A Trust-based Multiagent System

Richard Seymour and Dr Gilbert L. Peterson
Air Force Institute of Technology
Department of Electrical and Computer Engineering
Air Force Institute of Technology, 2950 Hobson Way
Wright-Patterson AFB, OH 45433, United States
Email: richard.seymour.1@us.af.mil and gilbert.peterson@afit.edu

***Abstract*—Cooperative agent systems often do not account for sneaky agents who are willing to cooperate when the stakes are low and take selfish, greedy actions when the rewards rise. Trust modeling often focuses on identifying the appropriate trust level for the other agents in the environment and then using these levels to determine how to interact with each agent. Adding trust to an interactive partially observable Markov decision process (I-POMDP) allows trust levels to be continuously monitored and corrected enabling agents to make better decisions. The addition of trust modeling increases the decision process calculations, and solves more complex trust problems that are representative of the human world. The modified I-POMDP reward function and belief models can be used to accurately track the trust levels of agents with hidden agendas. Testing demonstrates that agents quickly identify the hidden trust levels to mitigate the impact of a deceitful agent.**

I. Introduction

The concept of trust is central to agent interactions 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 [1]–[3]. 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^{TM} 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 plays 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 around it. 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. This paper extends the traditional I-POMDP framework to fully incorporate trust modeling. The enhanced trust modeling allows the agents to quickly recognize and adapt to behavior changes to maximize their performance.

The addition of trust modeling into the I-POMDP dynamically alters an agent's reward function and indirectly alters the other agent's belief models concerning an agent. Within an I-POMDP environment, an agent's actions are governed by its reward function. A trustworthy agent performs cooperative actions that achieve the highest collective reward while an untrustworthy agent subverts the collective good to achieve higher personal rewards. Each agent maintains belief models that are expanded to include the estimated trust level of the agents it interacts with. When the agents do not act in accordance with their model, their trust rating is changed affecting future interactions between agents. Testing of a trust-based I-POMDP simulation illustrates that the framework quickly identifies and reacts to hidden trust levels preventing additional betrayal. The testing includes a direct comparison between the trust-based I-POMDP and another trust model.

II. Trust

In a cooperative environment, autonomous agents require an implicit level of trust to work together. An agent chooses to cooperate if it anticipates that it will receive the highest expected reward by working with another agent. If one agent does not trust another agent, the prospect of a reduced expected reward causes that agent to avoid cooperating. If trust completely breaks down within the system, all agents may choose to work independently resulting in cooperative tasks not being accomplished.

The typical obstacle with trust modeling is an agent's ability to determine the appropriate level of trust for other agents within the environment. Quickly and accurately determining the correct trust rating allows the agent to maximize its expected reward and minimize the damage caused by a deceitful agent. Failure to identify the proper trust rating results in reduced task accomplishment and lower individual rewards. Several techniques have been used to establish trust ratings.

One common approach builds a network of trusted agents [3], [4]. 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 into

a single trust rating. If one of polled agents does not have a recommendation about the unknown agent, it will poll its own trust network for recommendations. While this method is not demonstrated in this paper, it is a useful trust rating system 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.

A second approach uses a series of nonbinding interactions between agents to determine trust [1]. The agents communicate their intentions to one another prior to acting. This technique mimics the human ability to get a feeling for whether or not to trust a new acquaintance. This paper utilizes nonbinding interactions to help determine when agent trust levels fluctuate.

Trust vectors [5] model complex domains by tracking multiple trust values for a given agent. The 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. If an agent is trustworthy on cleaning tasks but deceitful on purchasing tasks, the other agents can identify these differences and choose to cooperate on future cleaning tasks. A trust vector can also contain a history of an agent's actions with a decay rate to reduce the impact of actions further in the past. This paper utilizes trust vectors for comparison testing against our algorithm.

Trust ratings based on fuzzy sets [6] 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.

Experience based models [7] 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.

Adaptive trust modeling [8] 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.

All of the trust techniques use the current trust value in the decision process. This neglects the possibility that an agent cooperates on small tasks to build a high trust rating and takes a greedy approach when the stakes are higher. In a dynamic trust environment, trust values can fluctuate due to adversary hacking, software/hardware error, greed, or some other reason. If trust values were to change, the same techniques can be reused to evaluate the new trust level, but the agent must quickly identify the change in trust. Failure to identify the change leaves the agent open to exploitation by the other agents.

