Putting MAIDs in Context

Mark Crowley
Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4
crowley@cs.ubc.ca

Abstract

Multi-Agent Influence Diagrams (MAIDs) are a compact modelling language for representing game theoretic settings. We show various examples of games where this representation shows clear advantages in every area, as well as types of games where MAIDs do not have an advantage over extensive form trees. We also propose modifications that will close this gap and maintain the strengths of this representation.

Introduction

Many approaches to problems in Artificial Intelligence research come down to finding ways to represent a large problem in a way that is more compact or easier to use. In decision theory *influence diagrams* (Howard & Matheson 2003) have been used for two decades to help greatly in the modelling of single agent decisions. *Bayesian Networks* (BNs), (Pearl 1988) which themselves are simplifying models for chained Bayesian inference, are frequently extended with influence diagram semantics. This has allowed the efficient and general algorithms for BNs to be made available for decision problems modelled with influence diagrams. Koller (Koller & Milch 2001) first introduced a method for further generalizing to represent networks of decisions for multiple agents using *Multi-Agent Influence Diagrams* (MAIDs) which allow these techniques to apply to game theoretic situations. This paper gives an overview of the features and structure of MAIDs, and considers the benefits and drawbacks of the original definition. We argue that these drawbacks are not without remedy, and that there are improvements which can be made. We will show both good and bad examples of uses of MAIDs and propose extensions to the modelling language to reduce the negative effects while still maintaining the most important benefits on computation of equilibria and modelling simplicity.

Multi Agent Influence Diagrams

We begin by briefly providing an overview of Multi-Agents Influence Diagrams (MAIDs). For a more thorough description see (Koller & Milch 2001).

A MAID is a modification of influence diagrams, (Howard & Matheson 2003) well known in decision analysis and Artificial Intelligence research. An Influence diagram is a directed acyclic graph containing nodes representing the variables of the decision problem each having their own domain of values. The set of parents of a node X is denoted *Pa(X)* and we refer to one particular assignment of values to those parent values as $\mathbf{pa} \in dom(Pa(X))$. These variables are of three types: *chance*, *decision* and *utility* variables (see *figure 1(ii)(iv)*). Chance and decision variables can have associated conditional probability tables (CPDs) that map each permutation of parents of that node (all **pa**'s) to a value in its domain with a certain probability. Chance variables always have a CPD while if a decision variable has one, then we call it a *decision rule*. Utility nodes have a similar table where each $\mathbf{pa} \in dom(Pa(U_i))$ has an associated value in $dom(U_i)$ with probability of either 1 or 0.

In a traditional influence diagram there is only one agent. In a MAID there are a set of agents A. Each decision node $D \in D_a$ or utility node $U \in U_a$ is associated with a particular agent $a \in A$. A *strategy* σ_a for agent a is an assignment of a decision rule to all of a's decisions. A *strategy profile* σ contains a strategy for all agents.

Once a strategy profile is determined, then decision nodes become identical to chance nodes in that they are defined by a probability distribution. We denote this MAID by $M_{[\sigma]}$ and can think of it as a Bayesian Network (BN)(Pearl 1988) with a joint probability distribution $P_{M[\sigma]}$. Then the expected utility to an agent a of a strategy σ is :

$$EU_a(\sigma) = \sum_{U \in U_a} \sum_{u \in dom(U)} P_{M[\sigma]}(U = u) \cdot u \qquad (1)$$

Now, given two strategies we can compare them using their expected utility in a BN.

Definition 1 *A strategy σ^*_ε giving decision rules for all decision nodes in the set $\varepsilon \in D_a$, is* optimal *for strategy profile σ if for a maid $M_{[-\varepsilon]}$, where all the decisions not in ε have decision rules assigned, σ^* has a higher expected utility than any other strategy over ε.*

In other words, σ^*_ε is the best solution to optimally completing the decisions in $M_{[-\varepsilon]}$.

