Multi-Agent Influence Diagrams for Representing and Solving Games

Daphne Koller
Computer Science Dept.
Stanford University
Stanford, CA 94305-9010
koller@cs.stanford.edu

Brian Milch
Computer Science Dept.
Stanford University
Stanford, CA 94305-9010
milch@cs.stanford.edu

Abstract

The traditional representations of games using the extensive form or the strategic (normal) form obscure much of the structure that is present in real-world games. In this paper, we propose a new representation language for general multi-player games — *multi-agent influence diagrams (MAIDs)*. This representation extends graphical models for probability distributions to a multi-agent decision-making context. MAIDs explicitly encode structure involving the dependence relationships among variables. As a consequence, we can define a notion of *strategic relevance* of one decision variable to another: D' is *strategically relevant* to D if, to optimize the decision rule at D, the decision maker needs to take into consideration the decision rule at D'. We provide a sound and complete graphical criterion for determining strategic relevance. We then show how strategic relevance can be used to detect structure in games, allowing a large game to be broken up into a set of interacting smaller games, which can be solved in sequence. We show that this decomposition can lead to substantial savings in the computational cost of finding Nash equilibria in these games.

1 Introduction

Game theory [Fudenberg and Tirole, 1991] provides a mathematical framework for determining what behavior is rational for agents interacting with each other in a partially observable environment. However, the traditional representations of games are primarily designed to be amenable to abstract mathematical formulation and analysis. As a consequence, the standard game representations, both the normal (matrix) form and the extensive (game tree) form, obscure certain important structure that is often present in real-world scenarios — the decomposition of the situation into chance and decision *variables*, and the dependence relationships between these variables. In this paper, we provide a representation that captures this type of structure. We also show that capturing this structure explicitly has several advantages, both in our ability to analyze the game in novel ways, and in our ability to compute Nash equilibria efficiently.

Our framework of *multi-agent influence diagrams (MAIDs)* extends the formalisms of *Bayesian networks* (BNs) [Pearl, 1988] and *influence diagrams* [Howard and Matheson, 1984] to represent decision problems involving multiple agents. MAIDs have clearly defined semantics as noncooperative games: a MAID can be reduced to an equivalent game tree, albeit at the cost of obscuring the variable-level interaction structure that the MAID makes explicit. MAIDs allow us to describe complex games using a natural representation, whose size is no larger than that of the extensive form, but which can be exponentially more compact.

Just as Bayesian networks make explicit the dependencies between probabilistic variables, MAIDs make explicit the dependencies between *decision* variables. They allow us to define a qualitative notion of *strategic relevance*: a decision variable D strategically relies on another decision variable D' when, to optimize the decision rule at D, the decision-making agent needs to take into consideration the decision rule at D'. This notion provides new insight about the relationships between the agents' decisions in a strategic interaction. We provide a graph-based criterion, which we call *s-reachability*, for determining strategic relevance based purely on the graph structure, and show that it is sound and complete in the same sense that d-separation is sound and complete for probabilistic dependence. We also provide a polynomial time algorithm for computing s-reachability.

The notion of strategic relevance allows us to define a data structure that we call the *relevance graph* — a directed graph that indicates when one decision variable in the MAID relies on another. We show that this data structure can be used to provide a natural decomposition of a complex game into interacting fragments, and provide an algorithm that finds equilibria for these smaller games in a way that is guaranteed to produce a global equilibrium for the entire game. We show that our algorithm can be exponentially more efficient than an application of standard game-theoretic solution algorithms, including the more efficient solution algorithms of [Romanovskii, 1962; Koller *et al.*, 1994] that work directly on the game tree.

2 Multi-Agent Influence Diagrams (MAIDs)

We will introduce MAIDs using a simple two-agent scenario:

Example 1 *Alice is considering building a patio behind her house, and the patio would be more valuable to her if she could get a clear view of the ocean. Unfortunately, there is a tree in her neighbor Bob's yard that blocks her view. Being somewhat unscrupulous, Alice considers poisoning Bob's tree, which might cause it to become sick. Bob cannot tell* *whether Alice has poisoned his tree, but he can tell if the tree is getting sick, and he has the option of calling in a tree doctor (at some cost). The attention of a tree doctor reduces the chance that the tree will die during the coming winter. Meanwhile, Alice must make a decision about building her patio before the weather gets too cold. When she makes this decision, she knows whether a tree doctor has come, but she cannot observe the health of the tree directly. A MAID for this scenario is shown in Fig. 1.*

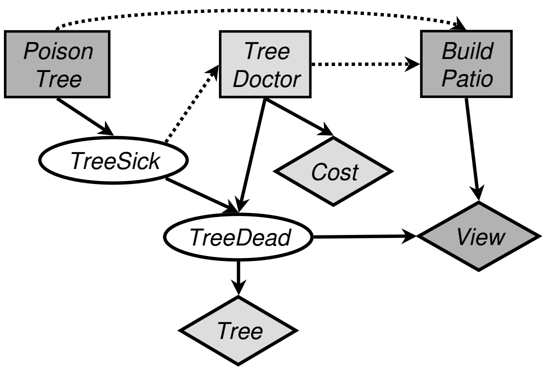


Figure 1: A MAID for the Tree Killer example; Alice's decision and utility variables are in dark gray and Bob's in light gray.

To define a MAID, we begin with a set $\mathcal{A}$ of agents. The world in which the agents act is represented by the set $\mathcal{X}$ of *chance variables*, and a set $\mathcal{D}_a$ of *decision variables* for each agent $a \in \mathcal{A}$. Chance variables correspond to decisions of nature, as in *Bayesian networks* [Pearl, 1988]. They are represented in the diagram as ovals. The decision variables for agent a are variables whose values a gets to choose, and are represented as rectangles in the diagram. We use $\mathcal{D}$ to denote $\bigcup_{a \in \mathcal{A}} \mathcal{D}_a$. The agents' utility functions are specified using *utility variables*: For each agent $a \in \mathcal{A}$, we have a set $\mathcal{U}_a$ of utility variables, represented as diamonds in the diagram. Each variable X has a finite set $dom(X)$ of possible values, called its *domain*. The domain of a utility variable is always a finite set of real numbers (a chance or decision variable can have any finite domain). We use $\mathcal{U}$ to denote $\bigcup_{a \in \mathcal{A}} \mathcal{U}_a$. and $\mathcal{V}$ to denote $\mathcal{X} \cup \mathcal{D} \cup \mathcal{U}$.

