Responding to Sneaky Agents in Multi-agent Domains *

Richard S. Seymour and **Dr Gilbert L. Peterson**
Department of Electrical and Computer Engineering
Air Force Institute of Technology, 2950 Hobson Way
Wright-Patterson AFB, OH 45433, United States

Abstract

This paper extends the concept of trust modeling within a multi-agent environment. 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. However, this type of modeling does not account for sneaky agents who are willing to cooperate when the stakes are low and take selfish, greedy actions when the rewards rise. 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, but 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.

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.

The typical trust modeling problem treats trust as a hidden rating (Rettinger, Nickles, and Tresp 2007; Wong and Sycara 2000; Song, Phoha, and Xu 2004). 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. In initial testing, the I-POMDP implementation quickly identifies and reacts to hidden trust levels preventing additional betrayal.

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 accom-

*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 United States Air Force Research Laboratory for their professional interest and support of this research effort.

plishment 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) (Kaelbling, Littman, and Cassandra 1996) 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) (Bernstein et al. 2002) and I-POMDPs (Doshi 2004), 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.

Trust

In a cooperative environment, autonomous agents require an implicit level of trust to work together. An agent chooses to cooperate if it's reward function identifies that the highest expected reward will come from 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 each of the 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 (Wang and Singh 2006; Song, Phoha, and Xu 2004). 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 (Rettinger, Nickles, and Tresp 2007). 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 (Ray and Chakraborty 2004) 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 (Azzedin, Ridha, and Rizvi 2007) 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.

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.

Trust I-POMDP

Within the I-POMDP framework, an agent's trust directly affects the reward of the agent and the IS of the other agents. The combination of these two factors govern the actions of the agents within the environment. A trust rating τ for each agent determines which actions lead to higher rewards at the given time. Each agent keeps a τ estimate for the other agents within the environment to track which agents are trustworthy. The Trust I-POMDP (TI-POMDP) tuple becomes

$$\langle IS_i, A, T_i, \Omega_i, O_i, R_i, \tau_i, \Theta \rangle \quad (3)$$

where $A, T_i, \Omega_i, and O_i$, are not directly changed from the I-POMDP model.

τ_i is the trust ranking for agent i. 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.

Θ is the probability distribution that τ_i transitions to τ_i'. Θ is responsible for modeling the trust fluctuations within the environment which captures the changing motives of the individual agents.

IS_i includes the τ_i, the true trust level of i. IS_i is also an expanded set of interactive states where M_j is a tuple $\langle f_j, h_j, \tau_j \rangle$. f_j now maps the possible histories of j's observations and the possible trust value of j to its action.

R_i is now dependent on τ_i as an agent's reward is directly tied to its trust level.

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 (Rettinger, Nickles, and Tresp 2007) 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.

The other agents within the environment adjust their individual belief models based on the actions of a given agent. If an agent carries out actions consistent with a trusted agent the other agents will associate higher probabilities to the interactive states that trust that agent.

If agent a's current belief model trusts agent b, agent a assigns higher probabilities to the states with a high τ_b. After the next action, if agent b is where agent a expected, agent a continues to trust agent b. If agent b is not where agent a expected, agent a alters (in this case reduces) τ_b during its belief model update in Equation 2. Agent a's belief model now assigns higher probabilities to the states with a lower τ_b.

The τ update is simple in the binary case. An untrusted agent that does not take an expected action becomes trusted, similarly a trusted agent becomes untrusted. When τ can take on more values, the update becomes more difficult. If agent a's belief model has a τ_b of 0.5 and agent b's next action is unexpected, does agent b become more or less trusted and by how much? Agent a must try to judge agent b's intent.

To judge agent b's intent, agent a runs two additional belief model updates from the previous interactive state. The first update assumes agent b is trustworthy while the second update assumes agent b should not be trusted. Agent a attempts to determine the accuracy of both model updates by measuring difference of each model's expected observations with agent a's actual observations. If the first model update is more accurate than the original model and the second model, then τ_b is increased. If the second model update is more accurate, τ_b is decreased.

It is important to note that the addition of trust expands the I-POMDP problem space. A problem with two agents and three trust values has twenty-seven times the state space as the same two agent problem without trust because every state now must account for all of the combinations of trusted, not trusted, and neither for the two agents and estimations of the other agent's belief model about the first agent. In a problem with n agents and m trust settings, the state space expands by a factor of $m^{n+(n-1)^2}$. Ultimately, this is an exponential expansion to a problem that is already NEXP-complete (Seuken and Zilberstein 2008).

