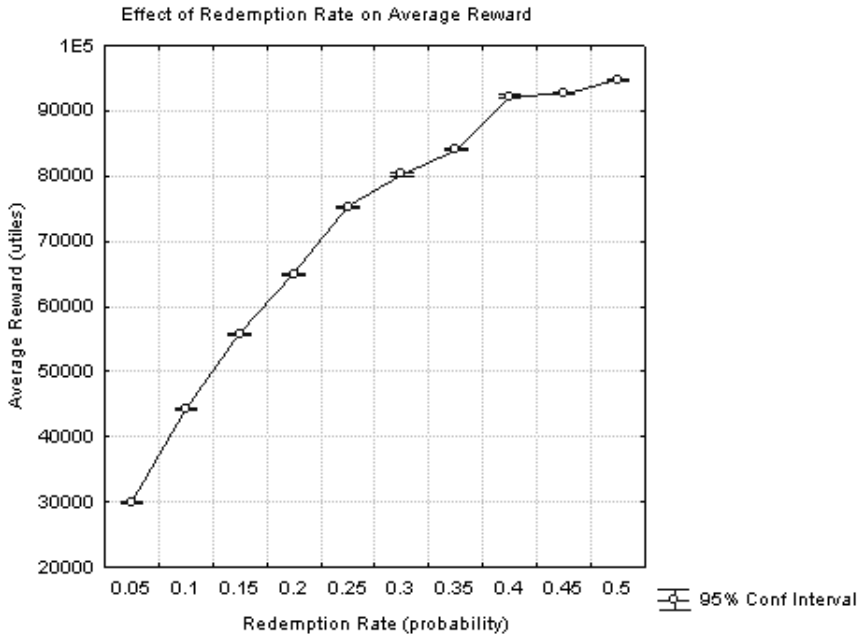

Figure 5.7: Average reward based on the probability of Agent redemption in the Environment.

For the rest of the factor analysis, only the resampling results are shown since the unequal variances affect the entire data set.

5.1.2 "Cooperates". "Cooperates" is the number of times Agents attempt to "cooperate" and the number of times the Agents successfully "cooperate." Figure 5.8 shows the number of times Agents try to "cooperate" and the number of times they succeed for each Agent size with 95 percent confidence intervals using resampling. The number of "cooperates" ranges from 0 to ∞. The graph indicates that Agents "cooperate" less as the number of Agents increases. The fewer "cooperates" are due to an increase in the number of Agents assigned to a task and a higher probability that one of the Agents does not trust another Agent. The success rate of "cooperates" also decreases as the number of Agents increases as seen in Table 5.2. The TI-POMDP becomes less "cooperative" in larger environments because more Agents are assigned to a task which increases the probability that the Agents do not trust at least one assigned Agent. In this particular domain, the average task has just over half the Agents assigned to it.

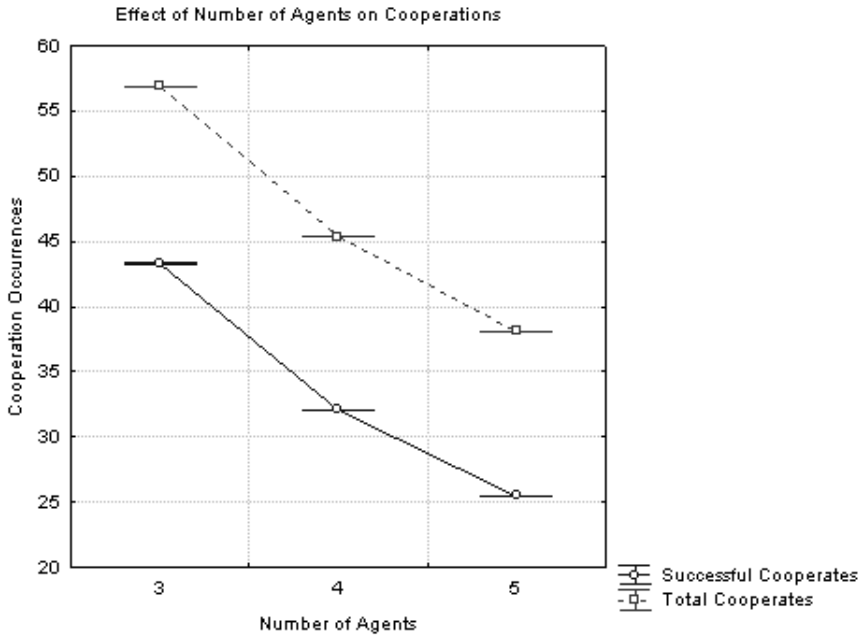

Figure 5.8: Number of cooperates and successful cooperates based on the number of Agents in the Environment.

Figure 5.9 shows the number of times Agents try to "cooperate" and the number of times they succeed for each reward level with 95 percent confidence intervals using resampling. The number of "cooperates" ranges from 0 to ∞. The graph shows a significant increase in both "cooperates" attempted and successful "cooperates" as reward level increases. This increase is due to Agents "cooperating" on lower level tasks to achieve higher rewards later. Table 5.2 shows the success rate of "cooperates" is steady for the first two levels, but increases at the third level. The TI-POMDP attempts to maximize its reward by "cooperating" more in environments with higher reward levels. The higher levels generate higher payoffs, so the Agents, either trustworthy or deceitful, "cooperate" to get to the maximum reward level.

Figure 5.10 shows the number of times Agents try to "cooperate" and the number of times they succeed for each Agent mix with 95 percent confidence intervals using resampling. The number of "cooperates" ranges from 0 to ∞. The graph indicates that Agent mix does not have a discernible impact on "cooperating." The

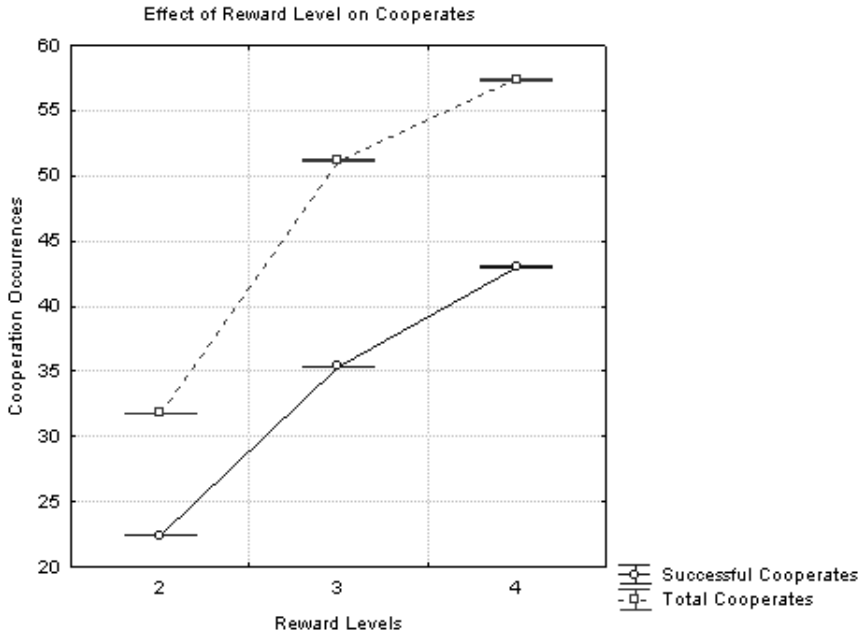

Figure 5.9: Number of cooperates and successful cooperates based on the number of reward levels.

