ACADEMIC PRESS

Available online at www.sciencedirect.com
SCIENCE DIRECT®

Games and Economic Behavior 45 (2003) 181–221

GAMES and Economic Behavior

www.elsevier.com/locate/geb

Multi-agent influence diagrams for representing and solving games ☆

Daphne Koller [a,*] and Brian Milch [b]

[a] *Stanford University, Computer Science Department, Gates Bldg. 1A, Stanford, CA 94305-9010, USA*
[b] *University of California Berkeley, USA*

Received 26 April 2001

Abstract

The traditional representations of games using the extensive form or the strategic form obscure much of the structure of real-world games. In this paper, we propose a graphical representation for noncooperative games—*multi-agent influence diagrams* (MAIDs). The basic elements in the MAID representation are *variables*, allowing an explicit representation of dependence, or relevance, relationships among variables. We define a decision variable D' as *strategically relevant* to D if, to optimize the decision rule at D, the decision maker needs to consider 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 decompose large games 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.

JEL classification: C72; D82

1. Introduction

Game theory provides a mathematical framework for determining what behavior is rational for agents interacting with each other in a partially observable environment. However, 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-

☆ This paper is a significantly expanded version of a paper first published in the Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01), Koller, Milch, 2001, pp. 1027–1034.

* Corresponding author.
E-mail addresses: koller@cs.stanford.edu (D. Koller), milch@cs.berkeley.edu (B. Milch).

0899-8256/$ – see front matter
doi:10.1016/S0899-8256(02)00544-4

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 representation is based on the framework of *probabilistic graphical models* (Pearl, 1988), used in many disciplines, including statistics and artificial intelligence. Probabilistic graphical models represent the world via a set of *variables*, that take on values in some (discrete or continuous) space. For example, in a simple economic model, we might have a continuous variable for each of several possible goods, indicating its supply at a given time. We might also have a discrete variable representing the amount of rainfall in a region over the last year (e.g., taking the values "drought," "low," "normal," or "high"). Each possible state (or trajectory) of the world is then an assignment of values to these variables. By representing the world in terms of these variables, we can make statements about the relationships between them. For example, we might know that the supply of oranges depends on the rainfall variable, whereas the supply of oil does not. The graphical model represents this structure using a directed graph structure, where the nodes represent the variables, and the edges represent the direct dependence of one variable on another. As we discuss in Section 2, these graphs, called *Bayesian networks* or *probabilistic networks*, have clear and formal semantics as a representation of a probability distribution over the state space defined by the variables. Furthermore, the graph structure itself makes explicit certain important aspects of the probability distribution, such as the conditional independence properties of the variables in the distribution.

Influence diagrams (Howard and Matheson, 1984) extend Bayesian networks to the decision-theoretic setting, where an agent has to make decisions in accordance with his preferences. In addition to chance variables, influence diagrams contain *decision variables*, which are variables whose value the agent selects as part of his strategy, and *utility variables*, which represent the agent's preferences.

In this paper, we define *multi-agent influence diagrams* (MAIDs), which represent decision problems involving multiple agents. We show that MAIDs have clearly defined semantics as noncooperative games, and can be reduced to an equivalent sequence form or normal form game, albeit at the cost of obscuring the variable-level interaction structure that the MAID makes explicit. MAIDs allow us to represent complex games in a natural way, whose size is no larger than that of the extensive form, but which can be exponentially more compact.

We show that MAIDs allow us to define a qualitative notion of dependence between *decision* variables. We define a 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. We also provide a polynomial time algorithm, which considers only the graph structure, 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 subgames in a way that is guaranteed to produce a global equilibrium for the entire game. As a special case, the algorithm generalizes the standard backward induction algorithm for game trees, showing a general subclass of games where backward induction can be applied, even in some games that are not perfect information. 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) and Koller et al. (1994) that work directly on the game tree.

The remainder of this paper is structured as follows. In Section 2, we review some of the key concepts in the framework of probabilistic graphical models that underlie our work. In Section 3, we present the framework of multi-agent influence diagrams, and in Section 4, we relate it to standard game-theoretic concepts. In Section 5, we define the notion of strategic relevance, and provide a criterion for determining strategic relevance from the graph structure. In Section 6 we show how to exploit strategic relevance to break up a game into smaller games, and compute equilibria more effectively. We discuss related work in Section 7. In Section 8, we conclude with a discussion of some extensions, including additional structure that can be induced from the MAID representation.

2. Bayesian networks

Our work builds on the framework of *Bayesian networks* (also known as probabilistic networks or belief networks) (Pearl, 1988) and on its decision-theoretic extension, *influence diagrams* (Howard and Matheson, 1984). In this section, we briefly review the Bayesian network framework, setting up much of the necessary foundation for the remainder of this paper.

2.1. Representation

A Bayesian network is a graphical representation of a distribution over the joint probability space defined by a set of variables. More precisely, consider a set of variables $X_1, \ldots, X_n$, where each X_i takes on values in some finite set $dom(X_i)$. Together, the variables define a cross-product space $\times_{i=1}^{n} dom(X_i)$. Our goal is to represent a joint distribution over this joint space. We use $\mathcal{X}$ to refer to the set of variables $X_1, \ldots, X_n$, and $dom(\mathcal{X})$ to refer to their joint domain.

Example 2.1. Consider a scenario where we are trying to reason about an alarm installed in a house. The alarm can go off either because of a burglary or because of a minor earthquake. If there is an alarm, then a neighbor might call. If there is an earthquake, the local radio station may report it. There are five variables in this domain, all of which are binary valued:

A (alarm); B (burglary); E (earthquake); C (phone call); R (radio report). Thus, the joint distribution has 32 entries.

As we can see, the number of states in the joint distribution grows exponentially with the number of variables: If we have n variables that take on binary values, the total number of states is 2^n. Even for fairly small n, the representation of this distribution as a list of numbers, one for each state, is impractical.

A Bayesian network (BN) represents the distribution using a graph, whose nodes represent the random variables and whose edges represent (in a very formal sense) direct influence of one variable on another. More precisely, we have the following definition for the Bayesian network representation:

Definition 2.1. A *Bayesian network* $\mathcal{B}$ over the variables $X_1, \ldots, X_n$ is a pair $(G, \Pr)$. G is a directed acyclic graph with n nodes, also labeled $X_1, \ldots, X_n$. For a node X, we use $Pa(X)$ to denote the *parents* of X in the graph, i.e., those nodes Y such that there is a directed edge from Y to X. Pr is a mapping that associates with each node X a *conditional probability distribution* (*CPD*) $\Pr(X \mid Pa(X))$, which specifies a probability distribution $\Pr(X \mid \mathbf{pa})$ over the values of X for each instantiation **pa** of $Pa(X)$.

The graph G for our Alarm example is shown in Fig. 1. This BN can be viewed as a compact representation of the symmetric probability tree in Fig. 2. The tree has 32 paths, each with five binary splits, one for each variable. Most simply, the splits occur in an order which is consistent with the order of nodes in the BN, e.g., B, E, A, C, R. Each branch from a split node is labeled with the probability that the branch is taken. For example, the $B = 1$ branch is labeled with $\Pr(B = 1) = 0.01$, and the $B = 0$ branch with 0.99. Note that the tree has a lot of duplication over the BN. For example, there are eight branches in the tree all labeled with the same probability $\Pr(R = 0 \mid E = 1) = 0.65$ (one for each assignment to B, A, C).

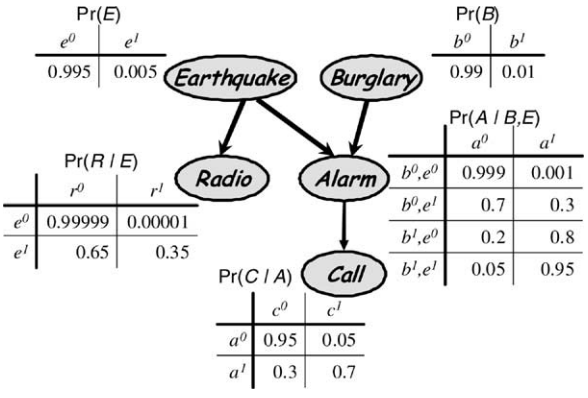

Fig. 1. Alarm BN; for these binary variables, we use the notation x^1 as shorthand for the event $X = 1$ and x^0 for the event $X = 0$.

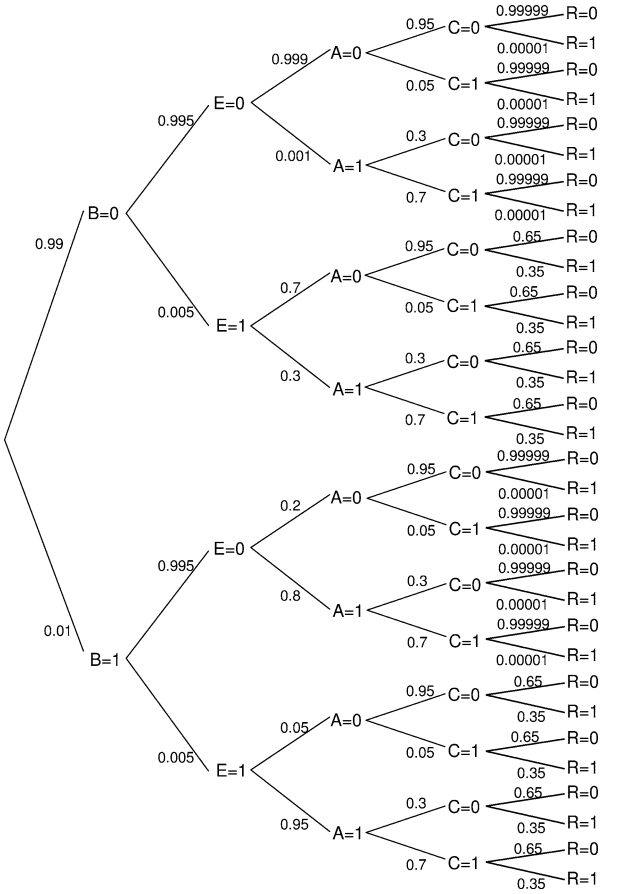

Fig. 2. Probability tree for the Alarm example.

The graph structure encodes our intuition about the way the world works, and thus lets us avoid writing the same probabilities many times:

- There either is or is not a burglary, but that event does not depend on any of the other variables in the model.
- Similarly, an earthquake either happens or does not happen, separately from everything else.
- The alarm can be set off either by a burglary or by an earthquake, and therefore it depends on both.
- Whether the neighbor calls depends only on whether the alarm went off: we can determine the probabilities of the $C = 0$ and $C = 1$ branches at a node in the probability tree just by looking at the value of A.
- The radio report depends only on whether there was an earthquake.

