Opponent Modeling in a PGM framework

Nicolaj
S?ndberg-Jeppesen
Dept. of Computer Science
Aalborg University
Aalborg, Denmark
nsj@cs.aau.edu

Finn V. Jensen
Dept. of Computer Science
Aalborg University
Aalborg, Denmark
fvj@cs.aau.edu

Yifeng Zeng
Dept. of Computer Science
Aalborg University
Aalborg, Denmark
yfzeng@cs.aau.edu

ABSTRACT

We consider the situation where two agents try to solve their own task in a common environment. In particular, we study a type of sequential Bayesian games with unlimited time horizon where two players share a visible scene, but the tasks (termed *assignments*) of the players are private information. We present a probabilistic graphical model (PGM), together with recursive modeling techniques, for representing this type of games. We introduce the type tree which can be used to calculate policies and to efficiently approximate the opponent's state of belief. Secondly, we propose a method for reasoning with a mixture of models when a true model of opponent agents is unknown in the game due to their private information. We demonstrate its performance in a *Grid* game.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Experimentation

Keywords

multiagent systems, opponent modeling, sequential Bayesian games, probabilistic graphical models

1. INTRODUCTION

It is a central problem in multiagent research to model the reasoning necessary when multiple agents, each with individual objectives, interact in the same environment. While each agent may change the state of the environment towards a more favorable state for itself, other agent's actions may change the state to a less favorable state. When planning under such conditions it is beneficial to take into account the other agent's reasoning.

A solution to the aforementioned planning problem may employ an opponent modeling method that equips each agent with a model which models the other agents until a predefined nesting level is met. Such solutions include recursive modeling methods (RMM) [6], nested influence diagrams (NIDs) [4], interactive POMDPs [5] and its graphical counterpart called interactive dynamic influence diagrams (I-DIDs) [2], and a general framework based on recursive influence diagrams (RIDs) [16]. Particularly, RIDs naturally inherit well-developed probabilistic graphical model (PGM) techniques for handling the problem complexity, and have been supplied with a set of approximate solutions to the planning problem of unlimited time horizons. In this paper, we further improve RIDs on adapting agent behavior to respond to co-inhabiting opponents.

Previous work on RIDs mainly focuses on approximate techniques to alleviate the complexity due to the curse of time horizon [16]. Exploiting that an agent will have partial information about his opponent, the approach reduces the size of the relevant past by using the concept of conditional independence in the structural representation. The RID modeling technique has been applied in a typical sequential Bayesian game where the planning involves interactions of two agents as well as unlimited time horizons. However, the current approach concerns only a single model of opponents. Consequently, the solution may become invalid in a real situation where true models of opponents are often unknown, particularly in a competitive multiagent setting.

In this paper, we adapt RIDs solutions to the planning problem in which an agent needs to consider a set of candidate models of its opponents in the game. We introduce a concise representation, called *type tree*, to recursively model possible types of opponents. The type tree also serves as a computational model in which we can update probability distributions over candidate models of opponents. The update employs a Bayesian mechanism for calculating new distributions given the emerging information about agents' behavior. The probability distribution will then weight policies generated by solving the set of models and the weighted policies become the predicated behavior of opponents in RIDs.

Meanwhile, we observe that a true model of opponents is often unknown in a problem domain. We may use a mixture of models and adapt them to the actual opponent. However, we may get some online behavior that is unexpected from any model in the model space. Consequently, the Bayesian update on the hypothesized model becomes invalid due to the inconsistency in the predicated behavior. Our solution is to add to the mixture a model which assumes random behavior which then becomes the only likely model in case all other models encounter conflicts. We implement a typical

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

sequential Bayesian game, called *Grid*, in which two agents compete for their individual rewards while sharing a common grid environment. We show favorable performance of the proposed techniques in the game and list some observations for further investigation.

2. GAME DESCRIPTION AND NOTATIONS

We consider games with the following characteristics. The *scene* is visible to all players. Each player has a set of *actions*, which have an effect on the state of the scene. The players act concurrently, and the joint effect of the two actions may be non-deterministic. Each player gets a *score*, which is a function of the state of the scene. In case of zero-sum games, the actual gain/loss is the score minus the other player's score. The score function is known only to the player. We call the score function the *assignment* of the player, and the assignment of each player does not change during the game. We assume everything to be finite (scene, actions, assignments). In this paper we consider games with only two players. We shall call this kind of games a *Simple Sequential Bayesian Game (SSBG)*.

Formally, an SSBG consists of two players ♡, ♠ and a *world* W with a finite set of states $\{w_0, w_1, w_2, \ldots, w_n\}$. We assume ♡ to be female and ♠ to be male. The players have finite sets of *moves*, $M^\heartsuit$ and $M^\spadesuit$, which affect W. The transition between world states at time t to time $t+1$ is determined by a probabilistic function τ, $\tau : W \times M^\heartsuit \times M^\spadesuit \times W \rightarrow \mathbf{R}$, where $\tau(w_1, m^\heartsuit, m^\spadesuit, w_2)$ is the probability of W_{t+1} being in state w_2 given that W_t was in state w_1 and the two players make the moves $m^\heartsuit$ and $m^\spadesuit$, respectively. Furthermore, each player draws an assignment from a finite set $\mathcal{A}$. The assignment is a particular score function reflected in utility numbers over states of the world. For the sake of notational convenience we shall throughout the paper assume that ♡ has received the assignment a_1. The structure of the game and the world state is always known by both players, but the actual assignment of the other player remains hidden, and the opponent's moves may also be hidden.

When both players have decided their moves, the game continues with the next time step. There is no prefixed limit on the number of moves, but the time for playing the game is so short that discounting is not relevant. That is, the players aim for maximizing the sum of the utilities gained after each move.

3. RECURSIVE INFLUENCE DIAGRAMS

We start with a brief review on recursive influence diagram (RID) and describe its solutions for addressing the problem of time horizon when determining optimal policies. We refer readers to [16] for more details.