Definition 2 *A strategy profile σ is a* Nash equilibrium *if σ_{D_a} is optimal for all $a \in A$.*

Extensive Form Games

A MAID can be converted into an extensive form game tree in a straightforward manner. It involves using the natural ordering of variables in the graph to split each level of the tree symmetrically over the domains of the variables. Nodes in the tree are joined into information sets if they correspond to the same instantiation of values of their parents *in the MAID*. Variables in the MAID graph can have several parents, all of which are at various heights above that variable in the game tree. This produces arbitrarily complicated information sets from straightforward graph structures (see *figure 1* for some simple examples).

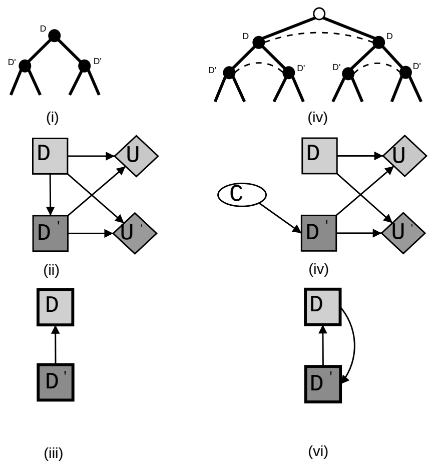

Figure 1: (i-iii)Tree, MAID and s-graph for a turn-based, perfect information game (iv-vi)Tree, MAID and s-graph for a simultaneous, imperfect information, Bayesian game

Strategic Relevance

Definition 3 *If* δ *is the decision rule for D under strategy profile* σ *then D* strategically relies *on D' if* δ *is optimal for* σ *but* δ *is not optimal for another* σ′ *that differs at decision D' only.* [1]

If we have such a δ then we know that the decision rule that is chosen at *D'* is essential for us to construct δ optimally. Therefore δ′ should be optimized first. We can construct a *strategic relevance graph (s-graph)* to represent these relationships, drawing a directed edge from *D'* to *D*. A child strategically relies on its parent in this graph.

Computing Equilibria

Once the s-graph is constructed, if it is acyclic, as in *figure 1(iii)*, then a simple algorithm to compute an optimal strategy profile is to randomly set a strategy for each agent and then optimize each decision in order starting with the decision from the s-graph that has no parents. Since it does not strategically rely on any other decisions, it can be optimized easily. It will contain an optimal decision rule which appears as a chance node for all other decisions. We then continue back through decisions until all decisions in the game are optimized.

In the more general situation where the s-graph is cyclic (as in *figure 1(vi)*) then we can locate the maximal strongly connected components (SCCs). These are maximal subsets of nodes such that each node has an edge to each other. This example has only one, the entire graph, but a later example will have several (*figure 5*). Each of these SCCs can be seen as a subgame that has all the information it needs to be optimal. Koller shows an algorithm similar to the one described above where each SCC, starting from the one with no outside strategic reliance, is solved by a standard game solver algorithm. These optimal decision rules are worked into the MAID and the next SCC is solved. They argue that this should be more efficient than solving the entire game since standard game solvers generally have performance that is worse than linear in the number of levels of the game tree. Here we are solving a larger number of small games.

[1](Koller & Milch 2001) also adds that there is no other δ′ that is identical to δ except on zero probability events, but it is not pertinent to our discussion here

Real World Applications

We have seen that any MAID representation can be turned into an extensive form game but it is not clear whether the opposite, turning arbitrary extensive form games into MAIDs, is always possible. It turns out that certain types of games that seem very natural in extensive form are very awkward in the new influence diagram notation.

Types of games that effectively use MAIDs

There are a few types of games where MAIDs offer clear advantages over extensive form game trees:

- The s-graph is acyclic giving a clear ordering to optimize decisions. This type of game is generally turn based with perfect or almost perfect information available to agents about decisions made by agents (or at least a signal about it). **SPACE** : more compact; usually simpler than a tree. **TIME** : no significant advantage on computing equilibrium, but we can use Bayesian inference for all sorts of interesting questions.
- The s-graph is cyclic with no SCC or the graph is fully connected. Now all the decisions depend, either directly or indirectly, on all others. **SIZE** : the MAID is still much more compact than the tree. **TIME** : no advantage for equilibrium since it must solve the whole game as one
- The s-graph has multiple SCCs. This MAID has regions of decisions that are fully connected and exposing their information to each other, connected by incomplete information between these regions. **SPACE** : big savings; the number of entries in the tables of the BN will be linear in the number of decisions whereas the trees will have exponential number of leaves. **TIME** : can solve 'subgames' given by the SCCs more efficiently since they are trees with less levels.