The CPDs associated with the nodes represent the local probability models for the different variables. One such model—$\Pr(B)$—specifies the prior probability of a burglary, i.e., it specifies the prior probabilities of the events $B = 1$ and $B = 0$. Similarly, $\Pr(E)$ specifies the prior probability of a minor earthquake. The probability of the alarm going off is a *conditional* probability distribution $\Pr(A \mid B, E)$; it specifies the probability of the alarm going off in any of the relevant circumstances: a burglary and an earthquake ($B = 1, E = 1$), a burglary and no earthquake ($B = 1, E = 0$), no burglary and an earthquake ($B = 0, E = 1$), and no burglary and no earthquake ($B = 0, E = 0$). The other models are similar. One possible choice of CPDs for this domain is shown in Fig. 1.[1]

2.2. Semantics

The semantics of a BN is as a joint distribution over $dom(\mathcal{X})$:

Definition 2.2. A BN $\mathcal{B} = (G, \Pr)$ over $X_1, \ldots, X_n$ defines a joint distribution over $X_1, \ldots, X_n$ via the *chain rule for Bayesian networks*:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} \Pr\big(X_i \mid Pa(X_i)\big). \tag{1}$$

The chain rule gives us the ability to compute the probability of any state in $dom(\mathcal{X})$. For example:

$$\begin{aligned} &P(B = 1, E = 0, A = 1, C = 1, R = 0) \\ &\quad = \Pr(B = 1)\Pr(E = 0)\Pr(A = 1 \mid B = 1, E = 0)\Pr(C = 1 \mid A = 1) \\ &\qquad \times \Pr(R = 0 \mid E = 0) = 0.01 \cdot 0.995 \cdot 0.8 \cdot 0.7 \cdot 0.99999 = 0.005566428. \end{aligned}$$

Thus, a BN is a full representation of a joint distribution. As such, it can be used to answer any query that can be answered with a joint distribution. In particular, we can assess the probability distribution over any variable conditioned on any assignment of values to a subset of others. For example, the prior probability that our neighbor calls is 0.0568. The probability that he calls in case of a burglary—$P(C = 1 \mid B = 1)$—is 0.57. Conversely, the probability of a burglary given that the neighbor called—$P(B = 1 \mid C = 1)$—is 0.1. Now, assume that we turn on the radio and hear a report of an earthquake in our neighborhood. The probability of an earthquake goes up substantially—$P(E = 1 \mid C = 1, R = 1) = 0.999$ as compared to $P(E = 1 \mid C = 1) = 0.021$. More interestingly, the probability of a burglary goes *down*—$P(B = 1 \mid C = 1, R = 1) = 0.027$. Note that burglary and earthquake are not mutually exclusive; indeed, they occur independently. Hence, there is nothing about the increased probability of an earthquake that, by itself, reduces our beliefs in a burglary. Rather, the earthquake provides an explanation for the neighbor's phone call; therefore, our basis for believing the alternative explanation (namely the burglary) is no longer as active.

[1] Our choice of probabilities is somewhat unrealistic, as the probabilities of various unlikely events (e.g., an earthquake) are higher than is plausible.

These are examples of conclusions that we can extract from the joint distribution. Of course, explicitly enumerating the joint distribution is a computationally expensive operation, except for the smallest networks. Fortunately, there are algorithms (e.g., Lauritzen and Spiegelhalter, 1988) that exploit the sparsity of the network structure to perform this type of reasoning without enumerating the joint distribution explicitly. These algorithms can work effectively even for very large networks, as long as there are not too many direct connections between the variables in the network (more precisely, as long as it is possible to construct a junction tree for the network that has small cliques—see (Lauritzen and Spiegelhalter, 1988) for details).

2.3. Independence

The BN structure induces a set of independence assumptions about the variables $X_1, \ldots, X_n$. These are assumptions that are a direct consequence of the definition of the joint distribution using the chain rule. They hold for any parameterization Pr of the graph G. These independencies are critical to our analysis, so we review them here.

We first define the basic notion of *conditional independence* of random variables.

Definition 2.3. Let P be a distribution over $\mathcal{X}$, and let $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ be three pairwise disjoint subsets of $\mathcal{X}$. We say that $\boldsymbol{X}$ *is conditionally independent of* $\boldsymbol{Y}$ *given* $\boldsymbol{Z}$ *in* P, denoted $P \models I(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$, if, for any $z \in dom(\boldsymbol{Z})$ such that $P(z) > 0$, and for any $\boldsymbol{x} \in dom(\boldsymbol{X})$, $\boldsymbol{y} \in dom(\boldsymbol{Y})$, we have that

$$P(\boldsymbol{x} \mid \boldsymbol{y}, z) = P(\boldsymbol{x} \mid z),$$

or, equivalently, that

$$P(\boldsymbol{y} \mid \boldsymbol{x}, z) = P(\boldsymbol{y} \mid z), \qquad P(\boldsymbol{x}, \boldsymbol{y} \mid z) = P(\boldsymbol{x} \mid z) \cdot P(\boldsymbol{y} \mid z).$$

This notion of conditional independence is quite strong, as it implies conditional independence for every assignment of values to the variables involved. Conversely, note that conditional independence is very different from the more standard notion of marginal independence. For example, in the distribution represented by Fig. 1, we have that C is conditionally independent of B given A—once we know whether the alarm sounded or not, knowledge about a possible burglary no longer gives us any information about the chances of a phone call. This independence is very natural; it reflects our intuition that the neighbor decides whether to call based only on hearing the alarm; he has no direct knowledge of the burglary. However, B and C are *not* marginally independent: the presence of a burglary makes the phone call more likely.

It turns out that we can use the graph structure to provide a qualitative criterion for determining independence properties of the distribution associated with the graph. At an intuitive level, we can show that probabilistic influence between the variables "flows" along paths in the graph, but can be "blocked" somewhere along the path. In our example above, influence can flow from B to C, but can be blocked if we condition on A.

The basic notion in this analysis is that of an *active path*, i.e., one along which influence can "flow." It helps to first analyze the simplest case, where one node X can influence

another Y via a third node Z. As we discussed, this depends on what our evidence is, i.e., what we are conditioning on. Let $\boldsymbol{E}$ be the set of variables on which we are conditioning; i.e., assume that our evidence is $\boldsymbol{E} = \boldsymbol{e}$ for some assignment $\boldsymbol{e}$, and we are trying to analyze the independencies in the distribution P conditioned on $\boldsymbol{E} = \boldsymbol{e}$. There are four cases to consider:

(1) The path has the form $X \rightarrow Z \rightarrow Y$. This case is the same as our example $B \rightarrow A \rightarrow C$. In this case, we say that the path is *active* if Z is not observed, i.e., $Z \notin \boldsymbol{E}$. If $Z \in \boldsymbol{E}$, we say that the path is *blocked*.
(2) The path has the form $X \leftarrow Z \leftarrow Y$. As probabilistic dependence is symmetrical, this case is precisely analogous to the first.
(3) The path has the form $X \leftarrow Z \rightarrow Y$. This case represents a situation where a common cause has two effects, for example, the earthquake causing both the alarm and the news report. It seems fairly intuitive that observing a news report about an earthquake will change our beliefs about whether the alarm has gone off. However, if we know that there is an earthquake, the news report gives us no additional information. In light of this intuition, we say that this path is active if $Z \notin \boldsymbol{E}$ and blocked otherwise.
(4) The path has the form $X \rightarrow Z \leftarrow Y$. This case, known as a *v-structure*, is the most interesting. An example in our simple network is the path from B through A to E. Note that, in the prior distribution P, B and E are marginally independent. Hence, when A is not observed, the path is not active, by contrast to the three previous cases. More interestingly, consider what happens when we observe that the alarm sounded. In this case, B and E become correlated. Intuitively, if we observe that an earthquake occurred, then that provides an explanation for the alarm, and reduces the probability of a burglary. (It is easy to verify this behavior by examining the probability distribution P given above.) Thus, the path is active if A is in $\boldsymbol{E}$. A similar phenomenon occurs if we observe C. These examples lead us to define a path $X \rightarrow Z \leftarrow Y$ to be active if Z *or one of* Z*'s descendants in* G is in $\boldsymbol{E}$, and blocked otherwise.

The definition of active for longer paths is a simple composition of the definition for the paths of length two. Intuitively, influence flows from X to Y via a long path if it flows through every intervening node. Thus, following Pearl (1988), we define a general active path as follows:

Definition 2.4. Let G be a BN structure, and $X_1 — \cdots — X_n$ an undirected path in G. Let $\boldsymbol{E}$ be a subset of nodes of G. The path $X_1 — \cdots — X_n$ is *active* given evidence $\boldsymbol{E}$ if

- whenever we have a configuration $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$, then X_i or one of its descendants is in $\boldsymbol{E}$;
- no other node along the path is in $\boldsymbol{E}$.

A path which is not active is *blocked*. We say that X and Y are *d-separated* in G given $\boldsymbol{E}$ if every path between them is blocked.

In our Alarm example, we have that R and B are d-separated, because the v-structure blocks the only possible path. On the other hand, R and B are not d-separated given C, as observing C activates the path. But, R and B *are* d-separated given E and C, as observing E blocks the path again. We note that d-separation can be computed in linear time, using, for example, Shachter's Bayes-Ball algorithm (Shachter, 1998).

So far, we have defined d-separation based purely on intuitive arguments. However, these intuitive arguments correspond to provably correct statements about independencies in the distribution. As shown by Verma and Pearl (1990), d-separation is *sound*: d-separation in G implies conditional independence for any Bayesian network $\mathcal{B} = (G, \Pr)$.

Theorem 2.1 (Verma and Pearl (1990): soundness). *Let G be a Bayesian network structure, and let X and Y be nodes in G and $\boldsymbol{E}$ a set of nodes such that $X, Y \notin \boldsymbol{E}$. If X and Y are d-separated in G given $\boldsymbol{E}$, then for any Bayesian network $\mathcal{B} = (G, \Pr)$, we have that $\mathcal{B} \models I(X; Y \mid \boldsymbol{E})$.*

In other words, the independencies that we derive qualitatively from the graph structure via d-separation hold for *every* parameterization of the network structure with CPDs.

The d-separation criterion is also *complete*, but in a weaker sense. It is not the case that if $\mathcal{B} \models I(X; Y \mid \boldsymbol{E})$ then we can necessarily detect that from the network structure. We might choose parameters that create spurious independencies, simply because two different probability expressions happen to be equal to each other. However, as shown by Geiger and Pearl (1990), if an independence does not follow from the d-separation criterion, then there is at least one counterexample to it.

