Reinforcement Learning for Adaptive Theory of Mind in the Sigma Cognitive Architecture

David V. Pynadath[1], Paul S. Rosenbloom[1,2], and Stacy C. Marsella[3]

[1] Institute for Creative Technologies
[2] Department of Computer Science
University of Southern California, Los Angeles CA, USA
[3] Northeastern University, Boston MA, USA

Abstract. One of the most common applications of human intelligence is social interaction, where people must make effective decisions despite uncertainty about the potential behavior of others around them. Reinforcement learning (RL) provides one method for agents to acquire knowledge about such interactions. We investigate different methods of multiagent reinforcement learning within the Sigma cognitive architecture. We leverage Sigma's architectural mechanism for gradient descent to realize four different approaches to multiagent learning: (1) with no explicit model of the other agent, (2) with a model of the other agent as following an unknown stationary policy, (3) with prior knowledge of the other agent's possible reward functions, and (4) through inverse reinforcement learning (IRL) of the other agent's reward function. While the first three variations re-create existing approaches from the literature, the fourth represents a novel combination of RL and IRL for social decision-making. We show how all four styles of adaptive Theory of Mind are realized through Sigma's same gradient descent algorithm, and we illustrate their behavior within an abstract negotiation task.

1 Introduction

Human intelligence faces the daily challenge of interacting with other people. To make effective decisions, people must form beliefs about others and generate expectations about the behavior of others to inform their own behavior. This cognitive capacity for *Theory of Mind* distinguishes social interaction from the decision-making that people do in isolation [21]. We therefore expect that a system capable of artificial general intelligence (AGI) would provide natural support for Theory of Mind. We are interested here in how Theory of Mind capabilities may be realized within *Sigma* (Σ), a nascent *cognitive system*—an integrated computational model of intelligent behavior—that is grounded in a *cognitive architecture*, a model of the fixed structure underlying a cognitive system [9].

In prior work, we have demonstrated this architecture's ability to support Theory of Mind within canonical examples from the game theory literature [13]. However, the games previously investigated made each agent's payoff functions common knowledge to both sides, a luxury not afforded in most social interactions. *Reinforcement learning* (RL) has proven a successful method for agents to make effective decisions in the

B. Goertzel et al. (Eds.): AGI 2014, LNAI 8598, pp. 143–154, 2014.
© Springer International Publishing Switzerland 2014

face of such uncertainty [19]. It is thus not surprising that the multiagent literature has tried a variety of knowledge structures and learning mechanisms to implement decision-making in such interactive environments [3].

Sigma's cognitive architecture provides a mechanism for gradient descent to update functional representations of an agent's knowledge, a mechanism that has supported a capability for reinforcement learning in prior work [15,17]. Here, we reuse this same general learning mechanism across different knowledge representations of a social interaction to arrive at four different models of the agent with which it is interacting:

Section 4: without explicitly modeling the other agent [10]
Section 5: with a stationary policy model of the other agent (a 1-level agent [5,6])
Section 6: with a set of possible reward functions for the other agent [4,12]
Section 7: by *inverse reinforcement learning* (IRL) of the other agent's reward [11]

The Sigma agent is able to leverage the same gradient-descent mechanism in learning these various models. It also leverages gradient descent to learn its own policy of behavior based on its current model of the other agent. In realizing the first three variations, we re-create existing multiagent decision-making mechanisms from the literature. In the fourth variation, we arrive at a novel combination of RL and IRL for multiagent decision-making. Thus, by examining permutations in the application of Sigma's gradient-descent mechanism, we are able to explore a broad space of multiagent learning, without any changes to the underlying cognitive architecture.

We examine the behavior of each of these variations within an abstract negotiation task (described in Section 3). By analyzing the resulting behaviors when interacting with the same fixed agents, we can compare the different models used (described in Section 8). We are thus able to show how the Sigma cognitive architecture can realize diverse adaptive social behavior by leveraging the same reinforcement learning mechanism with different models at the knowledge level.

