

Modeling how Humans Reason about Others with Partial Information

Sevan G. Ficici
School of Engineering and Applied Sciences
Harvard University
Cambridge, Massachusetts USA
sevan@eecs.harvard.edu

Avi Pfeffer
School of Engineering and Applied Sciences
Harvard University
Cambridge, Massachusetts USA
avi@eecs.harvard.edu

ABSTRACT

Computer agents participate in many collaborative and competitive multiagent domains in which humans make decisions. For computer agents to interact successfully with people in such environments, an understanding of human reasoning is beneficial. In this paper, we investigate the question of how people reason strategically about others under uncertainty and the implications of this question for the design of computer agents. Using a situated partial-information negotiation game, we conduct human-subjects trials to obtain data on human play. We then construct a hierarchy of models that explores questions about human reasoning: Do people explicitly reason about other players in the game? If so, do people also consider the possible states of other players for which only partial information is known? Is it worth trying to capture such reasoning with computer models and subsequently utilize them in computer agents? We compare our models on their fit to collected data. We then construct computer agents that use our models in one of two ways: emulating human behavior and playing best response to the model. After building our agents, we deploy them in further human-subjects trials for evaluation. Our results indicate that people do reason about other players in our game and also reason under uncertainty. Better models are shown to yield more successful computer agents.

Categories and Subject Descriptors

I.2.6 [Learning]: Knowledge acquisition; I.6.5 [Model Development]: Modeling methodologies; J.4 [Social and Behavioral Sciences]: Economics

General Terms

Experimentation, Human Factors, Performance, Design

Keywords

Human models, Uncertainty, Reasoning, Negotiation

1. INTRODUCTION

With increasing frequency, computer agents are participating in collaborative and competitive multi-agent domains

Cite as: Title, Author(s), *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX.

Copyright © 2008, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

in which humans reason strategically to make decisions. Examples include online auctions, financial trading, scheduling, and computer gaming (online and video). The deployment of computer agents in such domains requires that the agents understand something about human behavior so that they can interact successfully with people; the computer agents must be sensitive to both how people reason in strategic settings as well as the social utilities people employ to inform their reasoning. To date, these design requirements for computer agents have received relatively little attention.

Models of human reasoning have been shown to be helpful for agent design [7]. In their work, Gal et al. [7] used a situated full-information two-player negotiation game; in this game, one player proposes an exchange of resources and the other player responds by accepting or rejecting the proposal. Gal et al. [7] conducted human-subjects trials and subsequently learned a model of the responder's utility function; this model was utilized to construct a computer proposer agent that took human responder behavior into account. Nevertheless, Gal et al. [7] leave unexplored the questions of how humans reason strategically and under uncertainty: the responder of their game reasons only after the proposal is received, and so requires neither strategic reasoning (that is, reasoning by one player about what another player might do) nor reasoning under uncertainty about another player's state. Here, we are interested to focus on these two aspects of human reasoning.

In this paper, we investigate questions about how people reason strategically about others under uncertainty and the implications of these questions for the design of computer agents. For example, is human reasoning *reflexive*, where the behavior of other players is accounted for implicitly, that is, without explicit consideration of other players' possible actions or states? Or, do humans somehow try to reason strategically about other players by consulting models that they maintain about them? If so, then does such reasoning also consider the possible states of other players? And, if either of these possibilities is true, then is it worth trying to capture such reasoning with computer models and subsequently utilize them in computer agents?

Our investigation of these questions leads us to a variety of contributions. We construct a hierarchy of models, whereby models are differentiated not only by whether they include strategic reasoning, but also by whom that reasoning concerns. We provide learning algorithms for our models. The human-subjects experiments we conduct provide a wealth of data which we use to train and test our models and which can be used for further investigations. We ana-

lyze our models by comparing them on their fit to the data. We also construct computer agents that use our models in one of two ways: emulating human behavior and playing best response to the model. We evaluate our agents by their performance in further human-subjects trials. Our analyses provide insight into whether and how humans behave strategically under uncertainty and the issues that surround engineering computer agents to interact with humans.

We find a model’s ability to predict human behavior depends upon whether we model the human as using reflexive or strategic reasoning; further, if strategic reasoning is used, then the model’s performance also depends upon whom we model the human to be reasoning about. We also find that the benefits gained from using increasingly sophisticated models diminish, while the computational costs of such models increase. Surprisingly, we find that emulating human behavior is better than playing best response to a model.

While many fields relate to the goal of creating computer agents that take human strategic thinking into account, none have yet placed a spotlight on this goal. For example, classical game theory [4] precludes modeling agents as anything other than rational actors, severely restricting the types of reasoning we can capture. Further, the rationality assumption of classical game theory is well known to be violated in many real-world domains. In recognition of the fact that human decision making often deviates from full rationality, the field of behavioral economics [15, 2] seeks to explain the gap between actual human decision-making and that of classical game theory’s *homo economicus*. Nevertheless, the decision-making domains studied in behavioral economics are very abstract and lack *situatedness*; being situated entails interaction with and within an environment [12]. Further, behavioral economics does not concern itself with engineering computational agents that can interact successfully with human decision makers.

