International Journal of Approximate Reasoning 51 (2010) 587–599

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

A PGM framework for recursive modeling of players in simple sequential Bayesian games

Nicolaj Søndberg-Jeppesen, Finn V. Jensen *

Department of Computer Science, Aalborg University, Selma Lagerlöfs Vej 300, DK-9220 Aalborg, Denmark

ARTICLE INFO

Article history:
Received 10 December 2008
Accepted 11 May 2009
Available online 28 January 2010

Keywords:
Sequential Bayesian games
Influence diagrams
LIMIDs
Multiple agents
Recursive modeling method

ABSTRACT

We consider the situation where two agents try to solve each their own task in a common environment. In particular, we study simple sequential Bayesian games with unlimited time horizon where two players share a visible scene, but where the tasks (termed *assignments*) of the players are private information. We present an influence diagram framework for representing simple type of games, where each player holds private information. The framework is used to model the analysis depth and time horizon of the opponent and to determine an optimal policy under various assumptions on analysis depth of the opponent. Not surprisingly, the framework turns out to have severe complexity problems even in simple scenarios due to the size of the relevant past. We propose two approaches for approximation. One approach is to use Limited Memory Influence Diagrams (LIMIDs) in which we convert the influence diagram into a set of Bayesian networks and perform single policy update. The other approach is *information enhancement*, where it is assumed that the opponent in a few moves will know your assignment. Empirical results are presented using a simple board game.

1. Introduction

It is a central problem in Multi-agent 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. The Recursive Modeling Method (RMM) which was proposed by Gmytrasiewicz et al. [1] does that. They propose to equip each intelligent agent with a model in which each agent is equipped with a model which models the other agents. These nested models may again have models of all the rest of the agents in the environment which again contain nested models. The nesting of models continues until a predefined nesting level is met. At the deepest level the nesting is ended by a simpler kind of model which equip each agent in the environment with a "flat" model, without models of other agents.

We shall use RMM together with Influence Diagrams (IDs) for modeling a game scenario. In this scenario each agent intends to solve a task or assignment by taking certain actions, which change the state of a common environment. The characteristics of the scenario is that since the agents co-exist in the same environment, the actions performed by one agent affects the state of the scenario for all agents. The state of the environment will be known by all agents, but the assignments are private knowledge. The scenario may be a competition between the agents, they may cooperate in solving the same task or they may be working on solving each their own task without caring about the other agent's performance. No matter what, the success for each agent is highly dependent on its ability to model the other agents in the scenario.

* Corresponding author. Tel.: +45 99408903; fax: +45 99409798.
E-mail addresses: nicolaj@cs.aau.dk (N. Søndberg-Jeppesen), fvj@cs.aau.dk (F.V. Jensen).

0888-613X/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2010.01.015

Many of the RMM based approaches turn out to be PSPACE-hard [2]. IDs are also facing notorious complexity problems due to the no-forgetting assumption (the assumption that everything that has been known is remembered). When looking into the future, the decision maker has to consider way too many possible future information scenarios, all relevant for the current decision. The information to consider is the variables that will be instantiated in the future. Lauritzen and Nilsson [3] propose *Limited Memory Influence Diagrams* (LIMIDs) in which the decision maker is assumed to remember only a subset of the instantiated variables. This means that the decision maker assumes in the future to know less than actually will be the case. We call this approach *information reduction*.

Another approach is to assume to know more. For the kind of games we study, the player may assume that at some point in the future he will know his opponent's assignment. If so, all other past information is irrelevant. That is, he assumes that he will know more than actually will be the case. We call this approach *information enhancement*.

In this paper, we will exploit both approaches for calculating approximations to optimal strategies, and we report on a series of experiments with game scenarios where the solution of the resulting influence diagrams is intractable.

2. Background

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. Each player gets a *pay-off*, which is a function of the state of the scene. In case of zero-sum games, the actual gain/loss is the difference between the two pay-off functions. The pay-off function is known only to the player. We call the pay-off function the *assignment* of the player, and the assignment of the players do 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 *Simple Sequential Bayesian Games (SSBG)*.

Formally, an SSBG consists of two players P^1, P^2 and a *world* W with a finite set of states $w_1, w_2, \ldots, w_m$. We assume P^1 to be female and P^2 to be male. The players have finite sets of *moves*, M^1 and M^2, which affect W. The transition between world states at time t to time $t+1$ is determined by a probabilistic function τ, where $\tau : W \times M^1 \times M^2 \times W \to \boldsymbol{R}$, where $\tau(w_1, m_1, m_2, w_2)$ 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 making the moves m_1 and m_2, respectively.

Furthermore, each player has an *assignment*, which is a particular pay-off function reflected in utility numbers over states of the world. Thus, players P^1 and P^2 have a finite set of possible assignments A^1 and A^2, respectively. 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. However, a probability distribution of the opponent's assignment may be inferred by observing the changes of the world and through Bayesian inversion inferring a distribution of his/her moves. The pay-off function, which assigns pay-offs to each player at each time step, is a function of the current world state and the actual assignments.

2.1. Influence diagrams

We shall use the classical paradigms from Probabilistic Graphical Models (PGMs). A PGM is a directed acyclic graph with three types of nodes, *chance nodes* (circular nodes), *decision nodes* (rectangular nodes), and *utility nodes* (diamond shaped nodes). A directed link into a chance node reflects (causal) impact, which may be of non-deterministic character, a link into a decision node represents information. That is, if C is a parent of the decision node D then the state of C is known by the decision maker when D is to be decided.

The quantitative part of a PGM consists of utility functions and conditional probabilities. For a utility node U with parents $pa(U)$ we specify the utility as a function of $pa(U)$. For a chance node C with parents $pa(C)$ we specify $P(C|pa(C))$, the conditional probability distribution of C given $pa(C)$.

A PGM is an *influence diagram (ID)*, if the utility nodes have no children and there is a directed path comprising all decision nodes. For an influence diagram we assume *no-forgetting*: the agent remembers everything from the past.

A *solution* to an ID is an *optimal strategy*: a strategy consists of a set of *policies*, one for each decision node. A policy, δ_D, for a decision node D is a function, which given the known past provides a decision. A strategy is *optimal* if it maximizes the decision maker's expected utility.

There are standard algorithms for solving IDs, and systems for specifying and solving IDs are commercially available [4–7]. We shall in this paper take these algorithms for granted.

The framework of IDs has been extended in various ways. In particular, Koller and Milch [8] introduced *Multi-Agent Influence Diagrams* (MAIDs), where the various acting agents are given decision and utility nodes of particular colors (or shadings). We shall use the MAID framework to represent the game scenario, and we address solution by converting the representation to a series of IDs.

2.2. The curse of time horizon

You may consider the solution of an influence diagram in the following way. Assume that you are confronted with an ID with three decisions D_1, D_2, and D_3. Before deciding D_i you observe O_i. When you are about to decide D_3, you know the states of O_1, D_1, O_2, D_2, and O_3, and you can rather easily compute the optimal decision. On the other hand, when you are about to

decide D_1 you need to know how you will act in the future, and right now you do not know the states of O_2, D_2, and O_3. That is, you need a tractable representation of the policy δ_{D_3} with domain $(O_1, D_1, O_2, D_2, O_3)$. *The curse of time horizon* is the problem that the domain of policies for future decisions grows exponentially with time and very fast become intractable. The problem is not that you cannot cope with the last decision; the problem is that you cannot compute the first decision because you do not have tractable representations of policies for future decisions. If you are so lucky that sufficiently many variables are irrelevant for the future decisions [9,?], the domain of the policy functions may have tractable size. If not, you may doom some variables irrelevant by assuming that you will not remember their states. This will give a so called LIMID [3]. A LIMID provides an approximate representation of future policies.