2 Sigma

Sigma's cognitive architecture is built upon *graphical models* [7]. Graphical models provide a general computational technique for efficient computation with complex multivariate functions by leveraging forms of independence. Sigma leverages this generality through a core knowledge structure—the *conditional*—that provides a deep blending of the forms of conditionality found in both rules and probabilistic networks.

Sigma's long-term memory comprises a set of these conditionals, which are jointly compiled into a single *factor graph* [8] at the level below. Memory access occurs by passing messages in this graph, via the *summary product algorithm* [8], until quiescence; that is, until there are no more messages to send. Each message is an n-dimensional piecewise linear function that is defined over an array of rectilinear regions. These piecewise linear functions can approximate arbitrary continuous functions as closely as desired, as well as be restricted to represent both discrete probability distributions and relational symbol structures. Working memory consists of a set of peripheral nodes in the graph that provide fixed *evidence* during solution of the long-term-memory graph.

The Sigma cognitive architecture provides a model of sequential action, during which operators are selected and applied to achieve goals (or maximize utilities) [14]. The core cognitive (decision) cycle in Sigma involves message passing until quiescence, with the results then used in deciding how to modify working memory. In prior work, we have demonstrated Sigma's ability to support multiple cognitive capabilities, such as problem solving [14], mental imagery [16], Theory of Mind [13], and learning [15].

Learning occurs by altering functions in conditionals at decision time. Sigma's learning mechanism was inspired by earlier work showing that gradient descent was possible in Bayesian networks, much as in neural networks, but without the need for an additional backpropagation mechanism [18]. In Sigma, gradient-descent learning modifies conditional functions, by interpreting incoming messages as gradients that are to be normalized, multiplied by the learning rate, and added to the existing function [17].

Reinforcement learning within Sigma leverages gradient descent to learn Q values over multiple trials, given appropriate conditionals to structure the computation as needed. One conditional proposes actions for selection, weighted by the Q values learned so far. To enable Q values to determine which action to choose, this proposal conditional is augmented to use them as operator weights (alternatively viewed as numeric preferences). For example, if the initial Q values are uniform, then the agent will choose its actions randomly in the early stages of learning.

If direct evidence were provided for the Q values, it would be trivial to use gradient descent to learn them without needing to invoke reinforcement learning. However, without such evidence, RL is the means by which rewards earned in later steps propagate backwards to serve as input for learning Q values for earlier steps. This occurs via a combination of: (1) learning to predict local rewards from the externally provided evidence in the current state and (2) learning to predict both future rewards and Q values by propagating backwards the next state's expected reward [17].

To learn to predict future rewards, we add a conditional that examines the current state and operator, along with the predicted next state (as given by the transition function) and that state's predicted local reward and future discounted reward. This conditional leverages an affine transformation to add the next state's predicted local reward to the distribution over its predicted future reward, and a coefficient to discount this sum. RL then results from using the messages that are passed back to the conditional functions as gradients in learning Q values and discounted future rewards.

3 Negotiation

We investigate different applications of Sigma's gradient-descent learning mechanism within a bilateral negotiation task. Two agents, A and B, make a series of offers and counteroffers to arrive at an allocation of two apples and two oranges. On its turn, each agent selects an operator (Operator-A and Operator-B), either making an offer or accepting the current offer on the table. The current offer on the table is represented by a number of Apples and Oranges, the combination of which represent the fruits that A would receive if the offer were accepted. B would receive the fruits *not* allocated to A.

In this paper, we adopt the perspective of A, who receives a Reward-A that increases linearly with the number of total fruits. An agent can offer only one type of fruit at a

time, so both agents' operators, Operator-A and Operator-B, contain: {offer 0 apples, offer 1 apple, offer 2 apples, offer 0 oranges, offer 1 orange, offer 2 oranges, accept}. If either agent chooses to accept the current offer, then the negotiation terminates with a final allocation based on the current offer on the table. It is straightforward to encode this deterministic transition function within Sigma conditionals.