III. Multi-agent Domains

Multi-agent environments allow a number of autonomous agents the opportunity to achieve an expanded set of goals through cooperation. While a single agent may not possess all of the requisite skills to perform a complex task, a group of agents working together can accomplish it. Task accomplishment requires some level of coordination between the agents to ensure each agent performs its portion of the overall task.

A partially observable Markov decision process (POMDP) [9] allows a single agent to cope with uncertainty about its current state while operating in a stochastic environment. Several methods, including decentralized POMDPs (DEC-POMDP) [10] and I-POMDPs [11], extend this model to multi-agent environments by tying a series of individual POMDPs together. The DEC-POMDP utilizes a single group reward for all of the agents which works well in a cooperative environment. The I-POMDP uses individual reward functions for each agent which are required in a trust modeling domain.

An I-POMDP, consists of the tuple

$$\langle IS_i, A, T_i, \Omega_i, O_i, R_i \rangle \tag{1}$$

for each agent i within the environment, where IS_i is the set of interactive states $S \times M_j$, S is the set of environment states, and M_j is the set of models of agent j. Each model m_j consists of the pair $\langle f_j, h_j \rangle$ 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.

An agent's state belief is a distribution over S. 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, which is:

$$b_i^t(s^t) = \beta O_i(o_i^t, s^t, a_i^{t-1}) \sum_{s^{t-1} \epsilon S} b_i^{t-1}(s^{t-1}) T_i(s^t, a_i^t, s^{t-1}) \tag{2}$$

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.

IV. Trust-based I-POMDP

The trust-based I-POMDP (TI-POMDP) is a modified version of the I-POMDP. The TI-POMDP maintains the basic components of the I-POMDP, and adds 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.

Each agent maintains a trust belief model, 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 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.

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 $IS_i \times A \rightarrow R$ which is the expected reward agent i receives from taking action a in state is.

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 and observing the changed environment, an agent updates its state belief model before attempting to decide on its next action.

The state belief update 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.

The trust model, τ, is a component of the state belief, but only is updated based on the observations prior to the state belief update. The agent decides on a current trust model and then creates its current state belief distribution. The new trust model is based on the trust model from the previous state belief. The transition to the new trust model occurs as the agent incorporates its current observations. The current observations include information about the actions taken by the other agents in the environment. An agent evaluates this action information to determine whether the other agent's actions were trustworthy or deceitful and updates its ratings of that agent accordingly.

The complexity of τ_i depends on the domain requirements. In the simple case, τ_i can be a single binary number representing whether agent i is trustworthy or an integer representing what level agent i attempts to betray. In the more complex case, τ_i can be a series of trust rankings corresponding to different types of tasks or dimensions within the domain such as a fuzzy set or a trust vector.

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 or reputation based). 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. 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.

In an environment with trust modeling an agent's reward function is a direct product of its trust rating. A trustworthy agent values cooperative tasks while an agent that is being untrustworthy values tasks that undermine cooperation. Within the I-POMDP framework, the reward function for an agent with multiple potential trust levels can be thought of as two or more separate reward functions. Each individual function directly corresponds to a specific trust rating for the agent.

The simplest case occurs when an agent can be either completely trustworthy or completely deceitful. The agent appears to have two reward functions that become inverses of one another for interactive states that are identical other than the agent's trust value. For instance, if agent a has the option of helping a trusted agent b move an object, the trustworthy agent a decides to move the object while the deceitful agent a decides not to move the object.

In a more complex case, agent a can appear to have a series of reward functions due to a larger range of trust ratings. Scenarios where an agent has multidimensional trust ratings [1] also increase the reward function complexity. Multidimensional trust occurs when an agent is trustworthy in some aspects within the environment, but not trustworthy in others. Ultimately, what appears to be two or more reward functions is actually one large reward function where the interactive state depends on the trust rating for the agent.

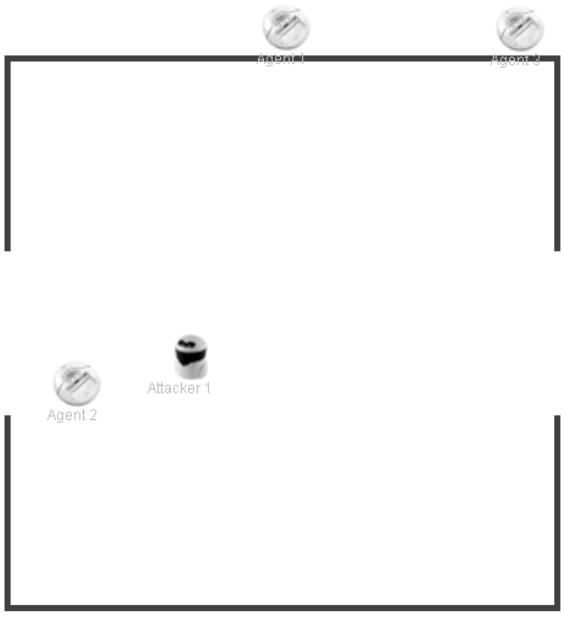

Fig. 1. An image of the Defender Simulation with three Agents and one Attacker.