There are other ways of approximating policies of future decisions. The main approach is to extract the information from the past into a smaller set of variables.

When dealing with PGM representations of games, you will be hit by the curse of time horizon, and much of the present paper is a report on various ways of coping with it.

3. Graphical representation of SSBGs

We adapt the framework of MAIDs to SSBGs. This is illustrated in Fig. 1. In Fig. 1 player P^1's nodes are lightly shaded, and P^2's nodes are darkly shaded.

The nodes $W_0, W_1, \ldots$ represent the world states at $t = 0,\ t = 1, \ldots$ the chance nodes A^1 and A^2 represent the players' assignments, the nodes with M^i-labels represent the moves by player P^i. The diamond shaped nodes, which are half lightly shaded and half darkly shaded represent the pay-off functions, which assign a utility to both players in each game step. The links from a W-node and the A-nodes to a U-node indicate that the utility is a function of the world state and the assignments. The links from the A^1-node to decision nodes represent that player P^1 knows her assignment. The links from W-nodes to decision nodes represent that the state of the world is always known. The dots at the right of the graph indicate that there is no time limit specified.

There might be constraints on which assignments the players can have simultaneously. For example, they might not have the same assignment. Therefore, A_1 and A_2 are connected to a constraint node, which is instantiated.

At $t = 0$ the game starts in W_0, where P^1 and P^2 each decide their moves concurrently knowing only their own assignment and the initial world state. P^1's and P^2's joint moves lead to a new state W_1, where player P^1 and P^2 again decide each their moves knowing W_0, W_1 and their own assignment and first move. When both players have decided their moves, the game continues with the next time step.

Even if there is a pre-specified time horizon, the standard methods for solving IDs [4–6] cannot be used. Consider the last time step. Both players have to come up with an optimal decision given the past. Part of the considerations for player P^1 will be an estimate of player P^2's move. However, P^2's move is dependent on an estimate of P^1's move. You end up with an infinite regression, which in game theory is solved by determining Nash equilibria [11]: a pair of policies for which each policy is optimal given that the other player uses the other policy.

3.1. The game seen in the eyes of P^1

In real world situations, players do not perform an infinite regression and determine Nash equilibria. The players will analyze the situation to a certain depth and with a certain look-ahead of moves, and in the depth analysis they will make some

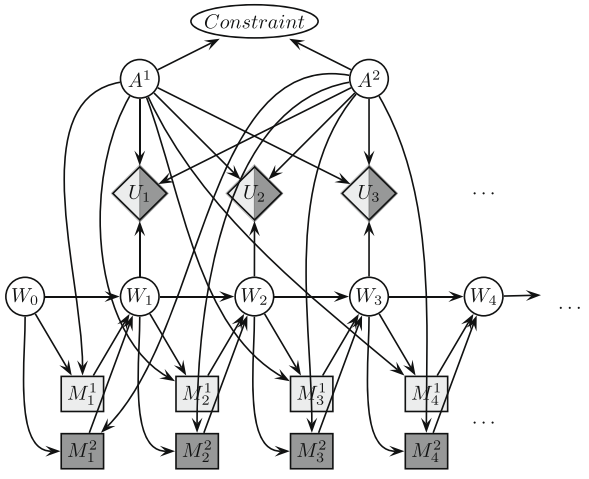

Fig. 1. A MAID representation of SSBG.

assumptions about the other players' analysis depth and look-ahead. This is called the *recursive modeling method* (RMM) [1], and we will incorporate RMM into the models by letting P^1 bound the recursive modeling to a certain level. More specifically, P^1 has a model incorporating the moves of player P^2, where she makes some assumptions on how many moves ahead he analyzes the situation, and in turn, how deep P^2 is assuming P^1's model to be. In other words, if P^1 makes these assumptions about the policies of P^2, the model in Fig. 1 can be transformed to an ID, where P^2's decision nodes are replaced by chance nodes, and $P(M^2_{i+1}|A^2, W_0, \ldots, W_i, M^2_0, \ldots, M^2_i)$ is the policy (see Fig. 2). Note that the node A^2 reflects that P^2's assignment is unknown to P^1. The model shall include prior probabilities for A^2.

In the simplest case P^1 assumes that P^2 just picks a move randomly. We will say that this player has a level 0 model since she in this case does the least effort to model P^2. In case P^1 assumes that P^2 has a level 0 model, we say that P^1 has a level 1 model. In general, when P^1 has a level i model, she assumes that P^2 has a level $i-1$ model.

At each level, P^1 may take different numbers of future time steps into account. If P^1 is only taking one future time step into account she will greedily pick a move that maximizes her expected utility in the next time step. If P^1 is taking two future time steps into account she will maximize the sum of her expected utility in the next and the following time step. In general, if P^1 is taking h future time steps into account she will maximize her expected sum of utility in the next h time steps. We shall call the number of future time steps taken into account the *time horizon* of P^1. Consequently, P^1 also must have an assumption about P^2's time horizon and she must also have an assumption about which time horizon P^2 assumes that P^1 has etc.

In order to capture P^1's modeling level and time horizon together with her assumptions about P^2's nesting depth and P^2's assumptions about P^1's nesting depth we give the following definition (inspired by [12]).

Definition 1 (*Player Representation*). A player representation P is a pair defined as follows:

1. $P = (h, NIL)$ represents a player with time horizon h and modeling level 0.
2. Given a player representation O, with modeling level $i-1$ and time horizon q. Let $h \geqslant q$. Then $P = (h, O)$ represents a player with time horizon h and modeling level i.

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

Thus, the simplest model, which is a level 0 model with time horizon 1 is denoted $(1, NIL)$; a $(2, (1, NIL))$ model is a level 1 model in which P^1 has time horizon 2 assuming that P^2 is a level 0 model with time horizon 1; a $(3, (2, (1, NIL)))$ model is a level 2 model in which P^1 has time horizon 3 assuming that P^2 is a level 1 model with time horizon 2 assuming that P^1 is a level 0 model with time horizon 1.

Fig. 2 shows how a $(2, (2, (1, NIL)))$ model for P^1 is represented as an ID. The leftmost ID represents the world as seen by P^1. In this ID, the nested model, namely $(2, (1, NIL))$ is used to fill in the conditional probability distributions for the nodes M^2_1 and M^2_2 representing P^2's decisions. The ID in the center represents this model, which represents the strategy assumed to be played by P^2. In this model, the conditional probability distributions for M^1_1 and M^1_2 are found by analyzing the rightmost ID, which represents the deepest model $(1, NIL)$. In the $(1, NIL)$ model, the chance node M^2_1 represents the completely random strategy, which P^2 is assumed to play on level 0.

Note that the leftmost ID in Fig. 2, contains extra arcs, namely the arc connecting the nodes M^2_1 and M^2_2 and the arc connecting W_0 and M^2_2. These arcs represent the extra information that (P^1 assumes that) P^2 will have in time step 1 when P^2 is taking decision M^2_2.