Combining the transition and reward functions can generate the reward for A's action selection one time step into the future. To predict long-term rewards further into the future, A can use its experiences in repeated negotiations to learn Q values, Q-A, across possible state-operator combinations [20]. We represent these Q values as a distribution over [0,10), the same scale as Reward-A. A can learn this distribution by deriving projected future rewards, Projected-A, from the observed immediate rewards and the state transitions it experiences.

To reason about the future rewards that result from B's action, A can use a variety of possible models. In Section 4, A naively treats this transition as simply part of the environment and does no modeling of B's operator selection. In Section 5, A models B as following a stationary policy of behavior (similar to fictitious play [2]) so that it can learn the distribution of actions underlying the state transitions during B's turn.

In Sections 6 and 7, A uses a Theory of Mind to assume that B behaves according to a reward function structured like its own. In Section 6, A assumes that this reward function is drawn from a finite set of candidates. Such prior knowledge is not always available, so, in Section 7, A uses inverse reinforcement learning to update a belief about B's reward function. In both cases, A can derive a policy of behavior from a hypothesized reward function for B, and use it to generate an expected probability distribution over B's operator selection and the implied state transitions.

We apply these variations in combination with two possible variations on B:

Cooperative: B wants the same outcome as A (i.e., A has all of the fruit).

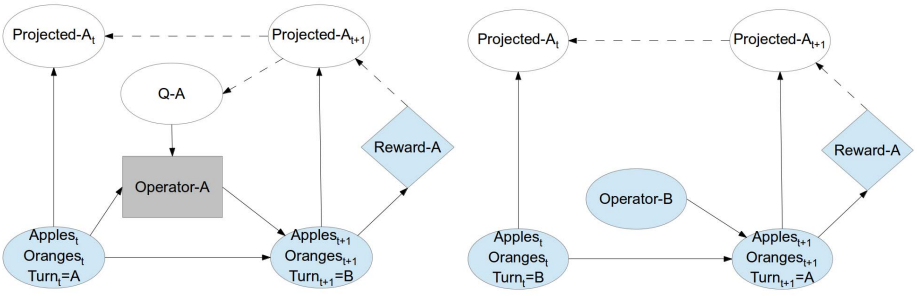
Competitive: B is happy to let A have the oranges, but it wants the apples for itself.

We provide A with a static target by not having B model A in return. In other words, B's behavior is a stationary function of the current offer on the table. The *cooperative* B will accept when the current offer is 2 apples and 2 oranges; otherwise, it will make an offer that moves the negotiation closer to that target offer, with two apples being offered before two oranges. The *competitive* B behaves similarly with respect to its target offer of 0 apples and 2 oranges.[1] Both versions of B make apples a higher priority than oranges, to break up the symmetry that would make many of the negotiation states equivalent. For example, a cooperative B prefers 2 apples and no oranges over no apples and 2 oranges. To complicate A's learning problem further, we make B's behavior nondeterministic by introducing a 10% chance of deviating from the optimal action to one of its suboptimal actions.

[1] A truly competitive B that wanted both the oranges and apples would be an uninteresting opponent, in that any action by A would not move B from its target offer of 0 apples and 0 oranges.

4 Learning with No Model of the Other Agent

We first build a Sigma agent that learns its Q values without any explicit modeling of B's behavior [10]. Figures 1a and 1b show an influence diagram representation of the conditionals that represent A's reasoning during A's and B's turns, respectively. On its turn, in Figure 1a, A uses its current Q values (Q-A) to inform its selection of Operator-A (a rectangular decision node) given the current state of the negotiation (Apples_t, etc.). It then observes the subsequent state (Apples_{t+1}, etc.) and the reward that results (Reward-A, a diamond-shaped utility node).