V. The TI-POMDP Simulation

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 and the amount of trust it has in the other agents. The defense scenario was 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 in this paper, future work can utilize this large domain.

After each task is completed, the agents gain insight into 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.

The simulation is a Java based application with a visual depiction of the agents operating in the environment. The environment consists of a simple building with two doors as shown in Figure 1.

Figure 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 listed below.

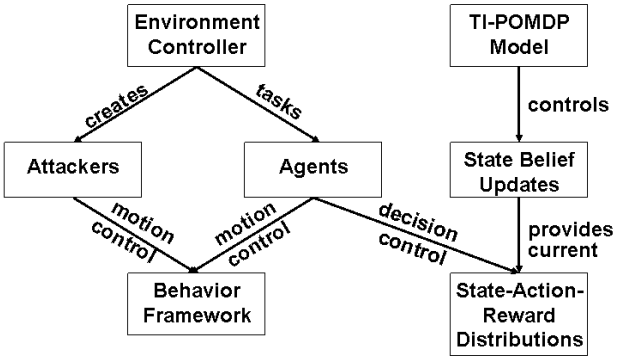

Fig. 2. The major components in the trust simulation.

A. Environment Controller

An Environment Controller initially creates and randomly distributes the Agents in the space. During the simulation, the Environment Controller creates and randomly distributes a 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. 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. These 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.

B. 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.

C. Agents

Agents have two mutually exclusive tasks, patrolling the environment and defeating Attackers. The patrolling behavior consists of randomly wandering the environment. Once assigned, 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 [12]. 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 engaging an Attacker decreases the Attacker's strength by 25 points every time it chooses an action. When an Agent decides to betray another Agent, it adds 15 points to the Attacker's strength. 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.

D. TI-POMDP

The TI-POMDP framework is updated when a task is first identified and Agents are assigned to it. Each Agent uses the framework to decide which action to take. After action execution, the TI-POMDP framework is updated based on the observations. 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."

- Cooperating Agents 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.
- Agents working alone always 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.
- Betraying Agents always add 15 points to the Attacker's strength.
- Concealing Agents 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.
- Redeeming Agents 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).

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 range from 0.05 to 0.5 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 assigned

Agents. 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.

- 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 reward level that it believes every other Agent is trustworthy to. 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 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 to change. Agent 2's betrayal level changes due to random corruption or redemption.

To understand the Agent's decision process start by assuming they only look at the immediate action and deciding whether to cooperate, each individual 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 and that they trust it. Otherwise, an untrustworthy Agent will "conceal" its actions. If the Agent chooses to "conceal" it actions, it does not help with the task, but it does not hinder it either. The other Agents

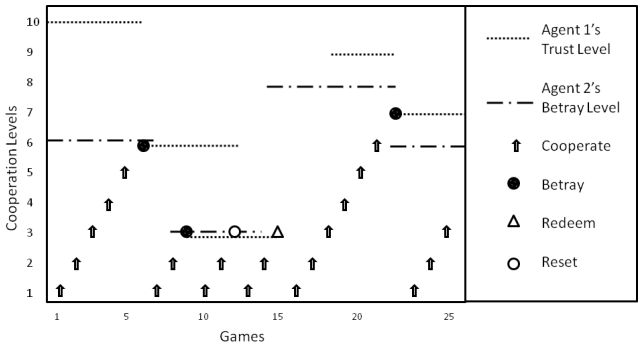

Fig. 3. The trust interactions between agents.

can not differentiate between "conceal" and a "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 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, resulting in the other Agents 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. Agents not assigned to the task have more difficultly updating their models. 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 a situations where one Agent incorrectly believes another Agent observed a betrayal which skews the first Agent's future expectation.

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.