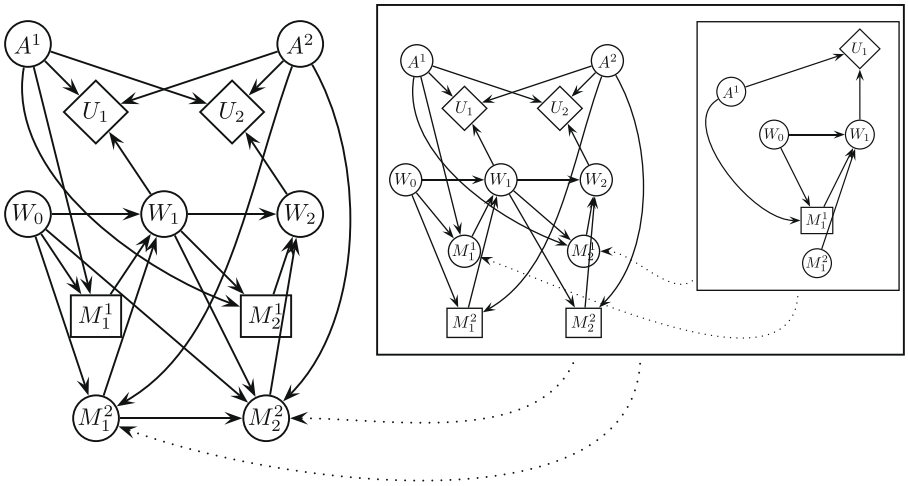

Fig. 2. An ID representing the $(2, (2, (1, NIL)))$ model.

Information from the past can be used to *learn* about the opponent's assignment. Look at the ID in the center in Fig. 2 with $t = 1$. If the information is that the variables are in states w_0, w_1, m_1^2, then P^2 can use the model to calculate $P(A^1|w_0, w_1, m_1^2)$. Thus, when deciding for the current move, the player has to take into consideration that she – as well as the opponent – in the future will learn and act accordingly.

The modeling of the learning opponent causes the dimensionality of the conditional probability table representing M_i^2 to grow heavily with the number of future world states. In this work we are addressing this problem.

4. Finding optimal policies

As can be seen from Fig. 2 you start off with a simple ID (level 0) assuming P^2 to play a random strategy regardless of her assignment. This yields a policy for P^1, which is a function of W_0 and A^1. This policy, $\delta_{1_1}(W_0, A^1)$, is used as CPT for the next ID representing P^2's decision problem (level 1). The solution of this ID is a pair of policies $\delta_1^2(A^2, W_0), \delta_2^2(A^2, W_0, M_1^2, W_1))$. Again, these policies are entered as CPTs to the top ID (level 2). Actually, the process is a bit more complex. The ID is used for calculating δ_1^2. However, when calculating the policy δ_2^2 we have to take one more time slice into account. We shall return to this point in the next section.

The resulting ID (level i) is a model which provides an optimal policy for the *first decision*. When the players have made their first move, the model is also used to establish an updated distribution of the opponent's assignment. P^2 has also learned from the first move. For the example in Fig. 2, the level 0 player does not learn during moves, and the recursion stops here.

Next, the model is prepared to calculate an optimal policy for the second move. This is done by giving the nodes A^2 and A^1 new prior distributions corresponding to the updated beliefs from the first moves. The level 1 ID now provides a new set of policies, which are entered to the level 2 ID as CPTs, and the model is used again to suggest a first decision (which in fact is the player's second move). The game continues so until some termination condition is met. Note that the model is always used only for the first decision in the model; the policies for the subsequent decisions are part of the player's considerations of future scenarios.

4.1. Addressing the curse of time horizon

Assume that we are dealing with a $(3, (2, (1, NIL))$ model. Then P^1's model contains four W-nodes and three nodes representing P^2's policies. Although P^2 has a short time horizon, he has a long lasting memory, which he uses to provide a probability distribution over A^1. This means that the policy for M_3^2 has the domain $(W_0, M_1^2, W_1, M_2^2, W_2, A^2)$, and we have a complexity problem. Alas, the complexity problem is even worse. When calculating δ_3^2, P^1 not only has to take care of the large space of possible past, she also has to include that P^2 will plan one time step ahead.

Fig. 3 represents the ID, which P^1 has to solve in order to estimate δ_3^2. Note that it contains a representation of P^1's policies. However, as P^1 expects P^2 to expect P^1 to expect P^2 to move completely randomly regardless of his assignment, P^2 will act as if P^1 does not learn from the past.

The IDs for estimating δ_2^2 and δ_1^2 are shown in Figs. 4 and 5.

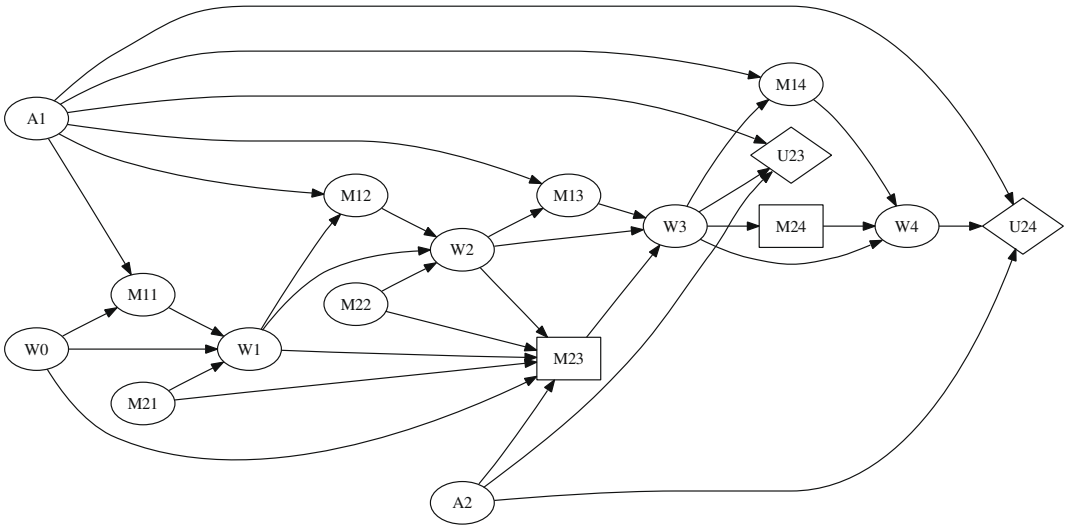

Fig. 3. P^1's model of P^2 for calculating δ_3^2 in a $(3, (2, (1, NIL)))$ model. The last two utility nodes for P^2 are included. The policy for M_3^2 has the domain $(W_0, M_1^2, W_1, M_2^2, W_2, A^2)$, and the policy for M_4^2 has the domain $(W_0, M_1^2, W_1, M_2^2, W_2, M2_3, W_3, A^2)$.

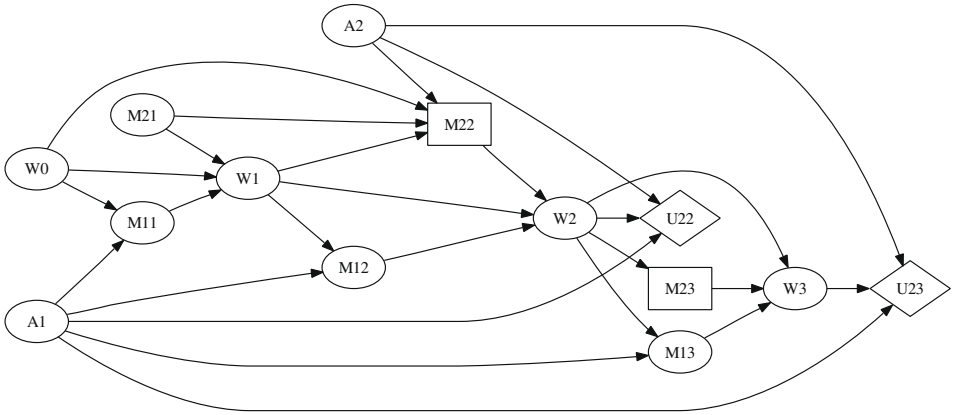

Fig. 4. P^1's model of P^2 for calculating δ_2^2 in a $(3,(2,(1,NIL)))$ model.