Theorem 2.2 (Geiger and Pearl (1990): completeness). *Let G be a Bayesian network structure, and let X and Y be nodes in G and $\boldsymbol{E}$ a set of nodes such that $X, Y \notin \boldsymbol{E}$. If X and Y are not d-separated in G given $\boldsymbol{E}$, then there exists a Bayesian network $\mathcal{B} = (G, \Pr)$ such that $\mathcal{B} \not\models I(X; Y \mid \boldsymbol{E})$.*

In fact, Meek (1995) has proved an even stronger version: in *almost all* Bayesian networks $\mathcal{B} = (G, \Pr)$ (i.e., in all except for a set of measure zero), we have that $\mathcal{B} \not\models I(X; Y \mid \boldsymbol{E})$.

Thus, Bayesian networks provide us with a formal framework for representing independence structure in a joint distribution. They allow us to exploit this structure in order to provide a compact representation of complex joint distributions. They also provide us with a qualitative method for determining the presence and absence of independence relations in the joint distribution.

3. Multi-agent influence diagrams (MAIDs)

Influence diagrams augment the Bayesian network framework with the notions of agents that make decisions strategically, to maximize their utility. Influence diagrams were introduced by Howard (Howard and Matheson, 1984), and have been investigated almost

entirely in a single-agent setting. In this section, we present *multi-agent influence diagrams* (MAIDs), which extend the influence diagram framework to the multi-agent case.

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

Example 3.1. 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 would cost her some effort but might cause the tree 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. 3.

To define a MAID more formally, 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 the Bayesian network formalism. 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. 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—they represent components of a utility function, not actual state variables that can influence other variables or be observed by agents. For each chance variable $X \in \mathcal{X}$, the MAID specifies a CPD $\Pr(X \mid Pa(X))$, as in a BN. 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

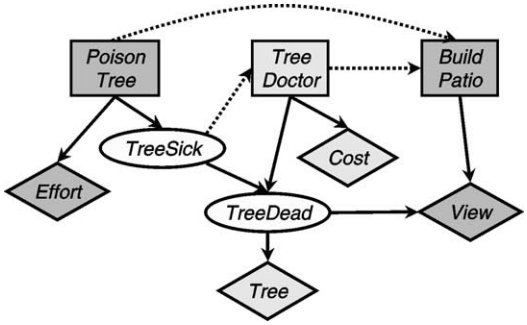

Fig. 3. A MAID for the Tree Killer example; Alice's decision and utility variables are in dark gray and Bob's in light gray.

D can be contingent only on these variables. (See Definition 3.1 below.) Edges into a decision variable are drawn as dotted lines. For a utility variable U, the MAID also specifies a CPD $\Pr(U \mid \mathbf{pa})$ for each instantiation $\mathbf{pa}$ of $Pa(U)$. 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, by breaking an agent's utility function into several variables, we are simply defining an additive decomposition of the agent's utility function (Howard and Matheson, 1984; Keeney and Raiffa, 1976).

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 3.1. A *decision rule* for a decision variable D is a function that maps each instantiation $\mathbf{pa}$ 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*.

Thus, a decision rule may be deterministic, or it may specify that there is some randomness in the agent's behavior.

Definition 3.2. A decision rule δ for D is *fully mixed* if, for every instantiation $\mathbf{pa}$ of $Pa(D)$ and every $d \in dom(D)$, we have $\delta(d \mid \mathbf{pa}) > 0$.

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 σ that assigns decision rules to a set $\mathcal{E}$ of decision variables induces a new MAID $\mathcal{M}[\sigma]$ where the elements of $\mathcal{E}$ have become chance variables. That is, each $D \in \mathcal{E}$ corresponds to a chance variable in $\mathcal{M}[\sigma]$ with $\sigma(D)$ 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 (recall that the utility variables are just BN variables with domains consisting of real numbers, and with deterministic CPDs). This BN defines a joint probability distribution $P_{\mathcal{M}[\sigma]}$ over all the variables in $\mathcal{M}$.

Definition 3.3. 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 $X \to Y$ if and only if $X \in Pa(Y)$;
- 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 $\sigma(D)$.

With this probability distribution, 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:

$$\mathrm{EU}_a(\sigma) = \sum_{(u_1,\ldots,u_m)\in dom(\mathcal{U}_a)} P_{\mathcal{M}[\sigma]}(u_1, \ldots, u_m) \sum_{i=1}^{m} u_i \tag{2}$$

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:

$$\mathrm{EU}_a(\sigma) = \sum_{U\in\mathcal{U}_a} \sum_{u\in dom(U)} P_{\mathcal{M}[\sigma]}(U = u) \cdot u. \tag{3}$$

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 3.4. 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 $\sigma'_{\mathcal{E}}$:

$$\mathrm{EU}_a\big((\sigma_{-\mathcal{E}}, \sigma^*_{\mathcal{E}})\big) \geqslant \mathrm{EU}_a\big((\sigma_{-\mathcal{E}}, \sigma'_{\mathcal{E}})\big).$$

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 3.5. 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.

The task of finding a Nash equilibrium for a game is arguably the most fundamental task in noncooperative game theory.

4. 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.

There are many ways to convert a MAID into a game tree, all of which are equivalent from a semantic perspective, but which differ dramatically in their computational costs. To understand this point, consider a MAID with n binary-valued chance variables $C_1, \ldots, C_n$, with C_i the sole parent of C_{i+1}, and a single decision D whose sole parent is C_n. One naive approach is to generate a symmetric tree where the root is a chance node splitting on C_1; its two children are both chance nodes splitting on C_2; etc. Each path down the tree will thus contain n splits (one for each of $C_1, \ldots, C_n$), and then have a final split for D, belonging to the agent. To preserve the information structure, the tree will have two information sets, one for $C_n = \mathit{true}$ and one for $C_n = \mathit{false}$; thus, all of the 2^{n-1} nodes where $C_n = \mathit{true}$ would be in the first information set. The advantage of this tree is that the probabilities associated with the chance-splits are simply extracted from the CPDs in the MAID. The disadvantage is that the size of the tree grows exponentially with the number of chance variables in the MAID, even when this blowup is extraneous.

A alternative approach (based on the construction of Pearl (1988, p. 311) is to generate a tree that has only the minimal set of splits. In our example, we only need a single split for C_n, and then another split for D. The probabilities of the C_n split must now be computed from the MAID using probabilistic inference; i.e., we want to compute $P(C_n)$ in the joint distribution over $C_1, \ldots, C_n$ defined by the MAID. Similarly, we must now compute the expected utilities at the leaves.

More generally, we need to split on a chance variable before its value is observed by some decision node. Furthermore, we need only split on chance variables that are observed at some point in the process. Thus, the set of variables included in our game tree is $\mathcal{G} = \mathcal{D} \cup \bigcup_{D \in \mathcal{D}} \mathit{Pa}(D)$. We present the construction below, referring the reader to (Pearl, 1988) for a complete discussion.

We begin by defining 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 $\mathit{inst}(N)$ 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 $\mathit{inst}(M)$ and $\mathit{inst}(M')$ assign the same values to $\mathit{Pa}(D)$.

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 $c \in \mathit{dom}(C)$, 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 (one where every decision rule satisfies Definition 3.2) for our MAID $\mathcal{M}$, and do inference in the BN $\mathcal{M}[\sigma]$ induced by this strategy profile. Specifically, we can compute:

$$P_{\mathcal{M}[\sigma]}\big(\mathit{inst}(N_c) \mid \mathit{inst}(N)\big). \tag{4}$$

We can do this computation for any instantiation $\mathit{inst}(N)$ that has nonzero probability under $P_{\mathcal{M}[\sigma]}$. Since σ is fully mixed, any instantiation that has zero probability under $P_{\mathcal{M}[\sigma]}$ will have zero probability under every other strategy profile as well.

Lemma 4.1. *The value of* (4) *does not depend on our choice of* σ.

Proof. We provide only a brief sketch of the proof. Note that if we split on a decision variable D before C, then the decision rule σ_D does not affect the computation of $P_{\mathcal{M}[\sigma]}(\mathit{inst}(N_c) \mid \mathit{inst}(N))$, because $\mathit{inst}(N)$ 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, because the order of splitting is a topological ordering of the MAID, $\mathit{inst}(N)$ cannot specify evidence on any of D's descendants. Therefore, σ_D cannot affect the computation. Hence, the value of this expression does not depend on our choice of σ. □

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:

$$\sum_{U \in \mathcal{U}_a} \sum_{u \in \mathit{dom}(U)} P_{\mathcal{M}[\sigma]}\big(U = u \;\big|\; \mathit{inst}(N)\big) \cdot u. \tag{5}$$

Lemma 4.2. *The value of* (5) *does not depend on our choice of* σ.

Proof. The basic idea here is that $\mathit{inst}(N)$ determines the values of D and $\mathit{Pa}(D)$ 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 $\mathit{dom}(D)$ for each pair $(D, \mathbf{pa})$, where $\mathbf{pa}$ is an instantiation of $\mathit{Pa}(D)$. 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 4.1. *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 certain games. As we now show, a MAID can be exponentially smaller than the extensive game it corresponds to.

Example 4.1. 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,

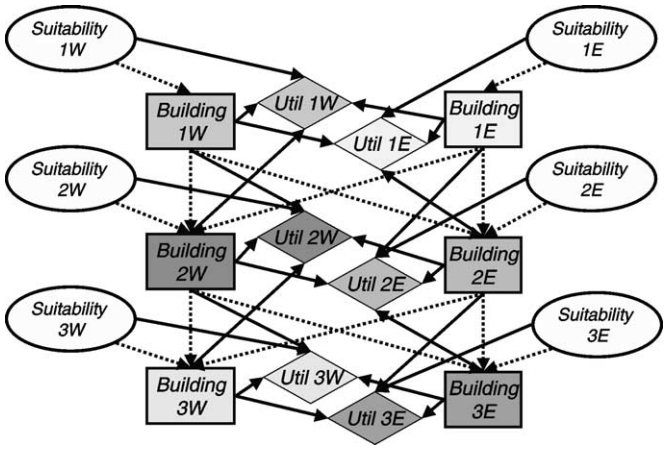

Fig. 4. A MAID for the Road example with $n = 6$.

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.

The MAID representation, shown in Fig. 4 for $n = 6$, is very compact. There are n chance nodes, corresponding to the private information about each agent's land; n decision variables; and n utility variables. The chance nodes have no parents, and each utility variable has at most five parents: the agent's private information, the agent's own decision, and the decisions of the agents north, south, and across the road from this agent. Thus, the size of the MAID—including CPDs for the chance and utility nodes—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.

