An Influence Diagram framework for acting under influence by agents with unknown goals

Nicolaj Søndberg-Jeppesen and Finn Verner Jensen
Department of Computer Science
Aalborg University
9220 Aalborg, Denmark

Abstract

We consider the situation where two agents try to solve each their own task in a common environment. We present a general framework for representing that kind of scenario based on Influence Diagrams (IDs). The framework is used to model the analysis depth and time horizon of the opponent agent 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 an algorithm based on Limited Memory Influence Diagrams (LIMIDs) in which we convert the ID into a Bayesian network and perform single policy update. 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. (1991) 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 number of tasks or assignments. 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 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 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.

Many of the RMM based approaches turn out to be PSPACE-hard (Gmytrasiewicz and Doshi, 2005). IDs are also facing notorious complexity problems due to the no-forgetting assumption (the assumption that everything that was known at a previous decision is also known at the current decision). Eventually the decision maker will have way too much information all of which being relevant for the current decision. Lauritzen and Nilsson (2001) propose Limited Memory Influence Diagrams (LIMIDs) in which the decision maker is assumed to have only a certain amount of memory. They propose an al-

gorithm called *single policy update*, which finds an approximation to the optimal policy. In this paper we shall propose an algorithm which is able to solve RMM based LIMIDs in multiagent scenarios.

2 Background

The kind of scenarios we are interested in consist of a finite set of *world states* $\boldsymbol{W}$ with states $w_1, w_2, \ldots, w_m$, and 2 *agents* P^1 and P^2. We assume P^1 to be female and P^2 to be male. The agents have finite sets of actions, say $\boldsymbol{Actions_{P^1}}$ and $\boldsymbol{Actions_{P^2}}$ with members $action^1_{P^1}, action^2_{P^1}, \ldots, action^k_{P^1}$ and $action^1_{P^2}, action^2_{P^2}, \ldots, action^k_{P^2}$ respectively. The transition between world states at time t to time $t+1$ is determined by a probabilistic function τ, where $\tau : \boldsymbol{W} \times \boldsymbol{Actions_{P^1}} \times \boldsymbol{Actions_{P^2}} \times \boldsymbol{W} \rightarrow \boldsymbol{[0;1]}$.
Furthermore, each agent has an assignment which reflects how much the agent prefers each world state by assigning a value to each state. Thus, agent P^1 and P^2, have a finite set of possible assignments $a_{P^1_1}, a_{P^1_2}, \ldots, a_{P^1_l}$ and $a_{P^2_1}, a_{P^2_2}, \ldots, a_{P^2_m}$ respectively. We will consider only scenarios where the world state is always known by all agents but the actual assignments of the other agents remain hidden. A probability distribution of the opponent's assignment can however, be obtained by observing his/her actions.
You may consider the scenario as a board game, where each player wishes to obtain certain pattern on the board. The players receive their pattern assignments by drawing cards from a deck. The payoff function, which assigns payoffs to each player at each time step, is a function of the current world state and the assignment of the two agents. We shall refer to the scenario as *covert interference*(CIF).

2.1 Influence Diagrams

We shall use the classical paradigms from Probabilistic Graphical Models (PGMs). A graphical model 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 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 of C given $pa(C)$.

A *solution* to an ID is an *optimal strategy*. A strategy consists of set of *policies*, one for each decision node. A policy for a decision node 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 (Shachter, 1986; Shenoy, 1992; Jensen et al., 1994; Hugin Expert A/S, 2007). 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 (2003) introduced *Multi-Agent Influence Diagrams* (MAIDs). The various acting agents are given decision and utility nodes of particular colors (or shadings).

2.2 Graphical representation of CIF

We adapt the framework of MAIDs to CIF. This is illustrated in Figure 1. In Figure 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 P^1-labels represent the moves by player P^1. The diamond shaped nodes, which are half lightly shaded and half darkly shaded represent the payoff matrixes, 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

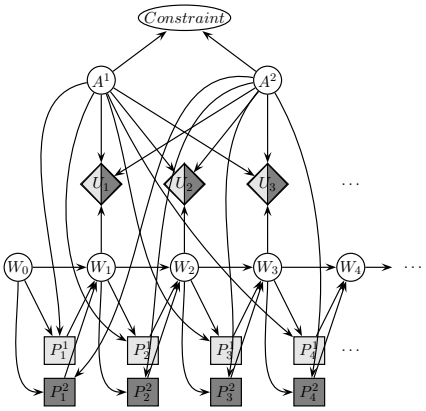


Figure 1: A MAID representation of CIF.