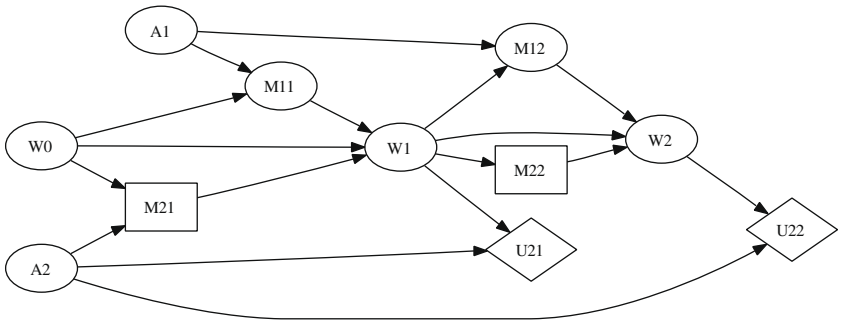

Fig. 5. P^1's model of P^2 for calculating δ_1^2 in a $(3,(2,(1,NIL)))$ model.

As the domain of δ_3^2 as well as the anticipated domain for δ_4^2 are very large, we look for shorter representations approximating them. There are in principle two ways; *information enhancement*: P^1 assumes that P^2 is more informed than he actually is at that state, and *information reduction*: she assumes that he is less informed.

4.1.1. Information enhancement

As the past information is used to estimate the opponent's assignment, P^2 will be most informed, if he knows P^1's assignment. If P^1 assumes that P^2 will know A^1 after two moves, then the ID in Fig. 3 is transformed to the ID in Fig. 6.

Although it seems to increase complexity it does not. A standard structural analysis ([9,10]) shows that W_0, M_1^2, W_1, M_2^2 are irrelevant for the decision M_3^2, and $W_0, M_1^2, W_1, M_2^2, W_2, M_3^2$ are irrelevant for M_4^2. This yields the approximated policies $\hat{\delta}_3^2(W_2, A^1, A^2)$ and $\hat{\delta}_4^2(W_3, A^1, A^2)$, and it corresponds to determining the policy for the first decision in the ID in Fig. 7.

The same approximation method can be used for M_3^2 in Fig. 4.

Now, the optimal policies determined from the IDs in Figs. 6, 4 and 5 are used as conditional probability tables in the ID for P^1. This is shown in Fig. 8.

Again, we are faced with the curse of time horizon: the policy for M_3^1 has a too large domain. Using information enhancement, P^1 will assume that, when she makes her third move, she will know P^2's assignment. This yields the ID in Fig. 9.

Again, structural analysis yields that W_0, M_1^1, W_1 and M_2^1 are irrelevant.

4.1.2. Information reduction

An example of information reduction is LIMIDs [3], where you simply remove some variables from the domain.

Syntactically, LIMIDs are like IDs with the only difference that no-forgetting is not assumed and information is represented directly as information links.

For SSBGs it holds that the variables (W_{i-1}, M_i, W_i) in combination provides information about the opponent's assignment. Therefore, in our framework, the memory is characterized by how many sets of these combinations are remembered. Hence we define a LIMID player representation as a player, which remembers a certain amount of time slices back. Like a regular player from Definition 1 a LIMID player representation has a time horizon and a model of the opponent. However, the LIMID player representation also has a constrained memory. If P^1 is a LIMID player representation with memory m, she will at decision P_i^1 remember her previous m moves $M_{i-m}^1, \ldots, M_{i-1}^1$, together with the previous m world states $W_{i-m}, \ldots, W_{i-1}$.

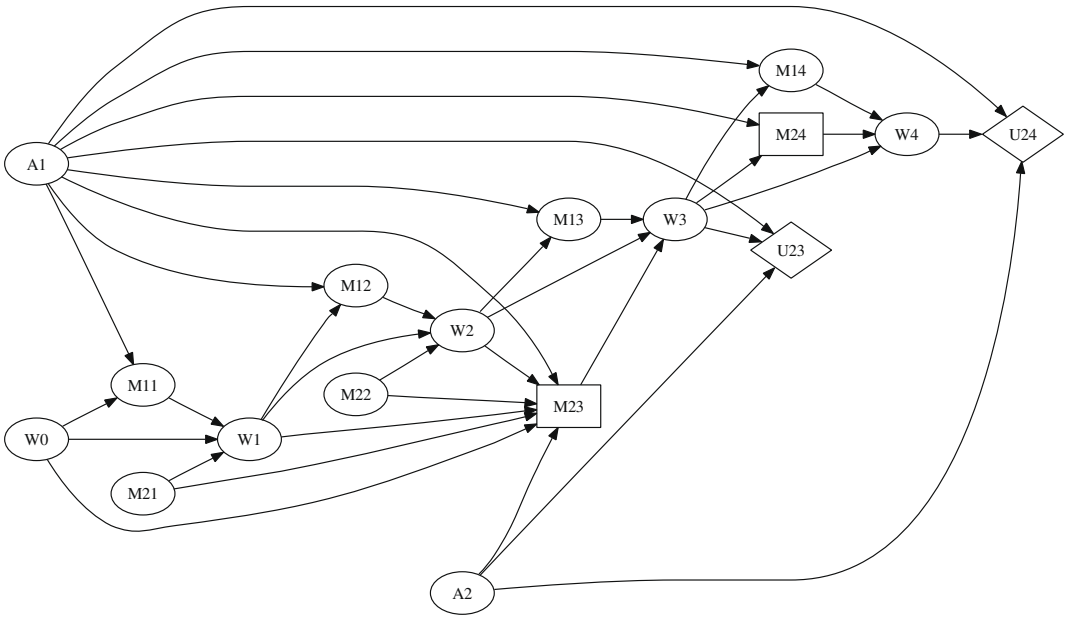

Fig. 6. Due to the link from A^1 to M_3^2, the nodes W_0, M_1^2, W_1, M_2^2 become irrelevant, and the domain of $\hat{\delta}_3^2$ is (W_2, A^1, A^2). The link from A^1 to M_4^2 reduces the domain of $\hat{\delta}_4^2$ to (W_3, A^1, A^2).

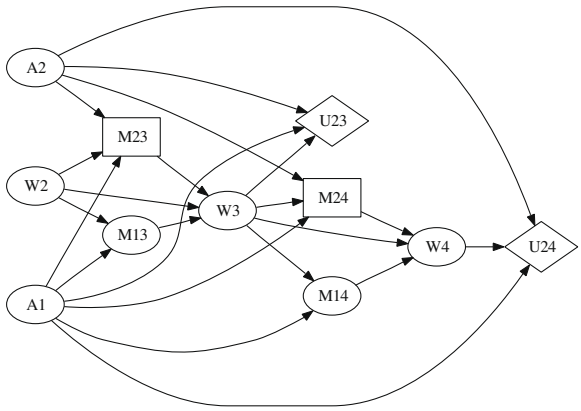

Fig. 7. The ID for determining the policies $\hat{\delta}_3^2(W_2, A^1, A^2)$ and $\hat{\delta}_4^2(W_3, A^1, A^2)$ in Fig. 6.