The regular I-POMDP is actually a special case of the TI-POMDP where each agent has a constant τ. This reduces the number of interactive states since each agent does not have states with a different τ values. The reduction in interactive states also reduces the size of the reward function.

Despite the I-POMDP problem complexity, several approximation techniques can provide timely solutions. Behavioral equivalence (Rathnasabapathy, Doshi, and Gmytrasiewicz 2006) collapses states into a manageable search space. Particle filtering (Doshi and Gmytrasiewicz 2005) uses particles to represent possible interactive states and

carries a subset of particles forward in time. A modified A^* search (Szer, Charpillet, and Zilberstein 2005) and dynamic programming (Hansen, Bernstein, and Zilberstein 2004) have been used to quickly prune dominated branches of the search tree. In addition, problem domain dimensionality can be reduced using principal component analysis (Roy, Gordon, and Thrun 2004). The reduced problem dimensionality of this paper did not require utilization of an approximation technique.

The Cooperative Tiger Game

To illustrate the nuances of the sneaky agent problem, the TI-POMDP is demonstrated on a modified version of the Tiger Game introduced by Kaelbling (1996) and expanded into a multiagent game by Doshi (Doshi 2004). 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 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 Cooperative Tiger Game (CTG) domain is limited to two agents to reduce the decision complexity and eliminate observability issues created by including a larger number of agents.

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 used in the demonstration had ten separate trust levels, corresponding to the number of times an agent cooperates before betraying the other agent. An agent's trust level could transition to any other trust level. For most of the simulations, the probability of transition was 0.05, but a 0.1 and 0.2 probability were also used for comparison in Table 1.

Figure 1 shows the changes in trust 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.

Figure 2 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.

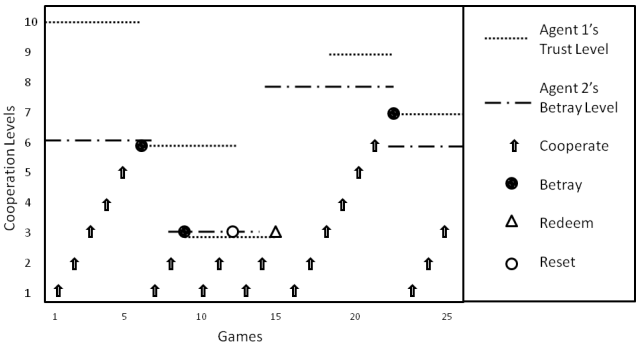

Figure 1: The trust interactions between agents playing the CTG.

I-POMDP Applied to the CTG

During the game, the individual agents must update their beliefs about the location of the tiger as well as their model of the other agent. The agents start each game with no prior knowledge about the location of the tiger. Hearing the tiger growl on a given side causes the agent to increase its belief that the tiger is on that side and decrease its belief in the tiger being on the other side. Attempting to open a door or hearing a door creak causes the agent to reset its belief about the location of the tiger to not prefer either side.

Agent a's model of agent b is updated after each turn. If agent b's actions correspond to the model, then agent a's model does not change. If agent b's actions do not correspond to the model, then agent a's model is changed, switching agent b's trust rating.

When neither agent is corrupted and their belief models trust the other agent, both agents listen until they agree on the reward's location and then open the corresponding door. This sequence of actions maximizes the expected reward for both agents and follows the belief update pattern utilized by Doshi.

As the game progresses agent b randomly becomes corrupted and the reward level reaches agent b's betrayal threshold. Agent b continues to listen until it can determine where the tiger is. If agent b believes that agent a trusts it, agent b announces that it wants to open the door with the reward. Once agent a agrees, agent a attempts to open the door with the reward and agent b opens the door with the tiger resulting in a large penalty for agent a and a double reward for agent b.

Based on the penalty, agent a determines that agent b lied and opened the door with the tiger. At this point, agent a updates its model to not trust agent b at the previous reward value and agent b's model is updated to assume that agent a no longer trusts it. The game effectively becomes capped as agent a will always reset the game prior to reaching the reward value it was betrayed at.