The field of multi-agent systems (MAS) [18, 11] is concerned with engineering computer agents that operate and interact in environments containing other agents. Nevertheless, much MAS research focuses on environments comprised of only computer agents, and so agents tend to be viewed as rational actors; this assumption simplifies the task of *recursive modeling* [8], where an agent models another agent as an entity that itself models other agents. Recursive models involving bounded-rational agents are also examined in MAS [17], but such models are not generally intended to capture the peculiarities of bounded rationality in human reasoning. Research on theories of mind and emotion [16, 13, 9, 1] typically involves no learning at all or no learning from real human data; the models used in such work are often used to generate simulations of behavior. Contrastingly, our model parameters are learned by training the model on human data; our computer agents use our models to interact with people. Thus, the enterprise of modeling strategic human decision-making for the purpose of engineering computer agents that are sensitive to human behavior stands apart from most MAS research.

2. THREE-PLAYER NEGOTIATION GAME

We require an environment that is appropriate for investigating how humans reason about others under uncertainty. The environment must be simple enough for analysis and agent engineering to be tractable, yet rich enough to reflect salient features of the real world. We desire an environment

that is situated, can provide partial information, and promotes reasoning about other players. The Colored Trails (CT) framework [10] meets our requirements.

CT is a highly configurable, situated multi-agent task environment that can be played by humans and computer agents. CT captures the important high-level features of decision-making found in many real-world environments; CT is sufficiently abstract to focus on high-level features, yet is simultaneously grounded in a situated task domain. The situated task activities presented by the CT environment distinguish CT from the games often used in behavioral economics, which tend to present highly abstract decision-making scenarios. Gal et al. [5] demonstrate a framing effect in which a game presented as a situated task activity elicits stronger concern with social factors such as fairness; the same underlying game presented in a more abstract payoff matrix form engenders behavior more in line with rational Nash equilibrium play. These results demonstrate the importance of eschewing highly abstract games in favor of situated activity if we are interested to learn about how people reason in real-world settings.

Using the CT environment, we construct a three-player partial-information negotiation game. Our game is played on a 4x4 board of colored squares; each square is one of five colors. Each of the three players has a piece on the board as well as a collection of four to eight colored chips (175 possible chip combinations) that can be used to move her piece; to move her piece to an adjacent square, a player must turn in a chip of the same color as the square. The objective of each player is to maximize her score by moving her piece as close as possible to, and preferably onto, a *goal square* while using as few chips as possible. We generate initial conditions such that players can usually improve their ability to approach the goal by trading chips.

Each game proceeds as follows. Each player is randomly assigned to one of the three roles in the game, denoted *proposer 1*, *proposer 2*, and *responder*. Each player knows the state of the board (board colors, goal location, locations of all player pieces) and the chips that she possesses. Proposers also know the chips possessed by the responder, but not by each other; this is the source of uncertainty in the game. The responder knows the chips possessed by both proposers. The proposers are allowed to exchange chips with the responder, but not each other. The proposers simultaneously formulate their proposals to exchange chips and submit them to the responder; any redistribution of chips between a proposer and the responder is valid. A proposal may also leave the chips unchanged. The responder then chooses to accept no more than one of the two proposals, or declines them both. After the responder’s decision is made, the CT system 1) informs the proposers of the outcome, 2) executes any accepted proposal, and 3) automatically moves all three players’ pieces to obtain the maximal possible score for each player, given the chips possessed after the negotiation. Landing onto the goal square earns a player 100 points; a player unable to reach the goal pays a penalty of 25 points for each square she is away from the goal. Each chip not used by a player for moving earns the player 10 points.

A number of factors may influence the offer a proposer ultimately makes. First, a proposer may need certain chips to improve its utility. But, the responder may also require certain chips, and these requirements may or may not be synergistic with the needs of the proposer. In addition,

because the responder can accept no more than one proposal, there exists a competitive relationship between the proposers. Therefore, a proposer may want to reason about what the other proposer may offer. Since proposers have partial information about each other (they know each other’s location on the board, but do not know about each other’s chips), reasoning under uncertainty is required. The behavior of a proposer explores the tension between fulfilling its own utility function and that of the responder in the face of unknown competition.

3. PLAYER MODELS

Our goal in this paper is to build computer agents that interact successfully with humans. Towards this end, we construct several models of how humans reason in our game. By examining the performance of a variety of models, we hope to identify prominent features of human reasoning and engineer effective computer agents. We concentrate on modeling the proposers in our game, since they have the most interesting decision-making task—our game invites proposers to employ strategic reasoning under uncertainty. Nevertheless, most of our proposer models will require models of the responder to operate, and so we construct responder models as well. Our player models fall into two broad categories: *reflexive* and *strategic*.

3.1 Reflexive Models

Our most basic models, the reflexive models, are elaborations of the architecture used in [7, 6]. All of our models (reflexive and strategic) make use of only three simple features that quantify proposal properties; these features are rather general and can be applied to almost any negotiation game. Though our game involves three players, each proposal specifies a pair-wise interaction between two players. Let *self-benefit* (SB) quantify the change in score a player will receive if a proposal is accepted, and *other-benefit* (OB) quantify the change in score the other player will receive if the proposal is accepted. Other features related to benefits were investigated with cross validation, but were found not to improve our models. In particular, we considered categorical discretizations of SB and OB to indicate whether benefit is positive, negative, or zero; these were believed to be useful to express “rational” play. We also considered two features that addressed the fairness of a proposal, one that corresponds to the Nash bargaining concept [14] and another that considers the context of alternative proposals that could have been made [3].

Given the construction of our negotiation game, a proposer may have two or more different chip-exchange proposals that she could make that would each yield the same set of benefit values if accepted; such proposals are thus equivalent with respect to our models. Indeed, the set of possible proposals that a proposer may make can be divided into many equivalence classes, with each class containing several possible proposals. All of our models operate by scoring proposal *classes*, not all individual proposals; more precisely, each class is represented by one member—called an *exemplar*—of the class, and the model scores that member. The probability that an equivalence class is selected may depend upon the number of proposals in the class. For example, it is plausible that, all else being equal, a class with more proposals in it will have more chance of being selected. Thus, the third feature used by our proposer models is the *class*

size (CS) of an exemplar proposal. We have found that proposer models that include the CS feature perform much better than those without it.

