Learning Models of Other Agents Using Influence Diagrams

Dicky Suryadi and Piotr J. Gmytrasiewicz

Department of Computer Science and Engineering, University of Texas at Arlington

Abstract. We adopt decision theory as a descriptive paradigm to model rational agents. We use influence diagrams as a modeling representation of agents, which is used to interact with them and to predict their behavior. We provide a framework that an agent can use to learn the models of other agents in a multi-agent system (MAS) based on their observed behavior. Since the correct model is usually not known with certainty our agents maintain a number of possible models and assign them probabilities of being correct. When none of the available models is likely to be correct, we modify one of them to better account for the observed behaviors. The modification refines the parameters of the influence diagram used to model the other agent's capabilities, preferences, or beliefs. The modified model is then allowed to compete with the other models and the probability assigned to it being correct can be arrived at based on how well it predicts the observed behaviors of the other agent.

1 Introduction

As research and development activities in agent-based systems progress, we can expect to see that in the future much of our computer-related works are done with the help of intelligent agents. In some systems, an agent may be working in the presence of other agents, either automated or human. These types of systems are called multi-agent systems (MAS), and are currently an active area of research. One important motivation behind the research is to find techniques that allow multiple agents to coordinate their actions so that individual rational actions do not adversely affect the overall system efficiency (Bond and Gasser, 1988). Effective coordination among agents in dynamic environments may be achieved by extending the agents' learning ability to recognize the capabilities, desires, and beliefs of other agents present in their environment (Gmytrasiewicz and Durfee, 1998).

Several papers have reported variety of techniques for constructing the models of agents. Kaminka et al. (1998) described a rule-based model for plan recognition task in the air combat simulation environment, while Carmel and Markovitch (1996) explored the use of finite automata to model the opponent agent's strategy. A series of papers reported works on recursive modeling method (RMM) for decision-theoretic agents, which uses deeper, nested models of other agents (Gmytrasiewicz et al., 1991, Vidal and Durfee, 1995, Gmytrasiewicz and Durfee, 1995, Gmytrasiewicz, 1996, Noh and Gmytrasiewicz, 1997). RMM represents an agent's decision situation in the form of a payoff matrix. In terms of belief, desire and intention (BDI) architecture, a payoff matrix contains a compiled representation of the agent's capabilities, preferences, and beliefs about the world. Beliefs about other agents are represented in terms of their own payoff matrices. These matrices form a hierarchical modeling structure, which is evaluated in the bottom-up fashion to determine the agent's action that maximize its expected utility.

Another decision-theoretic tool for constructing models of agents is an influence diagram, which represents information about an agent's capabilities, preferences, and beliefs in a more

explicit form. While RMM payoff matrices can be seen to summarize the information contained in the influence diagram (Noh and Gmytrasiewicz, 1997), this type of representation may provide better insight to the learning problem as it completely specifies all known random variables in the domain and their dependence relationships.

In our work, we seek to develop a method that can be used by an agent to learn the models of other agents. Our basic paradigm is to use decision-theoretic notion of rationality to describe and predict actions of other agents. Thus, we use influence diagrams as a modeling representation for agents. Given an initial model of an agent and a history of its observed behavior, we construct new models by refining the parameters of influence diagram in the initial model. The probabilities of a model being correct can be assigned based on how well it predicts the history of behavior.

The rest of the paper starts with an overview of influence diagrams, followed by an example on how they can be used to represent decision models of agents. Next, we get into the learning problem. We present a learning method for other agents capabilities and preferences, and we provide an example on how to apply it in a particular MAS domain. Finally, we present our conclusions and directions for further work.

2 Influence Diagrams

An Influence diagram (Howard and Matheson, 1984) is a graphical knowledge representation of a decision problem. It may be viewed as an extension to a Bayesian or belief network (Pearl, 1988), with additional node types for decisions and utilities. Influence diagrams have three types of nodes: nature node, decision node, and utility node. Just as in belief networks, nature nodes are associated with random variables or features, which represent the agent's possibly uncertain beliefs about the world. Decision node holds the choice of actions an agent has, thus represents the agent's capabilities. Utility node represent the agent's preferences. The links between the nodes summarize their dependence relationships.

Evaluation of the influence diagram is done by setting the value of the decision node to a particular choice of action, and treating the node just as a nature node with a known value that can further influence the values of other nodes. An algorithm for evaluating influence diagrams can be found in Russell and Norvig (1995).