Like a BN, a MAID defines a directed acyclic graph with its variables as the nodes, where each variable X is associated with a set of parents $Pa(X) \subset \mathcal{X} \cup \mathcal{D}$. Note that utility variables cannot be parents of other variables. For each chance variable $X \in \mathcal{X}$, the MAID specifies a *conditional probability distribution (CPD)*: a distribution $\Pr(X \mid \mathbf{pa})$ for each instantiation $\mathbf{pa}$ of $Pa(X)$. For a decision variable $D \in \mathcal{D}_a$, $Pa(D)$ is the set of variables whose values agent a knows when he chooses a value for D. Thus, the choice agent a makes for D can be contingent only on these variables. (See Definition 1 below.) For a utility variable U, the MAID also specifies a CPD $\Pr(U \mid \mathbf{pa})$ for each instantiation $\mathbf{pa}$ of $Pa(X)$. However, we require that the value of a utility variable be a deterministic function of the values of its parents: for each $\mathbf{pa} \in dom(Pa(U))$, there is one value of U that has probability 1, and all other values of U have probability 0. We use $U(\mathbf{pa})$ to denote the value of node U that has probability 1 when $Pa(U) = \mathbf{pa}$. The total utility that an agent a derives from an instantiation of $\mathcal{V}$ is the sum of the values of $\mathcal{U}_a$ in this instantiation; thus, we are defining an additive decomposition of the agent's utility function.

The agents get to select their behavior at each of their decision nodes. An agent's decision at a variable D can depend on the variables that the agent observes prior to making D — D's parents. The agent's choice of strategy is specified via a set of *decision rules*.

Definition 1 *A* decision rule *for a decision variable D is a function that maps each instantiation* **pa** *of $Pa(D)$ to a probability distribution over $dom(D)$. An assignment of decision rules to every decision $D \in \mathcal{D}_a$ for a particular agent $a \in \mathcal{A}$ is called a* strategy.

An assignment σ of decision rules to every decision $D \in \mathcal{D}$ is called a *strategy profile*. A *partial strategy profile* $\sigma_{\mathcal{E}}$ is an assignment of decision rules to a subset $\mathcal{E}$ of $\mathcal{D}$. We will also use $\sigma_{\mathcal{E}}$ to denote the restriction of σ to $\mathcal{E}$, and $\sigma_{-\mathcal{E}}$ to denote the restriction of σ to variables not in $\mathcal{E}$.

Note that a decision rule has exactly the same form as a CPD. Thus, if we have a MAID $\mathcal{M}$, then a partial strategy profile $\sigma_{\mathcal{E}}$ that assigns decision rules to a set $\mathcal{E}$ of decision variables induces a new MAID $\mathcal{M}[\sigma_{\mathcal{E}}]$ where the elements of $\mathcal{E}$ have become chance variables. That is, each $D \in \mathcal{E}$ corresponds to a chance variable in $\mathcal{M}[\sigma_{\mathcal{E}}]$ with $\sigma_{\mathcal{E}}(D)$ as its CPD. When σ assigns a decision rule to every decision variable in $\mathcal{M}$, the induced MAID is simply a BN: it has no more decision variables. This BN defines a joint probability distribution $P_{\mathcal{M}[\sigma]}$ over all the variables in $\mathcal{M}$.

Definition 2 *If $\mathcal{M}$ is a MAID and σ is a strategy profile for $\mathcal{M}$, then the* joint distribution for $\mathcal{M}$ induced by σ, *denoted $P_{\mathcal{M}[\sigma]}$, is the joint distribution over $\mathcal{V}$ defined by the Bayes net where:*

- *the set of variables is $\mathcal{V}$;*
- *for $X, Y \in \mathcal{V}$, there is an edge $X \to Y$ iff $X \in Pa(Y)$;*
- *for all $X \in \mathcal{X} \cup \mathcal{U}$, the CPD for X is $\Pr(X)$;*
- *for all $D \in \mathcal{D}$, the CPD for D is $\sigma(D)$.*

We can now write an equation for the utility that agent a expects to receive in a MAID $\mathcal{M}$ if the agents play a given strategy profile σ. Suppose $\mathcal{U}_a = \{U_1, \ldots, U_m\}$. Then:

$$\mathrm{EU}_a(\sigma) = \sum_{(u_1,\ldots,u_m) \in dom(\mathcal{U}_a)} P_{\mathcal{M}[\sigma]}(u_1, \ldots, u_m) \sum_{i=1}^{m} u_i \tag{1}$$

where $dom(\mathcal{U}_a)$ is the joint domain of $\mathcal{U}_a$.

Because the expectation of a sum of random variables is the same as the sum of the expectations of the individual random variables, we can also write this equation as:

$$\mathrm{EU}_a(\sigma) = \sum_{U \in \mathcal{U}_a} \sum_{u \in dom(U)} P_{\mathcal{M}[\sigma]}(U = u) \cdot u \tag{2}$$

Having defined the notion of an expected utility, we can now define what it means for an agent to optimize his decision at one or more of his decision rules, relative to a given set of decision rules for the other variables.