(a) on A's own turn (b) on B's turn

Fig. 1. Graph for computing and learning A's projected rewards with no model of B

In our negotiation model, the agents receive reward only when an offer is accepted by either party. A propagates this final reward back over the offer on the table (the dashed line from Reward-A to Projected-A_{t+1} in Figure 1a) and then backs up those values to the state leading to the final accepted offer (the dashed line from Projected-A_{t+1} to Projected-A_t). The messages that are passed back to the conditional functions then serve as gradients in learning the projected future rewards [17]. Similar gradient descent using the messages passed along the dashed line from Projected-A_{t+1} to Q-A then supports learning the Q values from these projected future rewards.

Figure 1b shows the graph for the rewards that A can expect when it is B's turn. In this case, the operator is an observation from A's perspective (as opposed to its own decision). The message passing and gradient descent proceed just as on A's turn.

As A encounters various negotiation states, Sigma's conditionals will generate messages for the current state and selected operator, leading to updates to the projected future rewards and, on A's turn, the Q values for those states. The agent can then use the learned Q values in its operator selection process by translating the distribution over Q values into an expected value for candidate operators, as shown in the forward direction in Figure 1a. A then chooses its operator through a *softmax* over these expected values, allowing it to choose high-value operators with high probability, while also allowing it to explore the negotiation space during learning.

Table 1 shows the expectation over A's Q values after 1000 cycles of interacting with a cooperative B, averaged over 60 runs. The boldfaced number represents the operator

having the highest expected Q value in each state. For the most part, A has learned to accept when the offer on the table reaches its best possible outcome: two apples and two oranges. However, the other operators in that state also have high values, because, in the cooperative setting, even if A changes the offer, B will seek to re-establish it on its subsequent turn. The optimal actions in the other states (offering either two apples or two oranges) all have similarly high values, although there are some states where another operator has a higher value. Again, B's cooperative nature means that accepting a suboptimal offer is the only really "bad" choice, but this learning method has still demonstrated an effective ability to learn discriminatory Q values across its operators.

Table 1. Q values with no explicit model of B, who is cooperative

Apples, Oranges	0,0	0,1	0,2	1,0	1,1	1,2	2,0	2,1	2,2
Q(Apples, Oranges, accept)	4.88	5.01	5.62	4.72	5.07	6.59	4.00	5.68	**9.49**
Q(Apples, Oranges, offer-0-apples)	6.05	5.85	5.74	5.39	6.50	7.26	6.39	7.20	9.46
Q(Apples, Oranges, offer-0-oranges)	5.54	5.39	5.28	6.01	5.80	6.72	**7.56**	8.00	9.46
Q(Apples, Oranges, offer-1-apples)	5.72	5.96	6.57	5.89	5.77	**7.72**	6.56	8.17	9.48
Q(Apples, Oranges, offer-1-oranges)	5.81	5.96	5.30	**6.23**	6.59	7.01	7.50	8.25	9.46
Q(Apples, Oranges, offer-2-apples)	**6.58**	**6.24**	6.27	5.88	**6.72**	7.62	7.22	**8.69**	9.49
Q(Apples, Oranges, offer-2-oranges)	5.67	5.43	**6.84**	5.93	6.21	7.15	7.47	8.55	9.49

5 Learning with a Probabilistic Model of the Other Agent

A's update of its projected future rewards in Figure 1b does not maintain any information about the likelihood of the observed transitions on B's turn. Early work on multiagent learning instead leveraged an assumption that other agents may be following a stationary policy of behavior, represented as a probability distribution over operator selection conditioned on the current state [5,6]. In this section, we enable A to model B as such a *0-level* agent by learning a stationary policy of B's behavior. We can represent this policy as a probability distribution, π-B, over B's operator selection, as a function of the current offer on the table. Figure 2 shows the graphical representation of the modified conditionals for A's computation that include this policy. On B's turn, A receives direct evidence of B's behavior that it uses to update the functional representation of π-B (the dashed line from Operator-B in Figure 2).