Eventually, agent b's corruption is removed and agent b must now reestablish its trust with agent a. When the reward approaches the level that a does not trust b, agent b can immediately announce that it is going to open a door prior to listening to determine which side holds the tiger. Agent b

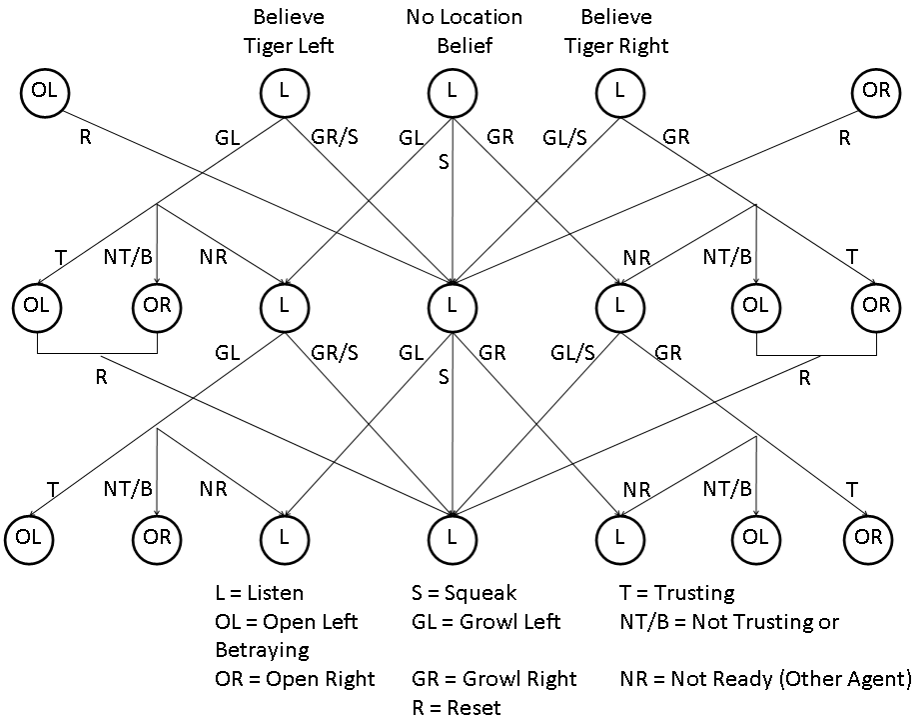

Figure 2: The state-action-observation transitions for the Cooperative Tiger Game

is guaranteed to take a penalty, either for releasing the tiger or for trying to open the reward without agent a's help, but agent b's trust rating is restored and the two agents start cooperating again.

To aid agent decisions, a horizon was used to look ahead. An agent examined the possible outcomes for a specified number of decisions and chose the decision path that led to the highest expected reward. An agent's decision was based on maximizing its future reward, not the necessarily the immediate reward for that decision. A horizon of one results in an agent always making a greedy, immediate choice while a larger horizon attempts to find a better end game solution at the cost of increased computation. All agents used the same horizon level to eliminate the horizon effect where one agent consistently outperforms another by conducting a deeper search.

A CTG simulation was used to demonstrate the algorithm. The agents played one hundred iterations of a fifty game series using a horizon of ten games. Each time the agents cooperated, the reward level increased, while any other outcome reset the reward level to one. If the reward level was greater than or equal to an agent's betrayal level, the agent attempted to betray. If the reward level was greater than or equal to an agent's trust level of the other agent, the agent refused to cooperate. During simulation, the agents cooperated 77 percent of the time, betrayed each other 8.5 percent of the time, and spent 14.5 percent of the simulation either redeeming themselves or reseting the game. Agents consistently cooperated when the stakes were low (the first three consecutive games), but rarely went past five consecutive games before betraying each other or refusing to cooperate. Agents did have trouble detecting when the other agent was corrupted to lower level in the game. If the agents had cooperated five consecutive times before agent a was corrupted, and agent a's new betrayal level was set at three, then agent a would betray agent b during the next (sixth) game. Agent b believed agent a's corruption level was six giving agent a the opportunity to betray agent b at during the third game after the reward reset.