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 e.g. they might not be able to 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 actions 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 actions knowing only W_1 and each their own assignments. When both players have decided their actions the game continues with time $t = 1$. At $t = 1$ P^1 still knows only her own assignment but if she knows P^2's policy she can estimate P^2's assignment in A^2.

Even if there is a pre-specified time horizon, the standard methods for solving IDs ((Shachter, 1986; Shenoy, 1992; Jensen et al., 1994)) 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 (Nash, 1950).

We consider the situation, where we wish to construct a computer program to play against human players. As we cannot expect human players to play Nash equilibria, the computer shall exploit that, and therefore it usually shall not play Nash-equilibria either.

2.3 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 analyse the situation to a certain depth and with a certain lookahead of moves, and in the depth analysis they will make some assumptions about the other players' analysis depth and look-ahead. This is called the *recursive modeling method* (RMM) (Gmytrasiewicz et al., 1991), 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 Figure 1 can be transformed to an ID, where P^2's decision nodes are replaced with chance nodes, and $P(P^2_{i+1}|A^2, W_0, \ldots, W_i, P^2_0, \ldots, P^2_i)$ is the policy (see Figure 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 uses 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. As we let P^1 be the computer and P^2 the human, we assume P^1 to have a larger analysis depth than P^2.

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 2 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 P^1 takes into account P^1's *time horizon*. 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.

Definition 1. A player P is a pair defined as follows:

1. $P = (h, NIL)$ is a player with time horizon h and modeling level 0.
2. Given a player O, with modeling level $i-1$, $P = (h, O)$ is a player with time horizon h and modeling level i.

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. Note that, it is possible to define models in which P^1 assumes that P^2 has a longer time horizon than herself. Scenarios where players want to maximize a short time gain playing against players with a long time horizon are common at, for example, stock exchanges.

Figure 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 in the nodes P_1^2 and P_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 P_1^1 and P_2^1 are found by analyzing the rightmost ID, which represents the deepest model $(1, NIL)$. In the $(1, NIL)$ model the chance node P_1^2 represents the completely random strategy which P^2 is assumed to play on level 0. Note that the leftmost ID in Figure 2, contains extra arcs, namely the arc connecting the nodes P_1^2 and P_2^2 and the arc connecting W_0 and P_2^2. These arcs represent that P^2 in time step 1 when P^2 is taking decision P_2^2 will remember W_0 and P_1^2 due to the no-forgetting-assumption (the assumption that whatever was known when taking a decision at time $i < t$ is also known at time t).

The not-forgotten information can be used to *learn* the opponent's assignment. Look at the ID in the center in Figure 2. At t=1 P^2 can use his model of P^1 to learn P^1's assignment in A^1. P^2 will calculate, $P(A^1|W_0, W_1, P_1^2)$ which can be found using the the ID. Thus, the no forgetting assumption has an impact as P^2 can be assumed to have learned something about the state of A^1.

The modeling of the learning opponent causes the dimensionality of the conditional probability table representing P_i^2 to grow heavily with the number of future world states to take into account. In this work we are addressing this problem by reducing the complexity introduced by the no-forgetting assumption in IDs.

3 Agents with limited memory

The no-forgetting assumption results in notorious complexity problems for IDs. In the study reported in (Søndberg–Jeppesen and Jensen, 2008) we used IDs directly in situations like the model in Figure 2 and we ran into serious complexity problems. In our framework we have seen that not only does it impact the decision maker when solving the ID, it also causes the chance nodes representing the opponent's decisions to have intractably large domains.
To address this problem we apply LIMIDs which provide a way to address the complexity problems in IDs by assuming that the decision maker

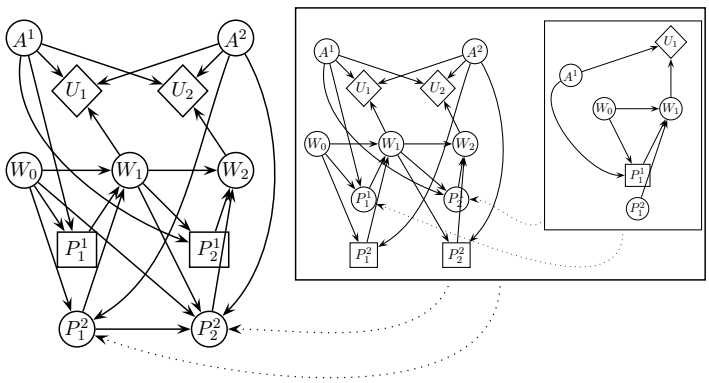


Figure 2: An ID representing the $(2,(2,(1,NIL)))$ model.