Anti-air Defense Domain. As an example on how to use influence diagram to represent an agent's decision model, we present the anti-air defense domain, adapted from Noh and Gmytrasiewicz (1997). The scenario shown in Figure 1 depicts an MAS with two agents, in the roles of the defending units, against two incoming missiles in a 20x20 grid world. Each defending unit has only one interceptor and has to decide which of the missiles to intercept. When a missile reaches the ground, it inflicts damage in proportion to its warhead size. The goal of each unit is to minimize overall damage to the ground. To coordinate their decisions the units have to model each other to avoid redundant targeting of the same threat. Figure 2, shows an influence diagram representation that unit B1 may have to model decision-making of unit B2.

3 Learning Problem

The purpose of learning models of other agents is to have models that can predict other agents' behavior correctly, so that the predictions will lead the learning agent to arrive at the best decision

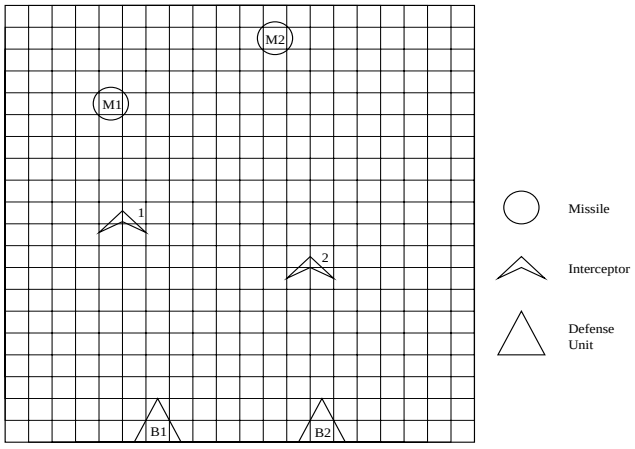

Figure 1. The anti-air defense domain.

in term of meeting the goal efficiently. The agent should already have a multi-agent reasoning system and maintain models of other agents in its knowledge base. Due to uncertainty, the agent assigns probabilities to the models, which are updated based on a history of the other agents' observed behavior. If a model is not accurate, its probability will gradually decrease. This signals the need for a better model, which can be obtained by learning.

In most situations, information about other agents only come from our observation of their behavior. Let us define a history of an agent's behavior as a set of its observed actions during a particular time frame, in which the data of the world states are known. Given only a history of behavior, how do we learn a better model? Our idea is to modify the initial model by refining parameters in influence diagram that are associated with capabilities, preferences and beliefs of other agent, based on the history of its behavior. The refinement can be done in stages with the order as above, according to the increasing level of complexity. There can be a number of modified models that can be generated. We will allow the modified models to compete with each other and other models maintained by the learning agent. The probability of each model being correct can be arrived based on how well the model predicts the history of behavior.

As we mentioned, we model the other agents as influence diagrams. From the point of view of learning, influence diagrams inherit the main feature of learning in belief networks, with additional aspects that relate to the character of decision and utility nodes. As a modeling representation tool, influence diagram is able to express an agent's capabilities, preferences, and beliefs, which are required if we want to predict the agent's behavior. Whereas learning problem in belief network is limited to learning the beliefs, influence diagram allows us to extend the description with learning the capabilities, that is, finding the correct possible value assignments for the decision node, and learning the preferences, which is finding the correct utility function.

The rest of this section describe the method we use to perform the refinement in an influence diagram. We discuss the strategy for learning the capabilities and the beliefs, but we concentrate on a method for learning the preferences.

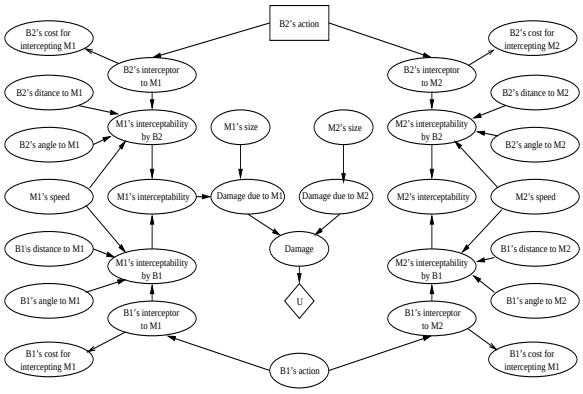

Figure 2. Decision model of defense unit B2.