"cooperate" success rate in Table 5.2 does not significantly change with the Agent mix. Once again, the TI-POMDP is not affected by a heterogeneous group of Agents.

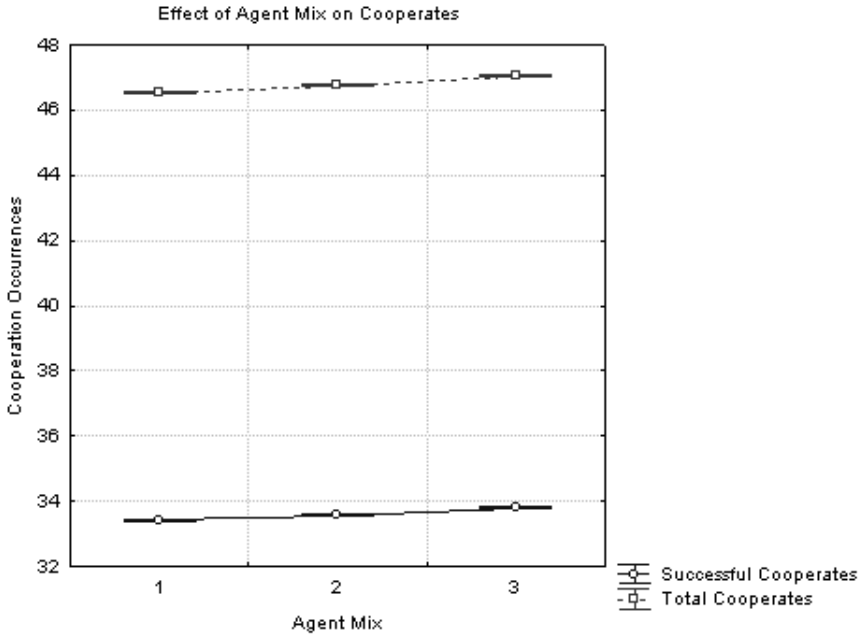

Figure 5.10: Number of cooperates and successful cooperates based on the mix of the Agents in the Environment.

Figure 5.11 shows the number of times Agents try to "cooperate" and the number of times they succeed for each corruption rate with 95 percent confidence intervals using resampling. The number of "cooperates" ranges from 0 to ∞. The graph shows that corruption rate decreases "cooperate" attempts and successes exponentially. As all Agents become less trustworthy, the Agents become less willing to "cooperate." Table 5.2 shows that the success rate of "cooperates" decreases to 0.6 before leveling off. The TI-POMDP becomes less likely to choose to "cooperate" as the probability of Agent corruption increases.

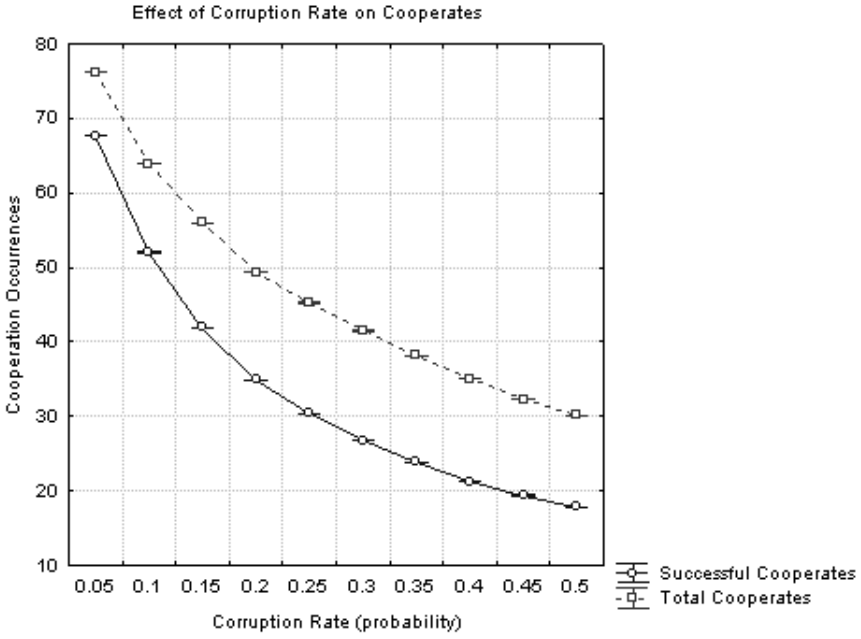

Figure 5.11: Number of cooperates and successful cooperates based on the probability of Agent corruption in the Environment.

Figure 5.12 shows the number of times Agents try to "cooperate" and the number of times they succeed for each redemption rate with 95 percent confidence intervals using resampling. The number of "cooperates" ranges from 0 to ∞. The graph shows an increase in the number of "cooperates" and the number of successful "cooperates" as redemption rate increases. While both attempts and successes increase as the redemption rate increases, the attempts increase more. This results in a lower success rate at higher redemption rates. Table 5.2 shows the diminishing success rate that

Table 5.2: Cooperation success rate for each factor setting.

Setting	Number of Agents	Reward Level	Agent Mix	Corruption Rate	Redemption Rate
1	76.1	70.6	71.9	88.8	79.6
2	70.6	69.1	71.8	81.4	75.6
3	66.9	74.9	71.8	74.9	74.6
4	-	-	-	70.7	72.4
5	-	-	-	67.1	71.7
6	-	-	-	64.4	71.2
7	-	-	-	62.5	70.3
8	-	-	-	60.6	70.0
9	-	-	-	60.2	70.2
10	-	-	-	59.2	70.2

levels off at 0.7. The TI-POMDP is able to attempt more "cooperates" because the Agents are less likely to be corrupt for long periods of time.

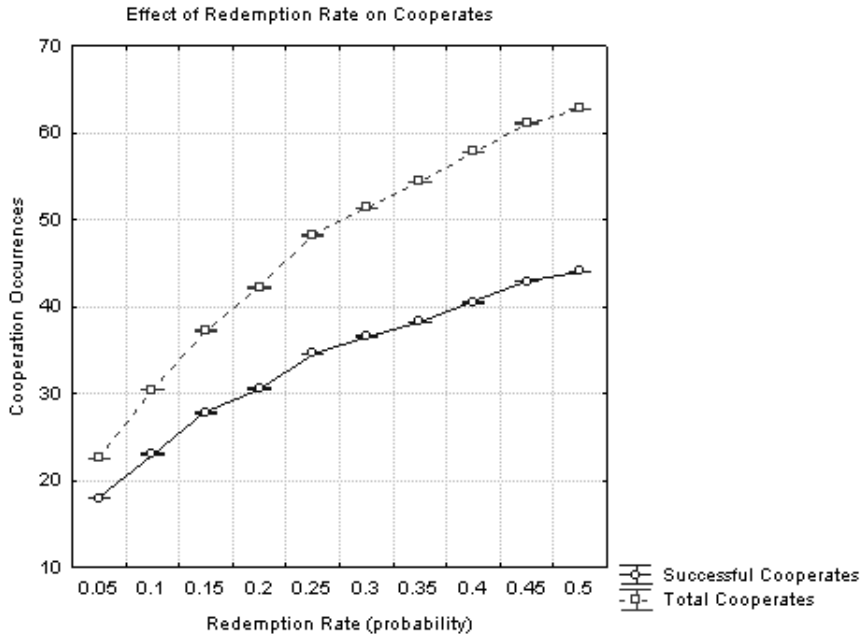

Figure 5.12: Number of cooperates and successful cooperates based on the probability of Agent redemption in the Environment.