3.1 The RID Representation

An RID is a dynamic influence diagram [17] that models the agent's subjective decision-theoretic reasoning and its reasoning about other agent's reasoning. Figure 1 shows an RID modeling an SSBG as seen in the eyes of ♡. The model consists of 3 influence diagrams, namely the A-Model, B-Model and the λ-Model, representing ♡'s model, ♡'s model of ♠ and ♡'s model of ♠'s model of ♡ respectively.

RIDs follow the same notations and conventions as influence diagrams [7] with chance nodes, decision nodes and utility nodes represented as circular nodes, rectangular nodes and diamond shaped nodes, respectively. The nodes labeled by $A^\heartsuit$ and $A^\spadesuit$ represent the assignments; the nodes labeled with M represent moves; the U-labeled nodes represent score functions, and the Ws represent the world states. The transition function is represented by the conditional probability $P(W_1|W_0, M^\heartsuit, M^\spadesuit)$. Looking at the A-Model, when deciding for the first move ♡ knows her assignment $A^\heartsuit$ and the world state W_0, and when deciding for the second move she will also know W_1 as well as her own first move, $M_1^\heartsuit$.

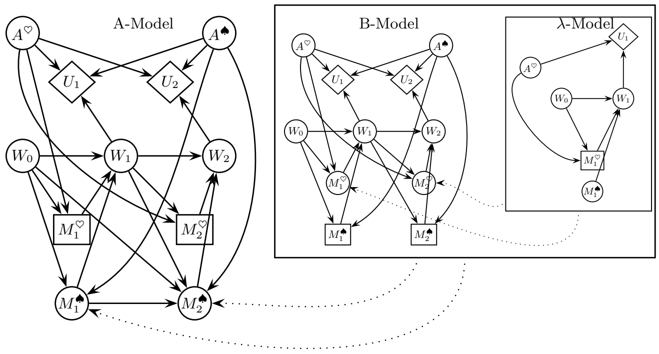


Figure 1: An ID framework for opponent modeling which contains a set of nested models.

The connection between the A-Model, the B-Model and the λ-Model is as follows: in the A-Model, ♡'s own decisions are represented as decision nodes while she represents ♠ decisions as chance nodes. That means that the policy for ♠ in each of his decisions must be represented as a conditional probability distribution. To find these, ♡ consults the B-Model - which in this case is another RID but this time representing the game in the eyes of ♠. In the B-Model, ♠'s decisions are represented as decision nodes and ♡'s decisions as chance nodes. The conditional probability distributions representing ♡'s decisions in the B-Model are found by consulting the λ-Model which in this case is an ordinary influence diagram where an assumption about the opponent's policy has been made, i.e. that ♠ in this case is playing completely at random. The model in the λ-Model can be solved using standard algorithms for solving IDs (see for instance [14, 15, 8]) and when a policy is found in the λ-Model it can be represented as a conditional probability distribution in the $M_1^\heartsuit$ and $M_2^\heartsuit$ nodes of the B-Model. Then we can proceed by solving the B-Model and obtain policies for $M_1^\spadesuit$ and $M_2^\spadesuit$ which are then represented as conditional probability distributions in the $M_1^\spadesuit$ and $M_2^\spadesuit$ nodes of the A-Model. Finally, we can solve the A-Model and get a policy for $M_1^\heartsuit$ and $M_2^\heartsuit$.

3.2 The Player Representation

It is clear from the above discussion that RIDs have inherent assumptions about how many future time steps ♡ is taking into account, how many future time steps ♡ thinks ♠ is taking into account and finally how many future time steps ♡ thinks ♠ thinks ♡ is considering. Furthermore, RIDs have inherent assumptions about how deep ♡'s nesting level is. In Figure 1 the A-Model, the B-Model and the λ-Model takes 2, 2 and 1 future time steps into account respectively, and the nesting level of the A-Model is 3. In order to capture ♡'s nesting level and time horizon in a concise way, we give the following definition (inspired by [1]).

Definition 1. *Player Representation*

A player representation P is a pair defined as follows: NIL represents a level 0 player, which moves completely at random.

1. *$P = (h, NIL)$ represents a player with time horizon h and nesting level 1 (assuming the opponent to be a NIL player).*
2. *Given a player representation O, with nesting level $l-1$ and time horizon q. Let $h \geq q$. Then $P = (h, O)$ represents a player P with time horizon h and nesting level l.*

When obvious from the context, we will use the term 'player' rather than 'player representation'.

For example, Figure 1 represents a (2,(2,(1,NIL))) player where the B-Model represents a (2,(1,NIL)) and the λ-Model represents a (1,NIL) model.

3.3 Solutions

RIDs provide different ways of addressing the curse of time horizon in the models by either removing arcs or adding arcs in the influence diagrams. When removing arcs RIDs use limited memory influence diagrams (LIMIDs) which were proposed in [11] and when new arcs are introduced the idea of information enhancement is used[16]. More specifically, an approximation using information enhancement in SSBGs can be obtained by assuming that at some point in the future, a set of private information is revealed to a relevant player. The technique can be implemented by altering the structural representation of RIDs. By doing this, we can reduce the computational complexity on solving RIDs. For the example of the B-Model in Figure 1, an approximation can be obtained by adding an arc from $A^{\heartsuit}$ to $M_2^{\spadesuit}$, ($\heartsuit$ assumes $\spadesuit$ to know her assignment when deciding for the second move). The result is that W_0 and $M_1^{\spadesuit}$ become irrelevant to the decision made in $M_2^{\spadesuit}$ (see [9] for more information). In experiments with games where different models using LIMIDs and information enhancement respectively were compared to the exact models, the amount of consumed memory was significantly reduced while acceptable performance was still achieved.

4. TYPE TREE

In this section we introduce the type tree. A type tree is a data structure that can be used to represent $\heartsuit$'s probability distribution over $\spadesuit$'s assignment. It can also represent $\heartsuit$'s beliefs of $\spadesuit$'s belief of $\heartsuit$'s assignment etc. We show how to use the type tree to calculate policies, and how to update the type tree when observations are received.