A Good Example For this and following examples we make a few extensions to the MAID syntax in order to make graphs more readable. First, is the use of notation of the form D_i^a to indicate the i^{th} decision for agent a. This will be used for utility and chance variables as well, the original definition uses colours to indicate the owning agent. We also add dotted boxes such as those used in UML for package grouping to allow one arrow to a dotted box to indicate that there is in fact one arrow going to each variable in that box. If an arrow needs to go directly to a particular variable it will cross into the box and touch that variable.

Example 1 *Consider the game shown in* figure 2(i) *a multi-round racing game where there are* n *agents racing and* r *rounds. Each agent* a *is in a lane with three cars to choose from that are always used for that lane. They have private information C_r^a about which car they prefer to drive which effects their utility for that race U_r^a which can one of three values denoting their finishing position in the race and this is effected by the cars chosen by the other agents. In the next round each agent shifts over one lane (mod the number of lanes) and the car used in that lane last round has a slightly better utility since its tires are still warm.*

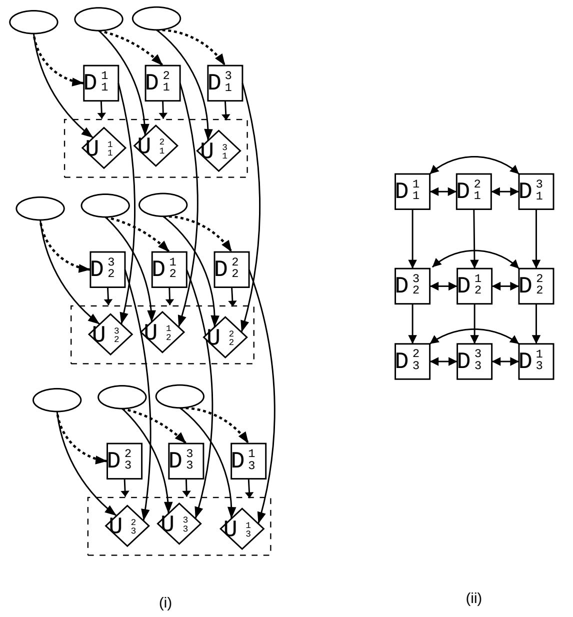

Figure 2: MAID for the racecar example

The extensive form tree for this game has *nr* decisions with 3 possible choices each, and *n* chance nodes indicating which of the 3 cars is preferred, giving 3^{2nr} leaves. For *n=r=3* as in our figure this is $3^{18} \approx 4 \times 10^8$. For the MAID we count the sizes of the CPDs for all variables as follows:

- 9 decisions $\times$ 3 choices $\times$ 3-valued-parent = 81
- 3 utility nodes $\times$ 3 ranks $\times$ 4 $\times$ 3-valued-parents = 108
- 6 utility nodes $\times$ 3 ranks $\times$ 5 $\times$ 3-valued-parents = 270

This yields 459 entries needed in our CPDs for the MAID, which in the general case is $O(n^2 r)$. This is clearly a lot more compact than the tree representation. In addition, as we see in *figure 2(ii)*, the s-graph has one SCC for each round of the race, thus allowing us to optimize decisions for each round starting with the last one in a backward induction equivalent manner. Each of these round games will only be *n* levels deep making each step of the solution faster.

Types of games that ineffectively use MAIDs

While this seems positive, there are unfortunately many natural games where MAIDs, in the form presented so far, are less helpful. One of the main drawbacks of the modelling language is that it does not support symmetric games.

Definition 4 *An* asymmetric game *is a game where at least one level of the extensive form tree is not filled straight across with nodes or there are two nodes n_i and n_j on the same level such that either*

1. *the agent associated with n_i is different than the one for n_j*
2. *n_i and n_j have a different number of branches beneath them*
3. *n_i and n_j have the same number of branches but not the same choices associated with them*

A MAID is always implicitly symmetric. Remember, that to transform a MAID to a game tree we made each variable into a level of the tree. This is not true in the reverse. Each level in the tree no longer contains only a single variable, but can contain mixtures of different variables or different domains for those variables. Since each node in a MAID represents all possibilities and values of one variable, the CPD table will always reflect that and lose an opportunity to be more compact. Generally an asymmetric tree can be exponentially more compact than its equivalent MAID.