A can now use its beliefs about B's policy to update its projected future rewards. As shown in Figure 2, the backup of Projected-A_{t+1} to Projected-A_t is now weighted by the probability of the observed action, as specified by π-B. Thus, A can minimize the impact that observations of B's unlikely suboptimal actions will have on its learning. In such cases, the low probability for π-B of Operator-B will scale the message along the link to Projected-A_t. This scaling allows A to incorporate its accumulated expectations about B into its learning of Q values associated with the state of the negotiation. On A's own turn, these expectations provide no information, so A backs up future rewards just as in Figure 1a.

Table 2 shows the probability of B's true optimal action (which it chooses with 90% probability) within A's learned π-B policy. Across all states and both versions of B,

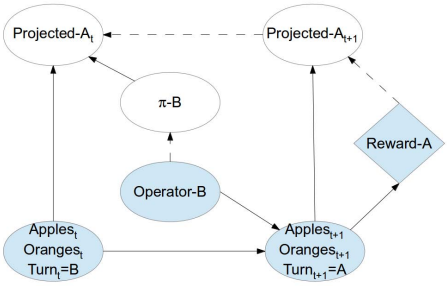


Fig. 2. Graph for A's projected rewards on B's turn with a stationary policy model of B

the optimal action has the highest probability. However, even after 1000 cycles, the values still deviate a significant amount from the correct 90% value. We postpone an investigation of the impact of this learning on A's performance until Section 8.

Table 2. Learned probability of B's optimal action

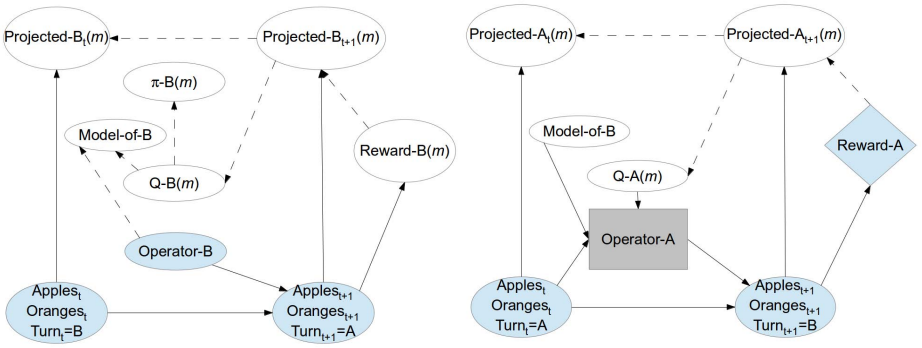
Apples,Oranges	0,0	0,1	0,2	1,0	1,1	1,2	2,0	2,1	2,2
π-B(Operator-B*) in cooperative	0.64	0.63	0.62	0.42	0.49	0.56	0.57	0.64	0.53
π-B(Operator-B*) in competitive	0.70	0.58	0.44	0.60	0.55	0.59	0.41	0.55	0.63

6 Learning with a Set of Reward Functions for the Other Agent

Although the modified conditionals of Section 5 explicitly model B, they treat it as an unknown stochastic process, rather than using a Theory of Mind to treat it as an agent-driven one. For example, A's Theory of Mind might inform it that B is also seeking to maximize its own reward function. We can hypothesize that B belongs to one of a finite set of possible models, each with a corresponding reward function. Such a representation would correspond to Bayesian approaches to multiagent modeling that use sets of models to specify the beliefs that one agent has about another [4,12].

To represent such models, we introduce variables for the Reward, Projected future rewards, and Q values of B, analogous to A's. However, these variables have an additional parameter, $m \in$ Model-of-B, representing the possible reward functions of B (e.g., *cooperative* vs. *competitive*). On B's turn, A will observe Operator-B and the resulting negotiation state. Figure 3a is a graph of A's model of B's decision-making analogous to Figure 1a for A's. Using this graph, A can deterministically derive the values of Reward-B from each of the candidate reward functions applied in the resulting state. It can then propagate that derived value back to Projected-B, which is propagated, in turn, back to Q values for B (Q-B), all contingent on each candidate model, m.