5.1.3 "Betrays". "Betrays" includes the number of times Agents attempt to "betray" and the number of times the Agents successfully "betray." Figure 5.13 shows the number of times Agents try to "betray" and the number of times they succeed for each Agent size with 95 percent confidence intervals using resampling. The number of "betrays" ranges from 0 to ∞. The graph shows a constant linear decrease Agent "betrayals" as the number of Agents increases. The fewer "betrays" are due to an increase in the number of Agents assigned to a task and a higher probability that one of the Agents assumes it is not the only Agent attempting to "betray." The success rate of "betrays" also decreases as the number of Agents increases as seen in Table 5.3. The TI-POMDP is less likely to attempt to "betray" in larger environments because the tasks have more Agents assigned to them, increasing the probability that at least one Agent is already not trusted.

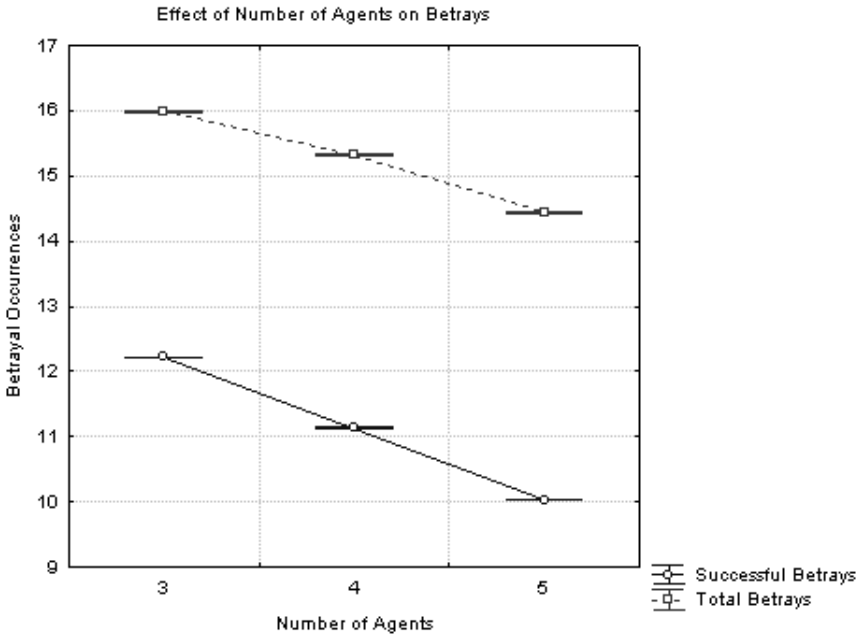

Figure 5.13: Number of betrays and successful betrays based on the number of Agents in the Environment.

Figure 5.14 shows the number of times Agents try to "betray" and the number of times they succeed for each reward level with 95 percent confidence intervals using resampling. The number of "betrays" ranges from 0 to ∞. The graph shows a sig-

nificant increase in both "betrays" attempted and successful "betrays" as the reward levels go from 2 to 3, but the trends decrease as the level goes to 4. The changes are due to Agents building trust on low reward level tasks to "betray" on high reward level tasks. As the number of levels continues to increase, Agents are more apt to wait for the highest reward levels, which occur rarely and the Agent may be redeemed prior to reaching those levels. Table 5.3 shows the success rate of "betrays" is steady for the first two levels, but decreases at the third level. After an initial jump, the TI-POMDP actually decides to "betray" less as the reward level increases because the "cooperating" to get to higher reward levels is in the Agent's best interests. While the Agents "cooperate" on several tasks to drive up the reward level they can be redeemed before they reach the level they wanted to "betray" at. Additionally, since the Agents complete a set number of tasks, they "cooperate" on a large number of low reward tasks to get to the high reward.

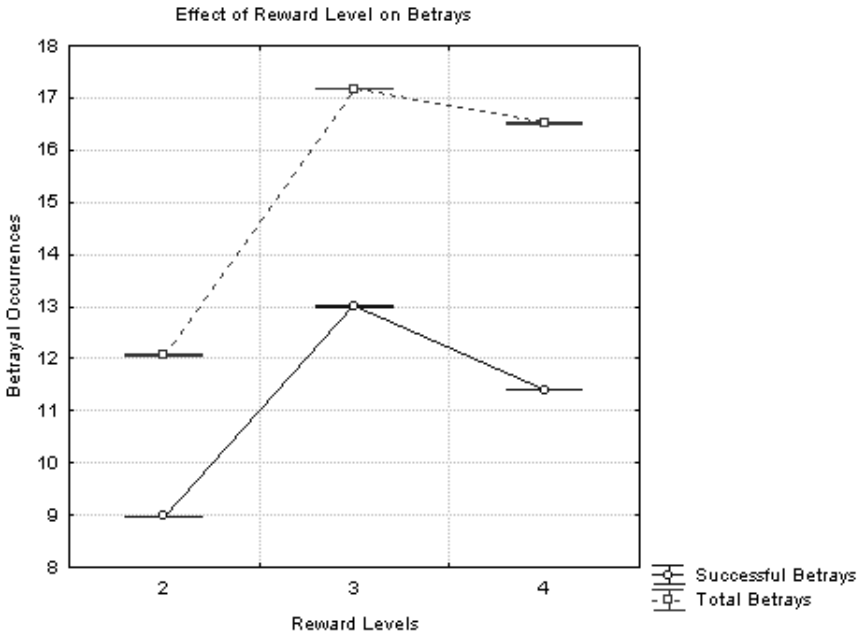

Figure 5.14: Number of betrays and successful betrays based on the number of reward levels.

Figure 5.15 shows the number of times Agents try to "betray" and the number of times they succeed for each Agent mix with 95 percent confidence intervals using

resampling. The number of "betrays" ranges from 0 to ∞. The graph indicates that Agent mix does not have a noticeable impact on "betraying." The "betray" success rate in Table 5.3 does not significantly change with the Agent mix either. The TI-POMDP is not influenced by these heterogeneous groups of Agents.

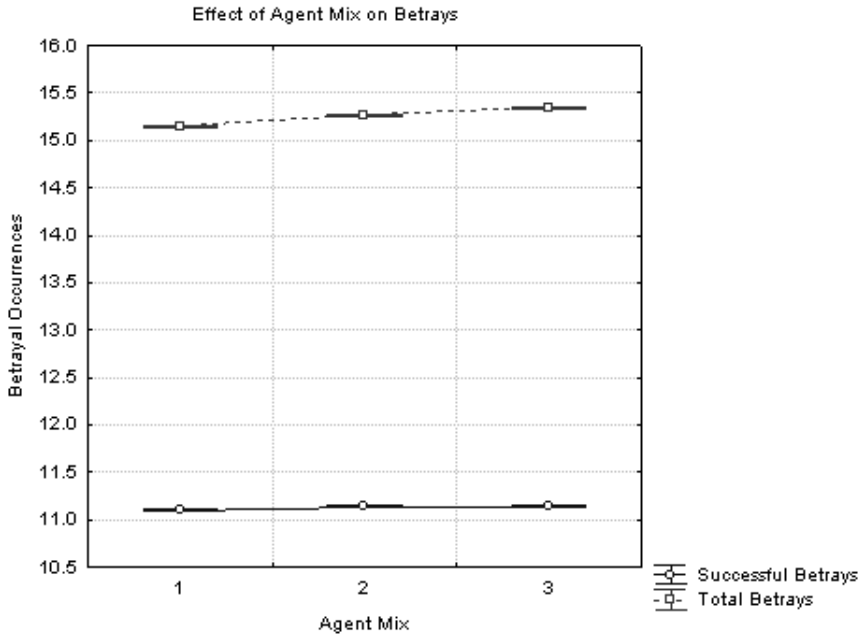

Figure 5.15: Number of betrays and successful betrays based on the mix of the Agents in the Environment.