4.1 A Player's type

Each player has private information. In the beginning of the game, the private information is the assignment taken from $\mathcal{A}$. Together with the assignment, each player holds a belief of the opponent's assignment. In our context a belief is a subjective probability distribution. Each player also has a belief of the opponent's beliefs. During the game the players make observations and use them to update their beliefs. We assume that the players update their subjective probabilities in a rational way. Furthermore, the players will during the game collect further private information, namely their own moves. This means that each player should extend their beliefs to cover beliefs of the opponent's moves and the opponent's belief of her moves.

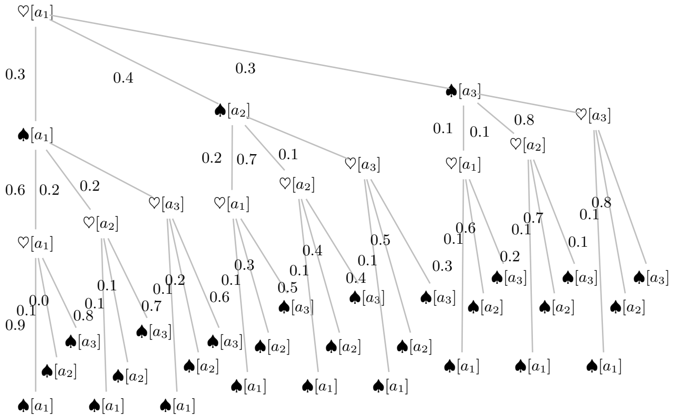


Figure 2: The tree representing the level 3 initial type for $\heartsuit$. $\heartsuit$ knows her own assignment a_1. From the root there is a branch to a sub tree for each possible $\spadesuit$ assignment. The edges are labeled with the probability assigned to each subtree.

We will adopt the notion of *type* from game theory, and by a player's type refer to her private information as well as her beliefs [3].

Definition 2. *Initial Types*

Let O and P be players, and let $\mathcal{A}$ be a finite set of assignments.

A level 1 initial type T for O is a pair $(a, \Delta(\mathcal{A}))$, representing a level 1 player, where

- *i $a \in \mathcal{A}$ is the assignment of O*
- *ii $\Delta(\mathcal{A})$ is a probability distribution over $\mathcal{A}$ representing O's belief of P's assignment.*

If the two players draw their assignments from different sets, it can be represented through 0-probabilities in $\Delta(\mathcal{A})$.

A level l initial type for O is defined recursively as follows:

- *iii Let $\mathcal{U}$ be a set of level $l-1$ initial types for P, one for each member of $\mathcal{A}$; let $\Delta(\mathcal{U})$ be a probability distribution over $\mathcal{U}$, and let a be O's assignment. Then $T = (a, \Delta(\mathcal{U}))$ is a level l initial type for O.*

An initial type can be depicted as a tree. Figure 2 depicts an initial type as a type tree containing a level 3 initial type with $\mathcal{A} = \{a_1, a_2, a_3\}$. The root has been labeled with $\heartsuit$'s assignment, namely a_1. The root has 3 subtrees one for each possible $\spadesuit$ assignment. Each link is labeled with $\heartsuit$'s belief of $\spadesuit$ having the particular assignment. The same is done recursively down from the roots of the subtrees.

A type is a distribution over types, but it is also implicitly a distribution over the opponent's assignments. In Figure 2 this distribution can be read from the edges going out from the root. We shall call this distribution the player's (or type's) *primary belief.* The player's *secondary belief* is her belief of what the opponent believes of her assignment. This can also be read from the type.

From the type in Figure 2 we can easily read $\heartsuit$'s primary belief, namely $P(\mathcal{A}) = (0.3, 0.4, 0.3)$. To find her secondary

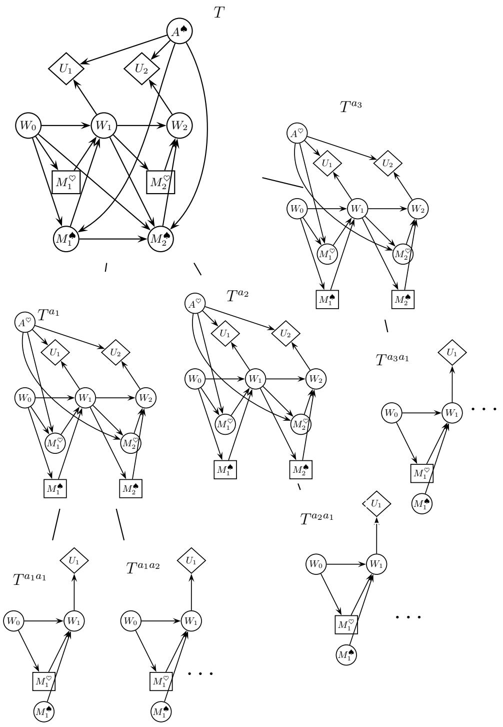


Figure 3: An influence diagram representation of the type tree for player ♡. The tree is used to calculate the ♡'s policies δ_1 and δ_2. Probabilities on the links have been omitted in this figure.

belief, we need to go a bit deeper into the structure. With probability 0.3 ♠ has the belief (0.6, 0.2, 0.2), with probability 0.4 he has the belief (0.2, 0.7, 0.1), and with probability 0.3 he has the belief (0.1, 0.1, 0.8). Altogether, this yields the belief (0.29, 0.37, 0.34).

We will also use *conditional beliefs*: if the opponent has type t, what is then his primary or secondary beliefs? In Figure 2, for example, we have that given that ♠ has assignment a_1 then (♡ believes that) his secondary belief is (0.84, 0.10, 0.06); given the assignments a_2 and a_3 the secondary beliefs are (0.51, 0.10, 0.39), and (0.13, 0.10, 0.77), respectively.

A type holds private information and beliefs relevant for decisions, and in the initial state the assignment is the only private information relevant for deciding the move. As mentioned above, the move performed by the opponent may actually be hidden to the player. Therefore, the types of the players should be extended with moves taken, belief of the opponent's move, belief of what the opponent beliefs, etc. This causes a combinatorial explosion (the curse of current history). We could extend the definition of type to also include the situations after moves. However, we will refrain from that, as we later will show how to use an initial type as an approximation.