Definition 3 *Let $\mathcal{E}$ be a subset of $\mathcal{D}_a$, and let σ be a strategy profile. We say that $\sigma^*_{\mathcal{E}}$ is* optimal *for the strategy profile σ if, in the induced MAID $\mathcal{M}[\sigma_{-\mathcal{E}}]$, where the only remaining decisions are those in $\mathcal{E}$, the strategy $\sigma^*_{\mathcal{E}}$ is optimal, i.e., for all strategies $\sigma'_{\mathcal{E}}$:*

$$EU_a((\sigma_{-\mathcal{E}}, \sigma^*_{\mathcal{E}})) \geq EU_a((\sigma_{-\mathcal{E}}, \sigma'_{\mathcal{E}}))$$

Note that, in this definition, it does not matter what decision rules σ assigns to the variables in $\mathcal{E}$.

In the game-theoretic framework, we typically consider a strategy profile to represent rational behavior if it is a *Nash equilibrium* [Nash, 1950]. Intuitively, a strategy profile is a Nash equilibrium if no agent has an incentive to deviate from the strategy specified for him by the profile, as long as the other agents do not deviate from their specified strategies.

Definition 4 *A strategy profile σ is a* Nash equilibrium *for a MAID $\mathcal{M}$ if for all agents $a \in \mathcal{A}$, $\sigma_{\mathcal{D}_a}$ is optimal for the strategy profile σ.*

3 MAIDs and Games

A MAID provides a compact representation of a scenario that can also be represented as a game in strategic or extensive form. In this section, we discuss how to convert a MAID into an extensive-form game. We also show how, once we have found an equilibrium strategy profile for a MAID, we can convert it into a behavior strategy profile for the extensive form game. The word "node" in this section refers solely to a node in the tree, as distinguished from the nodes in the MAID.

We use a straightforward extension of a construction of [Pearl, 1988] for converting an influence diagram into a decision tree. The basic idea is to construct a tree with splits for decision and chance nodes in the MAID. However, to reduce the exponential blowup, we observe that we do not need to split on every chance variable in the MAID. A chance variable that is never observed by any decision can be eliminated by summing it out in the probability and utility computations. We present the construction below, referring the reader to [Pearl, 1988] for a complete discussion.

The set of variables included in our game tree is $\mathcal{G} = \mathcal{D} \cup \bigcup_{D \in \mathcal{D}} Pa(D)$. We define a total ordering $\prec$ over $\mathcal{G}$ that is consistent with the topological order of the MAID: if there is a directed path from X to Y, then $X \prec Y$. Our tree $\mathcal{T}$ is a symmetric tree, with each path containing splits over all the variables in $\mathcal{G}$ in the order defined by $\prec$. Each node is labeled with a partial instantiation $inst(N)$ of $\mathcal{G}$, in the obvious way. For each agent a, the nodes corresponding to variables $D \in \mathcal{D}_a$ are decision nodes for a; the other nodes are all chance nodes. To define the information sets, consider two decision nodes M and M' that correspond to a variable D. We place M and M' into the same information set if and only if $inst(M)$ and $inst(M')$ assign the same values to $Pa(D)$.

Our next task is to determine the split probabilities at the chance nodes. Consider a chance node N corresponding to a chance variable C. For each value $c \in dom(C)$, let N_c be the child of N corresponding to the choice $C = c$. We want to compute the probability of going from N to N_c. The problem, of course, is that a MAID does not define a full joint probability distribution until decision rules for the agents are selected. It turns out that we can choose an arbitrary fully mixed strategy profile σ for our MAID $\mathcal{M}$ (one where no decision has probability zero), and do inference in the BN $\mathcal{M}[\sigma]$ induced by this strategy profile, by computing

$$P_{\mathcal{M}[\sigma]}(inst(N_c) \mid inst(N)) \tag{3}$$

The value of this expression does not depend on our choice of σ. To see why this is true, note that if we split on a decision variable D before C, then the decision rule σ_D does not affect the computation of $P_{\mathcal{M}[\sigma]}(inst(N_c) \mid inst(N))$, because $inst(N)$ includes values for D and all its parents. If we split on D after C, then D cannot be an ancestor of C in the MAID. Also, by the topological ordering of the nodes in the tree, we know that $inst(N)$ cannot specify evidence on D or any of its descendants. Therefore, σ_D cannot affect the computation. Hence, the probabilities of the chance nodes are well-defined.

We define the payoffs at the leaves by computing a distribution over the utility nodes, given an instantiation of $\mathcal{G}$. For a leaf N, the payoff for agent a is:

$$\sum_{U \in \mathcal{U}_a} \sum_{u \in dom(U)} P_{\mathcal{M}[\sigma]}(U = u \mid inst(N)) \cdot u \tag{4}$$

We can also show that the value of (4) does not depend on our choice of σ. The basic idea here is that $inst(N)$ determines the values of D and $Pa(D)$ for each decision variable D. Hence, the agents' moves and information are all fully determined, and the probabilities with which different actions are chosen in σ are irrelevant. We omit details.

The mapping between MAIDs and trees also induces an obvious mapping between strategy profiles in the different representations. A MAID strategy profile specifies a probability distribution over $dom(D)$ for each pair $(D, \mathbf{pa})$, where $\mathbf{pa}$ is an instantiation of $Pa(D)$. The information sets in the game tree correspond one-to-one with these pairs, and a behavior strategy in the game tree is a mapping from information sets to probability distributions. Clearly the two are equivalent.

Based on this construction, we can now state the following equivalence proposition:

Proposition 1 *Let $\mathcal{M}$ be a MAID and $\mathcal{T}$ be its corresponding game tree. Then for any strategy profile σ, the payoff vector for σ in $\mathcal{M}$ is the same as the payoff vector for σ in $\mathcal{T}$.*

The number of nodes in $\mathcal{T}$ is exponential in the number of decision variables, and in the number of chance variables that are observed during the course of the game. While this blowup is unavoidable in a tree representation, it can be quite significant. In some games, a MAID can be exponentially smaller than the extensive game it corresponds to.