Table 1 shows the percentage of time the agents take a particular action. As the probability of an agent being corrupted increases, the agents cooperate slightly less while the number of betrayal, redemption, and reset occurrences increase. Table 2 shows the different reward levels obtained by three separate algorithms and the typical difference in rewards achieved by the two agents during the same game. All three algorithms used ten trust levels and a horizon of ten. The memory function in the trust vector approach reduced the effect of an event five percent each time step. 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 both I-POMDP and 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 I-POMDP algorithm didn't utilize the reset option, leading to an increase of betrayals. The vector trust had a large increase of resets because the memory function would make the agent suspicious. Table 3 shows the percentage of time the agents take

Agent Action	Agent Corruption Rate		
	0.05	0.1	0.2
Cooperate	79.5	76.8	74.7
Betray	7.4	8.8	9.4
Redeem	10.1	10.6	11.4
Reset	3.0	3.8	4.5

Table 1: Percentage of agent actions with various rates of agent corruption

	Average Reward	Average Reward Difference Between Agents
TI-POMDP	1.0	0.29
I-POMDP	0.57	0.37
Vector Trust	0.86	0.23

Table 2: Normalized average rewards and the average difference between agent rewards.

a particular action for each of the three algorithms with a corruption rate of 0.05.

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 I-POMDP framework a suitable method to capture trust modeling. A major drawback to I-POMDP approach is the intractable nature of the problem, but approximation techniques can provide satisfactory results. The cooperative tiger game demonstrates the trust based I-POMDP's ability to quickly 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. In addition, testing on larger problems that include more agents is needed to determine the computation requirements and limits of this method.

Agent Action	TI-POMDP	I-POMDP	Vector Trust
Cooperate	79.5	70.0	73.1
Betray	7.4	14.6	3.1
Redeem	10.1	15.4	2.4
Reset	3.0	0.0	21.4

Table 3: Percentage of agent actions for different trust algorithms

References

Azzedin, F.; Ridha, A.; and Rizvi, A. 2007. Fuzzy trust for peer-to-peer based systems. *Proc. WASET* 21:123–127.

Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The complexity of decentralized control of markov decision processes. *Math. Oper. Res.* 27(4):819–840.

Doshi, P., and Gmytrasiewicz, P. J. 2005. A particle filtering based approach to approximating interactive pomdps. In Veloso, M. M., and Kambhampati, S., eds., *AAAI*, 969–974. AAAI Press / The MIT Press.

Doshi, P. 2004. A framework for optimal sequential planning in multiagent settings. In McGuinness, D. L., and Ferguson, G., eds., *AAAI*, 985–986. AAAI Press / The MIT Press.

Hansen, E. A.; Bernstein, D. S.; and Zilberstein, S. 2004. Dynamic programming for partially observable stochastic games. In McGuinness, D. L., and Ferguson, G., eds., *AAAI*, 709–715. AAAI Press / The MIT Press.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1996. Planning and acting in partially observable stochastic domains. Technical Report CS-96-08.

Rathnasabapathy, B.; Doshi, P.; and Gmytrasiewicz, P. J. 2006. Exact solutions of interactive pomdps using behavioral equivalence. In Nakashima, H.; Wellman, M. P.; Weiss, G.; and Stone, P., eds., *AAMAS*, 1025–1032. ACM.

Ray, I., and Chakraborty, S. 2004. A vector model of trust for developing trustworthy systems. In Samarati, P.; Ryan, P. Y. A.; Gollmann, D.; and Molva, R., eds., *ESORICS*, volume 3193 of *Lecture Notes in Computer Science*, 260–275. Springer.

Rettinger, A.; Nickles, M.; and Tresp, V. 2007. Learning initial trust among interacting agents. In Klusch, M.; Hindriks, K. V.; Papazoglou, M. P.; and Sterling, L., eds., *CIA*, volume 4676 of *Lecture Notes in Computer Science*, 313–327. Springer.

Roy, N.; Gordon, G.; and Thrun, S. 2004. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*. To appear.

Seuken, S., and Zilberstein, S. 2008. Formal models and algorithms for decentralized decision making under uncertainty. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*.

Song, W.; Phoha, V. V.; and Xu, X. 2004. An adaptive recommendation trust model in multiagent system. In *IAT*, 462–465. IEEE Computer Society.

Szer, D.; Charpillet, F.; and Zilberstein, S. 2005. Maa*: A heuristic search algorithm for solving decentralized pomdps. In *UAI*, 576–590. AUAI Press.

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

Wong, H. C., and Sycara, K. P. 2000. Adding security and trust to multiagent systems. *Applied Artificial Intelligence* 14(9):927–941.