Example 4.2. Consider, for example, a standard centipede game, a perfect information two-player game. The agents take turns moving, each move consisting of either "right" or "down"; the game ends as soon as any agent decides to move "down." An agent's utility is equal to the number of times he moved right, plus 2 points if he moved down. We can view the agents' decisions as variables $D_1, \ldots, D_n$, each of which takes one of two values. An agent's utility depends on D_n, but only if none of $D_1, \ldots, D_{n-1}$ have the value "down." Thus, there must be a utility node with all of $D_1, \ldots, D_n$ as parents. So a naive representation of the MAID (with its CPDs) grows exponentially in this example, despite the fact that the tree only has n decisions, and can be represented very compactly.

It is possible to avoid this problem by representing CPDs and decision rules more compactly (Boutilier et al., 1996; Poole, 1997). Rather than representing CPDs as tables, we can represent them as trees, only considering combinations of parents that represent achievable paths in the game. Using ideas along these lines, it is fairly straightforward to provide a transformation from game trees to MAIDs which causes no blowup, i.e., so that the size of the MAID is the same as that of the game tree. The transformation is somewhat technical, and brings no real insight into MAIDs, so we omit it from this paper. However, it has the following important consequence: We can state that the MAID representation of a decision-making situation is no larger than the extensive form representation, and is exponentially smaller in many cases.

5. 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 3.1, when Alice is deciding whether to poison Bob's tree, she needs to compare the expected utilities of her two alternatives. The expected utility of poisoning the tree depends on the probability of the tree dying, given that it has been poisoned. However, this probability 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 *TreeDoctor* to determine the optimal decision rule for *PoisonTree*. In such situations, we will say that *PoisonTree* (*strategically*) *relies on TreeDoctor*, or that *TreeDoctor* is *relevant to PoisonTree*. On the other hand, *TreeDoctor* does not rely on *PoisonTree*. The cost of hiring a tree doctor does not depend on whether the tree was poisoned, and if we know whether the tree is sick, the probability of it dying is independent of whether it was poisoned. So Bob does not need to know the probability of poisoning to compute the expected utilities of his choices, given that he can observe *TreeSick*.

5.1. Definition of strategic relevance

We will now formalize this intuitive discussion of strategic relevance. Suppose we have a strategy profile σ, and we would like to find a decision rule δ 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.

Recall that in Definition 3.4, to determine whether δ 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. The decision rule 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: Which decision rules in σ are relevant for optimizing the decision rule at D? We begin with a draft definition of strategic relevance, then explain why it needs to be refined somewhat.

Draft Definition 5.1. Let D and D' be decision nodes in a MAID $\mathcal{M}$. *D strategically relies on D'* if there exist two strategy profiles σ and σ' and a decision rule δ for D such that:

- δ is optimal for σ;
- σ' differs from σ only at D';

but δ is not optimal for σ'.

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 reason why this definition is insufficient is one that arises in many other places in game theory—the problem of suboptimal decisions in response to probability zero moves by the other agent.

Example 5.1. Consider a very simple scenario in which Alice chooses whether to go north or south, and then Bob gets to observe her action and choose whether he will go north or south. Both agents receive positive utility if they end up in the same place, and negative utility otherwise. We can model this scenario with the MAID $\mathcal{M}$ shown in Fig. 5. Intuitively, Bob's decision D does not rely on Alice's decision D': regardless of Alice's decision rule, it is optimal for Bob to adopt a decision rule that says, "go north if Alice goes north; go south if she goes south." But suppose σ is a strategy profile in which Alice goes north with probability 1, and δ is a decision rule for D that has Bob go north regardless of Alice's action. Then δ is optimal for σ: even though going north would be a suboptimal action for Bob if Alice ever went south, Bob does not have an incentive to deviate to any other decision rule in $\mathcal{M}[\sigma]$ because the probability of Alice going south is zero. However, if σ' has Alice go south with probability 0.5, then δ is not optimal for σ'. So, by our draft definition, D relies on D'.

Intuitively, we do not want D to rely on D'. We now revise our definition of strategic relevance to exclude cases such as this one.

Definition 5.1. Let D and D' be decision nodes in a MAID $\mathcal{M}$. *D strategically relies on D'* if there exist two strategy profiles σ and σ' and a decision rule δ for D such that:

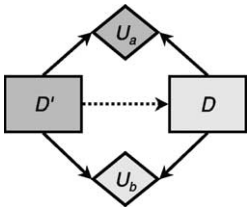

Fig. 5. A MAID in which Bob can observe Alice's action D' when he makes his decision D.

- δ is optimal for σ;
- σ' differs from σ only at D';

but no decision rule δ^* that agrees with δ on all parent instantiations $\mathbf{pa} \in dom(Pa(D))$ where $P_{\mathcal{M}[\sigma]}(\mathbf{pa}) > 0$ is optimal for σ'.

Continuing our example, we can construct a decision rule δ^* for D that differs from δ only on the parent instantiation $D' = south$, and this δ^* will be optimal for σ'. So under the revised definition, D does not rely on D'.

5.2. S-reachability

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.

To begin with, suppose we have a strategy profile σ for a MAID $\mathcal{M}$, and consider finding a decision rule for D that is optimal for σ. The following lemma specifies the maximization problems we must solve to find this optimal decision rule.

Lemma 5.1. *Let δ be a decision rule for a decision variable $D \in \mathcal{D}_a$ in a MAID $\mathcal{M}$, and let σ be a strategy profile for $\mathcal{M}$. Then δ is optimal for σ if and only if for every instantiation $\mathbf{pa}_D$ of $Pa(D)$ where $P_{\mathcal{M}[\sigma]}(\mathbf{pa}_D) > 0$, the probability distribution $\delta(D \mid \mathbf{pa}_D)$ is a solution to the maximization problem*:

$$\operatorname*{argmax}_{P^*} \sum_{d \in dom(D)} P^*(d) \sum_{U \in \mathcal{U}_D} \sum_{u \in dom(U)} P_{\mathcal{M}[\sigma]}(u \mid d,\ \mathbf{pa}_D) \cdot u \tag{6}$$

where $\mathcal{U}_D$ is the set of utility nodes in $\mathcal{U}_a$ that are descendants of D in the MAID.

Proof. By Definition 3.4, δ is optimal for σ if and only if δ maximizes agent a's expected utility in $\mathcal{M}[\sigma_{-D}]$. So δ is optimal for σ if and only if δ is a solution for:

$$\operatorname*{argmax}_{\delta^*} \mathrm{EU}_{\mathcal{M}}(\sigma_{-D}, \delta^*).$$

By (3), this is equivalent to:

$$\operatorname*{argmax}_{\delta^*} \sum_{U \in \mathcal{U}_a} \sum_{u \in dom(U)} P_{\mathcal{M}[(\sigma_{-D}, \delta^*)]}(U = u) \cdot u.$$

The events $(U = u, D = d, Pa(D) = \mathbf{pa}_D)$ for $d \in dom(D)$, $\mathbf{pa}_D \in dom(Pa(D))$ form a partition of the event $U = u$, so we can express $P_{\mathcal{M}[(\sigma_{-D}, \delta^*)]}(U = u)$ as the sum of the probabilities of these more specific events. Then breaking up the joint probability expression yields:

$$P_{\mathcal{M}[(\sigma_{-D},\delta^*)]}(U=u)$$
$$= \sum_{\mathbf{pa}_D \in dom(Pa(D))} P_{\mathcal{M}[(\sigma_{-D},\delta^*)]}(\mathbf{pa}_D)$$
$$\times \sum_{d \in dom(D)} P_{\mathcal{M}[(\sigma_{-D},\delta^*)]}(d \mid \mathbf{pa}_D) P_{\mathcal{M}[(\sigma_{-D},\delta^*)]}(U=u \mid d,\ \mathbf{pa}_D).$$

The CPD for a node in a Bayesian network has no effect on the prior distribution over its parents, so $P_{\mathcal{M}[(\sigma_{-D},\delta^*)]}(\mathbf{pa}_D) = P_{\mathcal{M}[\sigma]}(\mathbf{pa}_D)$. Also, $P_{\mathcal{M}[(\sigma_{-D},\delta^*)]}(d \mid \mathbf{pa}_D)$ is simply $\delta^*(d \mid \mathbf{pa}_D)$. And $P_{\mathcal{M}[(\sigma_{-D},\delta^*)]}(U=u \mid d, \mathbf{pa}_D) = P_{\mathcal{M}[\sigma]}(U=u \mid d, \mathbf{pa}_D)$, because a distribution does not depend on the CPD for D given values for D and its parents. So we find that:

$$P_{\mathcal{M}[\sigma_{-D},\delta^*]}(U=u) = \sum_{\mathbf{pa}_D \in dom(Pa(D))} P_{\mathcal{M}[\sigma]}(\mathbf{pa}_D)$$
$$\times \sum_{d \in dom(D)} \delta^*(d \mid \mathbf{pa}_D) P_{\mathcal{M}[\sigma]}(U=u \mid d,\ \mathbf{pa}_D).$$

Thus, δ is optimal for σ if and only if it is a solution for:

$$\operatorname*{argmax}_{\delta^*} \sum_{U \in \mathcal{U}_a} \sum_{u \in dom(U)} \sum_{\mathbf{pa}_D \in dom(Pa(D))} P_{\mathcal{M}[\sigma]}(\mathbf{pa}_D)$$
$$\times \sum_{d \in dom(D)} \delta^*(d \mid \mathbf{pa}_D) P_{\mathcal{M}[\sigma]}(U=u \mid d,\ \mathbf{pa}_D) \cdot u.$$

Rearranging the summations, we get:

$$\operatorname*{argmax}_{\delta^*} \sum_{\mathbf{pa}_D \in dom(Pa(D))} P_{\mathcal{M}[\sigma]}(\mathbf{pa}_D) \sum_{d \in dom(D)} \delta^*(d \mid \mathbf{pa}_D)$$
$$\times \sum_{U \in \mathcal{U}_a} \sum_{u \in dom(U)} P_{\mathcal{M}[\sigma]}(U=u \mid d,\ \mathbf{pa}_D) \cdot u.$$

Because we can set $\delta^*(D \mid \mathbf{pa}_D)$ separately for each $\mathbf{pa}_D$, it is clear that we have a separate maximization problem for each $\mathbf{pa}_D$. If $P_{\mathcal{M}[\sigma]}(\mathbf{pa}_D) = 0$, the maximization problem is trivial: all distributions $\delta^*(D \mid \mathbf{pa}_D)$ yield a value of zero. Thus, it is necessary and sufficient that for all $\mathbf{pa}_D$ such that $P_{\mathcal{M}[\sigma]}(\mathbf{pa}_D) > 0$, the distribution $\delta(D \mid \mathbf{pa}_D)$ be a solution of the following maximization problem:

$$\operatorname*{argmax}_{P^*} \sum_{d \in dom(D)} P^*(d) \sum_{U \in \mathcal{U}_a} \sum_{u \in dom(U)} P_{\mathcal{M}[\sigma]}(U=u \mid d,\ \mathbf{pa}_D) \cdot u.$$