Let each proposal $O = \langle O_{SB}, O_{OB}, O_{CS} \rangle$ be represented by a vector of feature values; let $\mathbf{w} = \langle w_{SB}, w_{OB}, w_{CS} \rangle$ be the vector of corresponding feature weight parameters. Let ϕ denote the *status-quo*, which for the proposer represents the proposal that no chips change hands, and for the responder represents the option of rejecting both offers. Let $U : \mathbb{O} \rightarrow \mathbb{R}$ be a linear utility function that maps the space of offers \mathbb{O} to the reals \mathbb{R} . Our initial definition of our utility function is

$$U(O) = w_{SB} \cdot O_{SB} + w_{OB} \cdot O_{OB}. \quad (1)$$

Human behavior varies: people have different utility functions. Thus, we use mixture models to cluster human play into different behavioral types. We have T types; an individual of type t_i uses utility function U^{t_i} with weight vector \mathbf{w}^{t_i} . Let ρ^{t_i} be the proportion of individuals of type t_i .

Humans also select offers non-deterministically; we are prone to make errors. Further, since our models will not be perfect, we care to have our models attach probabilities to different outcomes. To accommodate these points, we use a multinomial logit (soft-max) function to obtain a probability distribution over a decision maker’s options. Let \mathcal{O} be a set of options to choose from. The probability that an individual of type t_i will select the m -th proposal in \mathcal{O} is

$$\Pr(\text{selected} = O^m | \mathcal{O}, t_i) = \frac{e^{U^{t_i}(O^m)} \cdot (O_{CS}^m)^{w_{CS}}}{K}. \quad (2)$$

The denominator K is a normalization constant so that we obtain probabilities. The numerator is the product of two terms. The first term belongs to the multinomial logit function, which gives the soft-max behavior over our utility function. The second term models how the class size of O^m may influence the probability of the proposal being selected. When $w_{CS} = 0$, the player is modeled to be indifferent with respect to class size. When $w_{CS} > 0$, the player is modeled to be more likely to select proposals belonging to larger classes. When $w_{CS} < 0$, the player is modeled to be more likely to select proposals belonging to smaller classes. Conveniently, the product of these two terms is mathematically identical to moving the second term into our function U as follows:

$$U(O) = w_{SB} \cdot O_{SB} + w_{OB} \cdot O_{OB} + w_{CS} \cdot \ln(O_{CS}). \quad (3)$$

This second version of function U captures all three of our proposal features and simplifies our learning algorithms (described below). Though the class-size feature does not pertain to benefits in score, it remains that proposals giving higher values of U are more likely to be selected, according to the model. The final form of our reflexive model is thus

$$\Pr(\text{selected} = O^m | \mathcal{O}, t_i) = \frac{e^{U^{t_i}(O^m)}}{\sum_k e^{U^{t_i}(O^k)}}. \quad (4)$$

Taking an expectation over all behavioral types gives

$$\Pr(\text{selected} = O^m | \mathcal{O}) = \sum_i \Pr(\text{selected} = O^m | \mathcal{O}, t_i) \rho^{t_i}. \quad (5)$$

3.1.1 Responder Model R

A reflexive model of player behavior assumes that, in the decision making process, the player reasons explicitly about

the choices she has, but not about other players. Such an assumption is clearly appropriate for modeling the responder of our game, since the responder’s decision making requires neither strategic thinking nor reasoning under uncertainty; the responder simply reacts to the decisions that are made by the proposers. The responder has three choices $\mathcal{O} = \{O, \bar{O}, \phi\}$, where O and \bar{O} are the two proposer offers; ϕ is the null offer, which the responder selects if it cares for neither proposer offer. Since it has only three choices, the notion of class size is not particularly meaningful to the responder; we define each of the responder’s three choices to have a class size of one. Our model of responder behavior is a mixture model precisely of the form described by Equation (5); the parameters $w_{SB}^{t_i}$ and $w_{OB}^{t_i}$ are easily interpreted to represent the responder’s utility function. We denote our reflexive responder model $R\{\}$; the empty curly braces indicate a reflexive model.

3.1.2 Proposer Model $P\{\}$

Our simplest model of proposer behavior, denoted $P\{\}$, is also reflexive; this model assumes that a proposer does not explicitly reason about the responder and other proposer, even though such reasoning is meaningful in our game. Here, we define $\mathcal{O} = \{O^1 \dots O^M\}$ to be the set of equivalence class exemplars, where M is the number of proposal classes that exist given the game state. In practice, a proposer may have on the order of 40–60 classes, with each class containing 1–25 equivalent chip-exchange proposals.

Like the responder model, the reflexive proposer model is described by Equation (5), and $w_{SB}^{t_i}$, $w_{OB}^{t_i}$, and $w_{CS}^{t_i}$ are the model’s parameters. Unlike the responder model, however, the benefit weights are not so easily interpreted to represent the proposer’s utility function. This is because, to the extent that a proposer implicitly reasons about other players, model training will cause the weights to represent, as best they can, some amalgam of the proposer’s utility function and implicit reasoning process.

3.2 Strategic Models

The reflexive models explicitly reason about the options they have, but not about other players in the game. Our game’s structure makes a reflexive model appropriate for the responder; but, it is possible that a reflexive model of proposer behavior can be improved upon. We introduce new models of proposer behavior that use strategic reasoning.