Example 2 *Suppose a road is being built from north to south through undeveloped land, and n agents have purchased plots of land along the road. As the road reaches each agent's plot, the agent needs to choose what to build on his land. His utility depends on what he builds, on some private information about the suitability of his land for various purposes, and on what is built north, south, and across the road from his land. The agent can observe what has already been built immediately to the north of his land (on both sides of the road), but he cannot observe further north; nor can he observe what will be built across from his land or south of it.*

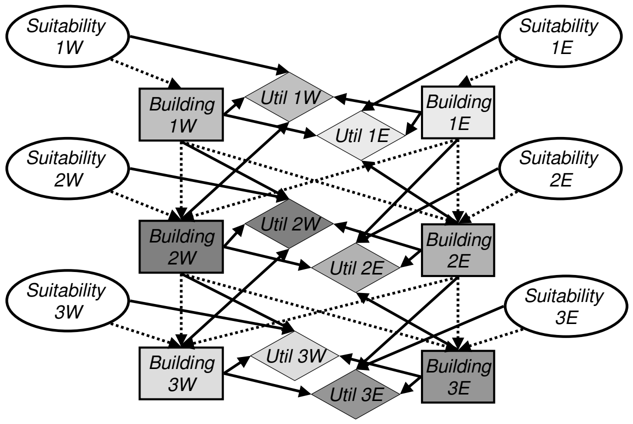


Figure 2: A MAID for the Road example with $n = 6$.

The MAID representation, shown in Fig. 2 for $n = 6$, is very compact. There are n chance nodes, corresponding to the private information about each agent's land, and n decision variables. Each decision variable has at most three parents: the agent's private information, and the two decisions regarding the two plots to the north of the agent's land. Thus, the size of the MAID is linear in n. Conversely, any game tree for this situation must split on each of the n chance nodes and each of the n decisions, leading to a representation that is exponential in n. Concretely, suppose the chance and decision variables each have three possible values, corresponding to three types of buildings. Then the game tree corresponding to the Road *MAID has 3^{2n} leaves.*

A MAID representation is not always more compact. If the game tree is naturally asymmetric, a naive MAID representation can be exponentially larger than the tree. We return to the problem of asymmetric scenarios in Section 6.

4 Strategic Relevance

To take advantage of the independence structure in a MAID, we would like to find a global equilibrium through a series of relatively simple local computations. The difficulty is that, in order to determine the optimal decision rule for a single decision variable, we usually need to know the decision rules for some other variables. In Example 1, when Alice is deciding whether to poison the tree, she needs to compare the expected utilities of her two alternatives. However, the probability of the tree dying depends on the probability of Bob calling a tree doctor if he observes that the tree is sick. Thus, we need to know the decision rule for *CallTreeDoctor* to determine the optimal decision rule for *PoisonTree*. In such situations, we will say that *PoisonTree (strategically) relies on CallTreeDoctor*, or that *CallTreeDoctor* is *relevant to PoisonTree*. On the other hand, *CallTreeDoctor* does not rely on *PoisonTree*. Bob gets to observe whether the tree is sick, and *TreeDead* is conditionally independent of *PoisonTree* given *TreeSick*, so the decision rule for *PoisonTree* is not relevant to Bob's decision.

We will now formalize this intuitive discussion of strategic relevance. Suppose we have a strategy profile, and we would like to find a decision rule for a single decision variable $D \in \mathcal{D}_a$ that maximizes a's expected utility, assuming the rest of the strategy profile remains fixed.

According to Definition 3, to determine whether a decision rule δ for D is optimal for σ, we construct the induced MAID where all decision nodes except D are turned into chance nodes, with their CPDs specified by σ. Then δ is optimal for σ if it maximizes a's expected utility in this single-decision MAID. The key question that motivates our definition of strategic relevance is the following: What other decision rules are relevant for optimizing the decision rule at D?

Definition 5 *Let D be a decision node in a MAID $\mathcal{M}$, δ be a decision rule for D, and σ be a strategy profile such that δ is optimal for σ. D* strategically relies on *a decision node D' in $\mathcal{M}$ if there is another strategy profile σ' such that σ' differs from σ only at D', but δ is not optimal for σ', and neither is any decision rule δ' that agrees with δ on all parent instantiations $\boldsymbol{pa} \in dom(Pa(D))$ where $P_{\mathcal{M}[\sigma]}(\boldsymbol{pa}) > 0$.*

In other words, if a decision rule δ for D is optimal for a strategy profile σ, and D does not rely on D', then δ is also optimal for any strategy profile σ' that differs from σ only at D'. The last clause of this definition is needed to deal with a problem that arises in many other places in game theory — the problem of suboptimal decisions in response to observations that have zero probability (such as observing an irrational move by another agent).

Relevance is a numeric criterion that depends on the specific probabilities and utilities in the MAID. It is not obvious how we would check for strategic relevance without testing all possible pairs of strategy profiles σ and σ'. We would like to find a qualitative criterion which can help us determine strategic relevance purely from the structure of the graph. In other words, we would like to find a criterion which is analogous to the d-separation criterion for determining conditional independence in Bayesian networks.

First, the optimality of the decision rule at D depends only on the utility nodes $\mathcal{U}_D$ that are descendants of D in the MAID. The other utility nodes are irrelevant, because the decision at D cannot influence them. Now, consider another decision variable D'. The decision rule at D' is relevant to D only if it can influence the probability distribution over the utility nodes $\mathcal{U}_D$. To determine whether the CPD for a node can affect the probability distribution over a set of other nodes, we can build on a graphical criterion already defined for Bayesian networks, that of a *requisite probability node*:

Definition 6 *Let G be a BN structure, and let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be sets of variables in the BN. Then a node Z is a* requisite probability node *for the query $P(\boldsymbol{X} \mid \boldsymbol{Y})$ if there exist two Bayesian networks $\mathcal{B}_1$ and $\mathcal{B}_2$ over G, that are identical except in the CPD they assign to Z, but $P_{\mathcal{B}_1}(\boldsymbol{X} \mid \boldsymbol{Y}) \neq P_{\mathcal{B}_2}(\boldsymbol{X} \mid \boldsymbol{Y})$.*

As we will see, the decision rule at D' is only relevant to D if D' (viewed as a chance node) is a requisite probability node for $P(\mathcal{U}_D \mid D, Pa(D))$.