Figure 5.16 shows the number of times Agents try to "betray" and the number of times they succeed for each corruption rate with 95 percent confidence intervals using resampling. The number of "betrays" ranges from 0 to ∞. The graph shows that corruption rate increases "betray" attempts and successes up to a point before the attempts and successes decrease. The peak of the "betrays" occurs with corruption rates between 0.15 and 0.3. In this range, the probability is that only one Agent is corrupted at a time. As all Agents become less trustworthy, the Agents become less willingly to "betray" since they do not trust the other Agents. Table 5.3 shows that the success rate of "betray" decreases to 0.64. The environment corruption rate influences the TI-POMDP's usage of "betrayals." In very trustworthy and very corrupt environments, the TI-POMDP does not "betray." In the very trustwor-

thy environment, there are not any corrupt Agents (Utopian society). In the very untrustworthy environment, everyone is corrupt so there is no benefit to betrayal (honor among thieves). The TI-POMDP does try to "betray" in slightly corrupt environments because only a few Agents become untrustworthy at any given time.

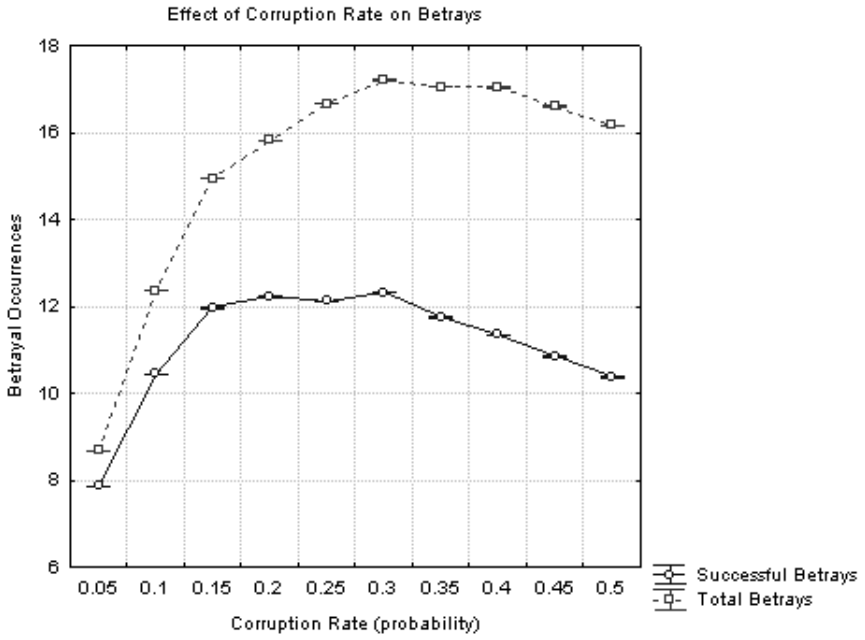

Figure 5.16: Number of betrays and successful betrays based on the probability of Agent corruption in the Environment.

Figure 5.17 shows the number of times Agents try to "betray" and the number of times they succeed for each redemption rate with 95 percent confidence intervals using resampling. The number of "betrays" ranges from 0 to ∞. The graph shows an increase in the number of "betrays" and the number of successful "betrays" as redemption rate increases. As the probability of an Agent getting redeemed increases, newly corrupted Agents are able to betray the newly redeemed Agents. Table 5.3 shows success rate staying around 0.73. The environment redemption rate increases the TI-POMDP's usage of the "betray" decision because Agents are more trusting of each other when a given Agent is only corrupted for one or two actions.

As an excursion test, if the probability of corruption is 0.99 and the probability of redemption is 0.01, the system stops "cooperating" and "betraying" completely.

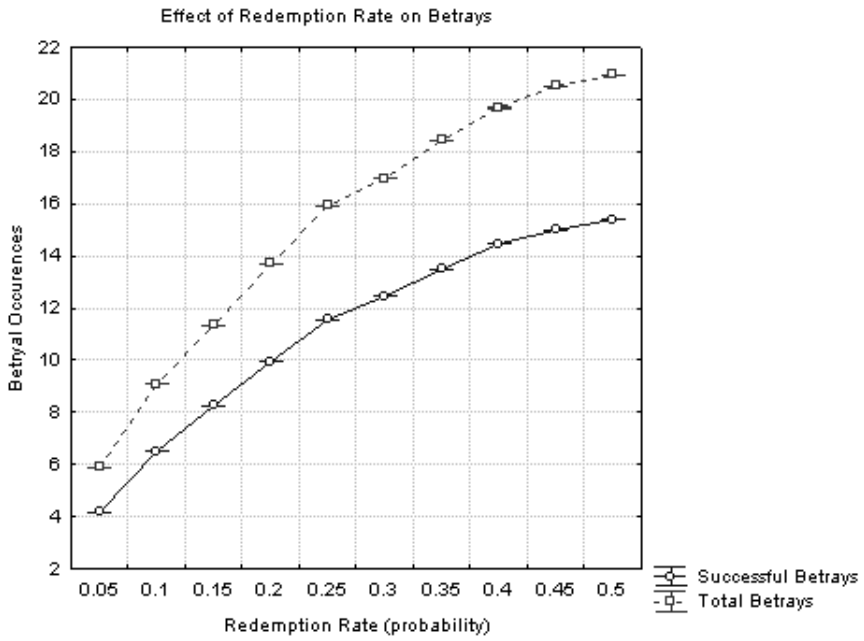

Figure 5.17: Number of betrays and successful betrays based on the probability of Agent corruption in the Environment.

Table 5.3: Betrayal success rate for each factor setting.

Setting	Number of Agents	Reward Level	Agent Mix	Corruption Rate	Redemption Rate
1	76.3	74.3	73.3	90.6	71.1
2	72.7	75.7	73.0	84.5	71.7
3	69.4	69.0	72.6	80.0	72.7
4	-	-	-	77.3	72.5
5	-	-	-	72.7	72.5
6	-	-	-	71.6	73.2
7	-	-	-	68.9	73.2
8	-	-	-	66.6	73.5
9	-	-	-	65.3	73.1
10	-	-	-	64.1	73.5

The Agents always choose to "work alone" or "conceal" and the average reward is -2573 because the Agents lose 1 point every time they act and the majority of their actions are "conceal" which does not quickly destroy an attacker.

5.2 Performance Test Results

The performance test measure the time required to complete 50 tasks by 3, 4, or 5 Agents. Each test run uses a homogeneous group of Agents. For each task, the Environment Controller selects a randomly sized subset of the group of Agents to perform the task. After each action, Agents have 0.25 probability of becoming more corrupt or trustworthy. The reward level setting for these runs is 2.