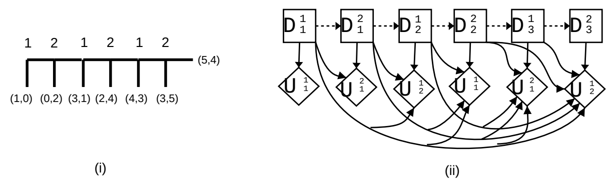

Figure 3: Tree and MAID for the asymmetric centipede game

This is potentially a huge drawback since many natural games are asymmetric. To get an intuition about this we will look at a simple example. Consider the standard centipede game with structure and payoffs as shown in *figure 3(i)* in extensive form. This is a highly asymmetric game where each agent has the option to end the game at any time. The naive representation of this as a MAID in *figure 3(ii)* is not encouraging. As we can see, for each decision we need to associate a utility node that depends on all previous decisions in order to account for the permutations where the game is actually over. Since each utility node has a CPD table over its domain, given the domains of all its parents, we can see that the size of the last utility node alone is prohibitive:

$$P(U_6|D_1,D_2,D_3,D_4,D_5,D_6) = 2^6$$

Moreover, $2^5 - 1$ of these are useless in deciding D_6 since the game will be stopped by an earlier decision. Only two rows in the whole table have non-zero payoffs, where all $D_1 \ldots D_5$ go across and D_6 can actually make a difference to the payoff.

A Bad Example There are even problems when there is no way for an agent to end the game prematurely, as long as there is some persistent asymmetry.

Example 2 *Suppose* n *game theorists are chosen as contestants on a popular tv gameshow called* Survival*. The game involves placing all the players on a remote (but scenic) island where they participate in multiple rounds of voting to see who is kicked off the island. If there is a tie in a vote, then amongst those that are tied the one with the lowest index is removed. In the final round, a game of chance determines the winner between the remaining two players. The winner receives 1 billion dollars, all others receive nothing.*

The game tree for this example would be full but asymmetric. After n levels it is possible to have nodes for different agents side by side. This is because in a particular subtree, after everyone has voted once, some agent a_i will have been voted off and will no longer get to cast votes. However, in another subtree it might be a different agent that was voted off and the tree beneath it will reflect this situation. This is straightforward to represent with a tree in a compact manner. It is still very large as the tree will have $O(n!)$ leaves.

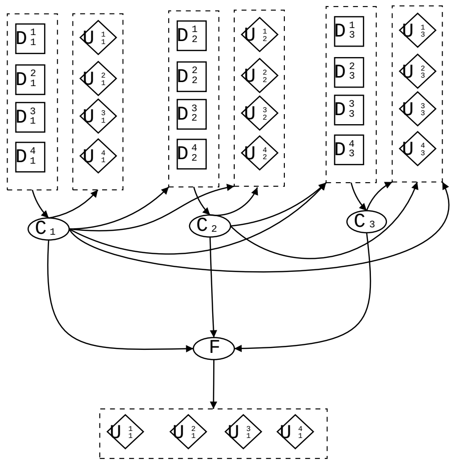
Figure 4: MAID for the Survival example

Our MAID representation however, (*see figure 4*), is actually worse. We see again that utilities in later rounds need all the information about previous decisions. Decisions only see a masked signal from a deterministic chance node C_i that indicates who was voted off, this does not change the complexity. There is no way in the standard BN representation to express different contexts when a decision need not be made at all, as we can in the tree. So the CPD tables of the utility nodes will have their values set to simply ignore votes from a player D_i^a, if any $C_1 \ldots C_{i-1}$ has a value of a_i. This leads to an enormous amount of duplication for situations that will never occur. The utility nodes in the last round have *n-1* inputs, each one with n possible values yielding a table with n^{n-1} entries. Things could hardly be worse.