Now, let $\mathcal{U}_D$ be the utility nodes in $\mathcal{U}_a$ that are descendants of D, and let $\mathcal{U}_{\widetilde{D}} = \mathcal{U}_a - \mathcal{U}_D$. Since a node in a BN is independent of its nondescendants given its parents, we know that for $U \in \mathcal{U}_{\widetilde{D}}$, $P_{\mathcal{M}[\sigma]}(U=u \mid d,\ \mathbf{pa}_D) = P_{\mathcal{M}[\sigma]}(U=u \mid \mathbf{pa}_D)$. Thus, we can split up the summation over $\mathcal{U}_a$ into two summations, then move the summation over $\mathcal{U}_{\widetilde{D}}$ outside the summation over $dom(D)$. The resulting maximization problem is:

$$\operatorname*{argmax}_{P^*}\Bigg(\Bigg(\sum_{U\in\mathcal{U}_{\widetilde{D}}}\sum_{u\in dom(U)} P_{\mathcal{M}[\sigma]}(U=u\mid \mathbf{pa}_D)\cdot u\Bigg) + \Bigg(\sum_{d\in dom(D)} P^*(d)\sum_{U\in\mathcal{U}_D}\sum_{u\in dom(U)} P_{\mathcal{M}[\sigma]}(U=u\mid d,\ \mathbf{pa}_D)\cdot u\Bigg)\Bigg).$$

The summation over $\mathcal{U}_{\widetilde{D}}$ is irrelevant to the maximization, so we can remove it, yielding:

$$\operatorname*{argmax}_{P^*}\sum_{d\in dom(D)} P^*(d)\sum_{U\in\mathcal{U}_D}\sum_{u\in dom(U)} P_{\mathcal{M}[\sigma]}(U=u\mid d,\ \mathbf{pa}_D)\cdot u,$$

which is precisely (6), as required. □

So, to be optimal for a strategy profile σ, a decision rule δ only has to satisfy (6). If the expression being maximized in (6) is independent of the decision rule that σ assigns to another decision variable D', then D does not rely on D'.

Consider, for example, the *TreeDoctor* decision variable in our example. To find an optimal decision rule for this variable, we only need to evaluate two probabilistic queries: $P_{\mathcal{M}[\sigma]}(\mathit{Tree} \mid \mathit{TreeSick}, \mathit{TreeDoctor})$ and $P_{\mathcal{M}[\sigma]}(\mathit{Cost} \mid \mathit{TreeSick}, \mathit{TreeDoctor})$. The second query is obviously trivial since *TreeDoctor* is the sole parent of *Cost*; the first can be evaluated without referring to the decision rules for *PoisonTree* or *BuildPatio* (because of the independence relations in the MAID). Thus, if we change σ to another strategy profile σ' that assigns different decision rules to *PoisonTree* or *BuildPatio*, δ will still be optimal for σ'.

The problem of determining which nodes' CPDs might affect the evaluation of a probabilistic query is a standard one in the Bayesian network literature, so that we can build on a graphical criterion already defined for Bayesian networks, that of a *requisite probability node*:

Definition 5.2. 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 $\boldsymbol{Y}$. More formally:

Lemma 5.2 (Geiger et al., 1990). *Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two Bayesian networks over the same set of variables, that are identical except in the CPDs they assign to a set of nodes $\boldsymbol{Z}$. Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be arbitrary sets of nodes in these networks. Suppose there is no $Z \in \boldsymbol{Z}$ such that if a new parent $\widehat{Z}$ were added to Z, there would be an active path from $\widehat{Z}$ to $\boldsymbol{X}$ given $\boldsymbol{Y}$. Then $P_{\mathcal{B}_1}(\boldsymbol{X} \mid \boldsymbol{Y}) = P_{\mathcal{B}_2}(\boldsymbol{X} \mid \boldsymbol{Y})$.*

Based on this lemma and (6), 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 5.3. 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.

As we now show, 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 5.1 (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 strategically rely on D'.*

Proof. Let σ be a strategy profile for $\mathcal{M}$, and let δ be a decision rule for D that is optimal for σ. By Lemma 5.1, for every $\mathbf{pa}_D \in dom(Pa(D))$ such that $P_{\mathcal{M}[\sigma]}(\mathbf{pa}_D) > 0$, the distribution $\delta(D \mid \mathbf{pa}_D)$ must be a solution of the maximization problem:

$$\operatorname*{argmax}_{P^*} \sum_{d \in dom(D)} P^*(d) \sum_{U \in \mathcal{U}_D} \sum_{u \in dom(U)} P_{M[\sigma]}(u \mid d,\ \mathbf{pa}_D) \cdot u. \tag{7}$$

Now, let σ' be any strategy profile for $\mathcal{M}$ that differs from σ only at D'. We must construct a decision rule δ^* for D that agrees with δ on all $\mathbf{pa}_D$ where $P_{\mathcal{M}[\sigma]}(\mathbf{pa}_D) > 0$, and that is optimal for σ'. By Lemma 5.1, it suffices to show that for every $\mathbf{pa}_D$ where $P_{\mathcal{M}[\sigma']}(\mathbf{pa}_D) > 0$, $\delta^*(D \mid \mathbf{pa}_D)$ is a solution of:

$$\operatorname*{argmax}_{P^*} \sum_{d \in dom(D)} P^*(d) \sum_{U \in \mathcal{U}_D} \sum_{u \in dom(U)} P_{M[\sigma']}(u \mid d,\ \mathbf{pa}_D) \cdot u. \tag{8}$$

If $P_{\mathcal{M}[\sigma]}(\mathbf{pa}_D) = 0$, then our choice of $\delta^*(D \mid \mathbf{pa}_D)$ is unconstrained; we can simply select a distribution that satisfies (8). For other $\mathbf{pa}_D$, we must let $\delta^*(D \mid \mathbf{pa}_D) = \delta(D \mid \mathbf{pa}_D)$, but we know $\delta(D \mid \mathbf{pa}_D)$ is a solution of (7). Assume for contradiction that $\delta(D \mid \mathbf{pa}_D)$ is not also a solution of (8). Then the two optimization problems must be different. Specifically, there must be some $d \in dom(D)$, $U \in \mathcal{U}_D$, and $u \in dom(U)$ such that:

$$P_{M[\sigma]}(u \mid d,\ \mathbf{pa}_D) \neq P_{M[\sigma']}(u \mid d,\ \mathbf{pa}_D).$$

But the induced MAIDs $\mathcal{M}[\sigma]$ and $\mathcal{M}[\sigma']$ are Bayesian networks that differ only in the CPD that they assign to D'. Because D' is not s-reachable from D, we know there would not be an active path from a new parent $\widehat{D'}$ of D' to U given D and $Pa(D)$. So by Lemma 5.2

$$P_{M[\sigma]}(u \mid d,\ \mathbf{pa}_D) = P_{M[\sigma']}(u \mid d,\ \mathbf{pa}_D)$$

and we have a contradiction. So δ^* must be optimal for σ', and thus 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 5.2 (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'.*

Proof. Our goal is to construct a set of parameters for this MAID structure where D relies on D'. In other words, we want a construction of a MAID, and two strategy profiles differing only at D', such that the optimal decision rule at D will be different in the two cases. We begin by assuming that all nonutility variables are binary (with $\{0, 1\}$ as their domains); we show at the end that this assumption is easy to relax.

Since D' is s-reachable from D, we know that there is a path from an imaginary parent $\widehat{D'}$ of D' to some utility node U that is a descendant of D, and this path is active given D and $Pa(D)$. Let $Y_0, \ldots, Y_{m+1}$ be this active path from $\widehat{D'}$ to U, with $Y_0 = \widehat{D'}$ and $Y_{m+1} = U$. Let Y_k be the first node on this path such that Y_k is not a descendant of D but Y_{k+1} is a descendant of D. Since $Y_0 = \widehat{D'}$ has no parents, it cannot be a descendant of D; conversely, U is, by assumption, a descendant of D. Hence, such a node Y_k has to exist. We will refer to the directed path from D to Y_{k+1} as $X_0 = D, \ldots, X_\ell, X_{\ell+1} = Y_{k+1}$.

We first prove that the path $Y_k, \ldots, Y_{m+1}$ is also a directed path $Y_k \to \cdots \to Y_{m+1}$. By contradiction, assume otherwise, and let i be the first place in the path where Y_{i+1} is not a child of Y_i. Then, we have a triple of nodes $Y_{i-1} \to Y_i \leftarrow Y_{i+1}$, for $i \geqslant k$. For the path from Y_0 to Y_{m+1} to be active in a given context, Y_i or one of its descendants must be observed in that context. In other words, Y_i or one of its descendants must be in the set $D \cup Pa(D)$. But note that, as Y_{k+1} is a descendant of D, and all edges on the path from Y_k to Y_i are downstream edges, Y_i is also a descendant of D. Hence, we conclude Y_i is a descendant of D, but it or one of its descendants is either D or a parent of D, violating the assumption that the graph is acyclic, and reaching the desired contradiction.

As a consequence, we conclude that U must be a descendant of Y_k. Therefore, there exists a path from D to U of the form

$$X_0 = D, \ldots, X_\ell, X_{\ell+1} = Y_{k+1}, Y_{k+2}, \ldots, Y_{m+1} = U$$

with one of the two segments potentially empty. We split the remainder of the proof into three cases, based on the value of k. The three cases are illustrated in Fig. 6.

In all cases, we construct a parameterization of the MAID $\mathcal{M}$ and two strategy profiles σ_1 and σ_2 that differ only at D'. We always set the CPDs for the nodes along the paths $X_0, \ldots, X_\ell$ and $Y_{k+2}, \ldots, Y_m$ to copy the values of their parents along the path. That is,

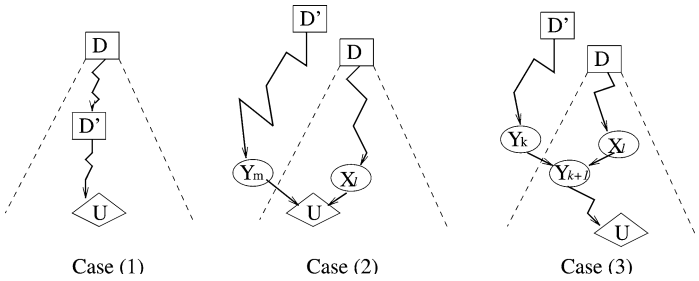

Fig. 6. The three cases of the completeness proof.

for $i \in \{1, \ldots, \ell\}$, $X_i = X_{i-1}$ with probability 1, and for $i \in \{k+1, \ldots, m\}$, $Y_i = Y_{i-1}$ with probability 1. These CPDs are specified by $\mathcal{M}$ for chance nodes, and by σ_1 and σ_2 for decision nodes. The CPD for Y_{k+1} will vary from case to case. We also set the utility functions in $\mathcal{M}$ so that if the agent who controls D is a, all utility nodes in $\mathcal{U}_a - \{U\}$ take on the value zero given all instantiations of their parents.