Geiger *et al.* [1990] provide a graphical criterion for testing whether a node Z is a requisite probability node for a query $P(\boldsymbol{X} \mid \boldsymbol{Y})$. We add to Z a new "dummy" parent $\widehat{Z}$ whose values correspond to CPDs for Z, selected from some set of possible CPDs. Then Z is a requisite probability node for $P(\boldsymbol{X} \mid \boldsymbol{Y})$ if and only if $\widehat{Z}$ can influence $\boldsymbol{X}$ given $\boldsymbol{Y}$.

Based on these considerations, we can define *s-reachability*, a graphical criterion for detecting strategic relevance. Note that unlike d-separation in Bayesian networks, s-reachability is not necessarily a symmetric relation.

Definition 7 *A node D' is* s-reachable *from a node D in a MAID $\mathcal{M}$ if there is some utility node $U \in \mathcal{U}_D$ such that if a new parent $\widehat{D'}$ were added to D', there would be an active path in $\mathcal{M}$ from $\widehat{D'}$ to U given $Pa(D) \cup \{D\}$, where a path is active in a MAID if it is active in the same graph, viewed as a BN.*

We can show that s-reachability is sound and complete for strategic relevance (almost) in the same sense that d-separation is sound and complete for independence in Bayesian networks. As for d-separation, the soundness result is very strong: without s-reachability, one decision cannot be relevant to another.

Theorem 1 (Soundness) *If D and D' are two decision nodes in a MAID $\mathcal{M}$ and D' is not s-reachable from D in $\mathcal{M}$, then D does not rely on D'.*

As for BNs, the result is not as strong in the other direction: s-reachability does not imply relevance in *every* MAID. We can choose the probabilities and utilities in the MAID in such a way that the influence of one decision rule on another does not manifest itself. However, s-reachability is the most precise graphical criterion we can use: it will not identify a strategic relevance unless that relevance actually exists in some MAID that has the given graph structure. We say that two MAIDs have the same graph structure when the two MAIDs have the same sets of variables and agents, each variable has the same parents in the two MAIDs, and the assignment of decision and utility variables to agents is the same in both MAIDs. The chance and decision variables must have the same domains in both MAIDs, but we allow the actual utility values of the utility variables (their domains) to vary. The CPDs in the two MAIDS may also be different.

Theorem 2 (Completeness) *If a node D' is s-reachable from a node D in a MAID, then there is some MAID with the same graph structure in which D relies on D'.*

Since s-reachability is a binary relation, we can represent it as a directed graph. As we show below, this graph turns out to be extremely useful.

Definition 8 *The* relevance graph *for a MAID $\mathcal{M}$ is a directed graph whose nodes are the decision nodes of $\mathcal{M}$, and which contains an edge $D \rightarrow D'$ if and only if D' is s-reachable from D.*

The relevance graph for the Tree Killer example is shown in Fig. 4(a). By Theorem 1, if D relies on D', then there is an edge from D to D' in the relevance graph.

To construct the graph for a given MAID, we need to determine, for each decision node D, the set of nodes D' that are s-reachable from D. Using an algorithm such as Shachter's Bayes-Ball [Shachter, 1998], we can find this set for any given D in time linear in the number of nodes in the MAID.

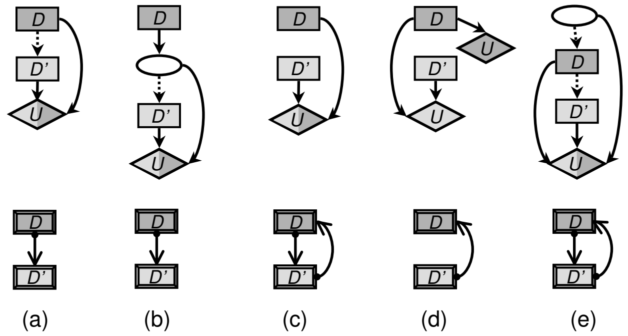


Figure 3: Five simple MAIDs (top), and their relevance graphs (bottom). A two-color diamond represents a pair of utility nodes, one for each agent, with the same parents.

By repeating the algorithm for each D, we can derive the relevance graph in time quadratic in the number of MAID nodes.

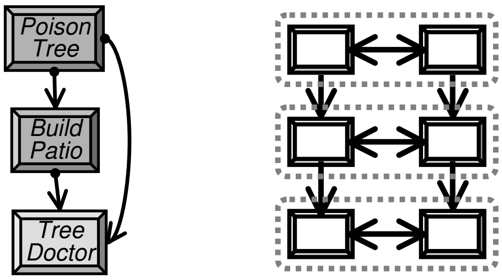
Recall our original statement that a decision node D strategically relies on a decision node D' if one needs to know the decision rule for D' in order to evaluate possible decision rules for D. Although we now have a graph-theoretic characterization of strategic relevance, it will be helpful to develop some intuition by examining some simple MAIDs, and seeing when one decision node relies on another. In the five examples shown in Fig. 3, the decision node D belongs to agent a, and D' belongs to agent b. Example (a) represents a perfect-information game. Since agent b can observe the value of D, he does not need to know the decision rule for D in order to evaluate his options. Thus, D' does not rely on D. On the other hand, agent a cannot observe D' when she makes decision D, and D' is relevant to a's utility, so D relies on D'. Example (b) represents a game where the agents do not have perfect information: agent b cannot observe D when making decision D'. However, the information is "perfect enough": the utility for b does not depend on D directly, but only on the chance node, which b can observe. Hence D' does not rely on D. Examples (c) and (d) represent scenarios where the agents move simultaneously, and thus neither can observe the other's move. In (c), each agent's utility node is influenced by both decisions, so D relies on D' and D' relies on D. Thus, the relevance graph is cyclic. In (d), however, the relevance graph is acyclic despite the fact that the agents move simultaneously. The difference here is that agent a no longer cares what agent b does, because her utility is not influenced by b's decision. In graphical terms, there is no active path from D' to a's utility node given D.