4.2 Type tree with RIDs

Each internal node in a type tree represents an acting agent, and we have to find the policy for each of them. So, we attach an influence diagram to each internal node. Figure 3 shows a type tree of influence diagrams representing the RIDs from Figure 1.

4.2.1 Notations

A node in the type tree of IDs has a label (an assignment), and it has an ID attached. A node is identified through a sequence of assignments $s = \{a_{j1} \dots a_{jk}\}$, where the assignment of the root is not included, and the node is denoted T^s. That is, in Figure 3, the node with an ID for ♠ with assignment a_2 is denoted T^{a_2}, and its child node with assignment a_1 is denoted $T^{a_2a_1}$. Note that the label of the node is the last assignment in the sequence s. Note also that if k is even, then the player is ♡, and if k is odd, then the player is ♠. We shall also let T^s refer to the node's attached ID.

The probabilities and policies from the various IDs are also indexed by s. $\delta^{a_2a_3}$ is the policy for $T^{a_2a_3}$, $\delta_1^{a_2}$ is the policy for the first decision in T^{a_2}, and P^{a_3} is a probability table from the ID T^{a_3}.

4.2.2 Solving the type tree with IDs

The tree is solved from the leaves and up to the root. Taking the type tree in Figure 3, we first calculate the nine optimal policies $\delta^{a_ia_j}(W_0)$ in the leaves. They are used to determine $P^{a_i}(M^\heartsuit|W_0, a_j)$:

$$P^{a_i}(M^\heartsuit|W_0, a_j) = \delta^{a_ia_j}(W_0). \qquad (1)$$

Next, the 3 IDs T^{a_i} are solved to provide $\delta_1^{a_i}(W_0)$ and $\delta_2^{a_i}(W_0, W_1, M_1^\spadesuit)$, and we set

$$P(M_1^\spadesuit|W_0, a_i) = \delta_1^{a_i}(W_0) \qquad (2)$$

and

$$P(M_2^\spadesuit|W_0, W_1, M_1^\heartsuit, a_i) = \delta_2^{a_i}(W_0, W_1, M_1^\heartsuit). \qquad (3)$$

Finally, we solve the ID in the root to get the policies $\delta_1(W_0)$ and $\delta_2(W_0, W_1, M_1^\heartsuit)$ (for the assignment a_1).

4.3 Updating the type tree

When both players have taken a move and the resulting world state is observed, they shall exploit the new information to update their beliefs in order to determine their next move. In our setting this means that ♡ has to update her type tree. In the description below we take outset in the structure in Figure 3, but the methods cover all structures.

4.3.1 Public moves

We shall first assume that the players are informed about the opponent's moves. Then the private information does not vary during the game, and the structure of the type is invariant. Hence, ♡ shall only update beliefs of assignments.

At the leaf level there is no updating to do, as at the leaves the opponent is assumed to play completely at random and an even distribution over assignments is always assumed. So, the updating consists in updating her primary belief and the conditional beliefs. With respect to influence diagrams, the updating consists in updating the prior probabilities for $A^\spadesuit$ in the A-model and the priors for $A^\heartsuit$ in the 3 B-models.

Assume now that after the first move ♡ has observed: $d = \{W_0 = w_0, M_1^\heartsuit = m^\heartsuit, W_1 = w_1, M_1^\spadesuit = m^\spadesuit\}$. To determine the new belief of $A^\spadesuit$, she can enter the observations as evidence in the A-model influence diagram, and belief updating in the model provides $P(A^\spadesuit|d)$. The formula for the calculation is

$$P(A^\spadesuit|m^\heartsuit, w_0, w_1, m^\spadesuit) \simeq P(m^\spadesuit|w_0, A^\spadesuit)P(A^\spadesuit), \qquad (4)$$

where $\simeq$ means that the two vectors of numbers are proportional. The right hand side is transformed to a probability distribution by normalizing it: divide each number by the sum of all numbers. Note that the two (conditional) probability tables of the right hand side are part of model A.

The updated probabilities for $A^\spadesuit$ are new labels on the links from the root in the type tree.

Next, she has to update the conditional primary beliefs. Take the subtree T^a, where $\spadesuit$ has the assignment a. $\spadesuit$'s evidence is $e = \{W_0 = w_0, M^\heartsuit = m^\heartsuit, W_1 = w_1, M^\spadesuit = m^\spadesuit\}$. It is entered to the B-model T^a, and propagation yields $P^a(A^\heartsuit|e)$. The calculation is similar to the calculation of the primary belief, and the formula is

$$P^a(A^\heartsuit|w_0, m^\heartsuit, w_1, m^\spadesuit) \simeq P^a(m^\heartsuit|w_0, A^\heartsuit)P^a(A^\heartsuit), \tag{5}$$

where the two probability tables of the right hand side are part of the B-model.

To determine her next move, $\heartsuit$ changes the prior probabilities for the assignments in the influence diagrams and does the same for the first move.

4.3.2 *Private moves*

If the opponent's moves are not disclosed, then her updated belief of $A^\spadesuit$ is based on $c = \{W_0 = w_0, M^\heartsuit = m^\heartsuit, W_1 = w_1\}$. Again, $P(A^\spadesuit|c)$ is found through belief updating in the A-model. The calculation formula is

$$\begin{aligned} P(A^\spadesuit|m^\heartsuit, w_0, w_1) \simeq & \textstyle\sum_{M^\spadesuit} P(w_1|w_0, m^\heartsuit, M^\spadesuit) \\ & P(M^\spadesuit|w_0, A^\spadesuit)P(A^\spadesuit), \end{aligned} \tag{6}$$

where the probability tables of the right hand side are part of the A-model.