has limited memory (Lauritzen and Nilsson, 2000). Syntactically, LIMIDs are like IDs with the only difference that the only things known at a decision node D are nodes with arcs going into D, $(pa(D))$. This means that all the information that the decision maker is assumed to remember when making the decision must be expressed explicitly by information arcs. In our framework we introduce a LIMID player as a player with a certain memory. Like a regular player from Definition 1 a LIMID Player has a time horizon and a model of the opponent, however the LIMID Player also has a certain memory. If P^1 is a LIMID player with memory m, she will at decision P_i^1 remember her previous m decisions $P_{i-m}^1 \dots P_{i-1}^1$, together with the previous m world states $W_{i-m} \dots W_{i-1}$.

Definition 2. A LIMID player L is a triple defined as follows:

1. $L = (h, m, NIL)$ is a LIMID player with time horizon h, memory m and modeling level 0.

2. Given a player or a LIMID player O, with modeling level $i-1$, $L = (h, m, O)$ is 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.

Single policy updating is an iterative algorithm which is guaranteed to converge towards a local maximum policy (Lauritzen and Nilsson, 2000).

Before we describe how the single policy updating algorithm works, we will describe how a policy can be found at each decision by first converting an ID into a Bayesian network.

3.1 Finding optimal policies

We convert an ID into a Bayesian network by converting the utility nodes and the decision nodes into chance nodes. This method was first proposed by (Cooper, 1988).

First replace each decision node P_i^1 with a chance node with the parents $pa(P_i^1)$, the relevant information nodes for the decision. The possible outcome of this new node is the possible decisions in the decision node. Eventually, this node shall receive a probability distribution representing an optimal policy at this node. Initially the node has uniform priors $P(P_i^1 = d|pa(P_i^1)) = 1/N_i$ for all decisions d in P_i^1 where N_i is the number of possible decisions.

Next, each Utility node U_i is replaced by a two-state chance node NU_i with the same set of parents but with possible outcomes y and n. For convenience we will write nu_i instead of $NU_i = y$ and $\overline{nu}_i$ instead of $NU_i = n$ throughout the rest of the paper. We need to scale the utility values from U_i into the interval $[0;1]$. Let $\mathcal{X}_{pa(NU_i)}$ denote the possible configurations of the variables in the set $pa(NU_i)$, with $x_{pa(NU_i)} \in \mathcal{X}_{pa(NU_i)}$. Now nu_i receives its probability distribution according to the function:

$$P(nu_i|x_{pa(NU_i)}) = \frac{U_i(x_{pa(NU_i)}) - u_{min}}{u_{max} - u_{min}}, \quad (1)$$

where $U_i(x)$ is the utility value for a configuration x according to U_i, while u_{max} and u_{min}

are the maximum and minimum values of U_i. Since we have assumed that the utility function is the same for all U_i the sum of normalized utilities and the sum of utilities are maximal for the same decision.

We shall show how this Bayesian network can serve to find an optimal policy.

Theorem 1. *Given a player with time horizon h and ID $\mathcal{I}$ which has been converted to a Bayesian network $\mathcal{B}$, and let $\boldsymbol{e} = pa(P_i^1)$. The optimal policy $\delta_{P_i^1}(pa(P_i^1))$, can be determined in the following way,*

- *If P_i^1 is the last decision (i.e. $i = h$)*

$$\delta_{P_i^1}(\boldsymbol{e}) = \operatorname*{argmax}_d P(d, |nu_i, \boldsymbol{e}). \quad (2)$$

- *If P_i^1 is not the last decision (i.e. $i < h$) but the CPTs in $P_{i+1}^1, \ldots, P_h^1$ have been replaced with their optimal policies, then $\delta_{P_i^1}(pa(P_i^1))$ is:*

$$\delta_{P_i^1}(\boldsymbol{e}) = \operatorname*{argmax}_d \Big(\sum_{j=i}^{h} P(d, \boldsymbol{e}|nu_j) P(nu_j|\boldsymbol{e}) \Big). \quad (3)$$

The proof for Theorem 1 is a trivial rewrite of the original in (Cooper, 1988) and has been omitted in this paper.