One might conclude that a decision node D' never relies on a decision node D when D is observed by D', but the situation is more subtle. Consider example (e), which represents a simple card game: agent a observes a card, and decides whether to bet (D); agent b observes only agent a's bet, and decides whether to bet (D'); the utility of both depends on their bets and the value of the card. Even though agent b observes the actual decision in D, he needs to know the decision rule for D in order to know what the value of D tells him about the chance node. Thus, D' relies on D; indeed, when D is observed, there is an active path from D that runs through the chance node to the utility node.

5 Computing Equilibria

The computation of a Nash equilibrium for a game is arguably the key computational task in game theory. In this section, we show how the structure of the MAID can be exploited to provide efficient algorithms for finding equilibria in certain games. The key insight behind our algorithm is the use of the relevance graph to break up the task of finding an equilibrium into a series of subtasks, each over a much smaller game. Since algorithms for finding equilibria in general games have complexity that is superlinear in the number of levels in the game tree, breaking the game into smaller games significantly improves the complexity of finding a global equilibrium.

Our algorithm is a generalization of existing backward induction algorithms for decision trees and perfect information games [Zermelo, 1913] and for influence diagrams [Jensen *et al.*, 1994]. The basic idea is as follows: in order to optimize the decision rule for D, we need to know the decision rule for all decisions D' that are relevant for D. For example, the relevance graph for the Tree Killer example (Fig. 4(a)) shows that to optimize *PoisonTree*, we must first decide on the decision rules for *BuildPatio* and *TreeDoctor*. However, we can optimize *TreeDoctor* without knowing the decision rules for either of the other decision variables. Having decided on the decision rule for *TreeDoctor*, we can now optimize *BuildPatio* and then finally *PoisonTree*.



(a) (b)

Figure 4: Relevance graphs for (a) the Tree Killer example; (b) the Road example with $n = 6$.

We can apply this simple backward induction procedure in any MAID which, like the Tree Killer example, has an acyclic relevance graph. When the relevance graph is acyclic, we can construct a topological ordering of the decision nodes: an ordering $D_1, \ldots, D_n$ such that if $i < j$, then D_i is not s-reachable from D_j. We can then iterate backward from D_n to D_1, deriving an optimal decision rule for each decision node in turn. Each decision D_i relies only on the decisions that succeed it in the order, and these will have been computed by the time we have to select the decision rule for D_i.

The relevance graph is acyclic in all perfect-information games, and in all single-agent decision problems with perfect recall. There are also some games of imperfect information, such as the Tree Killer example, that have acyclic relevance graphs. But in most games we will encounter cycles in the relevance graph. Consider, for example, any simple two-player simultaneous move game with two decisions D_1 and D_2, where both players' payoffs depend on the decisions at both D_1 and D_2, as in Fig. 3(c). In this case, the optimality of one player's decision rule is clearly intertwined with the other player's choice of decision rule, and the two decision rules must "match" in order to be in equilibrium. Indeed, as we discussed, the relevance graph in such a situation is cyclic.

However, we can often utilize relevance structure even in games where the relevance graph is cyclic.

Example 3 *Consider the relevance graph for the* Road *example, shown in Fig. 4(b) for $n = 6$ agents. We can see that we have pairs of interdependent decision variables, corresponding to the two agents whose lots are across the road from each other. Also, the decision for a given plot relies on the decision for the plot directly to the south. However, it does not rely on the decision about the land directly north of it, because this decision is observed. None of the other decisions affect this agent's utility directly, and therefore they are not s-reachable.*

Intuitively, although the last pair of nodes in the relevance graph rely on each other, they rely on nothing else. Hence, we can compute an equilibrium for the pair together, regardless of any other decision rules. Once we have computed an equilibrium for this last pair, the decision variables can be treated as chance nodes, and we can proceed to compute an equilibrium for the next pair.

We formalize this intuition in the following definition:

Definition 9 *A set $\mathcal{S}$ of nodes in a directed graph is a* strongly connected component *(SCC) if for every pair of nodes $D \neq D' \in \mathcal{S}$, there exists a directed path from D to D'. A* maximal *SCC is an SCC that is not a strict subset of any other SCC.*

The maximal SCCs for the Road example are outlined in Fig. 4(b).

We can find the maximal SCCs of a relevance graph in linear time, by constructing a *component graph* whose nodes are the maximal SCCs of the graph [Cormen *et al.*, 1990]. There is an edge from component $\mathcal{C}$ to component $\mathcal{C}'$ in the component graph if and only if there is an edge in the relevance graph from some element of $\mathcal{C}$ to some element of $\mathcal{C}'$. The component graph is always acyclic, so we can define an ordering $\mathcal{C}_1, \ldots, \mathcal{C}_m$ over the SCCs, such that whenever $i < j$, no element of $\mathcal{C}_i$ is s-reachable from any element of $\mathcal{C}_j$.

We can now provide a divide-and-conquer algorithm for computing Nash equilibria in general MAIDs.