VI. TI-POMDP Testing

A comparison test is used to measure the difference between the TI-POMDP model and a Trust Vector model. The Trust Vector model uses a history decay function that reduces the

TABLE I
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	Cooperations	94.1	91.3	85.0
	Success Rate	83.5	70.0	61.0
	Betrays	15.9	31.4	40.0
	Success Rate	79.9	70.0	67.0
Trust Vector	Cooperations	43.6	38.9	33.9
	Success Rate	73.0	57.0	48.0
	Betrays	55.9	69.4	76.0
	Success Rate	13.0	17.0	17.0

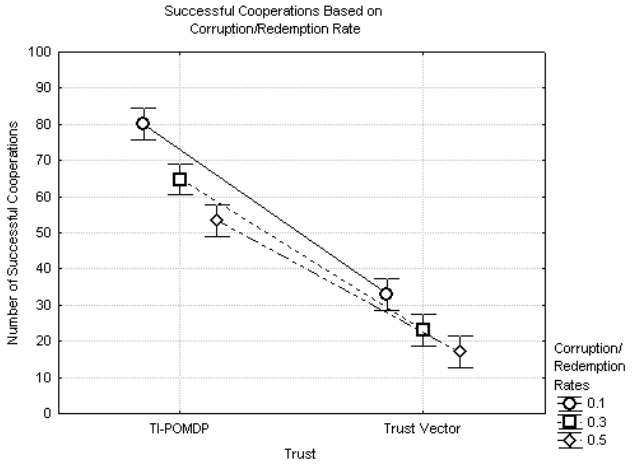

Fig. 4. The effect corruption/redemption rate has on "cooperate" success.

impact of previous actions by five percent each time step. The Trust Vector model is used because it handles the trust modeling in a very different manner from the TI-POMDP. The TI-POMDP already relies on experience modeling and the domain does not have the communication network required for reputation modeling.

Both models are tested with 3, 4, and 5 Agents, three separate reward levels (1, 2, and 3) and three separate levels of corruptions/redemption (0.1, 0.3, and 0.5). The models are also tested with different mixes of Agents. The first mix is a homogeneous group of Agents. The second mix has two types of Agents, but the individual tasks may only require a single type of Agent. The final mix of Agents has two types of Agents and the individual tasks require representatives from both types. The number of times Agents choose to cooperate or betray each other are tracked.

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 I 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 achieves higher "cooperate" rates than the Trust Vector model. 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."

Figures 4 and 5 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 II shows the number of times Agents choose to "cooperate" or "betray" at each corruption/redemption rate.

TABLE II
AVERAGE NUMBER OF TIMES AGENTS CHOOSE "COOPERATE" OR "BETRAY" FOR EACH CORRUPTION/REDEMPTION RATE.

Model	Action	Corruption/ Redemption Rate		
		0 .1	0 .3	0 .5
TI-POMDP	Cooperate	94.1	91.3	85.0
	Betray	15.9	31.4	40.0
Trust Vector	Cooperate	43.6	38.9	33.9
	Betray	55.9	69.4	76.0

Table III illustrates the impact of the number of Agents, the number of reward levels, and the mix of Agents 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 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" on when the reward level is low. The sharp decrease in "cooperate" success between Agent mix 1 and 2 for the TI-POMDP appears to be a data anomaly as that trend is not present in the rest of the test.

Table IV shows the average number of each action chosen by the Agents. While both models have a large number of Agents working alone and concealing their actions, these individual behaviors account for nearly 80 percent of the Trust Vector actions versus 56 percent of the TI-POMDP actions. The lack of cooperation is the driving factor behind the Trust Vector requiring 1.4 times the number of actions as the TI-POMDP.

Table V shows the average reward of the Agents using

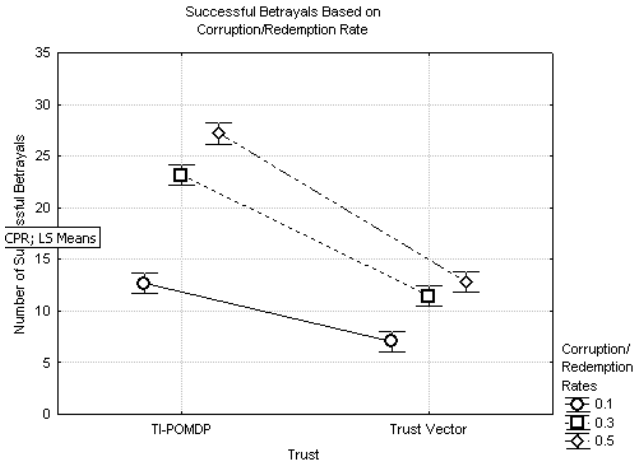

Fig. 5. The effect corruption/redemption rate has on "betray" success.