3.2.1 Proposer Model $P\{R^-\}$ and Responder R^-

More sophisticated than $P\{\}$, we can model a proposer as reasoning explicitly about the responder, but still not the other proposer. The existence of the other proposer is acknowledged, but the partial information about the other proposer that the game provides is ignored. Unfortunately, without explicit reasoning about the other proposer, we cannot utilize our responder model R , because it requires that we specify both proposers’ offers. To address this problem, we construct a modified responder model R^- that does not require two proposer offers. Our new proposer model then utilizes this modified responder model to reason about the responder, specifically about how likely the proposer’s offers are to be accepted by the responder.

The modified responder model R^- accepts two known choices $\mathcal{O} = \{O, \phi\}$ as input and represents the unknown \bar{O} with a fixed “generic” proposal that has utility v^{t_i} (v^{t_i} is

a model parameter in addition to feature weights \mathbf{w}^{t_i}):

$$\Pr(\text{selected} = O | O, \phi, t_i) = \frac{e^{U^{t_i}(O)}}{e^{v^{t_i}} + e^{U^{t_i}(\phi)} + e^{U^{t_i}(O)}}, \quad (6)$$

where v^{t_i} is the generic utility given by responder type t_i to the unknown proposal \bar{O} .

We denote our first strategic proposer model $P\{R^-\}$, to indicate that the *pre-learned* responder model R^- is embedded in the proposer model. The parameters to be learned in $P\{R^-\}$ are the proposer’s three utility function weights w_{SB} , w_{OB} , and w_{CS} . Proposer model $P\{R^-\}$ uses R^- to reason about how the responder might react to possible offers. We assume players are maximizing social utility relative to their models of other agents. Here, $P\{R^-\}$ uses R^- to compute expected utility EU , which is the utility of an offer if accepted times the probability of it being accepted by the responder; out of convenience, we choose to ignore the case where the responder rejects an offer, since the realized benefits (i.e., SB and OB) from a rejected offer are zero.

$$EU^{t_i}(O) = U^{t_i}(O) \cdot \Pr(\text{selected} = O | O, \phi) \quad (7)$$

Just as with our reflexive models, we assume that human behavior is noisy. Thus, we convert our expected utilities to probabilities using the multinomial logistic function. The probability of model $P\{R^-\}$ selecting offer $O^m \in \mathcal{O}$ is thus

$$\Pr(\text{selected} = O^m | \mathcal{O}, t_i) = \frac{e^{EU^{t_i}(O^m)}}{\sum_k e^{EU^{t_i}(O^k)}}. \quad (8)$$

3.2.2 Level- N Proposer Models

Our next model of proposer behavior asserts that a proposer explicitly reasons about the responder *and* other proposer; this model takes into account the partial information available about the other proposer to reason about what that proposer might offer and how its offers might affect the responder’s decision. To perform such reasoning, the proposer model must itself utilize models of the responder and other proposer. One such proposer model is $P\{R, P\{\}\}$. This model embeds our reflexive model of responder behavior R ; it also models the other proposer as being reflexive. Another such proposer model is $P\{R, P\{R^-\}\}$, which models the other proposer as reasoning about the responder but not the first proposer. We say $P\{R, P\{\}\}$ and $P\{R, P\{R^-\}\}$ are *level-one* models because they model the other proposer; the models of proposer behavior *embedded* in the level-one models do not model the other proposer ($P\{R^-\}$ only models the responder), and so are level-zero proposer models. As before, the embedded models are pre-learned, and so do not contribute any new parameters to the level-one models. The parameters of a level-one model remain the three weights w_{SB} , w_{OB} , and w_{CS} .

A *level-two* proposer model is more sophisticated than a level-one model; it states that a proposer (for clarity named P1) reasons about both the responder and other proposer (for clarity named P2), and further states that P1 believes that P2 itself reasons about P1. In the level-two model $P\{R, P\{R, P\{\}\}\}$, P1 believes that P2 models P1 as being reflexive; in level-two model $P\{R, P\{R, P\{R^-\}\}\}$, P1 believes that P2 models P1 as reasoning about the responder, but not P2. The parameters of a level-two model are the same three weights as in level one. In principle, we can cre-

ate a level- N proposer model by recursively embedding a pre-learned level- $(N - 1)$ model.

The expected utility $EU(O)$ of an offer O to a level- N model (say, $P\{R\}, P\{N - 1\}$) is

$$EU(O) = \sum_{\bar{C} \in \bar{\mathcal{C}}} \sum_{\bar{O} \in \bar{\mathcal{O}}|\bar{C}} \Pr(\bar{C}) \cdot \Pr(\text{selected} = \bar{O}|\bar{C}) \cdot \Pr(\text{selected} = O|O, \bar{O}, \phi) \cdot U(O). \quad (9)$$

Equation (9) operates as follows. We consider the set $\bar{\mathcal{C}}$ of all possible chipsets that the other proposer might have. For each such chipset $\bar{C} \in \bar{\mathcal{C}}$, we consider all possible offers \bar{O} that the other proposer could make. To calculate our expected utility for offer O , we need to consider several factors. First is the probability that the other proposer has a certain chipset \bar{C} . Then, given \bar{C} , we use our level $N - 1$ model of the other proposer to estimate the probability that it will make offer \bar{O} . We next use our model of the responder to estimate the probability that it will accept our proposal O over \bar{O} and ϕ . The product of these probabilities times our utility for O gives our expected utility for O . The expected utilities for all the offers in \mathcal{O} are then plugged into the softmax equation (8) to obtain a probability distribution over offers for the level- N model.