Figure 5.18 depicts the simulation time required to complete 50 tasks for 3, 4, and 5 Agents. The steep trend between 4 and 5 Agents is indicative of the exponential increase with respect to the overall number of Agents. Since τ contains $n + (n-1)^2$ ratings and each rating can take one of two values (the number of reward levels), the state space difference between 4 and 5 agents is $2^2 1 - 2^1 3 = 2088960$. The size difference between 5 and 6 agents reaches 2145286496. While the behavior equivalence approximation method reduces the amount of search space visited, the total search space for 6 Agents is 1000 times larger than the search space for 5 Agents ($2^3 1/2^2 1 = 2^8 = 1024$).

The bulk of the computation time comes from learning the model. Figure 5.19 shows the time required to complete 50 tasks for 3 t o10 Agents when they do not have to learn the policy model. In this case, the Agents just follow their current trust ratings of the other Agents. The trend indicates that executing the TI-POMDP updates is linear with respect to the number of Agents in the environment while learning the policy suffers from exponential growth.

5.3 Comparison Test Results

Two comparison tests are used to measure the difference between the TI-POMDP model and a Trust Vector model. The Trust Vector Model stores the five previous actions of the other Agents. The Trust Vector length of five is consistent with the TI-POMDP horizon of five. While the TI-POMDP is looking ahead five steps, the Trust Vector looks at the previous five steps. Trustworthy actions enter the model

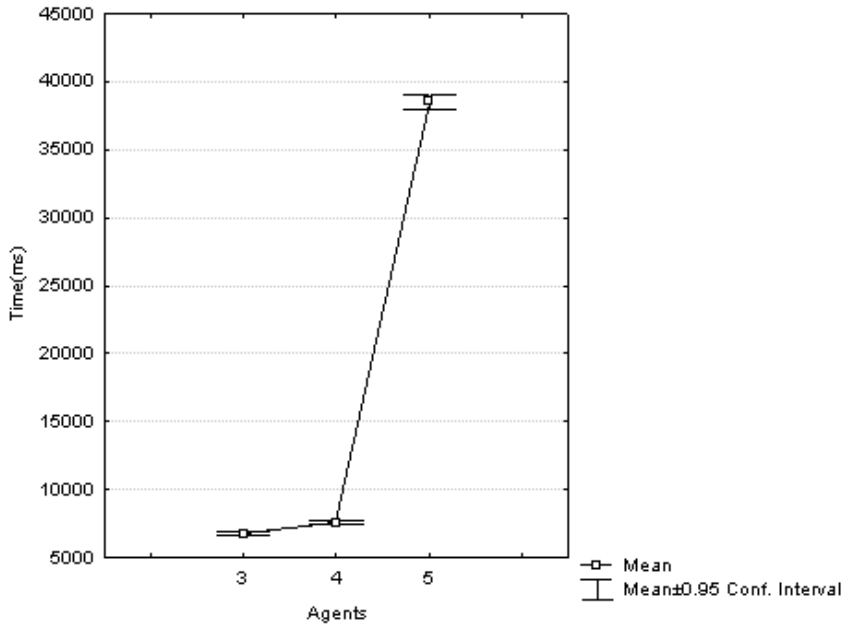

Figure 5.18: Time required for a number of Agents to complete 50 tasks.

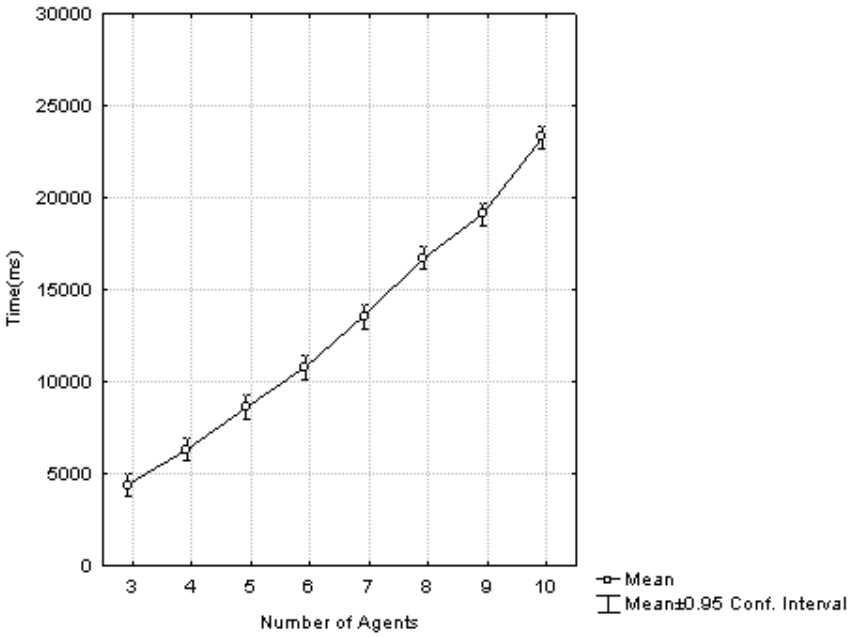

Figure 5.19: Time required for a number of Agents to complete 50 tasks without learning a decision model.

as a 1 while untrustworthy actions enter as a -1. Every time a new action occurs, the action is inserted into the first model position, shifting the previous actions over.

Table 5.4: Normalized average rewards and the average difference between agent rewards.

	Average Reward	Average Reward Difference Between Agents
TI-POMDP	1.0	0.29
Trust Vector	0.83	0.20

During the shift, a decay function is used to reduce the impact of previous actions on the current model. For this testing, the decay function reduces the weight of previous actions by five percent each time a new action is inserted. The five percent decay rate is consistent with the five percent reward decay used by the TI-POMDP. The Trust Vector model is used because it handles the trust modeling in a very different manner from the TI-POMDP which already relies on experience modeling. Additionally, the Defender simulation does not have the communication network required for reputation modeling.

The first test uses the CTG. Agents must locate the tiger and the reward and decide whether to cooperate with one another, betray each other, try to redeem themselves, or reset the game so the other agent cannot betray them. Table 5.4 shows the normalized reward levels obtained by two trust models and the typical difference in rewards achieved by the two agents during the same game. If the higher scoring agent achieved 100 points using the TI-POMDP algorithm, the other agent's score was twenty-nine points lower. On average, TI-POMDP agents scored higher than trust vector agents. The TI-POMDP algorithm reduced the number of reset and redemption occurrences at lower reward levels which allowed more cooperation and higher scores. The vector trust had a large increase of resets because the memory function would make the agent suspicious. Table 5.5 shows the percentage of time the agents take a particular action for both of the trust models.

The second test uses the Defender simulation. Both trust models are tested with 3, 4, and 5 Agents, three separate reward levels (1, 2, and 3) and three separate

Table 5.5: Percentage of agent actions for different trust algorithms.

Agent Action	TI-POMDP	Trust Vector
Cooperate	79.5	70.2
Betray	7.4	5.3
Redeem	10.1	1.6
Reset	3.0	22.9

levels of corruptions/redemption (0.1, 0.3, and 0.5). The models are also tested with the three different Agent mixes. The number of times Agents choose to cooperate or betray each other are tracked. The TI-POMDP is tested with an action horizon of 1 and 5.