Things are more complicated with respect to the conditional beliefs. If $\spadesuit$ has assignment a, he will use the evidence $c^\spadesuit = \{W_0 = w_0, A^\spadesuit = a, W_1 = w_1, M_1^\spadesuit = m^\spadesuit\}$, and he does not know $M_1^\heartsuit$. He would use the B-model to get $P^a(A^\heartsuit|w_0, w_1, m^\spadesuit)$ for which the formula is

$$\begin{aligned} P^a(A^\heartsuit|w_0, w_1, m^\spadesuit) \simeq & \textstyle\sum_{M_1^\heartsuit} P^a(w_1|w_0, m^\spadesuit, M_1^\heartsuit) \\ & P^a(M_1^\heartsuit|w_0, A^\heartsuit)P^a(A^\heartsuit). \end{aligned} \tag{7}$$

However, as $M_1^\spadesuit$ is not disclosed to $\heartsuit$, she cannot perform this calculation. She could condition on $M_1^\spadesuit$ and add nodes to the type tree, but that has to be done after each move, and the type tree will grow exponentially with the number of moves. On the other hand, if the policies are deterministic, then knowing the assignment will suffice. She has his policy represented in the A-model as $P(M_1^\spadesuit|W_0, A^\spadesuit)$, and $\spadesuit$'s calculations can be simulated through the formula

$$\begin{aligned} P^a(A^\heartsuit|w_0, w_1, m^\spadesuit) \simeq & \; P^a(A^\heartsuit) \textstyle\sum_{M_1^\heartsuit} P^a(M_1^\heartsuit|w_0, A^\heartsuit) \\ & \textstyle\sum_{M_1^\spadesuit} P(w_1|w_0, M_1^\spadesuit, M_1^\heartsuit)P(M_1^\spadesuit|w_0, a), \end{aligned} \tag{8}$$

where the 2 first probability tables of the right hand side are part of the B-model for assignment a while the 2 last probability tables are part of the A-model. Note that the evidence $m^\spadesuit$ does not appear at the right hand side.

If $\spadesuit$ has several optimal moves given his assignment and the world state, then $\heartsuit$ cannot perform a correct simulation of $\spadesuit$'s calculation. Instead, she can as an approximation use the average of his estimates: $\spadesuit$ will with probability $P(m^\spadesuit|w_0, a)$ use $P^a(A^\heartsuit|w_0, w_1, m^\spadesuit)$ for the updating. In average, the updating will be

$$\begin{aligned} P^a(A^\heartsuit|w_0, w_1) \simeq & \; P^a(A^\heartsuit) \textstyle\sum_{M_1^\heartsuit} P^a(M_1^\heartsuit|w_0, A^\heartsuit) \\ & \textstyle\sum_{M_1^\spadesuit} P(w_1|w_0, M_1^\spadesuit, M_1^\heartsuit)P(M_1^\spadesuit|w_0, a), \end{aligned} \tag{9}$$

Equation 9 is actually the same formula as the one for deterministic policies. In this way we avoid the curse of current history.

5. MIXTURE MODELS FOR ADAPTATION

So far we have assumed that $\heartsuit$ works with only one opponent model. In reality, she may work with a mixture of models and may not know the true model of $\spadesuit$. To react in an intelligent way, $\heartsuit$ needs to adapt the set of models to the actual behavior of $\spadesuit$. The adaptation becomes straightforward if the true $\spadesuit$ model is contained in the model set. Otherwise, $\heartsuit$ needs to handle the inconsistent observation of $\spadesuit$'s behavior due to his unexpected model.

5.1 General Adaptation

Formally, let $\Gamma_1, \ldots, \Gamma_k$ be models. Then a mixture model can be denoted as

$$\Gamma = \bigoplus_i \mu_i \Gamma_i, \tag{10}$$

where μ_i are positive reals for which $\sum_i \mu_i = 1$.

When calculating the policies in the Γ, you combine the policies from the Γ_i-models. Let $\delta_1, \ldots, \delta_k$ be policies provided by $\Gamma_1, \ldots, \Gamma_k$ respectively, then the combined policy, δ, is

$$\delta = \sum_i \mu_i \delta_i. \tag{11}$$

Updating of beliefs for the Γ_i-models is performed as described previously, and you calculate a combined belief as you calculated combined policies. You may consider the mixture as combining models with their probabilities. That is, μ_i is the modeler's (subjective) probability, $P(\Gamma_i)$, that the opponent plays in accordance with model Γ_i. Now, when information e has been collected, then the probabilities for the various models change. Let $P(e|\Gamma_i)$ denote the probability of the evidence e if $\spadesuit$ plays in accordance with model Γ_i. Then standard Bayesian conversion yields

$$P(\Gamma_i|e) \simeq P(e|\Gamma_i)P(\Gamma_i). \tag{12}$$

This means that if we can calculate $P(e|\Gamma_i)$ for all i, then we can use the collected information about $\spadesuit$'s moves to adapt the mixture to his actual reasoning.

For updating of beliefs you have the evidence $e = \{w_o, w_1, m_1^\heartsuit\}$ and the updating will for the $\spadesuit$-types be exactly as for single models, and in this way you get $P_i(A^\spadesuit|e)$, the updated belief of $\spadesuit$'s assignment given that he plays according to model Γ_i represented in the type T_i. However, we also need to update the probabilities of the 3 models. From the Bayesian inversion above we see that it is sufficient to calculate $P(w_0, w_1, m_1^\heartsuit|T_i)$, the probability of the evidence given the model with type T_i. For notational convinience, let P_i denote probabilities in the model with type T_i. Then

$$\begin{aligned} P_i(w_0, w_1, m_1^\heartsuit) \simeq & \textstyle\sum_{M^\spadesuit} P(w_1|w_0, M^\spadesuit, m^\heartsuit) \\ & \textstyle\sum_{A^\spadesuit} P_i(M^\spadesuit|w_0, A^\spadesuit)P_i(A^\spadesuit), \end{aligned} \tag{13}$$

where $P(w_1|w_0, M^\spadesuit, M^\heartsuit)$ is the transition rule in the game, $P_i(M^\spadesuit|w_0, A^\spadesuit)$ is δ_i^a for the various $\spadesuit$-assignments, and