3.1 Capabilities

An agent's capabilities are the alternative courses of actions it can execute to interact with the world. In an influence diagram, we represent them as possible value assignments to the agent's decision node. These values need to be exhaustive, which means that a model of other agent should cover all the possibilities of what the agent can do, even if they are not known or very unlikely. We can specify *Other* action value to represent all the unknown or unlikely actions not explicitly represented. At times, the agent may observed to be executing an action that is not explicitly present in our current model. In this case, we modify the model by specifying the action as explicit possible value of the decision node, and by updating the CPT's of the nodes that are influenced by the decision node.

For example, consider the agent as a defense unit B1 in the anti-air defense domain. It may be that initially B1 thinks that B2 is not equipped with long range interceptors. It would then model the capabilities of B2 as a set of decision node's values $\{SR\_M1, SR\_M2, Other\}$ in B1's model of B2, where SR_M1 and SR_M2 denote the actions of launching short range interceptors at missile M1 and M2, respectively. At some time, B1 could observe that B2 launches long range interceptor at one of the targets. Given this observation, B1 will modify its model of B2; the refined set of decision node's values is $\{SR\_M1, SR\_M2, LR\_M1, LR\_M2, Others\}$, where LR_M1 and LR_M2 explicitly represent the actions of launching long range interceptors at M1 and M2, which were thought unlikely before.

Conversely, sometimes the other agent's capabilities may need to be collapsed and included as part of the *Other* value. This may happen, for example, when B2 has become incapacitated and can no longer perform certain actions. From B1's perspective, it may notice that certain actions are missing from B2's history of behavior, especially when B1 believes that the missing action is the action B2 should take, if it is capable of it. To guarantee that the situation is caused only by an inaccuracy in modeling B2's capabilities, we need to identify the next preferable action given the model. If it agrees with the history of behavior, we can reason that the model still captures B2's beliefs and preferences except that somehow B2 cannot do the action that is missing from the history of behavior. B1's model of B2 is modified by removing the value of the missing action from the decision node.

3.2 Preferences

When a model of other agent cannot explain the history of its behavior, one of the possible reasons is that the agent's preferences are not accurately represented by the model. Our strategy will be to modify the model by refining the utility function so that every action in the history of behavior always maximizes utilities of the resulting states. Let $U(S)$ denote the utility of state S, which is given by a utility function.

The general structure of utility function is:

$$U(S) = f(X_1, ..., X_N) \tag{1}$$

where $X = \{X_1, ..., X_N\}$ is a set of features that directly influence the agent's preferences. In influence diagram, X is the set of parents of the utility node. The utility function f, is commonly postulated in multi-attribute utility theory (Keeney and Raiffa, 1976,Russell and Norvig, 1995) to be a weighted sum of the factors of values of features X_k, $k = 1, ..., N$. For simplicity, we assume that weighted factors depend linearly on the features X_k. We realize that this is a strong approximation, for example, in case of the utility of money, St. Petersburg paradox shows that it is not linear.

Therefore, we rewrite an agent's utility function as follows:

$$U(S) = w_1 x_1 + ... + w_N x_N \tag{2}$$

where each weight w_k corresponds to a feature X_k and represents the feature's measure of influence on the agent's utility, and x_k is the value of the feature X_k in state S. Our method of learning the agent's preferences will modify the weights w_k.

Let A^* denote an action that maximizes expected utility:

$$A^* = \arg\max_{a_i} \sum_{j=1}^{J} P(S_J|a_i, E) \times U(S_J) \tag{3}$$

where $A = \{a_1, ..., a_M\}$ is a set of the agent's alternative actions, and S_j are the possible outcome states given a, possibly non-deterministic, action a_i, with j ranging over J different outcomes. The background evidence E in the conditional probabilities represents the known data of the world which are provided in the history of behavior.