Sigma translates these learned Q values for each $m \in$ Model-of-B into policies of behavior for B: π-B, as in Section 5. Here, rather than learning the distribution directly, A uses a softmax to translate the Q values in Operator-B into a Boltzmann distribution



(a) Projected-B on B's turn (b) Projected-A on A's own turn

Fig. 3. Graph for projected rewards with two hypothesized reward functions for B

over selection, conditioned on the current state. A can incorporate this distribution into updating its projected future rewards on B's turn as previously illustrated in Figure 2.

Given Q-B(m) for each operator, A can update its beliefs about the likelihood of these models. The dashed lines to Model-of-B in Figure 3a represent the messages that translate the observed Operator-B and the Q-B values for that observation under each model, m, into a gradient over the distribution over models. For example, the models for which the observed action has higher Q values will have an increased likelihood in the posterior distribution. A will incorporate this distribution into an expected value over Q-A across the possible values for Model-of-B (shown in Figure 3b).

We gave A a Model-of-B containing our two candidate reward functions (cooperative and competitive) and let it interact with B of both types. After 1000 cycles, A's belief over Model-of-B is so certain that the likelihood of the incorrect model is less than 0.1%. We again postpone an investigation of the impact of this modeling on A's performance until Section 8.

7 Learning with IRL of the Other's Reward

We will not always have a priori knowledge of another agent's possible reward functions. However, we could assume that B has *some* fixed reward function that is guiding a rational decision-making process. *Inverse reinforcement learning* leverages such an assumption into an ability to reason backward from observed behavior to the observed agent's reward function [11]. Existing agents have used IRL to mimic an observed agent's behavior by inferring its underlying reward function [1]. In this work, we adapt this same mechanism within Sigma to arrive at a novel application of IRL to learning a policy for interacting with another (potentially adversarial) agent.

The graph in Figure 4 illustrates this IRL process in Sigma. A uses observations of Operator-B to learn a frequency count, π-B, just as in Section 5's stationary policy model. However, rather than use this policy directly, we leverage our knowledge that there is an underlying optimization process generating this policy. Via a process that

required a small extension to Sigma, the action likelihoods of π-B are re-interpreted in terms of the expected values of Q distributions. We then use a softmax to translate these Q values into a new Boltzmann distribution, π-Q-B (similar to Section 6's version of π-B).

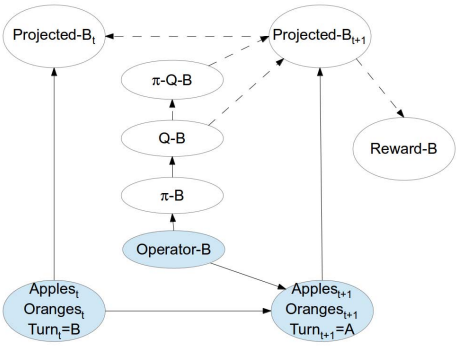


Fig. 4. Graph for IRL on B's turn

Just as A backs up projected reward in a future state onto the Q value for the operator that led to that state, we can now reason in the inverse direction for B. In other words, the learned Q value for B's selected operator implies the projected future reward in the resulting state. However, because of the potential noise and error in this projection, we weight each update by the likelihood, π-Q-B, of the observed operator.

To infer B's reward from these projected future rewards, we exploit the fact that when an agent accepts an offer, both agents receive an immediate reward and the negotiation terminates. Therefore, when B accepts an offer, its projected future rewards include the immediate reward from the accepted offer and nothing else, because there are no more future states. By observing the offers that B accepts, A can thus propagate its Projected-B values to a Reward-B model.