$P_i(A^\spadesuit)$ is the initial belief of ♠'s assignment given that he plays according to model i. Altogether we get

$$\begin{aligned} P(T_i|w_0,w_1,m^\heartsuit) \;&\simeq P(T_i)\textstyle\sum_{M^\spadesuit} P(w_1|w_0,M^\spadesuit,m^\heartsuit) \\ &\textstyle\sum_{A^\spadesuit} P_i(M^\spadesuit|w_0,A^\spadesuit)P_i(A^\spadesuit). \quad (14)\end{aligned}$$

5.2 Coping with Inconsistency

At time step t, ♡ receives an observation and needs to update her type tree T_t. For that she could use the update function (6), but it will only suffice if observation received at time step t is consistent with her type tree. If the observation is inconsistent with T_t (i.e. the observation has probability 0 according to her model), Bayesian update becomes invalid and ♡ will have no way of assessing the probabilities of T_{t+1}. Consequently, we are unable to update the probabilities over the model from that state and forward.

Actually, we are dealing here with a general problem in opponent modeling techniques. When ♡ has a model of ♠, the model of ♠ will inevitably contain a wrong model of ♡ (otherwise the model would be infinite) and inconsistent observations will eventually emerge. It is an open issue how this important problem must be handled so we propose a more thorough analysis of this problem as future research.

For the purpose of the experiments in this paper we propose to resolve the conflict by adding a baseline model, Γ_0 = NIL, that prescribes random behavior, into the mixture of models. As Γ_0 hypothesizes all possible actions that an opponent may take, updating the probability over the mixture of models can always be executed through the Bayesian function (6).

In spite of its simplicity, the approach using a mixture of models with additional randomization has some merit. It is possible to investigate whether one of the candidate models is the correct opponent model. One could apply an entropy based score to get an idea of whether there is a great degree of confusion in the mixture. If the entropy is high, it means that the player is uncertain which candidate model is the right one. That can be used in a future line of work where such a score is used to draw conclusions on whether the mixture is a good representation of the opponent model or whether other candidate models should be considered.

6. EXPERIMENTAL RESULTS

We implemented RID and the proposed type tree framework using the HUGIN[1] engine. To perform empirical experiments with the methods, we have implemented a game named *Grid* that is an abstraction of popular physical games in northern Europe. We demonstrate that our models are able to capture players' interactions and to improve players' performance when facing various types of opponents in the game. In the experiments we shall refer to the models listed in Table 1.

6.1 Game Example

In *Grid*, 2 players stand on each their side of a table with a ball placed on top of it. The objective is for each player to blow the ball to a favorable position. The players have each their assignment which determines the player's desired destination for the ball and the table is surrounded by walls such that the ball does not fall over the edge. The table is represented as an $m \times n$ grid and an example of a 4×2 *grid* game in Figure 4.

Table 1: Models used in the experiments.

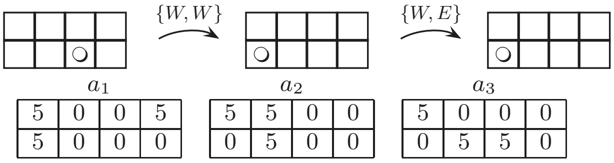
	Model
Γ_0	NIL
Γ_1	$(1, NIL)$
Γ_2	$(2, (1, NIL))$
Γ_3	$(3, (2, (1, NIL)))$
Γ_4	$(3, (3, (2, (1, NIL))))$

	Opponent Models
MIX$_1$	$\Gamma_0, \Gamma_1, \Gamma_2$
MIX$_2$	Γ_1, Γ_2
MIX$_3$	$\Gamma_1, \Gamma_2, \Gamma_3$

Initially, both players draw an assignment (from the set $\mathcal{A} = \{a_1, a_2, a_3\}$) and the assignment each player gets determines the pay-off function over the ball's positions in the grid. Pay-off functions corresponding to assignments a_1, a_2 and a_3, respectively, are shown in Figure 4.

In each turn the players observe the position of the ball and decide to blow it either up, down, right or left, (N, S, E and W). The effect on the ball is a combination of the directions in which the two players blow, and it is determined by the non-deterministic transition function.

When the ball is in its new position, the players are rewarded according to their assignments and the next turn begins. The real game scores are not revealed to the players. The game is a zero-sum game so ♡ is deducted what ♠ wins and vice versa. The game has unbounded time horizon, but it is interrupted by the game master after a number of turns.



5	0	0	5
5	0	0	0

5	5	0	0
0	5	0	0

5	0	0	0
0	5	5	0

Figure 4: An example of the game Grid. In the first move, both players choose to move W. In the second turn, ♡ and ♠ move W and E respectively, canceling each other's effect.

6.2 Comparison of players of different levels

In the first series of experiments we have measured the performance of player ♡ when she has the correct model of ♠ and compared with games where ♡ has made wrong assumptions about ♠'s nesting level. We have used the 4×2 board and the 3 assignments from the above example. In each encounter between the ♡ and ♠ models they play 100 games. To avoid one player having an advantage from one particular assignment we let ♡ and ♠ play with each possible configuration of assignments the same number of times. Furthermore, we have left out the configurations where ♡ and ♠ have the same assignment since these games will always end in a draw (0 – 0 in fact).

The results are summarized in Table 2. The column player ♡ with models Γ_i, ($i = 1,\ldots,4$) competes with the row player ♠ with models Γ_j, ($j = 0,\ldots,3$). Note that ♡ has the correct model of ♠ if $i = j-1$, while ♡ overestimates ♠ if $i > j-1$, and finally, ♡ underestimates ♠ if $i < j-1$.

As expected, ♡ wins when she has the correct assumptions about ♠'s nesting level, whereas her performance is reduced when ♡ overestimates the opponent.

[1] www.hugin.com

Table 2: Average scores and standard deviations (*italics*) obtained on different levels.