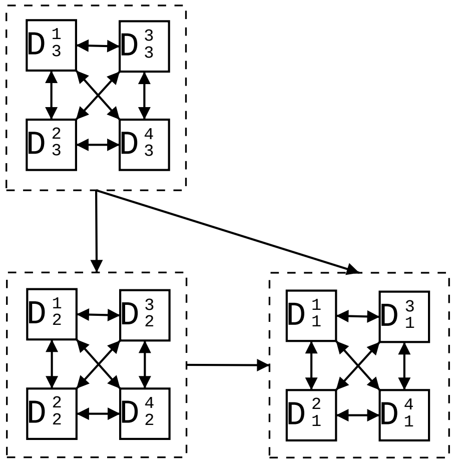
Figure 5: s-graph for the Survival example

Before looking at how to improve this situation, we should briefly consider the value of getting the MAID back down to at least the size of the tree. From the s-graph of *Survival* for *n=4*, in *figure 5*, we can see that there are well defined SCCs that could be taken advantage of using the algorithm from (Koller & Milch 2001). Remember, the arrows between dotted boxes indicate that all decisions in one SCC strategically rely on *all* decisions in the other SCC. However, since it is one way we can still perform backward induction by solving each round of the game, beginning with the last decision round, where there are 3 agents left. This would produce smaller trees, that is, with smaller n's to solve. Thus for n's where the space needed is still feasible, we can compute the equilibrium more efficiently than by solving the entire game tree.

Context Specifi c Independence

Much work in the last decade has focused on exploiting contextual information in Bayesian Networks. Lack of space does not allow a thorough explanation but (Boutilier *et al.* 1996), (Zhang & Poole 1999) and (Poole & Zhang 2003) provide good explanations and an interesting progression from specific to general solutions. The core observation is that some situations can be described as *contextually independent* of one another. A *context* is simply an instantiation of some actual values to a set of variables. In the games we have seen, it would correspond to a set of decisions to end the caterpillar game in the previous step, $D_{i-1}^2 = across$, which would make the value of D_i^1 irrelevant to the payoff (ie. $U_i^1 = 0$ in all cases).

More formally, two variables U and D are contextually independent given some context $C = c$ where $c \in dom(C)$ if:

$$P(U|D = d, C = c) = P(U|D = d', C = c)$$

for all $d, d' \in dom(U)$ such that $P(D = d, C = c) > 0$ and $P(D = d', C = c) > 0$. That is, once we know the assignments of the variables in *C*, the variable *D* will not influence the outcome of *U* as long as we are not dealing with zero probability events.

It is not surprising that this relationship is often expressed using trees. Boutilier (Boutilier *et al.* 1996) formulated contextual probability trees (CPTs) and showed efficient algorithms for network clustering and cut-set conditioning. The latter took up less space than traditional methods but actually took longer to compute. Further extensions in (Zhang & Poole 1999) were used by Guestrin et al. (Guestrin, Venkataraman, & Koller 1996) in extending MAID-like graphs to cooperative multiagent environments using factored MDPs. Poole (Poole & Zhang 2003) shows a more general approach holding for any contextually independent situation, and a modification of BNs to include contextual information, while still allowing for normal inference tasks.

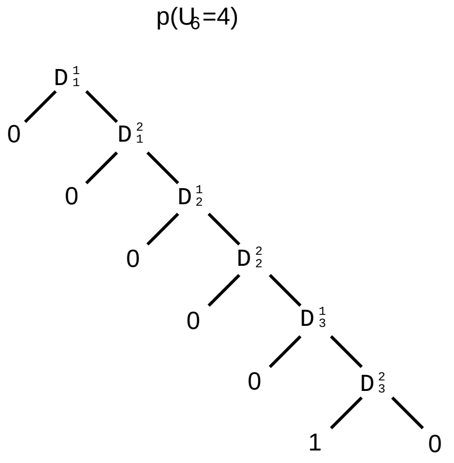

Figure 6: Context trees for the centipede game

Context Specifi c MAIDs

To see how this might be useful for MAIDs consider the centipede example again. We represented our MAID as a Bayesian network with CPD tables of size 2^n. Instead of a table, we will now use trees such as in *figure 6*. We need one tree for each value in the domain of each variable. The numbers on the leaves of the tree are probabilities. Since utility nodes are deterministic, they will always be 1 or 0; this would not be the case for chance or decision nodes. Three trees suffice to express U_6's distribution for a total of 21 probabilities. Zero is always implicitly in the domain of all utility variables for the case when they are not reached so we have two payoffs value plus a zero payoff. This is significantly better than the 2^6 needed originally. In Poole's formulation these tree are actually represented as sets of context-table pairs called *confactors*. In *table 1* we can see what these look like. Note that there is also room in this notation for assigning sets of values to variables which can compress the size even further.