We will construct σ_1 and σ_2 so that there is some instantiation $\mathbf{pa}_D$ of $Pa(D)$ such that $P_{M[\sigma_i]}(\mathbf{pa}_D) > 0$ for both $i = 1$ and $i = 2$. We can apply Lemma 5.1 to conclude that the optimal distribution for D given $\mathbf{pa}_D$, in the context of the strategy profile σ_i, must be

$$\operatorname*{argmax}_{P^*} \sum_{d \in dom(D)} P^*(d) \sum_{U' \in \mathcal{U}_D} \sum_{u \in dom(U')} P_{M[\sigma_i]}(U' = u \mid D = d,\ \mathbf{pa}_D) \cdot u.$$

As the other utility variables except for U are identically zero, the second summation disappears. Now, consider the utility value for a particular decision d for D, and recall that the chain from D to X_ℓ has the effect of copying D's value. Thus, we get that:

$$\begin{aligned}
&\sum_{u \in dom(U)} P_{M[\sigma_i]}(U = u \mid D = d,\ \mathbf{pa}_D) \cdot u \\
&\quad = \sum_{u \in dom(U)} P_{M[\sigma_i]}(U = u \mid X_\ell = d,\ D = d,\ \mathbf{pa}_D) \cdot u \\
&\quad = \sum_{\mathbf{pa}_U \in dom(Pa(U))} P_{M[\sigma_i]}(\mathbf{pa}_U \mid X_\ell = d,\ D = d,\ \mathbf{pa}_D) U(\mathbf{pa}_U).
\end{aligned}$$

Case (1). $k = 0$, that is, $Y_{k+1} = D'$. We let σ_1 specify a decision rule δ_1 for D' such that D' depends only on X_ℓ (ignoring any other parents), as follows:

$$\delta_1(D' = 1 \mid X_\ell = 1) = 0.75, \qquad \delta_1(D' = 1 \mid X_\ell = 0) = 0.25.$$

All the remaining CPDs in $\mathcal{M}$ and σ_1 are set arbitrarily. The utility function at U depends only on Y_m, as follows:

$$U(Y_m = 1) = 1, \qquad U(Y_m = 0) = 0.$$

So the expected utility for an action d at D given $\mathbf{pa}_D$ becomes:

$$\begin{aligned}
&\sum_{\mathbf{pa}_U \in dom(Pa(U))} P_{M[\sigma_i]}(\mathbf{pa}_U \mid X_\ell = d,\ D = d,\ \mathbf{pa}_D) U(\mathbf{pa}_U) \\
&\quad = P_{\mathcal{M}[\sigma_i]}(Y_m = 1 \mid X_\ell = d,\ D = d,\ \mathbf{pa}_D) \\
&\quad = P_{\mathcal{M}[\sigma_i]}(D' = 1 \mid X_\ell = d,\ D = d,\ \mathbf{pa}_D) \\
&\quad = P_{\mathcal{M}[\sigma_i]}(D' = 1 \mid X_\ell = d).
\end{aligned}$$

In the steps above, we can eliminate the summation because $U(Y_m = 0) = 0$ and $U(Y_m = 1) = 1$. Then we can replace Y_m with $D' = Y_{k+1}$ because of the way the CPDs are defined along the path $Y_{k+1}, \ldots, Y_m$. Finally, we can simplify the probability statement because D' is independent of D and $Pa(D)$ given X_ℓ. Given this equation, it is clear that in $\mathcal{M}[\sigma_1]$, the expected utility of $D = 1$ given $\mathbf{pa}_D$ is 0.75, while the expected utility of $D = 0$ given $\mathbf{pa}_D$ is 0.25. So the optimal decision rule is to choose $D = 1$ with probability 1. However, if we let σ_2 assign the following decision rule to D':

$$\delta_2(D' = 1 \mid X_\ell = 1) = 0.25, \qquad \delta_2(D' = 1 \mid X_\ell = 0) = 0.75$$

then the expected utilities are exactly reversed in $\mathcal{M}[\sigma_2]$. So the unique optimal decision rule is to choose $D = 0$ with probability 1. Since these decision rules differ on parent instantiations that have positive probability in $\mathcal{M}[\sigma]$, D relies on D'.

Next, we consider the cases where D' is not a descendant of D. Let $\mathcal{V}'$ be the set consisting of all of the chance and decision nodes in the MAID except for D and its descendants. This set is a "prefix" of the graph—if $X \in \mathcal{V}'$ then so are X's parents. Let $\mathcal{B}$ be a Bayesian network which duplicates the structure of the MAID over $\mathcal{V}'$, and includes the dummy parent $\widehat{D'}$ of D'. It is easy to see that the path from $\widehat{D'}$ to Y_k is an active path in $\mathcal{B}$ given $Pa(D)$: the set $\mathcal{V}'$ contains all of the nodes in the path, and all of the ancestors of any observed nodes (only $Pa(D)$ are observed). Hence, by the completeness result of Geiger et al. (1990), there must be a parameterization of the CPDs in $\mathcal{B}$, and some assignment $\mathbf{pa}_D$ to $Pa(D)$, such that $P_{\mathcal{B}}(\mathbf{pa}_D) > 0$, and:

$$P_{\mathcal{B}}\big(Y_k \,\big|\, \mathbf{pa}_D,\ \widehat{D'} = e_1\big) \neq P_{\mathcal{B}}\big(Y_k \,\big|\, \mathbf{pa}_D,\ \widehat{D'} = e_2\big)$$

where e_1 and e_2 are two possible values of $\widehat{D'}$. Since these two distributions are different, we can choose y to be some particular value of Y_k such that

$$\begin{aligned}
&P_{\mathcal{B}}\big(Y_k = y \,\big|\, Pa(D) = \mathbf{pa}_D,\ \widehat{D'} = e_1\big) = p_1, \\
&P_{\mathcal{B}}\big(Y_k = y \,\big|\, Pa(D) = \mathbf{pa}_D,\ \widehat{D'} = e_2\big) = p_2
\end{aligned}$$

and $p_1 > p_2$. Let p^* be some value between p_1 and p_2.

We use the CPDs of $\mathcal{B}$ to specify the CPDs and decision rules of all of the nodes in $\mathcal{M}$ that are not descendants of D. For any chance node except D', we simply copy the CPDs from $\mathcal{B}$ to $\mathcal{M}$. For any decision node except D', we simply use its CPD in $\mathcal{B}$ (where we viewed it as a chance node) as a decision rule. For D', we introduce two decision rules: δ_1, which is the CPD of D' conditioned on the context $\widehat{D'} = e_1$, and δ_2, which is the same CPD conditioned on the context $\widehat{D'} = e_2$. Let $\sigma_1(D') = \delta_1$ and $\sigma_2(D') = \delta_2$.

Case (2). $k = m$, that is, the only node in the path $Y_0, \ldots, Y_{m+1}$ that is a descendant of D is the utility node $Y_{m+1} = U$. In this case, the parents of U include Y_m and X_ℓ (the parent

of Y_{k+1} in the chain $D, \ldots, Y_{k+1}$). We set the utility function at U to ignore all parents except X_ℓ and Y_m, and for those variables, to take the following values:

$$U(X_\ell = 1,\ Y_m = 1) = p^*, \qquad U(X_\ell = 0,\ Y_m = 1) = 1,$$
$$U(X_\ell = 1,\ Y_m = 0) = p^*, \qquad U(X_\ell = 0,\ Y_m = 0) = 0.$$

Using the equations above, we find that the expected utility of choosing d given $\mathbf{pa}_D$ in $\mathcal{M}[\sigma_i]$ is:

$$\begin{aligned}
&\sum_{y=0,1} P_{\mathcal{M}[\sigma_i]}(Y_m = y \mid X_\ell = d,\ D = d,\ \mathbf{pa}_D) U(X_\ell = d,\ Y_m = y) \\
&\quad = \sum_{y=0,1} P_{\mathcal{M}[\sigma_i]}(Y_m = y \mid \mathbf{pa}_D) U(X_\ell = d,\ Y_m = y) \\
&\quad = \sum_{y=0,1} P_{\mathcal{B}}(Y_m = y \mid \widehat{D'} = e_i,\ \mathbf{pa}_D) U(X_\ell = d,\ Y_m = y)
\end{aligned}$$

where the first equality follows from the fact that Y_m is independent of D and X_ℓ given $Pa(D)$, and the last from the definition of the MAID using the BN. Note that, as $k + 1 = m + 1$, then $m = k$. So for the case where $d = 1$, we get an expected value of p^*, and for the case where $d = 0$ we obtain an expected value of p_i. Since $p_1 < p^* < p_2$, we have that the optimal decision rule for D relative to the strategy profile σ_1 must have $\delta(D \mid \mathbf{pa}_D)$ be 1 with probability 1, whereas the optimal decision rule for D relative to the strategy profile σ_2 must have $\delta(D \mid \mathbf{pa}_D)$ be 0 with probability 1. Since $\mathbf{pa}_D$ has positive probability in $\mathcal{M}[\sigma]$, we have shown that D strategically relies on D'.

Case (3). $1 < k < m$, that is, the two paths from D to U and from $\widehat{D'}$ to U intersect in the middle. This case is only slightly more complicated. In this case, Y_{k+1} has both Y_k and X_ℓ as parents. We define the CPD for Y_{k+1} as follows: Y_{k+1} depends only on the two parents X_ℓ and Y_k; its dependence on these two parents is as follows:

$$\Pr(Y_{k+1} = 1 \mid X_\ell = 1,\ Y_k = 1) = p^*, \qquad \Pr(Y_{k+1} = 1 \mid X_\ell = 0,\ Y_k = 1) = 1,$$
$$\Pr(Y_{k+1} = 1 \mid X_\ell = 1,\ Y_k = 0) = p^*, \qquad \Pr(Y_{k+1} = 1 \mid X_\ell = 0,\ Y_k = 0) = 0.$$

The utility node U now depends only on its parent Y_m. We have

$$U(Y_m = 1) = 1, \qquad U(Y_m = 0) = 0.$$

A straightforward analysis, similar to the one above, shows that, in the context $\mathbf{pa}_D$ and relative to the strategy profile σ_i, the expected utility of the decision d is p^* for $d = 1$ and p_i for $d = 0$. Hence, we again conclude that D strategically relies on D'.