Algorithm 1

Given a MAID $\mathcal{M}$
a topological ordering $\mathcal{C}_1, \ldots, \mathcal{C}_m$ of the component graph derived from the relevance graph for $\mathcal{M}$

1 *Let σ^0 be an arbitrary fully mixed strategy profile*
2 *For $i = 0$ through $m - 1$:*
3 *Let τ be a partial strategy profile for $\mathcal{C}_{(m-i)}$ that is a Nash equilibrium in $\mathcal{M}\left[\sigma^i_{-\mathcal{C}_{(m-i)}}\right]$*
4 *Let $\sigma^{i+1} = (\sigma^i_{-\mathcal{C}_{(m-i)}}, \tau)$*
5 *Output σ^m as an equilibrium of $\mathcal{M}$*

The algorithm iterates backwards over the SCC's, finding an equilibrium strategy profile for each SCC in the MAID induced by the previously selected decision rules (with arbitrary decision rules for some decisions that are not relevant for this SCC). In this induced MAID, the only remaining decision nodes are those in the current SCC; all the other decision nodes have been converted to chance nodes. Finding the equilibrium in this induced MAID requires the use of a subroutine for finding equilibria in games. We simply convert the induced MAID into a game tree, as described in Section 3, and use a standard game-solving algorithm [McKelvey and McLennan, 1996] as a subroutine. Note that if the relevance graph is acyclic, each SCC consists of a single decision node. Thus, step 3 involves finding a Nash equilbrium in a single-player game, which reduces to simply finding a decision rule that maximizes the single agent's expected utility.

In proving the correctness of Algorithm 1, we encounter a subtle technical difficulty. The definition of strategic relevance (Def. 5) only deals with the optimality of a single decision rule for a strategy profile. But in Algorithm 1, we derive not just single decision rules, but a complete strategy for each agent. To make the leap from the optimality of single decision rules to the optimality of whole strategies in our proof, we must make the standard assumption of *perfect recall* — that agents never forget their previous actions or observations. More formally:

Definition 10 *An agent a has* perfect recall *with respect to a total order $D_1, \ldots, D_n$ over $\mathcal{D}_a$ if for all $D_i, D_j \in \mathcal{D}_a$, $i < j$ implies that $D_i \in Pa(D_j)$ and $Pa(D_i) \subset Pa(D_j)$.*

We can now prove the correctness of Algorithm 1.

Theorem 3 *Let $\mathcal{M}$ be a MAID where every agent has perfect recall, and let $\mathcal{C}_1, \ldots, \mathcal{C}_m$ be a topological ordering of the SCCs in the relevance graph for $\mathcal{M}$. Then the strategy profile σ^m produced by running Algorithm 1 with $\mathcal{M}$ and $\mathcal{C}_1, \ldots, \mathcal{C}_m$ as inputs is a Nash equilibrium for $\mathcal{M}$.*

To demonstrate the potential savings resulting from our algorithm, we tried it on the Road example, for different numbers of agents n. Note that the model we used differs slightly from that shown in Fig. 2: In our experiments, each agent had not just one utility node, but a separate utility node for each neighboring plot of land, and an additional node that depends on the suitability of the plot for different purposes. The agent's decision node is a parent of all these utility nodes. The idea is that an agent gets some base payoff for the building he builds, and then the neighboring plots and the suitability node apply additive bonuses and penalties to his payoff. Thus, instead of having one utility node with $3^5 = 243$ parent instantiations, we have 4 utility nodes with $3^2 = 9$ parent instantiations each. This change has no effect on the structure of the relevance graph, which is shown for $n = 6$ in Fig. 4(b). The SCCs in the relevance graph all have size 2; as we discussed, they correspond to pairs of decisions about plots that are across the road from each other.

Even for small values of n, it is infeasible to solve the Road example with standard game-solving algorithms. As we discussed, the game tree for the MAID has 3^{2n} leaves, whereas the MAID representation is linear in n. The normal form adds another exponential factor. Since each agent (except the first two) can observe three ternary variables, he has 27 information sets. Hence, the number of possible pure (deterministic) strategies for each agent is 3^{27}, and the number of pure strategy profiles for all n players is $(3^{27})^{(n-2)} \cdot (3^3)^2$. In the simplest interesting case, where $n = 4$, we obtain a game tree with 6561 terminal nodes, and standard solution algorithms, that very often use the normal form, would need to operate on a game matrix with about 4.7×10^{27} entries (one for each pure strategy profile).

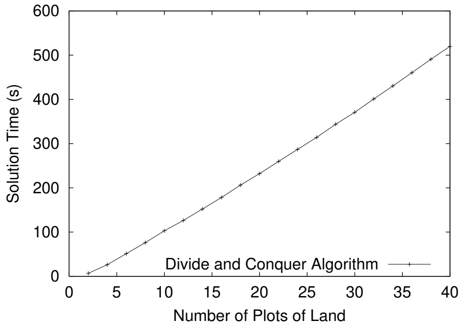


Figure 5: Performance results for the Road example.

Solving the Road game either in its extensive form or in the normal form is infeasible even for $n = 4$. By contrast, our divide-and-conquer algorithm ends up generating a sequence of small games, each with two decision variables. Fig. 5 shows the computational cost of the algorithm as n grows. We converted each of the induced MAIDs constructed during the algorithm into a small game tree, and used the game solver GAMBIT [2000] to solve it. As expected, the time required by our algorithm grows approximately linearly with n. Thus, for example, we can solve a Road MAID with 40 agents (corresponding to a game tree with 3^{80} terminal nodes) in 8 minutes 40 seconds.

6 Discussion and Future Work

We have introduced a new formalism, multi-agent influence diagrams (MAIDs), for modeling multi-agent scenarios with imperfect information. MAIDs use a representation where variables are the basic unit, and allow the dependencies between these variables to be represented explicitly, in a graphical form. They therefore reveal important qualitative structure in a game, which can be useful both for understanding the game and as the basis for algorithms that find equilibria efficiently. In particular, we have shown that our divide-and-conquer algorithm for finding equilibria provides exponential savings over existing solution algorithms in some cases, such as the Road example, where the maximal size of an SCC in the relevance graph is much smaller than the total number of decision variables. In the worst case, the relevance graph forms a single large SCC, and our algorithm simply solves the game in its entirety, with no computational benefits.