♡\♠	Γ_0	Γ_1	Γ_2	Γ_3
Γ_1	23.2(*12.3*)	–	–	–
Γ_2	12.85(*15.6*)	12.1(*19.1*)	–	–
Γ_3	4.77(*17.7*)	-6.59(*17.1*)	13.75(*17.0*)	–
Γ_4	14.9(*14.2*)	-16.5(*19.2*)	3.71(*18.5*)	35.8(*13.9*)

If we look at the results in the eyes of ♠ we can see the consequences of underestimating the opponent. For instance, in the column labeled Γ_1, ♠ is assuming ♡ to play with the Γ_0 model, whereas ♡ is really playing with models Γ_2, Γ_3 and Γ_4 respectively. ♠ is generally loosing when ♡ has the correct model of him but that does not seem to be true when ♡ has overestimated ♠. For instance, ♠ has won on average 6.59 points with the Γ_1 model against ♡ with the Γ_3 model. Arguably, the results seem to indicate that the Γ_1 model is a rather successful choice against an overestimating opponent. However, it is not the best choice. Look for instance at the games against the Γ_3 model. The Γ_1 model wins with 6.59 points on average against the Γ_3 model but the Γ_4 wins with 35.8 points on average against the same model. Hence the best result seems to be obtained when a player knows the correct model of the opponent.

Overall the results indicate that the best results are obtained when the opponent's model is known. However, if the model of the opponent is unknown it may be a good choice to use a Γ_1 model. The advantage of this observation is even better if we do not know the opponent's model but we have information that the opponent does certainly not use the Γ_2 model. If, however, the opponent is using a Γ_2, model the Γ_1 is a bad choice. In that case a Γ_3 model is the optimal choice. Generally it may be a good policy to play a combination of the Γ_1 and the Γ_3 models if the opponent's model is unknown. The scheme for adaptation we have proposed here may be a good scheme for playing with such a combination of models.

6.3 Adaptation

In the second series of experiments we used mixtures of models to allow the player to adapt to the opponent's model. We used the mixture models, MIX$_1$, MIX$_2$ and MIX$_3$ from Table 1. The model Γ_0 plays randomly, so he can always be expected to pick any action. All mixture models will start with uniform priors. MIX$_2$ and MIX$_3$ will add Γ_0 with probability 1 in case they observe an inconsistent observation.

We report the results of the experiments in Table 3. Player ♡ plays with the 3 mixtures from Table 1 against player ♠ playing with Γ_0, Γ_1, Γ_2 and Γ_3 respectively. Note that when ♡ uses MIX$_1$ and MIX$_2$ against Γ_3 she is playing against a model which is not included in her mixture. ♡'s performance does not seem to be very good in these cases. However, playing with MIX$_1$ is better than playing a pure Γ_2 against the Γ_3 (See Table 2).

The results also show the influence from including Γ_0 in the mixture. Against Γ_1 and Γ_2 MIX$_1$ has performed worse than MIX$_2$ and MIX$_3$. Not surprisingly, MIX$_1$ shows the best performance against Γ_0. This is because MIX$_1$ will adapt faster to the Γ_1 policy than to the other models. Table 4 shows the average probability assigned to Γ_1 after 10 moves for each of the experiments. It shows that MIX$_1$ has a strong tendency to conclude that the opponent is using the Γ_0 model (and hence it will use the Γ_1 model). Furthermore, MIX$_1$ has a tendency to loose when ♠ uses Γ_2 but it has a tendency to win against Γ_3. The explanation may be that Γ_1 looses to Γ_2 but it wins against Γ_3 (see Table 2).

Table 3: Average scores and standard deviations (*italics*) obtained by the mixture models in Table 1.

♡\♠	Γ_0	Γ_1	Γ_2	Γ_3
MIX$_1$	20.5(*12.4*)	2.05(*14.7*)	-0.35(*15.4*)	2.9(*19.7*)
MIX$_2$	11.2(*15.3*)	7.35(*18.6*)	10.4(*16.0*)	-1.7(*22.2*)
MIX$_3$	12.6(*13.7*)	2.85(*15.8*)	10.3(*17.1*)	9.25(*24.0*)

Table 4: Average beliefs assigned to Γ_0 in models MIX$_1$, MIX$_2$ and MIX$_3$ after 10 moves against Γ_0, Γ_1, Γ_2 and Γ_3 respectively.

♡\♠	Γ_0	Γ_1	Γ_2	Γ_3
MIX$_1$	1.00(*0.00*)	0.85(*0.02*)	0.77(*0.02*)	0.99(*0.00*)
MIX$_2$	0.62(*0.05*)	0.02(*0.00*)	0.04(*0.01*)	0.41(*0.06*)
MIX$_3$	0.53(*13.7*)	0.11(*0.01*)	0.01(*0.0*)	0.01(*0.00*)

Table 4 shows that the MIX$_2$ model and the MIX$_3$ model do not have the same tendency to conclude that the opponent plays randomly. This is actually a small disadvantage if the opponent actually plays randomly. Against the Γ_0 model MIX$_2$ and MIX$_3$ have only assigned probabilities 0.62 and 0.53 respectively to the Γ_0 model. This in turn explains the lower performance of the MIX$_2$ and MIX$_3$ models against the Γ_0 model compared to the MIX$_1$ model.

The results are comparable to the results of Table 2. Take for instance the results of MIX$_3$. The performance against the Γ_0 model is in the best case 23.2 when ♡ plays a pure Γ_1 model and it is 12.6 if ♡ uses MIX$_3$. Against the Γ_3 model ♡ wins on average 35.8 points when she plays with the pure Γ_4 model but it has dropped to 9.25 with MIX$_3$. These are however comparisons with games where the pure models have not overestimated nor underestimated the opponent's model. A more fair comparison would be to compare the performance of e.g. MIX$_3$ with the performance of the pure Γ_4 model. In this case the performance against the Γ_1 model has increased from -16.5 to 2.85 and against the Γ_2 the performance has increased from 3.71 to 10.3.

Overall the proposed adaptation scheme seems to provide good performance and may be a fruitful approach when the exact opponent models are unknown.