At the beginning of the proof, we assumed that all the variables are binary. To extend the proof to variables with arbitrary domains, we simply choose one value in the domain of each variable and label it "1." Then, we replace all references in the proof to $X = 0$ (where X is some variable) with $X \neq 1$. The probability mass assigned to $X \neq 1$ in a CPD is uniformly distributed over the values in $dom(X) - \{1\}$. It is easy to check that the proof still holds. □

5.3. The relevance graph

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 5.4. 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 $D' \to D$ if and only if D' is s-reachable from D.[2]

The relevance graph for the Tree Killer example is shown in Fig. 8(a). By Theorem 5.1, if D relies on D', then D' is a parent of D in the 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. 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. 7, 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

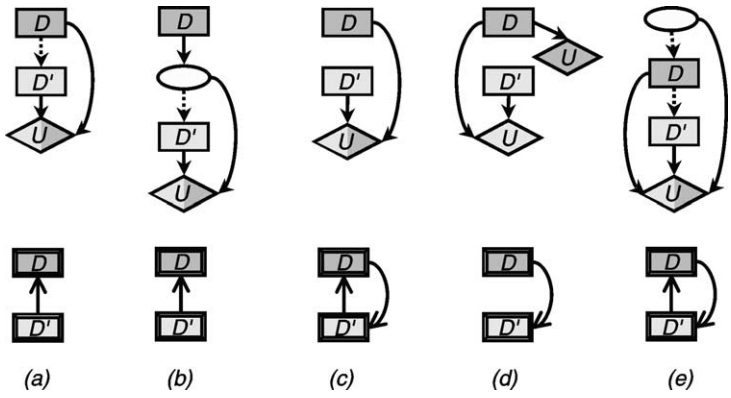

Fig. 7. Five simple MAIDs (top), and their relevance graphs (bottom). A two-color diamond represents a pair of utility nodes, one for each agent, with the same parents.

[2] The edges in this definition are the reverse of those in (Koller and Milch, 2001); the definition was changed to make the parent relationship more analogous to the parent relationship in BNs.

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 (a hypothetical parent of D) that runs through the chance node to the utility node.

6. Computing equilibria using divide and conquer

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 algorithms 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 will significantly improve the complexity of finding a global equilibrium. We begin by discussing the relatively simple case where the relevance graph is acyclic, then we extend our algorithm to handle MAIDs with cyclic relevance graphs.

6.1. Backward induction and acyclic relevance graphs

Our algorithm for acyclic relevance graphs is a generalization of existing backward induction algorithms for decision trees and perfect information games (Zermelo, 1913) and for influence diagrams (Cooper, 1988; Shachter, 1990; Shenoy, 1992; 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. 8(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*.

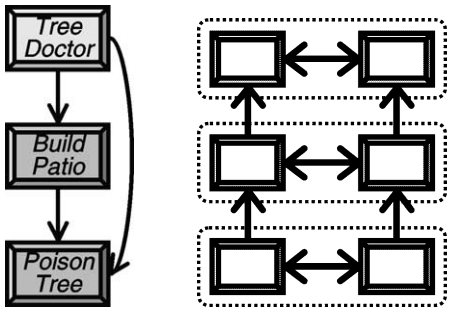

Fig. 8. Relevance graphs for (a) the Tree Killer example; (b) the Road example with $n = 6$.

In any acyclic relevance graph, we can construct a topological ordering of the decision nodes: an ordering $D_1, \ldots, D_n$ such that if D_i is s-reachable from D_j, then $i < j$. For instance, the relevance graph for the Tree Killer example has only one topological ordering: *TreeDoctor*, *BuildPatio*, *PoisonTree*. Then we can iterate over the decision nodes in this order, deriving an optimal decision rule for each node in turn. Each decision D_j relies only on the decisions that precede it in the order, and these will have already been processed by the time we have to select the decision rule for D_j. The formal description of the algorithm is as follows:

Algorithm 6.1.
Given a MAID $\mathcal{M}$ with an acyclic relevance graph, a topological ordering $D_1, \ldots, D_n$ of the relevance graph for $\mathcal{M}$.
1. *Let σ^0 be an arbitrary fully mixed strategy profile for $\mathcal{M}$.*
2. *For $i = 1$ through n:*
3. *Let δ be a decision rule for D_i that is optimal for σ^{i-1}.*
4. *Let $\sigma^i = (\sigma^{i-1}_{-D_i}, \delta)$.*
5. *Output σ^n as an equilibrium of $\mathcal{M}$.*

All the individual steps in this algorithm are trivial except for step 3: finding a decision rule δ for D_i that is optimal for σ^{i-1}. By Lemma 5.1, it is sufficient to find a δ such that for every instantiation $\mathbf{pa}_{D_i}$ of $Pa(D_i)$ where $P_{M[\sigma^{i-1}]}(\mathbf{pa}_{D_i}) > 0$, the probability distribution $\delta(D_i \mid \mathbf{pa}_{D_i})$ is a solution of:

$$\operatorname*{argmax}_{P^*} \sum_{d \in dom(D_i)} P^*(d) \sum_{U \in \mathcal{U}_{D_i}} \sum_{u \in dom(U)} P_{M[\sigma^{i-1}]}(u \mid d,\ \mathbf{pa}_{D_i}) \cdot u.$$

It is clear that in order to maximize the expression, it is sufficient to find a value $d^* \in dom(D_i)$ that maximizes:

$$\sum_{U \in \mathcal{U}_{D_i}} \sum_{u \in dom(U)} P_{M[\sigma^{i-1}]}(u \mid d,\ \mathbf{pa}_{D_i}) \cdot u.$$

Then we let P^* assign probability 1 to d^* and 0 to the other possible values of D_i. This analysis also shows that the resulting strategy profile σ^n is always a pure strategy profile.

To perform this computation, we can use BN inference to obtain the distribution over the parents of each utility node U given each combination of a parent instantiation $\mathbf{pa}_{D_i}$ and a value $d \in dom(D_i)$. From that, we can easily compute the expected utility.

To prove that this algorithm is correct, we must show that regardless of the fully mixed strategy profile σ^0 we start with, the final strategy profile σ^n is a Nash equilibrium for $\mathcal{M}$.

We begin with a lemma. Consider a MAID with just three decision nodes, where the topological ordering of the nodes in the relevance graph is D_1, D_2, D_3. Suppose δ is a decision rule for D_1 that is optimal for σ^0. By our construction of the ordering, neither D_2 nor D_3 is s-reachable from D_1. So changing σ^0 at either D_2 or D_3 will not affect the optimality of δ. But one might worry that changing σ^0 at *both* D_2 and D_3 (as we do when we derive σ^n in Algorithm 6.1) might cause δ to lose optimality. The following lemma shows that such a thing cannot happen: if we have a set of decisions none of which are individually relevant, then the entire set is not relevant.

Lemma 6.1. *Let σ be a strategy profile, D be a decision node, and δ be a decision rule for D that is optimal for σ. Let σ' be another strategy profile such that whenever $\sigma'(D') \neq \sigma(D')$, then D' is not s-reachable from D. Then there is some decision rule δ^* for D such that δ^* agrees with δ on all $\mathbf{pa} \in dom(Pa(D))$ where $P_{\mathcal{M}[\sigma]} > 0$, and δ^* is optimal for σ'.*

The proof of this lemma involves a straightforward modification of the proof of Theorem 5.1. Instead of considering a single decision node D', we consider the entire set $\mathcal{Z} = \{D' : \sigma'(D') \neq \sigma(D')\}$. Then we take advantage of the fact that Lemma 5.2 can apply to an entire set of nodes $\boldsymbol{Z}$.

Lemma 6.1 does not state that δ itself is optimal for σ'; it only asserts the existence of a δ^* that is appropriately similar to δ and is optimal for σ'. The shift from δ to δ^* will become inconvenient. However, the following lemma shows that we can avoid this shift if σ is fully mixed on the nodes where it differs from σ'.

Lemma 6.2. *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. Let σ' be another strategy profile such that whenever $\sigma'(D') \neq \sigma(D')$, then D' is not s-reachable from D. If σ is fully mixed on all D' where $\sigma'(D') \neq \sigma(D')$, then δ is also optimal for σ'.*

Proof. Since no decision node where σ and σ' differ is s-reachable from D, Lemma 6.1 tells us there is some decision rule δ^* for D that differs from δ only on parent instantiations that have zero probability in $\mathcal{M}[\sigma]$, and this δ^* is optimal for σ'. Since σ is fully mixed on every D' where it differs from σ', the set of $\mathbf{pa} \in dom(Pa(D))$ such that $P_{\mathcal{M}[\sigma]}(\mathbf{pa}) > 0$ is a superset of the set of $\mathbf{pa} \in dom(Pa(D))$ such that $P_{\mathcal{M}[\sigma']}(\mathbf{pa}) > 0$. So δ^* could only differ from δ on parent instantiations that have zero probability in both $\mathcal{M}[\sigma]$ and $\mathcal{M}[\sigma']$. But the distributions over D conditioned on parent instantiations that have zero probability in $\mathcal{M}[\sigma']$ cannot affect $\mathrm{EU}_a(\sigma')$ for any a. Since δ differs from δ^* only on these zero-probability parent instantiations, and δ^* is optimal for σ', δ must also be optimal for σ'. □

Note that neither of the preceding results depends on the relevance graph being acyclic. Indeed, we require acyclicity only to ensure that there exists a topological sort of the relevance graph that we can provide as input to Algorithm 6.1. At this point, we can finally prove the correctness of this algorithm.

It is easy, using Lemma 6.2, to show that each individual decision rule derived in the algorithm is optimal for the final strategy profile σ^n. That is, no agent can gain by deviating from σ^n on a single decision node. However, to guarantee the Nash equilibrium property, we must show that the agent cannot gain by deviating at any combination of his decision nodes simultaneously. In our Tree Killer example, we know that Alice cannot improve her decision rule at *BuildPatio*, nor can she improve her decision rule at *PoisonTree* given her decision rule at *BuildPatio*, but we need to show that she cannot improve her utility by deviating at both decisions simultaneously. This is what we prove in the following theorem.

Theorem 6.1. *Let $\mathcal{M}$ be a MAID whose relevance graph is acyclic, and let $D_1, \ldots, D_n$ be a topological ordering of the relevance graph for $\mathcal{M}$. Then the strategy profile σ^n produced by running Algorithm* 6.1 *with $\mathcal{M}$ and the ordering $D_1, \ldots, D_n$ as inputs is a Nash equilibrium for $\mathcal{M}$.*

Proof. To show that σ^n is a Nash equilibrium, we must show that for all agents $a \in \mathcal{A}$ and all other strategies σ'_a for a:

$$\mathrm{EU}_a(\sigma^n) \geqslant \mathrm{EU}_a\big((\sigma^n_{-a}, \sigma'_a)\big).$$

Consider any agent a. We proceed by induction on the number k of decisions where σ_a and σ'_a differ. If $k = 0$, then $\sigma^n_a = \sigma'_a$, so obviously $\mathrm{EU}_a(\sigma^n) = \mathrm{EU}_a((\sigma^n_{-a}, \sigma'_a))$. As an inductive hypothesis, suppose that whenever σ'_a differs from σ^n_a on k or fewer nodes, $\mathrm{EU}_a(\sigma^n) \geqslant \mathrm{EU}_a((\sigma^n_{-a}, \sigma'_a))$.