3.3 Emulating and Best-Response Agents

We utilize the models described above in two ways to construct computer agents that play proposers in our game. Our first approach uses a proposer model to *emulate* the behavior of human proposers; the computer agent queries the model to learn which proposal in \mathcal{O} , according to the model, a human is most likely to make, and makes that proposal. In our second approach, the computer agent uses a proposer model along with responder model R to form a *best response* (BR) to the expected behavior of the other proposer; the agent strategizes to maximize its expected benefit. When the BR computer agent uses a level- N proposer model, the agent employs a pattern of reasoning that is identical to a level- $N + 1$ proposer model, except that the BR agent's utility function is entirely selfish: $w_{SB} > 0$, $w_{OB} = 0$, $w_{CS} = 0$.

4. LEARNING

Models are trained by gradient descent. Let $g(\text{selected} = O^*|\mathcal{O})$ be the probability that a model assigns to the proposal O^* that was actually selected by a human player, given the set of options \mathcal{O} . The error function F that we minimize measures negative log likelihood of the data (containing D decision instances), given a model:

$$F = - \sum_{d=1}^D \ln(g(\text{selected} = O^{*d}|\mathcal{O}^d)). \quad (10)$$

The derivative of the error function with respect to some model parameter $w_i^{t_i}$ (or v^{t_i} in R^-) is

$$\frac{\partial F}{\partial w_i^{t_i}} = - \sum_{d=1}^D \frac{\frac{\partial g}{\partial w_i^{t_i}}}{g(\text{selected} = O^{*d}|\mathcal{O}^d)}. \quad (11)$$

Let α be our learning rate. The weight update equation for some model parameter $w_i^{t_i}$ is

$$w_i^{t_i} \leftarrow w_i^{t_i} - \alpha \cdot \frac{\partial F}{\partial w_i^{t_i}}. \quad (12)$$

To update the probability ρ^{t_i} of type t_i in the mixture, we multiply by the negative of the gradient, which turns out to be equivalent to using Bayes' rule:

$$\rho^{t_i} \leftarrow - \frac{\partial F}{\partial \rho^{t_i}} \cdot \rho^{t_i} = \sum_{d=1}^D \frac{g(\text{selected} = O^{*d}|\mathcal{O}^d, t_i) \cdot \rho^{t_i}}{g(\text{selected} = O^{*d}|\mathcal{O}^d)}. \quad (13)$$

Equation (11) requires that we further calculate the partial derivative of function g , which represents the behavior of our model. Though g varies with each model, we find that the derivative of g has a similar structure over all models. The partial derivative of g with respect to some parameter $w_i^{t_i}$ for our reflexive models (and R^-) is:

$$\frac{\partial g(O^*|\mathcal{O}, t_i)}{\partial w_i^{t_i}} = g(O^*|\mathcal{O}, t_i) \left(O_i^* - \sum_k O_i^k \cdot g(O^k|\mathcal{O}, t_i) \right) \quad (14)$$

The partial derivative of g with respect to v^{t_i} in R^- is:

$$\frac{\partial g(O^*|\mathcal{O}, t_i)}{\partial v^{t_i}} = - \frac{e^{U^{t_i}(O^*)} \cdot e^{v^{t_i}}}{(e^{v^{t_i}} + \sum_k e^{U^{t_i}(O^k)})^2} \quad (15)$$

The partial derivative of g with respect to some parameter $w_i^{t_i}$ for model $P\{R^-\}$ is:

$$\frac{\partial g(O^*|\mathcal{O}, t_i)}{\partial w_i^{t_i}} = g(O^*|\mathcal{O}, t_i) \left(O_i^* \cdot \Pr(O^*|\mathcal{O}) - \sum_k O_i^k \cdot g(O^k|\mathcal{O}, t_i) \cdot \Pr(O^k|\mathcal{O}) \right) \quad (16)$$

The partial derivative of g with respect to some parameter $w_i^{t_i}$ for a level- N model is:

$$\frac{\partial g(O^*|\mathcal{O}, t_i)}{\partial w_i^{t_i}} = g(O^*|\mathcal{O}, t_i) \left(O_i^* \cdot Z(O^*) - \sum_k O_i^k \cdot Z(O^k) \cdot g(O^k|\mathcal{O}, t_i) \right), \quad (17)$$

where

$$Z(O) = \sum_{\bar{C} \in \bar{\mathcal{C}}} \sum_{\bar{O} \in \bar{\mathcal{O}}|\bar{C}} \Pr(\bar{C}) \cdot \Pr(\text{selected} = \bar{O}|\bar{C}) \cdot \Pr(\text{selected} = O|O, \bar{O}, \phi). \quad (18)$$

Note that using Equation (17) entails the calculations performed in (9). For higher level models, this is recursive, making the cost of a level- N model grow exponentially with N . Thus, only small N are feasible; nevertheless, we do not expect human reasoning to correspond to large N .

5. HUMAN-SUBJECTS TRIALS

We recruited 69 human subjects to play our negotiation game for 15 rounds; over half of their total compensation was determined by the scores they accumulated over the rounds.

Subjects were randomly matched in each round. To emphasize that they were playing a sequence of one-shot games and not an iterated game, subjects performed an unrelated activity between rounds. We obtained a total of 268 games in which all three players were human subjects. Another 221 games involved a human responder deciding between two hand-crafted offers; the hand-crafted offers are not used to train proposer models.