6.4 Model Complexity

The most demanding computation is to solve the influence diagrams in the models. They are converted into a computational structure called a strong junction tree [8]. It is the number of cells in the strong junction tree that constrains how complex models we can solve.

Table 5 shows the average time used for each move and the total number of cells used by the influence diagrams in each model's type tree. Time is the average time used per move while space consumption is measured in the number of cells used in the underlying strong junction trees. The average time has been measured on a computer with a 2.5 GHz Intel Xeon CPU, with 32 GB RAM. The actual memory consumption of the Γ_2 model was 25 MB. The Γ_3 however,

Table 5: Time (Avg t) and space consumption (JT Size) of the fixed sized models.

	Avg t	JT Size
Γ_1	0.001 sec.	2,361
Γ_2	0.003 sec.	22,861
Γ_3	24 sec.	7,445,513
Γ_4	283 sec.	25,679,069

consumed 1.3 GB while the Γ_4 could not run since it requires more RAM than available on the system. Approximation methods proposed in [16] were applied in order to implement the Γ_4 model.

7. RELATED WORK

RIDs contribute to the growing line of research on probabilistic graphical models for multiagent decision making including nested influence diagrams (NIDs) [4], multiagent influence diagrams (MAIDs) [10] and interactive dynamic influence diagrams (I-DIDs) [2]. Both NIDs and MAIDs provide an analysis of the game from an external viewpoint, and adopt Nash equilibrium as the solution concept. However, the deficiencies of equilibrium (non-unique and incomplete) prevent the applications of NIDs and MAIDs in a general setting. Recently, I-DIDs emerge as an alternative framework for solving the sequential multiagent decision making problem. The technique represents the problem from the perspective of an individual decision maker and allows a general setting of both cooperative and competitive agents. Due to the computational complexity, current research on I-DIDs still focuses on the planing problem of limited time horizons [18]. Analogous to I-DIDs, RIDs appear as one PGM framework for solving sequential Bayesian games, and mainly target at a more natural planning problem in which the time horizons are unknown or unlimited. More importantly, RIDs can resort to many effective PGM techniques, like the information enhancement method [16] for overcoming the curse of time horizon, therefore resulting in more efficient solutions.

In parallel, other frameworks for modeling the multiagent decision-making problem exist. Most notable among them is the decentralized POMDP (Dec-POMDP) [13]. This framework is suitable for cooperative settings only and focuses on computing the joint solution for all agents in the team. Many Dec-POMDP techniques [12] exploit the conditional independence concept to factorize the state space with the purpose of providing scalable solutions.

8. DISCUSSION AND CONCLUSION

The curse of current history emerges in games where the moves of the opponents are not disclosed. The player will have to represent all moves taken along with each possible belief state of the opponents for each possible opponent move resulting in a combinatorial explosion. In this paper we first introduce the notion of type tree and show how they can be used to address the curse of current history. Secondly, we observe that the true model of the opponent is often unknown in opponent modeling scenarios. To address this problem, we apply a mixture of models. As experiments also have shown, the success of the player depends on whether the player makes correct assumptions about the opponent's model. With a mixture of models this requirement can be relaxed.

The possible lines of future research are many. However, we have identified the following as the most interesting topics to pursue: (i) The area of modeling inconsistency in opponent modeling is not yet fully understood and more work is needed in this area. (ii) In this paper we have not investigated in depth how a good set of candidate opponent models can be selected. It is possible to come up with some good advice based on the present experiments but a general investigation has yet to be performed. (iii) We have assumed that the opponent is playing with a singleton model. It must be investigated how to handle situations when the opponent is allowed also to use a mixture of models.

9. REFERENCES

[1] D. Carmel and S. Markovitch. Incorporating opponent models into adversary search. In *AAAI*, pages 120–125, 1996.

[2] P. Doshi, Y. Zeng, and Q. Chen. Graphical models for interactive pomdps: Representationsand solutions. *JAAMAS*, 18(3):376–416, 2009.

[3] D. Fudenberg and J. Tirole. *Game Theory.* The MIT Press, Cambridge, MA, 1991.

[4] Y. Gal and A. Pfeffer. A language for modeling agents' decision making processes in games. In *AAMAS*, pages 265–272, 2003.

[5] P. J. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multi-agent settings. *JAIR*, 24:49–79, 2005.

[6] P. J. Gmytrasiewicz, E. H. Durfee, and D. K. Wehe. A decision theoritic approach to coordinating multiagent interactions. In *IJCAI*, pages 62–68, 1991.

[7] R. A. Howard and J. E. Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, pages 721–762, 1984.

[8] F. Jensen, F. V. Jensen, and S. L. Dittmer. From influence diagrams to junction trees. In *UAI*, pages 367–374, 1994.

[9] F. V. Jensen and E. Gatti. Information enhancement for approximate representation of optimal strategies for influence diagrams. In *PGM*, pages 102–109, 2010.

[10] D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. In *IJCAI*, pages 1027–1034, 2001.

[11] S. L. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47:1235–1251, 2001.

[12] J. Pajarinen and J. Peltonen. Efficient planning for factored infinite-horizon dec-pomdps. In *IJCAI*, pages 325–331, 2011.

[13] S. Seuken and S. Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *JAAMAS*, pages 190–250, 2008.

[14] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):597–609, 1986.

[15] P. P. Shenoy. Valuation-based systems for Bayesian decision analysis. *Operations Research*, 40(3):463–484, 1992.

[16] N. Søndberg-Jeppesen and F. V. Jensen. A pgm framework for recursive modeling of players in simple sequential bayesian games. *IJAR*, 5(51):587–599, 2010.

[17] J. A. Tatman and R. D. Shachter. Dynamic programming and influence diagrams. *IEEE Trans. on Systems, Man, and Cybernetics*, 20(2):365–379, 1990.

[18] Y. Zeng and P. Doshi. Exploiting model equivalences for solving interactive dynamic influence diagrams. *JAIR*, (43):211–255, 2012.