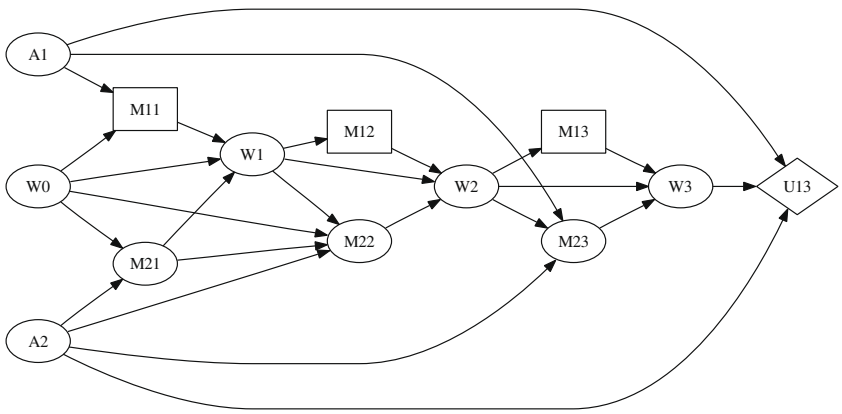

Fig. 8. The ID for P^1, where the last policy for P^2 is simplified through information enhancement.

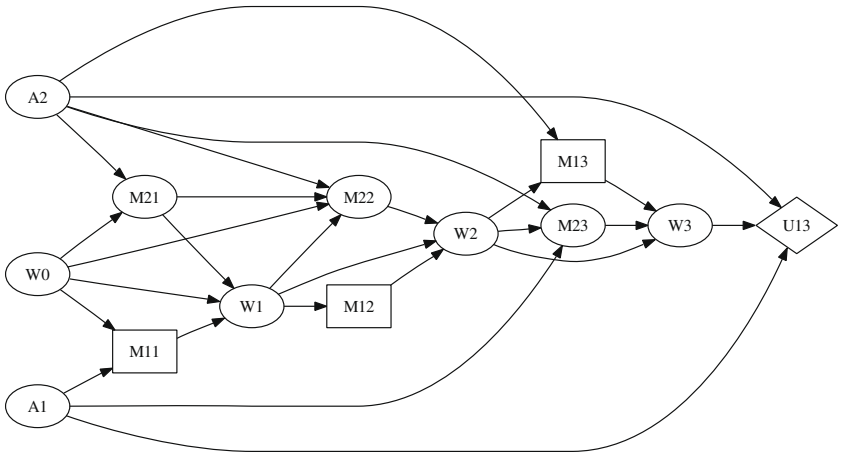

Fig. 9. The link from A_2 to M_3^1 reduces the domain of $\hat{\delta}_3^1$ to (W_2, A^1, A^2).

Definition 2 (*LIMID Player Representation*). A LIMID player representation L is a triple defined as follows:

1. $L = (h, m, NIL)$ represents a LIMID player with time horizon h, memory m and modeling level 0.
2. Given a player representation or a LIMID player representation O, with modeling level $i-1$ and time horizon q. Let $h \geqslant q$. $P = (h, m, O)$ represents a LIMID player with time horizon h, memory m, and modeling level i.

The second entry allows LIMID players to assume that their opponents are ordinary players.

[3] describe a solution algorithm for solving LIMIDS called *single policy updating*. Single policy updating is an iterative algorithm, which is guaranteed to converge towards a local optimum policy.

Another approach is to introduce dummy variables holding extracts of the information variables. As mentioned earlier, for SSBGs the past is used to estimate the probability distribution over the opponent's assignments. So we look for an efficient way of extracting important information about this distribution. This is best illustrated by using the LIMID specification (all information links must be explicit). Fig. 10 shows the ID from Fig. 4 in LIMID specification.

For each set of nodes (W_{i-1}, M_i^2, W_i) we may introduce a node representing what may be learned about P^1's assignment. What has been learned from the various time slices is collected in a conclusion. As an information reducing approximation, we assume that only the conclusion is observed (see Fig. 11).

The L-nodes may hold information on possible assignments. Then L has a state for each set of assignments, and the conclusion node represents set intersection. There is a risk of combinatorial explosion, though, as the number of states in the L-

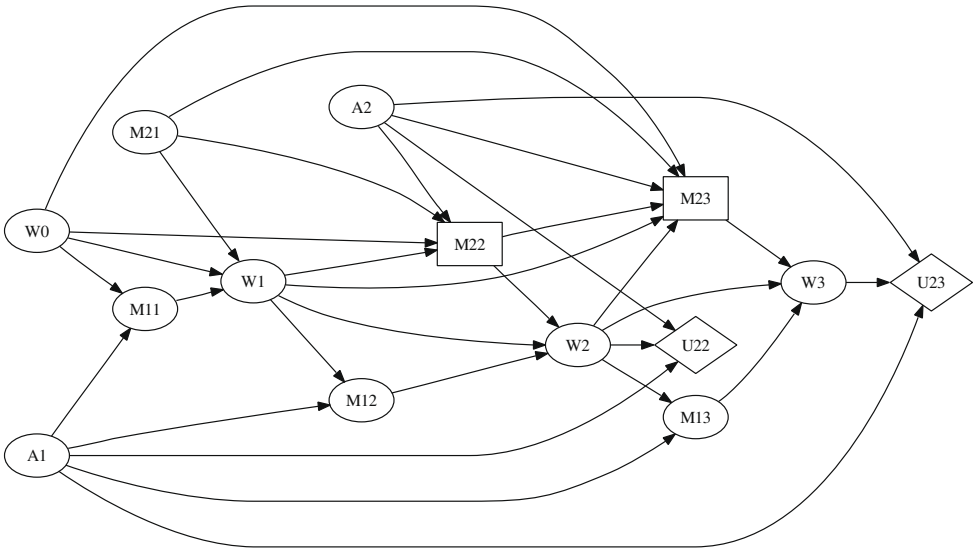

Fig. 10. P^1's model of P^2 for calculating δ_2^2. All information is explicit through links.

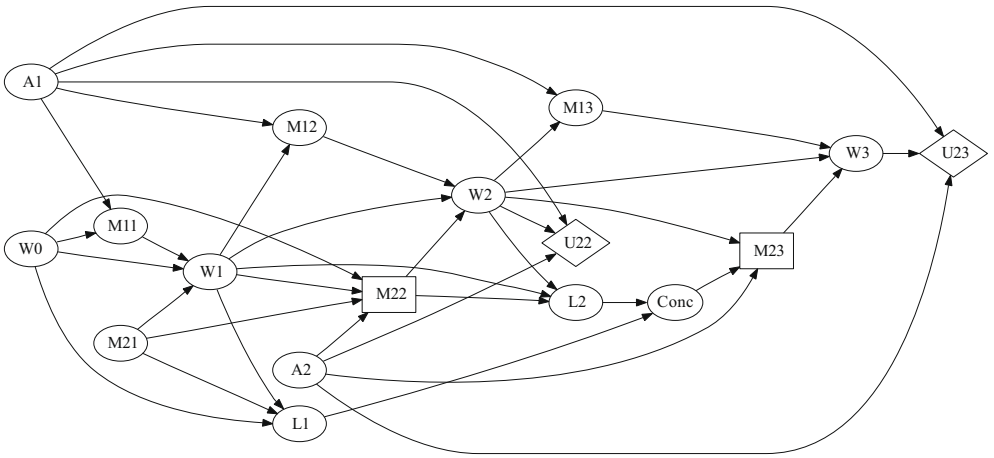

Fig. 11. Fig. 10 extended with nodes representing what has been learned. No-forgetting is not assumed.

nodes increases exponentially with the number of assignments. If this is a problem, you may have the *L*-nodes hold whether there are at most two possible assignments left, or whether up to two assignments are deemed impossible.