After junction tree propagation with evidence nu_i, the clique containing P_i^1 will hold $P(P_i^1, \boldsymbol{e}, nu_i)$. Hence,

$$\delta_{P_i^1}(\boldsymbol{e}) = \operatorname*{argmax}_{P_i^1} \frac{P(P_i^1, \boldsymbol{e}, nu_i)}{P(\boldsymbol{e}, nu_i)}.$$

When you have an ID with only one decision, then Theorem 1 can be used for an efficient calculation of an optimal policy. We will exploit this in the next section.

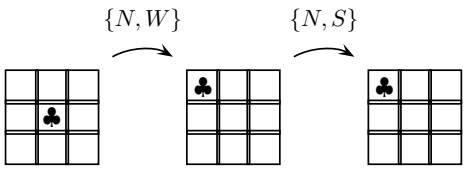
3.2 A LIMID framework

Single policy updating for LIMIDs consists in a series of policy estimates for IDs with one decision. The decision nodes are represented as chance nodes with the information parents as parents. You start off with some policy for all decisions, and then you systematically update the policies for each decision. In each iteration, each local policy $\hat{\delta}_{d_i}(pa(d_i))$ is calculated according to the above trick with the last decisions first. The algorithm iterates until convergence (i.e. no policies change in two consecutive iterations) or for a predefined number of iterations.

- Given a LIMID Player with horizon h, memory m and ID $\mathcal{I}$.
- Convert $\mathcal{I}$ into a Bayesian network $\mathcal{B}$.
- for each decision node P_i^1 in $\mathcal{B}$,
  - add $P_{i-2}^1, P_{i-3}^1, \ldots, P_{i-m}^1$ as parents to P_i^1, and
  - add $W_{i-2}, W_{i-3}, \ldots, W_{i-m}$ as parents to P_i^1.
- Repeat until convergence:
  - For each decision P_i^1, for $i = h, h-1, \ldots, 0$:
  - update $\hat{\delta}_{P_i^1}$ according to Theorem 1.

4 Experimental results

In order to measure the performance of our proposed algorithm we present a simple game which we call Grid. Grid is played by two players P^1 and P^2 that are moving the same single piece on an $m \times m$ grid. Initially, both players are randomly given an assignment, which is a reward function for the positions in the grid. The players may have the same assignment. In each turn the players observe the position of the piece (i.e. the state of the game board) and decide to move it either up, down, right or left, (N, S, E and W). The actions the players have chosen are carried out simultaneously and the resulting effect on the piece is the 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 $m \times m$ 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 at least one of the actions cannot be carried out, the single



Assignment 1			Assignment 2		
8.55	6.82	1.15	-1.66	0.42	6.84
-6.82	0.0	4.12	-0.51	0.0	-0.42
-8.55	-1.15	-4.12	-6.84	0.51	1.66

Figure 3: 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, cancelling each other's effect.

action which can be carried out (if any) is carried out. 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 competition so P^1 is punished when P^2 is rewarded and vice versa. This is obtained by subtracting P^2's reward from P^1's reward and vice versa. The game continues for a predetermined number of turns.

An example of Grid with $m = 3$ and 2 possible assignments is shown in Figure 3. Initially, player P^1 and P^2 are assigned Assignment 1 and Assignment 2 respectively. In the first move, P^1 decides to move N while P^2 decides to move W. From the resulting new state, P^1 gets the reward 8.55 while P^2 gets the reward -1.66. Now the scores are $8.55 - -1.66 = 10.21$ to P^1 and $-1.66 - 8.55 = -10.21$ to P^2. In the second move P^1 decides to move N while P^2 decides to move S in which case the board state is not changed. The scores are now 20.42 to P^1 and -20.42 to P^2.

We have implemented Grid together with our proposed algorithm using (Hugin Expert A/S, 2007). The experiments have been performed on a 1.6 GHz PC with 1 GB of RAM.

4.1 Comparison of memory limitations

In the first experiments we have investigated how deep a time horizon players are allowed to get on our system. We start with a 3×3 instance of Grid with 5 assignments and create both a player with a regular ID and a LIMID player to see which values of h we can use in the models before our system runs out of memory. The results are summarized in Table 1. As expected the LIMID allows significantly larger values of h than IDs do.