Although the possibility of extending influence diagrams to multi-agent scenarios was recognized at least fifteen years ago [Shachter, 1986], the idea seems to have been dormant for some time. Suryadi and Gmytrasiewicz [1999] have used influence diagrams as a framework for learning in multi-agent systems. Milch and Koller [2000] use multi-agent influence diagrams as a representational framework for reasoning about agents' beliefs and decisions. However, the focus of both these papers is very different, and they do not consider the structural properties of the influence diagram representation, nor the computational benefits derived from it. Nilsson and Lauritzen [2000] have done related work on limited memory influence diagrams, but they focus on the task of speeding up inference in single-agent settings. MAIDs are also related to La Mura's [2000] game networks, which incorporate both probabilistic and utility independence. La Mura defines a notion of strategic independence, and also uses it to break up the game into separate components. However, his notion of strategic independence is an undirected one, and thus does not allow as fine-grained a decomposition as the directed relevance graph used in this paper, nor the use of a backward induction process for interacting decisions.

This work opens the door to a variety of possible extensions. On the representational front, it is important to extend MAIDs to deal with asymmetric situations, where the decisions to be made and the information available depend on previous decisions or chance moves. Game trees represent such asymmetry in a natural way, whereas in MAIDs (as in influence diagrams and BNs), a naive representation of an asymmetric situation leads to unnecessary blowup. We believe we can avoid these difficulties in MAIDs by explicitly representing context-specificity, as in [Boutilier *et al.*, 1996; Smith *et al.*, 1993], integrating the best of the game tree and MAID representations.

Another direction relates to additional structure that is revealed by the notion of strategic relevance. In particular, even if a group of nodes forms an SCC in the relevance graph, it might not be a fully connected subgraph; for example, we might have a situation where D_1 relies on D_2, which relies on D_3, which relies on D_1. Clearly, this type of structure tells us something about the interaction between the decisions in the game. An important open question is to analyze the meaning of these types of structures, and to see whether they can be exploited for computational gain. (See [Kearns *et al.*, 2001] for results in one class of MAIDs.)

Finally, the notion of strategic relevance is not the only type of insight that we can obtain from the MAID representation. We can use a similar type of path-based analysis in the MAID graph to determine which of the variables that an agent can observe before making a decision actually provide relevant information for that decision. In complex scenarios, especially those that are extended over time, agents tend to accumulate a great many observations. The amount of space needed to specify a decision rule for the current decision increases exponentially with the number of observed variables. Thus, there has been considerable work on identifying irrelevant parents of decision nodes in single-agent influence diagrams [Howard and Matheson, 1984; Shachter, 1990; 1998]. However, the multi-agent case raises subtleties that are absent in the single-agent case. This is another problem we plan to address in future work.

Acknowledgements This work was supported by Air Force contract F30602-00-2-0598 under DARPA's TASK program and by ONR MURI N00014-00-1-0637 under the program "Decision Making under Uncertainty".

References

[Boutilier *et al.*, 1996] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proc. 12th UAI*, pages 115–123, 1996.

[Cormen *et al.*, 1990] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[Fudenberg and Tirole, 1991] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[Gambit, 2000] GAMBIT software, California Institute of Technology, 2000. http://www.hss.caltech.edu/gambit/Gambit.html.

[Geiger *et al.*, 1990] D. Geiger, T. Verma, and J. Pearl. Identifying independence in Bayesian networks. *Networks*, 20:507–534, 1990.

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

[Jensen *et al.*, 1994] F. Jensen, F.V. Jensen, and S.L. Dittmer. From influence diagrams to junction trees. In *Proc. 10th UAI*, pages 367–373, 1994.

[Kearns *et al.*, 2001] M. Kearns, M.L. Littman, and S. Singh. Graphical models for game theory. Submitted, 2001.

[Koller *et al.*, 1994] D. Koller, N. Megiddo, and B. von Stengel. Fast algorithms for finding randomized strategies in game trees. In *Proc. 26th STOC*, pages 750–759, 1994.

[LaMura, 2000] P. LaMura. Game networks. In *Proc. 16th UAI*, pages 335–342, 2000.

[McKelvey and McLennan, 1996] R.D. McKelvey and A. McLennan. Computation of equilibria in finite games. In *Handbook of Computational Economics*, volume 1, pages 87–142. Elsevier Science, Amsterdam, 1996.

[Milch and Koller, 2000] B. Milch and D. Koller. Probabilistic models for agents' beliefs and decisions. In *Proc. 16th UAI*, 2000.

[Nash, 1950] J. Nash. Equilibrium points in n-person games. *Proc. National Academy of Sciences of the USA*, 36:48–49, 1950.

[Nilsson and Lauritzen, 2000] D. Nilsson and S.L. Lauritzen. Evaluating influence diagrams with LIMIDs. In *Proc. 16th UAI*, pages 436–445, 2000.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, 1988.

[Romanovskii, 1962] I. V. Romanovskii. Reduction of a game with complete memory to a matrix game. *Soviet Mathematics*, 3:678–681, 1962.

[Shachter, 1986] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.

[Shachter, 1990] R. D. Shachter. An ordered examination of influence diagrams. *Networks*, 20:535–563, 1990.

[Shachter, 1998] R. D. Shachter. Bayes-ball: The rational pastime. In *Proc. 14th UAI*, pages 480–487, 1998.

[Smith *et al.*, 1993] J. E. Smith, S. Holtzman, and J. E. Matheson. Structuring conditional relationships in influence diagrams. *Operations Research*, 41(2):280–297, 1993.

[Suryadi and Gmytrasiewicz, 1999] D. Suryadi and P.J. Gmytrasiewicz. Learning models of other agents using influence diagrams. In *Proc. 7th Int'l Conf. on User Modeling (UM-99)*, pages 223–232, 1999.

[Zermelo, 1913] E. Zermelo. Über eine Anwendung der Mengenlehre auf der Theorie des Schachspiels. In *Proceedings of the Fifth International Congress on Mathematics*, 1913.