Using cross validation, we determined that our models of responder behavior best fit the data with two types; we then used two types for our proposer models, as well. We trained two responder models R and R^- , two level-zero proposer models $P\{\}$ and $P\{R^- \}$, and the level-one proposer model $P\{R, P\{R^- \}\}$. We constructed six computer proposer agents from our three proposer models. Three agents were emulators, using proposer models $P\{\}$, $P\{R^- \}$, and $P\{R, P\{R^- \}\}$. The other three agents played a best response to our three proposer models.

We recruited an additional 59 human subjects to test the performance of our models and agents. We selected initial game states that caused our models and agents to behave differently in order to magnify differences in performance. Each game state was used seven times. In one copy, all players were human. In the other six copies, proposer 1 was one of our agents, and the other two players were human. This allows us to compare the performance of the different agents on the same game states.

6. RESULTS

6.1 Learned Models

In this section, we examine how well our models fit our data from a number of perspectives. The top half of Table 1 summarizes model fit with respect to a hold-out data set that comes from the same cohort of subjects used to generate our training data. Each column reports performance of one of our models; the column labeled ‘Random’ gives figures for random guessing. The figures in each row are averages over the entire data set.

The first row gives the negative log likelihood (NLL) of the data, given a model; lower values indicate better fit. Our models were trained to minimize NLL. All of our models outperform random guessing, and the figures improve as our models become more complex. To speed learning of the level-one proposer model $P\{R, P\{R^- \}\}$, we initialized its parameters w_{SB} , w_{OB} , and w_{CS} with the utility function weights of the level-zero model $P\{R^- \}$. We found that simply transposing these weight values from $P\{R^- \}$ to the level-one architecture alone gave marked improvement; indeed, subsequent learning was unable to further improve the level-one model. The improvement in NLL is statistically significant as we move from $P\{\}$ to $P\{R^- \}$, and from $P\{R^- \}$ to $P\{R, P\{R^- \}\}$ ($p < 0.012$ and $p < 0.014$, respectively, using the sign-rank test); the difference between $P\{R, P\{R^- \}\}$ and $P\{R, P\{R, P\{R^- \}\}\}$ is not statistically significant. The second row gives mean squared error. All differences are statistically significant (all $p < 0.0054$) except for that between $P\{R, P\{R^- \}\}$ and $P\{R, P\{R, P\{R^- \}\}\}$.

The third row indicates for each model the highest probability given to an option $O \in \mathcal{O}$; this indicates how peaked a distribution the model produces. The strategic models are more peaked, suggesting a higher confidence. The fourth row indicates the proportion of options in \mathcal{O} that are given

a higher probability by the model than the option actually chosen by the human in the data instance; zero would indicate that the actually chosen option was always most preferred by the model.

Even if the option most preferred by a model is different from the option selected by a human, the model’s choice might still be quite similar to the human’s. Arguably, a direct comparison between human and model proposals is the best way to ascertain the fidelity of our models. We measure similarity between two proposals by their benefit structures. Let O^* be the proposal actually made by a human proposer in a game; let \hat{O} be the proposal most preferred by a proposer model in a game. The average difference in benefit between \hat{O} and O^* is given in rows 5 and 8 of Table 1 for self-benefit (ΔSB) and other-benefit (ΔOB), respectively; a positive number indicates that \hat{O} gives a higher benefit than O^* . The average delta in SB, where ‘self’ is the proposer, shrinks as the models become more complex; these changes are statistically significant: $p < 7e - 09$ (t -test) from $P\{\}$ to $P\{R^- \}$, and $p < 0.026$ from $P\{R^- \}$ to $P\{R, P\{R^- \}\}$. This pattern is not found for OB. Nevertheless, examining the distributions of benefit deltas between \hat{O} and O^* over our hold-out data set, we see that the ΔOB distribution for model $P\{\}$ is multimodal; the most common value for ΔOB is approximately -50 . In contrast, the distributions for the strategic models are more unimodal, and the most common values for ΔOB are near zero. In this sense, the most preferred proposals of the strategic models are better approximations of human behavior.

Rows 6 and 9 give the mean *absolute* differences between the benefit structures of \hat{O} and O^* . Fit improves as the models become more complex. The improvement from model $P\{\}$ to $P\{R^- \}$ is significant: $p < 1.6e - 04$ (sign-rank test) for SB and $p < 0.028$ for OB; the improvement from $P\{R^- \}$ to $P\{R, P\{R^- \}\}$ is weaker: $p < 0.24$ for SB and $p < 0.2$ for OB. Rows 7 and 10 present the average *expected* absolute difference between the benefit structures; these expectations are calculated from the model’s probability distribution over all proposals in \mathcal{O} . Here, the improvement from model $P\{\}$ to $P\{R^- \}$ is statistically weak ($p < 0.39$ for SB and $p < 0.26$ for OB); the improvement from $P\{R^- \}$ to $P\{R, P\{R^- \}\}$, however, is significant ($p < 0.002$ for SB and $p < 0.025$ for OB). Thus, the improvement from $P\{\}$ to $P\{R^- \}$ is stronger when we are concerned with the most likely offer than when we care about the entire distribution; the opposite is the case when we go from $P\{R^- \}$ to $P\{R, P\{R^- \}\}$. In any case, the totality of our measurements suggest that our models improve as they become more complex. The right-most column of Table 1 gives data for the level-two proposer model $P\{R, P\{R, P\{R^- \}\}\}$. Due to the excessive computational cost of training a level-two model, we borrow the parameter weights from our level-zero model $P\{R^- \}$. Moving the weights from level-zero to the level-two architecture also gave a marked improvement over level zero. In several measurements (rows 3, 4, 5, and 8) the level-two model gives the best results; row 6 shows the level-two model to perform worse than $P\{R^- \}$. Overall, the level-two model does not appear to be better than the level-one model; further, the level-two models require much more computer time to operate. The bottom half of Table 1 presents measurements on a second test set, generated from a cohort of subjects that were not used to obtain training data. More complex models generally perform better.