Table 3 presents the expectation over the reward function that A's IRL generates after 1000 cycles with cooperative and competitive Bs. The bold entries show that the learned Reward-B successfully identifies B's optimal outcomes. Furthermore, both versions of B value apples (positively or negatively) more than oranges, which the learned Reward-B captures as well. In particular, the cooperative entries with two apples and the competitive entries with zero constitute six out of the seven highest expected rewards. We will examine the impact of IRL on A's performance in the next section.

Table 3. A's expectation about B's reward function after 1000 cycles in IRL agent

Apples,Oranges	0,0	0,1	0,2	1,0	1,1	1,2	2,0	2,1	2,2
cooperative	0.23	0.27	0.19	0.22	0.23	0.25	0.25	0.36	**0.41**
competitive	0.26	0.36	**0.40**	0.21	0.22	0.25	0.23	0.24	0.20

8 Experimental Comparison

We can compare the behavior of the four versions of A from Sections 4–7 within the same context by measuring cumulative statistics over their interactions with our two versions of B. To measure the flexibility of the learned policies, we also examine how effectively each variation of A can transfer what it has learned to a new situation. Therefore, after the first 1000 cycles with a cooperative (competitive) B, we switch B's reward function to be competitive (cooperative), and let A continue interacting for 250 more cycles.

Table 4. Comparison of four variations vs. cooperative and competitive versions of B

Modeling	No Model	Stochastic Policy	Reward Set	IRL
msgs/decision	445	483	675	587
msec/cycle	306	309	1,343	560
A's Reward (vs. cooperative B)	7.11	7.13	7.12	**7.17**
A's Reward (after switch)	5.82	5.80	**5.85**	5.82
A's Reward (vs. competitive B)	5.88	**5.88**	5.83	5.85
A's Reward (after switch)	7.00	6.96	**7.08**	6.99

Table 4 shows that moving from the "no model" case of Section 4 to the "stochastic policy" model of Section 5 has minimal impact on the number of messages and amount of time required for the graph to reach quiescence. On the other hand, when including a model of B's reward function (Section 6's "reward set" and Section 7's "IRL" variations), the bigger graphs lead to significant increases in the messages and time required. The "reward set" variation requires the most messages and time, due to the explicit Model-of-B and the parameterization of Reward-B, Projected-B, Q-B, and π-B as a function of the two possible models. The "IRL" variation avoids this doubling in the size of messages regarding the B nodes, so it is able to model B's reward function without as much of an increase in computational time incurred by the "reward set" method.

Table 4 also shows that all four methods achieve roughly the same reward, with a dropoff between the cooperative and competitive cases. We expect such a dropoff because of B's unwillingness to accept or offer A's optimal outcome in the latter case. Given the differences among our methods, the comparable performance is a bit surprising. On the other hand, all four variations use a similar structure for A's Q values, and B's stationary policy lends itself to RL. Sigma's message passing quickly converges to the best response, regardless of the knowledge representation. B's behavior is thus not complicated enough to draw out deeper differences among our learning methods.

The midstream switch of B from cooperative to competitive (or vice versa) does elicit some distinguishing characteristics. Table 4 shows that the "reward set" version of A does slightly better than the other three after both switches. During the first 1000 cycles, this variation has already learned Q-B for both cooperative and competitive Bs. It can thus quickly detect the switch through RL of Model-of-B and use its Q-B(m) to learn a modified Q-A. On the other hand, this advantage is still relatively small. Further analysis is thus needed to gain more insight into the differences across these algorithms.

9 Conclusion

We have applied Sigma's architectural capability for gradient descent across four different models of multiagent learning. Three of these models re-create multiagent RL algorithms from the literature, while the fourth represents a novel combination of RL and IRL for decision-making in combination with Theory of Mind about another agent. The four permutations of graphs represent only a subset of the possible structures that we could experiment with. By introducing new nodes or changing the link structure, we can alter A's knowledge-level representation. Re-application of the same architectural mechanism will enable reinforcement learning of that knowledge, allowing us to easily study the impact that the modified graph has on multiagent interaction.