The "cooperate" and "betray" actions are the high risk/high reward choices for the domain. Even though "betray" actions are considered bad, successfully "betraying" other "cooperating" Agents results in a large reward indicating that the "betraying" Agent made a good decision.

Table 5.6 illustrates the average number of times Agents choose to "cooperate" or "betray" over the course of 50 Attackers. Since each Attacker requires multiple actions before it is destroyed, Agents can take hundreds of total actions for 50 Attackers. Overall, the TI-POMDP model with a horizon of 5 achieves higher "cooperate" and "betray" success rates than the Trust Vector model. The Trust Vector and the TI-POMDP with a horizon of 1 decide to "cooperate" and "betray" more, but are less successful, which increases the number of required actions. As expected, the average number of times Agents "cooperate" decreases while "betrays" increases as the probability of changing trustworthiness (Corruption/Redemption Rate) increases. This is due to the fact that a corrupted Agent can immediately attempt to 'betray" while a redeemed Agent must first regain the trust of others before it can "cooperate."

Table 5.6: Number of times Agents choose to "cooperate" or "betray" using the TI-POMDP and Trust Vector models. The success rates indicate the percentage of "cooperates" and "betrays" that achieve the expected reward.

Model	Action	Corruption/Redemption Rate		
		0 .1	0 .3	0 .5
TI-POMDP Horizon 5	Cooperations	94 .1	91.3	85.0
	Cooperation Success Rate	83 .5	70.0	61.0
	Betrays	15 .9	31.4	40.0
	Betrayal Success Rate	79 .9	70.0	67.0
TI-POMDP Horizon 1	Cooperations	103.6	97.4	90.7
	Cooperation Success Rate	70 .2	58.6	47.9
	Betrays	19 .3	42.5	57.0
	Betrayal Success Rate	68 .7	60.2	55.8
Trust Vector	Cooperations	103.4	79.5	68.6
	Cooperation Success Rate	70 .7	63.9	57.1
	Betrays	42 .0	62.8	74.8
	Betrayal Success Rate	63 .4	60.5	58.6

Figures 5.20 and 5.21 show that successful "cooperate" actions decrease and successful "betray" actions increase as the corruption/redemption rate increases. The drop in "cooperate" success is a result of fewer "cooperate" attempts and the higher probability that an Agent immediately becomes corrupt after redeeming itself. The "betray" success increase comes from the larger number of "betray" attempts that occur as the corruption/redemption rate rises.

Table 5.7 illustrates the impact of the number of Agents, the number of reward levels, and the Agent complexity has on the success of "cooperate" and "betray" actions. While none of the factors have an affect on the "betray" success rate, two factors affect the "cooperate" success rate. "Cooperate" success decreases as the number of Agents increases. The decrease is due to larger numbers of Agents being

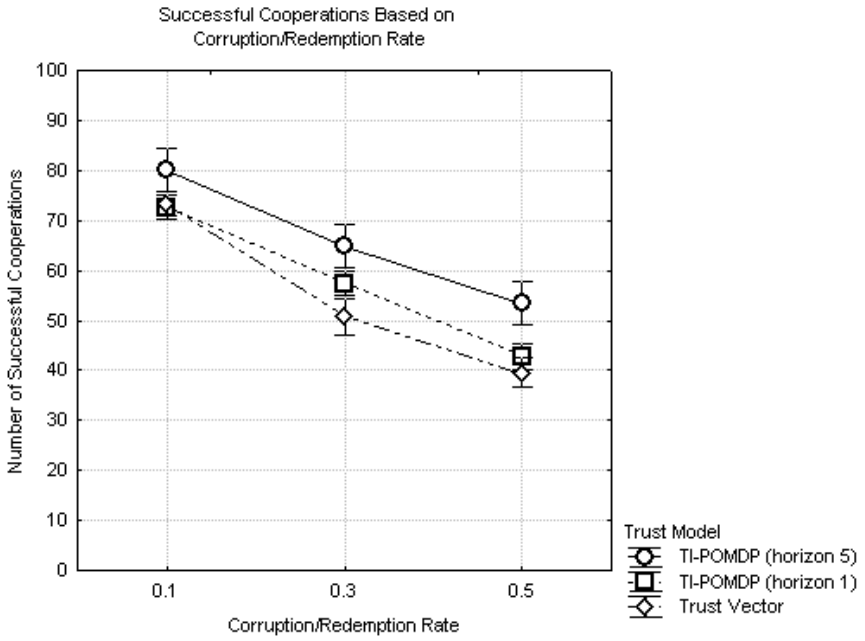

Figure 5.20: The effect corruption/redemption rate has on "cooperate" success.

assigned to tasks which increases the probability that one of them is untrustworthy. Increasing the number of reward levels improves the number of "cooperate" successes. This increase is due to untrustworthy Agents choosing to "cooperate" when the reward level is low. The sharp decrease in "cooperate" success between Agent mix 1 (homogeneous set of Agents) and 2 (1 group of Agents is 10 percent more likely to be corrupted than the other group of Agents) for the TI-POMDP appears to be a data anomaly as that trend is not present in the rest of the test.

Table 5.8 shows the average number of each action chosen by the Agents. While all models have a large number of Agents working alone and concealing their actions, these individual behaviors account for nearly 66 percent of the Trust Vector and TI-POMDP horizon 1 actions versus 56 percent of the TI-POMDP horizon 5 actions. The lack of cooperation is the driving factor behind the Trust Vector requiring 1.5 times the number of actions as the TI-POMDP.

Table 5.9 shows the average reward of the Agents using the two models for each corruption/redemption rate. The TI-POMDP achieves significantly higher rewards

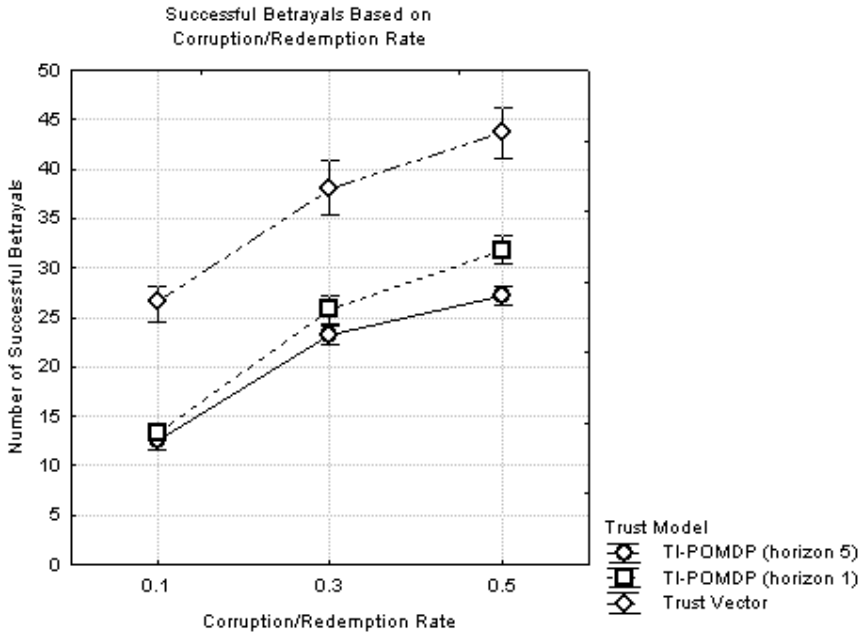

Figure 5.21: The effect corruption/redemption rate has on "betray" success.

due to the increased success of "cooperate" and "betray" actions (positive rewards instead of -100 rewards) and less use of "work alone" and "conceal" actions (-1 rewards). While the Trust Vector and the TI-POMDP with a horizon of 1 "cooperate" and "betray" more, those actions are less successful which reduces the overall reward.