It should be restated at this point that the approximating policies are not used directly by the players. They are only used when considering what will happen in the future.

5. Experimental results

To perform empirical experiments with the methods proposed here we have implemented a simple board game which we call Grid. Grid is an SSBG. The board is an $m \times n$ grid. There is exactly one piece on the board. In each turn, the players move simultaneously the piece from one cell to another.

Initially, both players are randomly given an assignment, which is a pay-off function for the positions in the grid. In each turn the players observe the position of the piece and decide to move it either up, down, right or left, (N, S, E and W). The effect on the piece is a combination of the two players' moves. If both actions can be carried out (i.e. the resulting position of the piece is still inside the $m \times n$ grid), the piece is first moved to the neighboring cell in the direction of P^1's decision and then to the neighboring cell in the direction of P^2's decision. If one of the actions cannot be carried out, the other action is carried out (if possible). The τ function adds noise to each player's decision; when the decision is taken, there is a probability that the game will act as if the player moved the piece left relative to the intended direction. For instance, if the player moves N it will with probability 0.2 be read as a W.

When the piece is in its new position, the players are rewarded according to their assignment and the next turn begins. The game is a zero-sum game so P^1 is deducted what P^2 wins and vice versa. This is obtained by subtracting P^2's pay-off from P^1's pay-off and vice versa. The game has an unbounded time horizon, but it is interrupted after a predetermined number of turns.

In our experiments we have chosen a 4×2 board with five possible assignments. The game together with the assignments are shown in Fig. 12.

As an example, let player P^1 and P^2 get Assignment 3 and Assignment 1, respectively. In the first move, (after taking the noise added by τ into account) P^1's move is read as N while P^2's move is read as W. From the resulting new state, P^1 gets the pay-off 5 while P^2 gets the pay-off -1. Now the scores are $5 - -1 = 6$ to P^1 and $-1 - 5 = -6$ to P^2. In the second move P^1's move is read as N while P^2's move is read as S in which case the board state is not changed. The scores are now 12 to P^1 and -12 to P^2.

5.1. Memory usage

In the first experiments we have investigated the memory required to construct the player's model and the memory required to represent the player's model. The memory required by a player is proportional to the memory required when the player's model has been compiled into a junction tree. Therefore, to measure the memory requirements, we measure the size of the constructed junction tree for the model's influence diagrams ([6]). Here, we shall refer to the size of a junction tree as the number of cells in all cliques in the junction tree. The results are shown in Table 1. The 3rd column shows the size of the junction tree of the top level influence diagram used by the model. This model, however, is not sufficient for the model to be useful for playing Grid. As has been pointed out in the previous section, it is necessary to estimate the opponent's policy, which is done by solving a number of IDs representing his policy. The model (2,(2,(1,NIL))) for instance, uses (2,(1,NIL)) to estimate P^2's policy in M_1^2 and a model resembling (3,(1,NIL)) to estimate P^2's policy in M_2^2 and finally the (1,NIL) model

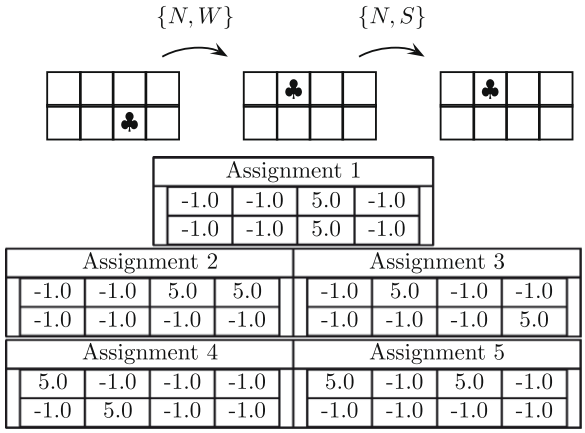

Fig. 12. An example of the game Grid. In the first move, P^1 chooses to move *N* while P^2 chooses to move *W*. In the second turn, P^1 and P^2 moves *N* and *S*, respectively, canceling each other's effect.

Table 1
Player models and the corresponding sizes of the required junction tree.

	Model	Top level model	Sum of required models
1	(1,NIL)	2559	2559
2	(2,(1,NIL))	22,190	24,749
3	(3,(1,NIL))	511,827	514,386
4	(4,(1,NIL))	16,079,864	16,082,423
5	(2,(2,(1,NIL)))	30,217	544,603
6	(3,(2,(1,NIL)))	1,012,406	17,628,846
7	(3,(3,(1,NIL)))	1,012,406	531,763,357
8	(4,(2,(1,NIL)))	37,560,163	568,333,304
9	(2,0,(1,NIL))	7150	9709
10	(3,(2,0,(1,NIL)))	511,827	549,609
11	(4,(2,0,(1,NIL)))	16,079,864	16,138,569

to represent P^2's concept of P^1's model. The 4th column in Table 1 shows the accumulated memory use for all the models required in order to construct the top level model.

The results in Table 1 further motivates the need for approximated models for solving SSBGs. The simplified models in rows 9, 10, 11 are based on LIMID players and require significant less memory than their complete memory counterparts.

5.2. Performance of LIMID players compared to regular players

In a second experiment we measure how well player P^1 with the approximated models plays *Grid* against P^2 playing with a (2,(1,NIL)) model. We compare these results with how well P^1 performs with an exact model. In all the models we shall assume that P^1 has the correct model of P^2, namely (2,(1,NIL)). We shall let P^1 use the approximated models presented in this paper. That is, P^1 will play with our proposed approximations of the (3,(2,(1,NIL))) model. We will denote with $(3,(2,(1,NIL)))^*$ a model that uses information enhancement when P^1 estimates the opponent's policy in M_3^2, i.e. she uses the model in Fig. 8. Also, $(3,(2,(1,NIL)))^{**}$ denotes a model where P^1 uses information enhancement when she estimates P^2's policy in M_3^2 and assumes that she knows his assignment when she decides M_3^1, i.e. the model in Fig. 9. $(3,(2,(1,NIL)))^{\dagger}$ is a model where P^1 uses information reduction when she estimates P^2's policy in M_3^2. More specifically, she assumes that when P^2 finds his policy in M_2^2 he has limited memory and remembers only the previous world state and his previous decision (see Fig. 10). $(3,(2,(1,NIL)))^{\dagger\dagger}$ makes the same assumptions with one addition, namely that also P^1 has limited her memory to the previous time slice in its 2nd decision.

It is likely that some combinations of assignments give one of the players an advantage. So, in stead of assigning the assignments to the players randomly we assign the assignment such that all combinations of assignments have been played the same number of times. In each experiment we let P^1 and P^2 play 200 games of each 10 moves. The results are summarized in Table 2. The column marked P^1 is P^1's average score per game. Notice that a positive number under P^1 means that on average P^1 has scored more than P^2. σ is the standard deviation of the scores and 'total size' is the total size of the required junction trees for the particular model.

As expected, the most advanced model performs best (row 1). But this model also has the highest memory requirements. The proposed approximations also gain positive scores with a significantly smaller amount of memory required. It is difficult to explain why the model $(3,(2,(1,NIL)))^*$ seems to perform worse than the simpler model $(3,(2,(1,NIL)))^{**}$.