Table 1: Fit of learned models to data test-sets.

	Random	$P\{\}$	$P\{R^-\}$	$P\{R, P\{R^-\}\}$	$P\{R, P\{R, P\{R^-\}\}\}$	
Subject Cohort A (108 Data Points)						
1	Negative Log Likelihood	3.728	3.0023	2.8401	2.7469	2.7544
2	Mean Squared Error	0.9474	0.8647	0.8307	0.8079	0.8083
3	Max Probability	0.027	0.165	0.225	0.225	0.228
4	Rank	N/A	0.165	0.140	0.133	0.133
5	$\overline{\Delta SB}$	N/A	44.1	13.8	9.5	2.0374
6	$ \overline{\Delta SB} $	N/A	49.537	33.889	31.759	34.120
7	$E[\overline{\Delta SB}]$	55.205	29.972	28.427	26.705	26.802
8	$\overline{\Delta OB}$	N/A	-10.6	13.2	19.2	-10.095
9	$ \overline{\Delta OB} $	N/A	56.898	48.056	45.000	45.787
10	$E[\overline{\Delta OB}]$	49.782	44.389	40.156	38.816	38.866
Subject Cohort B (120 Data Points)						
11	Negative Log Likelihood	3.6905	3.0204	2.9255	2.9059	2.9005
12	Mean Squared Error	0.9445	0.8651	0.8327	0.8137	0.8139
13	Max Probability	0.028	0.169	0.190	0.213	0.214
14	Rank	N/A	0.169	0.206	0.196	0.197
15	$\overline{\Delta SB}$	N/A	60.1	22.5	10.9	8.6531
16	$ \overline{\Delta SB} $	N/A	66.333	42.333	38.792	38.875
17	$E[\overline{\Delta SB}]$	46.556	31.686	28.686	27.741	27.797
18	$\overline{\Delta OB}$	N/A	-41.5	-11.5	6.3	-22.1217
19	$ \overline{\Delta OB} $	N/A	57.792	47.958	35.583	35.667
20	$E[\overline{\Delta OB}]$	43.319	34.683	39.633	36.303	36.540

6.2 Agent Performance

Using models $P\{\}$, $P\{R^-\}$, and $P\{R, P\{R^-\}\}$, we construct six computer agents to play the role of proposer 1. Each model is used to build an emulator agent and a best response (BR) agent. We then deploy these computer agents in follow-up human-subjects trials to test their performance. These trials are comprised of seven parallel games that each share the same initial game state. Each of the seven games has a different agent playing proposer 1: our six computer agents and one human. The responder and proposer 2 are human players. To make the most of our follow-up trials, we use initial game states (which specify board colors, chip distributions, and locations of the goal and player pieces) that cause at least the three emulators to make different proposals (often the BR agents make different offers, as well).

Table 2 summarizes the performance of our agents in the follow-up sessions. Each column of data corresponds to one of the seven agent types playing the role of proposer 1 (human, three emulator agents, and three BR agents). Row 1 indicates the expected number of offers made by an agent that were accepted by the responder. Note that the values in these rows are not integers. When two or more of the players playing the role of proposer 1 make the same offer, it is often the case that the responders in the respective games make different decisions; some responders accept the offer made by proposer 1, while others reject it. In such situations, we average the outcome over all players in the role of proposer 1 that make the same offer. Row 2 gives the expected total benefit to the player in the role of proposer 1. Row 3 gives the expected total benefit earned by the responder over all the offers it accepts from the agent playing proposer 1.

The performance of the emulator agents improves as they use models that better fit our human data. Further, the worst and best emulator agents were the worst and best of the entire set of players in the role of proposer 1. The worst performance was obtained from the emulator agent

using model $P\{\}$; this agent had by far the fewest offers accepted, earned the least total benefit, and also provided the least total benefit to the responder. The best performance came from the emulator using model $P\{R, P\{R^-\}\}$; this agent had the most offers accepted (over half—25 offers in 45 games), obtained the greatest total benefit, and provided the most benefit to the responder. Note also that our best emulator’s performance closely matches that of humans who played the role of proposer 1.

The difference in performance amongst our best-response agents is much smaller. Our worst model, $P\{\}$, produced the weakest BR agent. This BR agent viewed the other proposer as being reflexive. In contrast, when the BR agent views the other proposer as being strategic, performance improves. The two BR agents using models $P\{R^-\}$ and $P\{R, P\{R^-\}\}$ perform nearly the same. These two BR agents have fewer accepted offers than the human players, yet manage to earn nearly as much total benefit for themselves; at the same time, the BR agents provide little total benefit to the responder. Thus, the style of play shown by the BR agents is to make aggressive offers; these agents earn much more per *accepted* offer. Notably, even the best BR agent performed much worse than the best emulator, overall.

Given that the BR agents are designed to formulate best responses to the expected behavior of human players, we may be surprised that they did not perform better. In particular, the BR agent that uses model $P\{R^-\}$ to reason about the other proposer is employing the same *pattern* of reasoning as the emulator agent that follows the advice of model $P\{R, P\{R^-\}\}$. Both of these computer agents think about the other proposer using model $P\{R^-\}$, and both think about the responder using model R . Both agents will deterministically pick the move that maximizes expected utility. The only difference between these two agents is the utility function that is used. In the emulator agent, the utility function is that of model $P\{R, P\{R^-\}\}$, which is

Table 2: Performance of computer agents in follow-up sessions (45 Game States; 420 Data Instances).