In both the CTG and the Defender simulation, the TI-POMDP outperformed the trust vector with a decay function of five percent. The TI-POMDP requires 36.2 percent fewer actions than the trust vector model to accomplish the 50 tasks. This reduction in actions results in a 3.8 times higher reward.

5.4 Summary

The TI-POMDP testing focuses on how a specific domain influences the decision model's output, the impact of environment size on the computation time of the model, and a direct comparison of the TI-POMDP with a trust vector model. The factorial analysis shows that factors influencing the reward, either directly (reward level) or indirectly (number of Agents, corruption and redemption rates) impact the decisions

Table 5.7: Impact of simulation variables on "cooperate" and "betray" success for each corruption/redemption rate.

Model	Factor	Action	Factor Setting		
			1	2	3
TI-POMDP Horizon 5	Number of Agents	Cooperate	87.4	62.0	48.9
		Betray	20.9	21.7	20.3
	Reward Levels	Cooperate	41.7	69.5	87.0
		Betray	15.8	24.6	22.6
	Agent Complexity	Cooperate	75.4	55.4	67.5
		Betray	20.4	21.1	21.5
TI-POMDP Horizon 1	Number of Agents	Cooperate	93.2	81.7	77.6
		Betray	22.8	27.9	24.5
	Reward Levels	Cooperate	83.2	87.5	89.1
		Betray	21.5	22.9	21.4
	Agent Complexity	Cooperate	83.5	79.2	84.6
		Betray	22.3	24.7	27.1
Trust Vector	Number of Agents	Cooperate	70.1	64.2	54.2
		Betray	58.3	60.2	62.9
	Reward Levels	Cooperate	52.5	59.3	70.2
		Betray	59.0	61.6	62.2
	Agent Complexity	Cooperate	56.4	60.1	63.3
		Betray	57.3	61.9	64.7

and output of the TI-POMDP. Performance testing demonstrates that increases in environment size (number of Agents) exponentially increases the time required to learn decision policies and linearly increases the decision execution time. Comparison testing indicates that the TI-POMDP outperforms the trust vector method with a decay function of five percent in both the Defender simulation and the CTG.

Table 5.8: Average number of times agents choose each action.

Action	TI-POMDP Horizon 5	TI-POMDP Horizon 1	Trust Vector
Cooperate	90.1	97.2	88.8
Work Alone	91.8	127.3	194.2
Betray	29.1	39.6	55.7
Conceal	110.9	157.2	180.4
Redeem	39.4	38.3	46.8

Table 5.9: The average reward of the Agents using each model based on Corruption/Redemption Rate.

Model	Corruption/ Redemption Rate		
	0.1	0.3	0.5
TI-POMDP(Horizon 5)	1667.0	1111.7	688.0
TI-POMDP(Horizon 1)	-57.3	124.5	218.8
Trust Vector	-136.4	73.6	143.8

VI. Conclusions

A cooperative system allows a group of agents to perform complex tasks while working toward the common good. The agents decide to utilize the talents of other agents by relying on their willingness to cooperate which is a product of mutual trust between agents. Modeling the interaction of trust and agent decisions provides an analysis of the effectiveness of the cooperative system.

This research merges the high level decision reasoning of a multi-agent environment with the individual oversight of trust modeling. The end result is a decision framework that allows unique agents to evaluate the trustworthiness of one another. The agents determine whether to work with other agents that may have different capabilities and different trust models. This investigation does not examine the impact of an agent operating completely outside the framework or a group of agents working together to subvert the ability of the framework.

The TI-POMDP framework outlined in Chapter III accomplishes the first research objective in Chapter I. It models a complete multi-agent environment by adding a trust model to the I-POMDP decision process. The Agents analyze their current state, estimate the effect of their potential decisions, and select the decision that is most likely to have the greatest future benefit. A key component of the entire decision process is the Agent's level of trust regarding the other Agents it interacts with. This trust model helps determine the current state belief and controls the reward function of the Agent.

The Defender simulation in Chapter IV fulfills the second research objective in Chapter I. It creates a cooperative environment where Agents eliminate threats and obtain individual rewards through cooperation and betrayal. The simulation uses the TI-POMDP track the Agent's trust ratings and makes action decisions based on those ratings. Ultimately, an Agent's success in this environment is dependent on the accuracy of its trust model of the other Agents.

6.1 Research Contributions

This research presents a novel model for cooperative multi-agent environments where agents are potentially corrupt. It combines traditional trust modeling in Section 2.3 with a multi-agent decision process in Section 2.2 to model the interactions of a group of agents.

6.2 Research Conclusions

Simulation testing described in Chapter V includes factor analysis, performance testing, and comparison testing and illustrates the ability of the TI-POMDP to effectively reason within the simulation domain. The simulation factor analysis shows that the number of Agents in the environment and the environment corruption rate negatively impact Agent rewards and "cooperation" ability, while rewards levels and redemption rate have a positive impact. Comparison testing between the TI-POMDP and a trust vector model with five percent decay satisfies the third research objective in Chapter I. Agents using the TI-POMDP model achieved 3.8 times the average reward of Agents using a trust vector model.

The simulation demonstrates the TI-POMDP's ability to allow Agents to make decisions based on their level of trust and model of the environment. The Agents continuously update their trust models and state beliefs to select what is predicted to be the most beneficial action at a given time. The combination of trust modeling and decision making enables the group of Agents to "cooperate" with one another to achieve higher collective rewards.

6.3 Future Work

The future research with respect to the TI-POMDP framework can be divided into two basic categories, improving the individual components of the framework and finding the exploitable areas of the trust model. Both categories attempt to improve the decision quality of the TI-POMDP.

- Policy Learning: the behavioral equivalence [16] policy learning method used in this thesis is effective for small problems, but it did not scale well. Finding a better solution would allow this simulation to tackle more complex trust problems.
- Differing Agent Trust Models: the simulation tested uses identical trust models for all of the Agents. If one Agent's trust model is based on a trust vector representation [17] while another uses a reputation model [18] the strengths and weaknesses of the two models may affect the output of the simulation.
- Dynamic Trust Models: the Agents currently use a single trust model in their decision process. Utilizing multiple trust models, similar to adaptive trust modeling [10], could help an Agent make better decisions.
- TI-POMDP exploitation: the assumptions made by the simulation limit the environment. Expanding what the environment allows can significantly impact the performance of the TI-POMDP. The specific limitations include allowing Agents to influence tasks they are not assigned to, allowing rogue Agents to team up and cooperate together to drive up trust, moving the task assignment to Agents which allows false tasks, and allowing untrustworthy Agents to "cooperate" versus just "betray" or "conceal." Exploring the impact of these factors can give greater insight into the ability of the TI-POMDP to reason over more complex domains.

6.4 Final Remarks

Groups of autonomous agents can be leveraged to complete complex tasks beyond the individual agent capabilities. Success of these interactions requires coordination and cooperation between the agents, which implies a level of trust between them. Without accurate trust models an autonomous agent may be susceptible to exploitation, limiting their utility.