Table 1: The maximal time horizons possible on our system with different sizes of the Grid game.

Board	ID max h	LIMID max h ($m = 1$)
3×3	4	32
5×5	3	8
7×7	3	8
9×9	2	8

Table 2: Average scores and standard deviations (σ) after 100 Grid games between different models against (2,(2,(1,NIL))).

	Model	P^1	σ
1	(2,(2,(2,(1,NIL))))	5.03	*10.1*
2	(2,1,(2,(2,(1,NIL))))	-0.248	*8.66*
3	(3,(2,(2,(1,NIL))))	7.32	*10,7*
4	(3,2,(2,(2,(1,NIL))))	0.252	*8,27*

4.2 Performance of LIMID players compared to regular players

In a second experiment we have measured how well a LIMID player plays compared a normal player with the same time horizon. We performed experiments with a 3×3 instance of Grid with 5 assignments. Each model has played 100 games of each 10 moves as player P^1 against (2,(2,(1,NIL))) who played as player P^2. The results are summarized in Table 2. When P^1 is playing with the ID models she outperforms P^2 (rows 1 and 3). This is no surprise since she uses more advanced models than P^2. With the LIMID models however (rows 2 and 4) P^1 looses slightly when she plays with lookahead 2 and memory 1 while she wins slightly when she plays with lookahead 3 and memory 2. The σ column indicates a huge variation in the game outcomes.

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

Level	0	1	2	3	4
1	2.47	–	–	–	–
	5.41	–	–	–	–
2	-1.83	3.24	–	–	–
	6.39	*10.76*	–	–	–
3	-3.19	-4.50	9.29	–	–
	7.64	*10.9*	*10.5*	–	–
4	0.55	-4.60	-0.73	8.00	–
	7.06	*10.0*	*5.82*	*10.6*	–
5	0.572	1.18	-6.21	4.40	5.78
	6.36	*7.34*	*10.8*	*10.0*	*8.84*

4.3 Comparison of players of different levels

In a third experiment we have investigated what happens if P^1 has a wrong model of P^2. We do that by letting P^1 assume that P^2 is more intelligent than he really is. Table 3 shows the average scores of Grid games between players of different levels all with $h = 3$ on a 3×3 board. Level 0 refers to the (3,NIL) model, level 1 refers to the (3,(3,NIL)) model etc. The numbers in the cells refer to the score of the row player, i.e. in the first row, the (3,(3,NIL)) has scored on average 2.47 points in games against (3,NIL) who has scored on average -2.47. Again the players play 100 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 seems to be less optimal to assume that the opponent is more intelligent than he actually is. This is problematic since a player rarely knows exactly how intelligent the opponent is before a game begins. Rather it should be possible to learn the opponent's level of intelligence during play. We will address this problem in future research.

Acknowledgments

We want to thank the staff in the Machine Intelligence Group at the Department of Computer Science at Aalborg University. In particular, we are grateful to Zeng Yifeng for help and valuable comments during this work. We also thank the anonymous reviewers for their useful feedback on this work

References

G. F. Cooper. 1988. A method for using belief networks as influence diagrams. In *Fourth Wrokshop on Uncertainty in Artificial Intelligence*, pages 55–63.

Piotr J. Gmytrasiewicz and Prashant Doshi. 2005. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*, 24:49–79.

P. J. Gmytrasiewicz, E. H. Durfee, and D. K. Wehe. 1991. A decision theoritic approach to coordinating multiagent interactions. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI 91)*, pages 62–68.

Hugin Expert A/S. 2007. Hugin api reference manual version 6.7. http://download.hugin.com/documents/manuals6.7/api-manual.pdf.

F. Jensen, F. V. Jensen, and S. L. Dittmer. 1994. From influence diagrams to junction trees. In R.L. Mantaras and D. Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 367–374. Morgan Kaufmann.

D. Koller and B. Milch. 2003. Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45(1):181–221.

S. Lauritzen and D. Nilsson. 2000. Evaluating influence diagrams using LIMIDs. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 436–445.

J. Nash. 1950. Equilibrium points in N-person games. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 36, pages 48–49.

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

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

N. Søndberg–Jeppesen and F. V. Jensen. 2008. Acting under interference by other agents with unknown goals. In *Proceedings of the 10th Scandinavian Conference on Artificial Intelligence (SCAI).*