TABLE III
IMPACT OF SIMULATION VARIABLES ON "COOPERATE" AND "BETRAY" SUCCESS FOR EACH CORRUPTION/REDEMPTION RATE.

Model	Factor	Action	Factor Setting		
			1	2	3
TI-POMDP	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
	Mix of Agents	Cooperate	75.4	55.4	67.5
		Betray	20.4	21.1	21.5
Trust Vector	Number of Agents	Cooperate	21.8	26.4	25.0
		Betray	7 .4	10.8	12.0
	Reward Levels	Cooperate	10.8	27.2	35.0
		Betray	8 .0	11.1	12.2
	Mix of Agents	Cooperate	23.2	25.0	24.8
		Betray	9 .6	11.5	10.3

TABLE IV
AVERAGE NUMBER OF TIMES AGENTS CHOOSE EACH ACTION.

Action	TI-POMDP	Trust Vector
Cooperate	90.1	38.8
Work Alone	91.8	175.8
Betray	29.1	67.1
Conceal	110.9	208.3
Redeem	39.4	11.0

the two models for each corruption/redemption rate. The TI-POMDP achieves significantly higher rewards 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).

VII. CONCLUSION

The addition of trust to multi-agent environments allows modeling of higher complexity interactions between agents. Sneaky agents further increase the complexity by adding extra uncertainty to the environment as a helpful agent can quickly become a hindering agent. The reward function and state representation make the TI-POMDP framework a suitable method to capture trust modeling. The Defender Simulation demonstrates the TI-POMDP's ability to react to a corrupt agent, mitigate the damage inflicted, and maintain a consistent level of cooperation within the system. Future work for this research is to expand the problem domain to a less observable environment to test an agent's ability to pinpoint the cause of betrayal in a noisy environment.

TABLE V
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	1667.0	1111.7	688.0
Trust Vector	-196.8	-267.0	293.9

ACKNOWLEDGMENT

The views expressed in this paper are those of the authors and do not endorse, or reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government. The authors thank representatives of the AFRL/RYTC for their professional interest and support of this research effort.

REFERENCES

[1] A. Rettinger, M. Nickles, and V. Tresp, "Learning initial trust among interacting agents." in *CIA*, ser. Lecture Notes in Computer Science, M. Klusch, K. V. Hindriks, M. P. Papazoglou, and L. Sterling, Eds., vol. 4676. Springer, 2007, pp. 313–327.

[2] H. C. Wong and K. P. Sycara, "Adding security and trust to multiagent systems." *Applied Artificial Intelligence*, vol. 14, no. 9, pp. 927–941, 2000.

[3] W. Song, V. V. Phoha, and X. Xu, "An adaptive recommendation trust model in multiagent system." in *IAT*. IEEE Computer Society, 2004, pp. 462–465.

[4] Y. Wang and M. P. Singh, "Trust representation and aggregation in a distributed agent system." in *AAAI*. AAAI Press, 2006.

[5] I. Ray and S. Chakraborty, "A vector model of trust for developing trustworthy systems." in *ESORICS*, ser. Lecture Notes in Computer Science, P. Samarati, P. Y. A. Ryan, D. Gollmann, and R. Molva, Eds., vol. 3193. Springer, 2004, pp. 260–275.

[6] F. Azzedin, A. Ridha, and A. Rizvi, "Fuzzy trust for peer-to-peer based systems." *Proc. WASET*, vol. 21, pp. 123–127, 2007.

[7] K. S. Barber and J. Kim, "Belief revision process based on trust: Agents evaluating reputation of information sources," in *Proceedings of the workshop on Deception, Fraud, and Trust in Agent Societies held during the Autonomous Agents Conference*. London, UK: Springer-Verlag, 2001, pp. 73–82.

[8] K. K. Fullam, "Adaptive trust modeling in multi-agent systems: utilizing experience and reputation," Ph.D. dissertation, Austin, TX, USA, 2007, adviser-Barber,, Suzanne.

[9] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," Tech. Rep. CS-96-08, 1996.

[10] D. S. Bernstein, S. Zilberstein, and N. Immerman, "The complexity of decentralized control of markov decision processes," pp. 32–37.

[11] P. Doshi, "A framework for optimal sequential planning in multiagent settings." in *AAAI*, D. L. McGuinness and G. Ferguson, Eds. AAAI Press / The MIT Press, 2004, pp. 985–986.

[12] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 10, 1986.