Table 2
Average scores and standard deviations (σ) after 200 Grid games between P^1 playing with different models against P^2 playing with a (2,(1,NIL)) model. $(3,(2,(1,NIL)))^*$ uses information enhancement to estimate the opponent's policy in M_3^2 whereas $(3,(2,(1,NIL)))^\dagger$ uses information reduction to estimate the opponent's policy in M_3^2. For $(3,(2,(1,NIL)))^{\dagger\dagger}$ and $(3,(2,(1,NIL)))^{**}$ see the text below.

	Model	P^1	σ	Total size
1	(3,(2,(1,NIL)))	22.4	*29.3*	17,628,846
2	$(3,(2,(1,NIL)))^*$	4.8	*23.9*	903,108
3	$(3,(2,(1,NIL)))^\dagger$	9.9	*21.9*	1,065,801
4	$(3,(2,(1,NIL)))^{**}$	7.2	*23.3*	580,161
5	$(3,(2,(1,NIL)))^{\dagger\dagger}$	9.6	*20.6*	578,534

Overall, the two models using limited memory (†) seem to perform slightly better than the models based on information enhancement (*).

5.3. Comparison of players of different levels

In a third experiment we have investigated what happens if P^1 has overestimated or underestimated P^2's modeling level. We have chosen to do this experiment based on LIMID Players. At level 0 we will use a model with horizon 1, but we will use models with time horizon 2 and memory 2 on all levels above 0. So, when we investigate what happens when P^1 plays with a level 2 model against P^2 with a level 0 model P^1 will use (2,2,(2,2,(1,NIL))) while P^2 will use a (1,NIL) model. In this example P^1 has overestimated P^2's modeling level since P^1 has assumed P^2 to have a (2,2,(1,NIL)) model while he really has a (1,NIL) model. Moreover, P^2 can be said to underestimate P^1 because P^2 expects P^1 to use a (1,NIL) model while she really is using a (2,2,(2,2,(1,NIL))) model.

The results for P^1 with levels 1–7 against P^2 with levels 0 through 6 are shown in Table 3.

The numbers in the cells refer to the score of player P^1, i.e. in the first row, the (2,2,(1,NIL)) has scored on average 18.5 points in games against (1,NIL) who has scored on average −18.5. Each score is the average score of at least 50 games of each 10 moves. Standard deviations on the game outcomes are shown in italics. As expected, players win when they have the correct assumptions about the opponent, whereas it is likely to cause a disaster if P^1 over estimates P^2. For instance, with a level 7 model (row 7), P^1 wins only if P^2 is a level 6 player (which she assumes he is) or when P^2 is a level 2 player. In all the rest of the cases P^1 looses. Hence it seems not to be an advantage to overestimate the opponent.

If we look at the results in the eyes of P^2 we can see the consequences of underestimating the opponent. For instance, in the column labeled '1', P^2 is playing with a level 1 model assuming P^1 to be level 0, whereas P^1 is really playing with models of levels 2, 3, 4, 5, 6 and 7. P^2 is loosing when P^1 has the correct model of him as a level 2 player and when P^1 is playing as a level 6 player. When P^2 is using a level 2 model he is winning only if P^1 is playing as a level 6 player. Overall it seems to be a disadvantage to underestimate the opponent.

6. Related work and conclusions

As we have already briefly outlined, we are not the first to address how multiple agents reason about other agent's reasoning. Partially Observable Markov Decision Processes (POMDPs) have been extended to multi-agent scenarios. These extensions can be divided into two groups, objective and subjective approaches.

Table 3
Average scores and standard deviations (*italics*) obtained by players with $h = 2$ on different levels in a 4×2 instance of grid.

Level	0	1	2	3	4	5	6
1	18.5	–	–	–	–	–	–
	24.8	–	–	–	–	–	–
2	0.375	20.8	–	–	–	–	–
	29.27	*33.0*	–	–	–	–	–
3	−11.76	−14.5	11.5	–	–	–	–
	41.1	*36.8*	*37.8*	–	–	–	–
4	−7.92	−14.4	13.8	13.8	–	–	–
	36.2	*39.9*	*36.9*	*25.9*	–	–	–
5	8.64	−2.4	0.24	−0.96	12.24	–	–
	36.9	*35.2*	*18.9*	*11.8*	*34.5*	–	–
6	−4.68	6.24	−1.92	−6.72	−11.5	13.3	–
	33.7	*31.2*	*25.8*	*36.1*	*33.9*	*27.6*	–
7	−6.48	−15.24	7.8	−12.7	−16.3	−2.52	9.12
	42.1	*33.9*	*28.6*	*28.5*	*40.1*	*26.9*	*25.8*

Objective approaches solve the problem from a global perspective and find an optimal policy for each agent in the system. In this line of work we find stochastic games (or Markov games) [13]. They base their solution on *Nash Equilibria* (or more specifically *Markov perfect equilibria* or *Bayesian Equilibria*). These are equilibria derived from situations when all agents can be assumed to be equally informed about the game and when all players play completely rational. Bernstein et al. [14] proposed *decentralized POMDPs* (DEC-POMDPs) in which all agents share the same reward function i.e. the environment is cooperative. DEC-POMDPs suffer from notorious complexity problems and much work is currently being done addressing this issue (e.g. [15], [16] and many more). Moreover, DEC-POMDPs calculate one overall policy which, when found, must be distributed among the agents in the scenario. Critics of Objective approaches argue that they rely on the unrealistic assumption that all agents are equally informed about the game. Furthermore, in a Nash equilibrium, each agent will be indifferent about the various possible non-dominated actions as long as the other agents stick to their policies. Therefore, there is a problem of non-uniqueness and a problem of incompleteness. Non-uniqueness: there may be more than one equilibrium, and if so, the agent is still indifferent about which action he should choose. Incompleteness: in case there is evidence that the other agent deviates from the equilibrium strategy the first agent is still indifferent about the non-dominated actions.

Subjective approaches, on the contrary, focus on an individual agent's modeling of the environment and other agents and perform decision-theoretic reasoning from his perspective. Gmytrasiewicz and Durfee [17], propose a subjective approach. They propose the *Recursive Modeling Method* (RMM) in which they model an agent as a rational agent equipped with models of the other agents. The other agents are also assumed to be rational and they in turn may also be equipped with models of the other agents (including the first agent). Eventually the recursive modeling will be terminated by a "No information" model which does not have models of the other agents.

Gmytrasiewicz and Doshi [2] have proposed the *Interactive POMDP* (IPOMDP) which is a POMDP method that has emerged from RMM. In an IPOMDP the agent has to reason about the world and the other agents in a structure called an *interactive state*. The interactive state contains the actual state of the world and belief over the other agent's interactive states. The concept of interactive states coincides with Harsanyi's *type* [18]. A type encloses the player's belief of the true world state, his preferences (utility function) and his belief of the other players' types. This, in turn, includes their preferences and their beliefs of his beliefs and his preferences etc. Not surprisingly, IPOMDPs (like DEC-POMDPs) suffer from notorious complexity problems. Specifically, IPOMDPs suffer from the curse of dimensionality (the complexity of the belief representation) and the curse of time horizon. Finding ways for better scalability in IPOMDPs is an important research topic. Doshi and Gmytrasiewicz [19], for instance, address both complexity issues. They apply a particle filter for addressing the curse of dimensionality and a they sample a look-ahead tree for addressing the curse of time horizon. While Monte-Carlo methods often turn out to provide efficient solutions in many areas they still rely on sampling from a domain which may be overly complex. If, however, it would be possible to identify redundant information and in a novel way simplify the original models, all techniques (including Monte-Carlo techniques) would benefit from that.