Using Equation 2 and 3, we obtain:

$$A^* = \arg\max_{a_i} \sum_{j=1}^{J} P(S_j|a_i, E) \sum_{k=1}^{N} w_k x_k^j \tag{4}$$

where x_k^j is the value of the feature X_k in the state S_j. We can show that this equation is equivalent to:

$$A^* = \arg\max_{a_i} \sum_{k=1}^{N} w_k \chi_k^i \tag{5}$$

where χ_k^i denote the expected value of feature X_k given a_i and background evidences E, given by the summation of all possible values of X_k: $(x_{k,1}, ..., x_{k,r_k})$ times their conditional probabilities:

$$\chi_k^i = \sum_{l=1}^{r_k} x_{k,l} \times P(X_k = x_{k,l} | a_i, E) \tag{6}$$

From the history of other agent's behavior, we have a set of the agent's observed actions during a given time frame. Let $D = \{D(1), ..., D(T)\}$ denotes such set, where T is the total number of actions, and $D(t) \in A$, $t = 1, ..., T$. The refinement of utility function is accomplished by having an initial utility function derived from the initial model, and subsequently adjusting the weights w_k based on the set D, so that $D(t)$ is equal to A^* for every situation t.

The choice of initial utility function depends on the initial knowledge the learning agent has about the other agent. In the extreme case where B1 has absolutely no knowledge on B2's preferences, it may have to put all the known features in B2's utility function and assign zero to all the weights.

To adjust the weights we follow a procedure common in neural network learning. We can render Equation 5 into neural network structure as shown in Figure 3. The network represents a

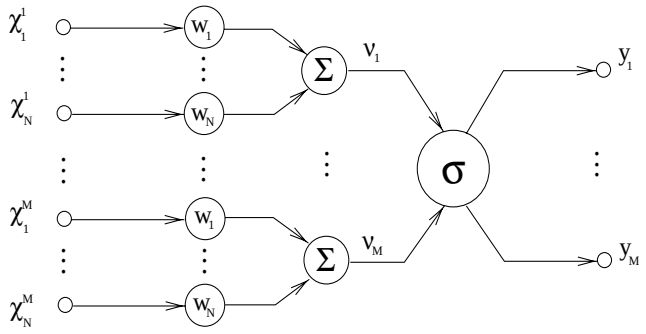

Figure 3. Network for learning the weights.

decision process at a decision situation t, where $\sigma(t)$ is the activation function that constitutes the output decision rule (Haykin, 1994):

$$y_i(t) = \begin{cases} 1 \; v_i(t) > v_j(t) \; for \; all \; j \neq i \\ 0 \; otherwise \end{cases} \tag{7}$$

where $i = 1, ..., M$. Let $d_i(t)$ represents the desired output value from $D(t)$ as follows:

$$d_i(t) = \begin{cases} 1 \; D(t) = a_i \\ 0 \; otherwise \end{cases} \tag{8}$$

Weight adjustment is done by applying a well-known gradient descent technique, namely *delta rule*(Widrow and Hoff Jr. (1960)). The idea is to minimize a cost function based on the error

signal $e_i(t)$ so that each weight adjustment brings the actual output value closer to the desired value. A commonly used cost function is the *mean-square-error* criterion:

$$E(t) = \frac{1}{2}\sum_{i=1}^{M} e_i^2(t) \tag{9}$$

where $e_i(t)$ is the error signal which is defined by:

$$e_i(t) = d_i(t) - y_i(t) \tag{10}$$

According to delta rule, the weight adjustment is proportional to the product of the error signal and the input unit. In our network, we use weight sharing to indicate the symmetry of the utility computation. We settle the adjustment for a shared weight as the sum of the correction of weights for all output units. We require normalization on χ before presenting it to the network, so that the resulting weights are normalized. The normalization is given by:

$$\aleph_k^i = \frac{\chi_k^i}{\sqrt{\sum_{s=1}^{M}\sum_{t=1}^{N}(\chi_t^s)^2}} \tag{11}$$

Thus, the final rule is given by:

$$\Delta w_k(t) = \eta \sum_{i=1}^{M} e_i(t)\aleph_k^i(t) \tag{12}$$

where $k = 1, ..., N$ and η is a positive constant that determines the rate of learning. The probabilities required to compute χ^i can be determined by evaluating the influence diagram model of the agent for every a_i.

Example. For demonstration of our method, let us consider a simple situation in the anti-air defense domain. Say that B1 knows there are at most two features that influence preferences of the unit B2. That is, the set of features that can influence B2's utility is $\{Damage, Cost\}$, where the possible value assignments for $Damage$ are $(0, 100, 500)$ and $Cost$ $(0, 1000, 5000)$.