In our survival example, we can take advantage of context trees or confactors for all the chance nodes and all the utility node after the first round. This will be compact since we will never duplicate any context-probability pair, so it will mimic the structure of the extensive form tree for the game in many little pieces. The space complexity will be *O(n!)* because each round *i* will have confactors for utility nodes over *i* parent variables and there are *n-1* rounds. This is still a lot of space, but it is back to the approximate size of the game tree representation plus overhead for representing confactors. As mentioned before, the advantage of this would be that given the same size of *n*, it should be more feasible to compute the equilibrium using the subgame backward induction. This is much more efficient than trying to solve the entire tree using standard techniques which often involves converting the game to normal form since this is truly unfeasible for this problem.

$$p(U_6|D_1\dots D_6)\left\{\left\langle D_1=d, \begin{array}{|c|c|}\hline \mathrm{U}_6 & \mathrm{Val}\\\hline 4&0\\\hline 3&0\\\hline 0&1\\\hline\end{array}\right\rangle,\dots\right.$$

$$\dots,\left\langle D_5=d, \begin{array}{|c|c|}\hline \mathrm{U}_6 & \mathrm{Val}\\\hline 4&0\\\hline 3&0\\\hline 0&1\\\hline\end{array}\right\rangle, \left\langle D_1\vee\dots\vee D_6=a, \begin{array}{|c|c|}\hline \mathrm{U} & \mathrm{Val}\\\hline 4&1\\\hline 3&0\\\hline 0&0\\\hline\end{array}\right\rangle$$

Table 1: Confactors for U_6 node

Conclusion

Multi Agent Influence Diagrams can be very useful for representing many game theoretic situations. This is especially obvious when the games to be represented have limited information hiding between agents and have symmetric styles of play. MAIDs provide a more expressive and more compact representation that can easily take advantage of Bayesian inference techniques to extract information. They also lead to a natural ordering that can be automatically extracted to optimize decisions leading to an efficient algorithm for computing the Nash equilibria of the game. In situations where there are cycles in the strategic reliance amongst decisions it is often possible to do even better by breaking the game into smaller subgames which can be solved faster apart than together. For games that have less restrictions, such as not allowing asymmetry, MAIDs can be exponentially worse in the amount of space they use than game trees. We showed several examples of this and proposed Context Specific MAIDs as a solution that takes advantage of recent advances in integrating the notion of context specific independence into Bayesian Networks. We provided arguments for why this would indeed bring the space complexity back down to no more than the game tree representation, and still clearly retains all the previous advantages of MAIDs. Future directions for this research include modifying and implementing MAID algorithms for use in Contextual Belief Networks and collecting data on the speed of equilibria computation using various approaches for asymmetric games. Guestrin (Guestrin, Venkataraman, & Koller 1996) have done some work along these lines using an older form of context specific inference for cooperative games.

References

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-specific independence in bayesian networks. In *Proc. 12th UAI*, 115–123.

Guestrin, C.; Venkataraman, S.; and Koller, D. 1996. Context-specific multiagent coordination and planning with factored mdps. *American Association for Artificial Intelligence*.

Howard, R. A., and Matheson, J. E. 2003. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, 721–762. Strategic Decisions Group.

Koller, D., and Milch, B. 2001. Multi-agent influence diagrams for representing and solving games. In *Seventeenth International Joint Conference on Artificial Intelligence*, 1027–1034.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann.

Poole, D., and Zhang, N. 2003. Exploiting contextual independence in probabilitistic inference. *Journal of Artificial Intelligence Research*.

Zhang, N., and Poole, D. 1999. On the role of context-specific independence in probabilistic reasoning. In *Proc. IJCAI*.