Therefore, we have investigated the possibility of expressing multi-agent interaction in the language of influence diagrams (IDs). IDs decompose the situation into multiple chance and decision variables and they allow to use the conditional independence between the variables for a less memory consuming representation. Suryadi and Gmytrasiewicz [20] proposed a simple way to model the interaction of multiple agents using influence diagrams. This work, however, modeled the opponent's policies as probability distributions depending on the past in the game and did no effort in performing a particular modeling of the other agents.

Also Koller and Milch [8] have proposed the Multi-Agent influence diagram (MAID) and more recently Gal and Pfeffer [21] have proposed Networks of Influence diagrams (NIDs) which are based on MAIDs. However, they focus on calculating equilibrium solutions.

Polich and Gmytrasiewicz [22] propose an influence diagram framework which avoids the equilibrium solutions by extending the IPOMDP framework and propose the *interactive dynamic influence diagram* (I-DID). I-DIDs introduce a new type of node to the ID language, namely the modeling node. A Modeling node contains the agent's models of other agent's which in turn may be other I-DIDs. Thus, the modeling node and the nodes representing the actual world states (in each time slice) constitute the agent's interactive state. I-DIDs suffer from the same complexity issues as IPOMDPs and currently much work is going on addressing this issue. Recently, Doshi et al. [23] in particular address the curse of time horizon by reducing the dimensionality of the state space through conjecturing that models with spatially close beliefs are likely to be behaviorally equivalent. They cluster the possible opponent models into clusters of similar models using k-means and keep k models.

More recently, Zeng and Doshi [24] have proposed an exact method that reduces the model space in two steps. First they identify behaviorally equivalent models by analyzing the models' policy trees and merge them together into a single policy graph. Secondly they group together, at each time step, the models with equivalent actions into action equivalence classes. Despite, still having to keep track of the different possible player models inside each action equivalence class, the approach reduces complexity because only the equivalence classes need to be represented in the modeling nodes.

The work in the present paper resembles the work done in the I-DID framework. We use dynamic influence diagrams for modeling the agent's subjective decision-theoretic reasoning and its reasoning about other agent's reasoning. We take outset in the MAID representation framework, but we do not assume the players to be completely rational. That is, we assume them to have a limited analysis depth and a limited time horizon in their reasoning. This makes it possible to build influence diagrams representing the opponent's policy directly as a chance node. Our work also suffers from complexity issues. We show different ways to address the curse of time horizon in the models by either removing arcs or adding arcs in the influence diagrams. When removing arcs we use limited memory influence diagrams (LIMIDs) which were proposed by Lauritzen

and Nilsson [3] and when we introduce new arcs we use the idea of information enhancement (the assumption that the agent is fully informed at some point in the future) which was proposed by Jensen [25]. It turns out that these methods can significantly reduce the amount of memory required to represent the models.

We have left an important point untouched: before a game begins a player rarely knows how intelligent the opponent is. It should be possible to adapt to the opponent's level of intelligence during play. It is likely that due to the relative success of the models that underestimated the opponent, it is possible to beat a player at any level with adaptation. A simple indication of wrong modeling is given when P^2 makes a move, which is impossible according to the model. For the current experiments we have chosen to keep the model and instead change the belief of the opponent's assignment. When a 0 probability is observed the player's belief of the opponent's assignment is perturbed slightly such that all assignment which were previously deemed impossible, are assigned a non zero probability in the player's belief. We plan to address this problem in future research in order to exploit inconsistence between models and observations for choosing correct models of the opponent.

Acknowledgements

We want to thank the staff in the Machine Intelligence Group at the Department of Computer Science at Aalborg University and the anonymous referees for useful comments and suggestions.

References

[1] P.J. Gmytrasiewicz, E.H. Durfee, D.K. Wehe, A decision theoretic approach to coordinating multiagent interactions, in: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI), 1991, pp. 62–68.
[2] P.J. Gmytrasiewicz, P. Doshi, A framework for sequential planning in multi-agent settings, Journal of Artificial Intelligence Research 24 (2005) 49–79.
[3] S.L. Lauritzen, D. Nilsson, Representing and solving decision problems with limited information, Management Science 47 (2001) 1235–1251.
[4] R.D. Shachter, Evaluating influence diagrams, Operations Research 34 (6) (1986) 597–609.
[5] P.P. Shenoy, Valuation-based systems for Bayesian decision analysis, Operations Research 40 (3) (1992) 463–484.
[6] F. Jensen, F.V. Jensen, S.L. Dittmer, From influence diagrams to junction trees, in: Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI), 1994, pp. 367–374.
[7] Hugin Expert A/S, Hugin api reference manual version 7.0,<http://download.hugin.com/documents/manuals7.0/api-manual.pdf>, 2008.
[8] D. Koller, B. Milch, Multi-agent influence diagrams for representing and solving games, Games and Economic Behavior 45 (1) (2003) 181–221.
[9] T.D. Nielsen, F.V. Jensen, Welldefined decision scenarios, in: Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI), 1999, pp. 502–511.
[10] R.D. Shachter, Efficient value of information computation, in: Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI), 1999, pp. 594–601.
[11] J. Nash, Equilibrium points in N-person games, in: Proceedings of the National Academy of Sciences of the United States of America, vol. 36, 1950, pp. 48–49.
[12] D. Carmel, S. Markovitch, Incorporating opponent models into adversary search, in: Proceedings of the Thirteenth National Conference on Artificial Intelligence, 1996, pp. 120–125.
[13] D. Fudenberg, J. Tirole, Game Theory, The MIT Press, Cambridge, MA, 1991.
[14] D. Bernstein, R. Givan, N. Immerman, S. Zilberstein, The complexity of decentralized control of markov decision process, Mathematics of Operations Research 27 (4) (2002) 819–840.
[15] D.S. Bernstein, E.A. Hansen, S. Zilberstein, Bounded policy iteration for decentralized POMDPs, in: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI), 2005, pp. 1287–1292.
[16] S. Seuken, S. Zilberstein, Improved memory-bounded dynamic programming for decentralized POMDPs, in: Proceedings of the Twenty-third Annual Conference on Uncertainty in Artificial Intelligence (UAI), 2007, pp. 344–351.
[17] P.J. Gmytrasiewicz, E.H. Durfee, Rational coordination in multi-agent environments, Autonomous Agents and Multi-Agent Systems 3 (4) (2000) 319–350.
[18] J.C. Harsanyi, Games with incomplete information played by "Bayesian" players, Management Science 14 (3) (1967) 159–182.
[19] P. Doshi, P. Gmytrasiewicz, Monte Carlo sampling methods for approximating interactive POMDPs, Journal of AI Research (JAIR) 34 (2009) 297–337.
[20] D. Suryadi, P. Gmytrasiewicz, Learning models of other agents using influence diagrams, in: Proceedings of the Seventh International Conference on User Modeling (UM), 1999, pp. 223–232.
[21] Y. Gal, A. Pfeffer, A language for modeling agents' decision making processes in games, in: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2003, pp. 265–272.
[22] K. Polich, P. Gmytrasiewicz, Interactive dynamic influence diagrams, in: Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2007, pp. 1–3.
[23] P. Doshi, Y.F. Zeng, Q. Chen, Graphical models for interactive POMDPs: representations and solutions, Journal of Autonomous Agents and Multiagent Systems 18 (3) (2009) 376–416.
[24] Y. Zeng, P. Doshi, Speeding up exact solutions of interactive influence diagrams using action equivalence, in: Proceedings of the Twenty First International Joint Conference on Artificial Intelligence (IJCAI), 2009, pp. 1996-2001.
[25] F.V. Jensen, Approximate representation of optimal strategies from influence diagrams, in: Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM), 2008, pp. 153–159.