B1 initially models B2's utility in inverse proportion to $Damage$ and independent to $Cost$. Therefore, the initial assignment of weights to features is:

$$w_1 = -1.0,\ w_2 = 0.0$$

This means that B1 thinks B2 only cares about minimizing damage to the ground, and not cost of intercepting the missiles. In this example we show how these weight are modified gives observation of B2's behavior that were different from the ones predicted by the initial model.

Let $A = \{I\_M1, I\_M2\}$ is a set of B2's possible actions, where I_M1 stands for intercept missile M1, and I_M2 intercept missile M2. In this case, assume that intercepting missile M1 would require a more sophisticated and more accurate interceptor. The following conditional probability distributions for $Damage$ and $Cost$:

$$\begin{aligned} P(Damage|I\_M1, E) &= (.20, .30, .50) \\ P(Damage|I\_M2, E) &= (.10, .10, .80) \\ P(Cost|I\_M1, E) &= (0, .30, .70) \\ P(Cost|I\_M2, E) &= (0, .60, .40) \end{aligned}$$

The above values reflect that using the advanced interceptor for missile M1 is more likely to reduce damage, but also more likely to result in a higher cost. The values of expected damage and cost are computed using Equation 6:

$$\begin{aligned}\chi_1^1 &= 0 \times .1 + 100 \times .1 + 500 \times .8 \quad = 280\\ \chi_2^1 &= 0 \times 0 + 1000 \times .3 + 5000 \times .7 = 3800\\ \chi_1^2 &= 0 \times .2 + 100 \times .3 + 500 \times .5 \quad = 410\\ \chi_2^2 &= 0 \times 0 + 1000 \times .6 + 5000 \times .4 = 2600\end{aligned}$$

where χ_1^1 and χ_2^1 are expected damage and cost given M1, and χ_1^2 and χ_2^2 are expected damage and cost given M2. The next step is to normalize χ, by applying Equation 11. Using the initial weights that indicate that B2 prefers to minimize damages but does not care about the costs, B1 arrives at a prediction that B2 would choose to intercept M1 and execute the action I_M1. Therefore, the values of network outputs according to B1's model of B2 are:

$$y_1 = 1,\, y_2 = 0$$

In the example case, however, B2 does something B1 did not expect and intercepts missile M2. Given that, B1 observes B2 takes action I_M2, the values of desired outputs, and the error signals are:

$$\begin{aligned}d_1 = 0, \quad d_2 = 1\\ e_1 = -1,\, e_2 = 1\end{aligned}$$

Using Equation 12 with the learning rate of $\eta = .1$, we have:

$$\begin{aligned}\Delta w_1 &= .1 \times (-1 \times 0.060 + 1 \times 0.089) = 0.029\\ \Delta w_2 &= .1 \times (-1 \times 0.821 + 1 \times 0.561) = -0.260\end{aligned}$$

The weights for the utility function in B1's model of B2 become:

$$w_1 = -0.971,\, w_2 = -0.260$$

As the result of considering this unexpected behavior of B2, therefore, B1 adjusted the weights it uses to model B2's utility function. Intuitively, the adjusted weights indicate that B2 does not care about the damages as much as B1 initially thought, and that it cares abut the costs of the defense operation as well.

In our experiments we simulated 100 decision situations, in which the positions and the sizes of the attacking missiles were generated randomly for each situation, and B2's behavior was guided by it's preference to minimize both damages and costs in equal proportion. After these iterations, the weights were further to the values:

$$w_1 = -0.777,\, w_2 = -0.561$$

This result confirms our intuition in this case; as B1 observed B2's behavior, its model of B2's utility function started resembling B2's actual preferences.

3.3 Beliefs

An agent's beliefs are represented in influence diagram as the nature nodes and the probabilistic relationships that exist among them. Given that we have knowledge on other agent's capabilities and preferences, and that our model still cannot predict its behavior, it is reasonable to assume that the model may not have the correct dependence links or probability assignments. We modify the model by refining them. The obvious difficulty is to determine where to begin the refinement. In a complex model, there can be a large number of possibilities that can account for the problem. What we need is to have a method that can trace down the elements in the belief system which need refinement. Although there are some good algorithms for learning the structure and local probability distributions, such as K2 algorithm (Cooper and Herskovits (1992)), the nature of our problem prevents us to use such algorithms.