Proposer 1 Played by:	Human	Emulator Agent			Best Response Agent		
Model Used by Agent:	N/A	$P\{\}$	$P\{R^-\}$	$P\{R, P\{R^-\}\}$	$P\{\}$	$P\{R^-\}$	$P\{R, P\{R^-\}\}$
1 $E[\#\text{accepted offers}]$	24.91	7.35	19.83	25.89	13.99	15.24	15.24
2 $E[\text{SB}]$	1692.75	574.36	1325.00	1738.99	1426.68	1574.60	1531.26
3 $E[\text{OB}]$	1851.00	162.93	1417.50	1869.58	527.26	571.85	497.68

learned from human data; in the BR agent, the utility function is selfish, i.e., to maximize the agent’s expected benefit (SB). Yet, the BR agent earned some 250 points less in total benefit than the emulator. This fact suggests that, for the purpose of formulating a best response, our model of responder behavior R did not generalize well to this cohort of human subjects. Nevertheless, the model R more than sufficed for the purpose of emulation. Model R misleads the BR agent into thinking that aggressive offers will be accepted; in contrast, the emulator agent prefers not to make aggressive offers, to begin with.

7. CONCLUSION

Our paper concerns strategic human reasoning about others under uncertainty and its implications for the design of computer agents intended to interact with humans in multi-agent settings. We conduct human-subjects trials in which people play a three-player partial-information negotiation game. We then construct a hierarchy of models to investigate how people reason. After training these models from human data, we deploy them in computer agents that interact with humans in follow-up trials. Our data indicate the following: 1) Human players are not reflexive; that is, humans do not base their decisions only upon the options they have. Instead, humans also reason about the other players in the game. 2) Human players reason about *both* of the other players in our game, and reason under uncertainty. 3) Models of human behavior that better fit our data enabled more successful computer agents. We can use our models to build effective computer agents that not only perform well in terms of the scores they earn, but also contribute to the social good by providing high utility to others. 4) Beyond a certain level of sophistication, more complex models yield diminishing returns and so may not be worth the additional computational effort. 5) Agents that emulate human behavior may be more robust to variations in behavior over different cohorts of humans than agents that play a best response to models of human behavior. Best response may be a more risky approach to playing our game.

If we want to build computer agents that can interact successfully with humans in a multiagent setting, then we must know something about how humans behave. In this paper, we have demonstrated that models of human reasoning about others can be effectively leveraged to construct successful computer agents. Our computer agents successfully emulated human performance. Similar demonstrations in other domains will be useful future work.

8. ACKNOWLEDGMENTS

The research reported in this paper was supported in part by NSF grant CNS-0453923 and AFOSR grant FA9550-05-1-0321. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF, AFOSR, or the U.S. Government.

9. REFERENCES

- [1] T. Bosse, Z. Memon, and J. Treur. A two-level BDI-agent model for theory of mind and its use in social manipulation. In *AISB 2007 Workshop on Mindful Environments*, 2007.
- [2] C. F. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton Univ. Press, 2003.
- [3] A. Falk, E. Fehr, and U. Fischbacher. On the nature of fair behavior. *Economic Inquiry*, 41(1):20–26, 2003.
- [4] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1998.
- [5] Y. Gal, B. J. Grosz, A. Pfeffer, S. M. Shieber, and A. Allain. The influence of task contexts on the decision-making of humans and computers. In *Proc. Sixth International and Interdisciplinary Conference on Modeling and Using Context*, 2007.
- [6] Y. Gal and A. Pfeffer. Predicting people’s bidding behavior in negotiation. In *AAMAS*, 2006.
- [7] Y. Gal, A. Pfeffer, F. Marzo, and B. J. Grosz. Learning social preferences in games. In *National Conference on Artificial Intelligence (AAAI)*, 2004.
- [8] P. J. Gmytrasiewicz and E. H. Durfee. Rational communication in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, 4(3):233–272, 2001.
- [9] J. Gratch and S. Marsella. Evaluating a computational model of emotion. *Autonomous Agents and Multi-Agent Systems*, 11(1):23–43, 2005.
- [10] B. J. Grosz, S. Kraus, S. Talman, B. Stossel, and M. Havlin. The influence of social dependencies on decision-making: Initial investigations with a new game. In *AAMAS*, 2004.
- [11] S. Kraus. *Strategic Negotiation in Multiagent Environments*. MIT Press, 2001.
- [12] C. Lueg and R. Pfeifer. Cognition, situatedness, and situated design. In *Conf. on Cognitive Tech.*, 1997.
- [13] S. Marsell, D. Pynadath, and S. Read. Psychsim: Agent-based modeling of social interactions and influence. In *ICCM 2004*, 2004.
- [14] J. Nash. The bargaining problem. *Econometrica*, 18:155–162, 1950.
- [15] M. Rabin. Psychology and economics. *Journal of Economic Literature*, 36:11–46, 1998.
- [16] M. Seif El Nasr, J. Yen, and T. R. Ioerger. Flame—fuzzy logic adaptive model of emotions. *Autonomous Agents and Multi-Agent Systems*, 3:219–257, 2000.
- [17] J. M. Vidal and E. H. Durfee. Recursive agent modeling using limited rationality. In *International Conference on Multi-Agent Systems*, 1995.
- [18] G. Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 2000.