Bibliography

1. Azzedin, Farag, Ahmad Ridha, and Ali Rizvi. "Fuzzy Trust for Peer-to-Peer Based Systems". *Proceedings of World Academy of Science, Engineering and Technology*, 21:123–127, 2007.

2. Barber, K. Suzanne and Joonoo Kim. "Belief Revision Process Based on Trust: Agents Evaluating Reputation of Information Sources". *Proceedings of the workshop on Deception, Fraud, and Trust in Agent Societies held during the Autonomous Agents Conference*, 73–82. 2001.

3. Bernstein, Daniel S., Robert Givan, Neil Immerman, and Shlomo Zilberstein. "The Complexity of Decentralized Control of Markov Decision Processes." *Mathematics of Operations Research*, volume 27, 819–840. 2002.

4. Boutilier, Craig. "Planning, Learning and Coordination in Multiagent Decision Processes". *Theoretical Aspects of Rationality and Knowledge*, 195–210. 1996.

5. Brooks, Rodney A. "A Robust Layered Control System for a Mobile Robot". *IEEE Journal of Robotics and Automation*, 2:14–23, 1986.

6. Damer, Steven and Maria Gini. "Achieving Cooperation in a Minimally Constrained Environment". *Proceedings of the Twenty-Third Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, 57–62. 2008.

7. Doshi, Prashant. "A Framework for Optimal Sequential Planning in Multiagent Settings." *Association for the Advancement of Artificial Intelligence*, 985–986. 2004.

8. Doshi, Prashant and Piotr J. Gmytrasiewicz. "A Particle Filtering Based Approach to Approximating Interactive POMDPs". *Association for the Advancement of Artificial Intelligence*, 969–974. 2005.

9. Efron, Bradley. "Bootstrap Methods: Another Look at the Jackknife". *Annals of Statistics*, 7:1–26, 1979.

10. Fullam, Karen K. *Adaptive Trust Modeling in Multi-agent Systems: Utilizing Experience and Reputation.* Ph.D. thesis, 2007. Adviser-Barber, Suzanne.

11. Fullam, Karen K., Tomas B. Klos, Guillaume Muller, Jordi Sabater, Andreas Schlosser, Zvi Topol, K. Suzanne Barber, Jeffrey S. Rosenschein, Laurent Vercouter, and Marco Voss. "A Specification of the Agent Reputation and Trust (ART) Testbed: Experimentation and Competition for Trust in Agent Societies". *Autonomous Agents and Multiagent Systems '05: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, 512–518. 2005.

12. Hansen, Eric A., Daniel S. Bernstein, and Shlomo Zilberstein. "Dynamic Programming for Partially Observable Stochastic Games". *Association for the Advancement of Artificial Intelligence*, 709–715. 2004.

13. Kaelbling, Leslie P., Michael L. Littman, and Anthony R. Cassandra. "Planning and Acting in Partially Observable Stochastic Domains". *Artificial Intelligence*, 101:99–134, 1998.

14. Montgomery, Douglas C. *Design and Analysis of Experiments*. 2004.

15. R., Falcone, Pezzulo G., and Castelfranchi C. "Quantifying Belief Credibility for Trust-based Decision". *Proceedings of the Autonomous Agents and Multi-Agent Systems-02 Workshop on "Deception, Fraud and Trust in Agent Societies"*, 41–48. 2002.

16. Rathnasabapathy, Bharaneedharan, Prashant Doshi, and Piotr J. Gmytrasiewicz. "Exact Solutions of Interactive POMDPs Using Behavioral Equivalence". *Autonomous Agents and Multiagent Systems*, 1025–1032. 2006.

17. Ray, Indrajit and Sudip Chakraborty. "A Vector Model of Trust for Developing Trustworthy Systems." *European Symposium on Research in Computer Security*, volume 3193 of *Lecture Notes in Computer Science*, 260–275. 2004.

18. Rettinger, Achim, Matthias Nickles, and Volker Tresp. "Learning Initial Trust Among Interacting Agents". *Cooperative Information Agents*, volume 4676 of *Lecture Notes in Computer Science*, 313–327. 2007.

19. Roy, Nicholas, Geoffrey J. Gordon, and Sebastian Thrun. "Finding Approximate POMDP solutions Through Belief Compression". *Journal of Artificial Intelligence Research*, 23:1–40, 2005.

20. Russell, Stuart Jonathan and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 2003.

21. Seuken, Sven and Shlomo Zilberstein. "Formal Models and Algorithms for Decentralized Decision Making Under Uncertainty". *Journal of Autonomous Agents and Multiagent Systems*, 17:190–250, 2008.

22. Song, Weihua, Vir V. Phoha, and Xin Xu. "An Adaptive Recommendation Trust Model in Multiagent System". *Intelligent Agent Technology*, 462–465. 2004.

23. Szer, Daniel, Franois Charpillet, and Shlomo Zilberstein. "MAA*: A Heuristic Search Algorithm for Solving Decentralized POMDPs". *Uncertainty in Artificial Intelligence*, 576–590. 2005.

24. Wang, Yonghong and Munindar P. Singh. "Trust Representation and Aggregation in a Distributed Agent System". *Proceedings of the Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference*, 16–20. 2006.

25. Wong, H. Chi and Katia P. Sycara. "Adding Security and Trust to Multiagent Systems". *Applied Artificial Intelligence*, 14:927–941, 2000.

26. Wu, Chao, Ming Jun Xin, and Wei Hua Li. "An Approach to Dynamic Model Combination on Solving Decision-Making Problem". *Proceedings of the 2006 Institute of Electrical and Electronics Engineers Asia-Pacific Conference on Services Computing*, 629–634. 2006.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
18–06–2009	Master's Thesis	June 2007 — June 2009

4. TITLE AND SUBTITLE

The Trust-based Interactive Partially Observable Markov Decision Process

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Richard S. Seymour, Capt, USAF

5d. PROJECT NUMBER

08-221

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCS/ENG/09-09

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RYTC (Juan M. Carbonell)
2241 Avionics Circle
Area B, Bldg 620
Wright Patterson Air Force Base, OH 45433
(937) 255–4709, ext 3157 juan.carbonell@wpafb.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RYTC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Cooperative agent systems are designed so that each is working toward the same common good. The problem is that software systems are complex and can be subverted by an adversary to either break the system or potentially worse, create sneaky agents who are willing to cooperate when the stakes are low and take selfish, greedy actions when the rewards rise. This research focuses on the ability of a group of agents to reason about the trustworthiness of each other and make optimal decisions about whether to cooperate. A TI-POMDP is developed to model the trust interactions between agents, enabling the agents to select the best course of action from the current state. The TI-POMDP is a novel approach to multiagent cooperation based on an I-POMDP augmented with trust relationships. Experiments demonstrate the TI-POMDP's ability to accurately track the trust levels of agents with hidden agendas, providing the agents the information needed to make decisions based on their level of trust and model of the environment. On average, agents achieved rewards 3.8 times higher using the TI-POMDP model compared to a trust vector model.

15. SUBJECT TERMS

multiagent system, trust modeling, partially observable Markov decision process

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Gilbert Peterson, AFIT/ENG
U	U	U	UU	101	19b. TELEPHONE NUMBER *(include area code)* (937)255–6565,ext4281;gilbert.peterson@afit.edu