Our idea for approaching the problem is to apply similar learning method in the previous subsection, only now we set the weights fixed and learn the expected values of the features which influence the utility. By using this process, it is possible to determine which of the feature nodes that are likely to be the source of inaccuracy in the model. We let Φ denote the set of of such feature nodes. There are several strategies to modify the model that are based on exploring the possible reasons that cause the model's failure to predict the behavior.

The first strategy assumes that the reason is due to incorrect local probability distributions of the features in Φ. Based on this assumption, the model is modified by refining CPT's on the feature nodes in Φ. The second strategy assumes that the reason is due to incorrect dependence relationships involving the features in Φ. The model is modified by refining the structure of influence diagram. The alternatives are to remove links from certain parent nodes, or to add more links from other nodes. The third strategy assumes that the problem is caused, not by the features in Φ, but by their parents. It is possible that certain parent nodes have incorrect probability distributions or incorrect dependence relationships. The model is modified by performing these strategies at the level of the parent nodes of Φ. Note that this strategy has a recursive property. It allows the refinement goes bottom up until there are no more parents or the evidence nodes are reached.

These strategies attempt to narrow down the search space of possible models. However, the number of models we come up with can still be unreasonably high, and we are yet to develop a method that can decrease the number further.

4 Conclusion and Further Work

In this paper, we developed a framework that can be used by an agent to learn models of other agents in a multi-agent system. The framework makes use of influence diagrams as a modeling representation tool. We addressed three strategies to create a new model of the other agent, which are based on learning its capabilities, preferences, and beliefs, given an initial model and the agent's behavior data.

We presented a method for learning the agents' capabilities, and our ideas on how to learn the beliefs, but we concentrated on learning of the other agent's preferences. Our method attempts to modify the other agent's utility function by incorporating a neural network learning technique, which involves the presentation of the history of other agent's behavior as the training set and a series of weight adjustments. The new model for the agent is created by replacing the initial

model's utility function with the one produced by our method. To assign the probability to the new model being correct, we allow it to compete with other models we have based on how well each model predicts the behavior.

We are currently working on the implementation of the learning algorithm and its integration with the agent's probabilistic frame-based knowledge base. We will provide the results from the experiment in our next paper.

References

Bond, A. H., and Gasser, L. (1988). *Readings in Distributed Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann Publishers.

Carmel, D., and Markovitch, S. (1996). Learning models of intelligent agents. *AAAI/IAAI* 1:62–67.

Cooper, G. H., and Herskovits, E. H. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning* 9:309–347.

Gmytrasiewicz, P. J., and Durfee, E. H. (1995). A rigorous, operational formalization of recursive modeling. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*, 125–132.

Gmytrasiewicz, P. J., and Durfee, E. H. (1998). Rational interaction in multiagent environments: coordination. *Submitted for publication, available at http://www-cse.uta.edu/ piotr/piotr.html*.

Gmytrasiewicz, P. J., Durfee, E. H., and Wehe, D. K. (1991). A decision-theoretic approach to coordinating multiagent interactions. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 166–172.

Gmytrasiewicz, P. J. (1996). An approach to user modeling in decision support systems. In *Proceedings of the Fifth International Conference on User Modeling*, 121–127.

Haykin, S. (1994). *Neural Network: A Comprehensive Foundation*. New York: Macmillan College Publishing Company.

Howard, R. A., and Matheson, J. E. (1984). Influence diagrams (article dated 1981). *Howard, R.A. and Matheson, J.E. (Eds.), Readings on the principles and applications of decision analysis* 2:719–762.

Kaminka, G., Tambe, M., and Hopper, C. (1998). The role of agent-modeling in agent robustness. In *Proceedings of the Conference on AI Meets the Real World*.

Keeney, R. L., and Raiffa, H. (1976). *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. New York: Wiley.

Noh, S., and Gmytrasiewicz, P. J. (1997). Agent modeling in antiair defense. In *Proceedings of the Sixth International Conference on User Modeling*, 389–400.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kauffman.

Russell, S. J., and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice-Hall.

Vidal, J. M., and Durfee, E. H. (1995). Recursive agent modeling using limited rationality. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*, 376–383.

Widrow, B., and Hoff Jr., M. E. (1960). Adaptive switching circuits. *IRE WESCON Convention Record* 96–104.