Our current experimental domain was able to validate the correctness of our various algorithms. However, an enriched domain would provide more insight into the differential impact of the various modeling assumptions. In particular, B's stationary policy of behavior forms a static target for A to learn. One very easy way to achieve a more realistic multiagent setting would be to expand our experiments into a true multiagent setting, by giving B the same learning capabilities we gave to A. By assigning different combinations of our learning algorithms to A and B, we will be able to better differentiate their adaptive behavior by applying them to a more complex target agent. With this enriched experimental setting, Sigma's general capability for multiagent learning opens up a promising line of future research into RL-based Theory of Mind.

Acknowledgments. This work has been sponsored by the Office of Naval Research and the U.S. Army. Statements and opinions expressed do not necessarily reflect the position or policy of the U.S. Government.

References

1. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: ICML, pp. 1–8. ACM (2004)
2. Brown, G.W.: Iterative solution of games by fictitious play. Activity Analysis of Production and Allocation 13(1), 374–376 (1951)
3. Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics 38(2), 156–172 (2008)
4. Gmytrasiewicz, P., Doshi, P.: A framework for sequential planning in multi-agent settings. JAIR 24, 49–79 (2005)
5. Hu, J., Wellman, M.P.: Multiagent reinforcement learning: theoretical framework and an algorithm. In: ICML, pp. 242–250 (1998)
6. Hu, J., Wellman, M.P.: Learning about other agents in a dynamic multiagent system. Journal of Cognitive Systems Research 2, 67–79 (2001)
7. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIT Press (2009)
8. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47(2), 498–519 (2001)
9. Langley, P., Laird, J.E., Rogers, S.: Cognitive architectures: Research issues and challenges. Cognitive Systems Research 10(2), 141–160 (2009)

10. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: ICML, vol. 94, pp. 157–163 (1994)
11. Ng, A.Y., Russell, S.J.: Algorithms for inverse reinforcement learning. In: ICML, pp. 663–670 (2000)
12. Pynadath, D.V., Marsella, S.C.: PsychSim: Modeling theory of mind with decision-theoretic agents. In: IJCAI, pp. 1181–1186 (2005)
13. Pynadath, D.V., Rosenbloom, P.S., Marsella, S.C., Li, L.: Modeling two-player games in the Sigma graphical cognitive architecture. In: Kühnberger, K.-U., Rudolph, S., Wang, P. (eds.) AGI 2013. LNCS (LNAI), vol. 7999, pp. 98–108. Springer, Heidelberg (2013)
14. Rosenbloom, P.S.: From memory to problem solving: Mechanism reuse in a graphical cognitive architecture. In: Schmidhuber, J., Thórisson, K.R., Looks, M. (eds.) AGI 2011. LNCS (LNAI), vol. 6830, pp. 143–152. Springer, Heidelberg (2011)
15. Rosenbloom, P.S.: Deconstructing reinforcement learning in Sigma. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS (LNAI), vol. 7716, pp. 262–271. Springer, Heidelberg (2012)
16. Rosenbloom, P.S.: Extending mental imagery in Sigma. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS (LNAI), vol. 7716, pp. 272–281. Springer, Heidelberg (2012)
17. Rosenbloom, P.S., Demski, A., Han, T., Ustun, V.: Learning via gradient descent in Sigma. In: ICCM (2013)
18. Russell, S., Binder, J., Koller, D., Kanazawa, K.: Local learning in probabilistic networks with hidden variables. In: IJCAI, pp. 1146–1152 (1995)
19. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
20. Watkins, C.J., Dayan, P.: Q-learning. Machine Learning 8(3-4), 279–292 (1992)
21. Whiten, A. (ed.): Natural Theories of Mind. Basil Blackwell, Oxford (1991)