Now suppose σ'_a differs from σ^n_a on $k+1$ nodes. Let j be the smallest index in $\{1, \ldots, n\}$ such that $\sigma'_a(D_j) \neq \sigma^n_a(D_j)$. Let $\delta = \sigma^n_a(D_j)$. By construction in Algorithm 6.1, δ is optimal for σ^{j-1}. Note that σ^{j-1} differs from $(\sigma^n_{-a}, \sigma'_a)$ only on $D_j, \ldots, D_n$. None of these nodes are s-reachable from D_j, because none of the nodes that come after D_j in the ordering are s-reachable from D_j, and neither is D_j itself. Also, σ^{j-1} agrees with σ^0 on $D_j, \ldots, D_n$, so it is fully mixed on these nodes. Thus, by Lemma 6.2, δ is also optimal for $(\sigma^n_{-a}, \sigma'_a)$. In particular, δ yields at least as much expected utility as $\sigma'_a(D_j)$ in $\mathcal{M}[\sigma^n_{-a}, (\sigma'_a)_{-D_j}]$. So:

$$\mathrm{EU}_a\big(\sigma^n_{-a}, (\sigma'_a)_{-D_j}, \delta\big) \geqslant \mathrm{EU}_a\big(\sigma^n_{-a}, \sigma'_a\big).$$

But the strategy $((\sigma'_a)_{-D_j}, \delta)$ differs from σ^n_a at only k decision nodes, so by the inductive hypothesis:

$$\mathrm{EU}_a(\sigma^n) \geqslant \mathrm{EU}_a\big(\sigma^n_{-a}, (\sigma'_a)_{-D_j}, \delta\big).$$

So by transitivity:

$$\mathrm{EU}_a(\sigma^n) \geqslant \mathrm{EU}_a\big(\sigma^n_{-a}, \sigma'_a\big). \qquad \square$$

Thus, we have shown that our induction process is guaranteed to find a Nash equilibrium. It is easy to show that it generalizes the notion of backward induction in perfect information games.

Definition 6.1. A MAID $\mathcal{M}$ is said to have *perfect information* relative to an ordering $D_1, \ldots, D_n$ if, for all j and $i < j$, we have that $D_i, Pa(D_i) \subseteq Pa(D_j)$.

This definition of perfect information does not require that all of the chance variables are observed, but only that any chance variable that is observed at one point in the game is also observed at subsequent points.

Lemma 6.3. *If D_i and D_j are two decisions such that $D_i, Pa(D_i) \subseteq Pa(D_j)$, then D_i is not s-reachable from D_j.*

Proof. By Definition 5.3, we must show that if a new parent $\widehat{D_i}$ were added to D_i, there would not be an active path from $\widehat{D_i}$ to any $U \in \mathcal{U}_{D_j}$ given D_j and $Pa(D_j)$. By assumption, $D_i, Pa(D_i) \subseteq Pa(D_j)$. Hence, D_i and all its parents are observed. Now consider any path starting at $\widehat{D_i}$. Since D_i is the only neighbor of $\widehat{D_i}$, the path must continue through D_i, which is observed. Thus, this path can be active only if D_i is at the middle of a v-structure. But the path would need to continue through some node in $Pa(D_i)$. These parents are all observed as well, and thus block the path. Hence, there is no active path starting at $\widehat{D_i}$, which means D_i is not s-reachable from D_j. □

Corollary 6.1. *If a MAID $\mathcal{M}$ has perfect information relative to some ordering $D_1, \ldots, D_n$, then the relevance graph for the MAID is acyclic, and $D_n, \ldots, D_1$ is a topological ordering for the graph.*

Thus, our backward induction algorithm ("backward" because the topological ordering of the relevance graph is the reverse of the perfect information ordering) applies to all perfect information games. However, we obtain acyclic relevance graphs in a wider range of situations. For example, the relevance graph of the Tree Killer example is acyclic, although the game does not have perfect information.

6.2. Divide and conquer in cyclic relevance graphs

Although acyclic relevance graphs arise even in games of imperfect information, 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. 7(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, and Algorithm 6.1 does not apply.

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

Example 6.1. Consider the relevance graph for the Road example, shown in Fig. 8(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 southernmost 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 southernmost pair, the decision variables can be treated as chance nodes, and we can proceed to compute an equilibrium for the next pair.

We can formalize this intuition using the following definition:

Definition 6.2. 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. 8(b).

We can find the maximal SCCs of a relevance graph in linear time using an algorithm based on depth-first search (see (Cormen et al., 1990, Section 23.5)). Then we can construct a *component graph* whose nodes are the maximal SCCs of the relevance graph. 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 (Cormen et al., 1990). Thus, we can find a topological ordering $\mathcal{C}_1, \ldots, \mathcal{C}_m$ over the maximal SCCs of the relevance graph, such that if some element of $\mathcal{C}_i$ is s-reachable from some element of $\mathcal{C}_j$, then $i < j$.

Based on this definition, and our intuition above, we can now provide a divide-and-conquer algorithm for computing Nash equilibria in general MAIDs.

Algorithm 6.2.

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 $i = 1$ through m:*
3. *Let τ be a partial strategy profile for $\mathcal{C}_i$ that is a Nash equilibrium in $\mathcal{M}[\sigma^{i-1}_{-\mathcal{C}_i}]$.*
4. *Let $\sigma^i = (\sigma^{i-1}_{-\mathcal{C}_i}, \tau)$.*
5. *Output σ^m as an equilibrium of $\mathcal{M}$.*

The algorithm iterates 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). 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 4, and use a standard game-solving algorithm (McKelvey and McLennan, 1996) as a subroutine. We assume in our correctness proofs that the game solution subroutine returns an exact equilibrium to the subgame.

It is important to understand exactly why the game trees passed to the game solver in step 3 of Algorithm 6.2 may be smaller than a game tree for the entire MAID. The induced MAID $\mathcal{M}[\sigma^{i-1}_{-\mathcal{C}_i}]$ is the same as the original MAID, except that all decision nodes outside of $\mathcal{C}_i$ have been converted to chance nodes. Recall from Section 4 that when we construct the game tree for a MAID, we only need to include the decision nodes and any chance nodes that are observed at those decisions. So the game tree for $\mathcal{M}[\sigma^{i-1}_{-\mathcal{C}_i}]$ splits only on the decision variables in $\mathcal{C}_i$ and their parents. In computing the probabilities of nature's moves and the payoffs at the leaves, we sum out the variables that are not included in the tree. Standard Bayesian network inference algorithms allow us to do these computations efficiently, as illustrated by our experimental results in Section 6.3.

Now we begin to prove the correctness of Algorithm 6.2. In the ith iteration of the main loop in this algorithm, the game solution subroutine gives us a partial strategy profile τ for $\mathcal{C}_i$ that is a Nash equilibrium in $\mathcal{M}[\sigma^{i-1}_{-\mathcal{C}_i}]$. Thus, for each agent a, the restriction of τ to $\mathcal{C}_i \cap \mathcal{D}_a$ (which we will call τ_a) is optimal for $(\sigma^{i-1}_{-\mathcal{C}_i}, \tau)$. Is τ_a also optimal for the final strategy profile σ^m? We can conclude from Lemma 6.2 that each individual decision rule in τ_a is still optimal for σ^m. But so far, our lemmas about how optimality is preserved when strategy profiles change have only dealt with the optimality of a single decision rule. Might the change from $(\sigma^{i-1}_{-\mathcal{C}_i}, \tau)$ to σ^m give agent a an incentive to deviate from τ_a on several decision nodes simultaneously?

We can answer this question in the negative if we assume that for each agent a, the relevance graph restricted to $\mathcal{D}_a$ (agent a's decision nodes) is acyclic. This condition is implied by the standard assumption of *perfect recall*—that agents never forget their previous actions or observations. More formally:

Definition 6.3. 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 $Pa(D_i) \subset Pa(D_j)$.

From Lemma 6.3, we can conclude the following:

Corollary 6.2. *If an agent a has perfect recall with respect to an ordering $D_1, \ldots, D_n$, then the relevance graph restricted to $\mathcal{D}_a$ is acyclic, and $D_n, \ldots, D_1$ is a topological ordering of the relevance graph restricted to $\mathcal{D}_a$.*

We now consider a set $\mathcal{C}_a$ of decision nodes belonging to a single agent. We show that if none of the decision nodes in another set $\mathcal{E}$ are s-reachable from any of the nodes in $\mathcal{C}_a$, then changing the decision rules for $\mathcal{E}$ cannot give the agent an incentive to change her decision rules on any set of nodes in $\mathcal{C}_a$. The difference between this result and Theorem 6.1 is subtle. In Theorem 6.1, we showed that Algorithm 6.1 yields a strategy σ^n_a for each agent a such that a has no incentive to deviate on any set of decision nodes. The following lemma is not about generating such strategies, but about how their optimality is preserved when irrelevant decision rules are changed.

Lemma 6.4. *Let σ be a strategy profile for a MAID $\mathcal{M}$, and let τ be a strategy for a set $\mathcal{C}_a$ of decision nodes belonging to a single agent a in $\mathcal{M}$, such that τ is optimal for σ. Assume that the relevance graph of $\mathcal{M}$ restricted to $\mathcal{D}_a$ is acyclic. Let σ' be another strategy profile for $\mathcal{M}$ that differs from σ only on a set of nodes $\mathcal{E}$ and possibly on $\mathcal{C}_a$ itself. If σ is fully mixed on each node in $\mathcal{E}$, and no node in $\mathcal{E}$ is s-reachable from any node in $\mathcal{C}_a$, then τ is also optimal for σ'.*

Proof. We begin by dispensing with the question of whether σ' differs from σ on some nodes in $\mathcal{C}_a$. Let $\widetilde{\sigma}$ be a strategy profile that agrees with σ' everywhere except on $\mathcal{C}_a$, and agrees with σ on $\mathcal{C}_a$. Then the partial strategy profile $\widetilde{\sigma}_{-\mathcal{C}_a}$ is the same as $\sigma'_{-\mathcal{C}_a}$. So to show that τ is optimal in $\mathcal{M}[\sigma'_{-\mathcal{C}_a}]$, it suffices to show that τ is optimal in $\mathcal{M}[\widetilde{\sigma}_{-\mathcal{C}_a}]$. So we have reduced our task to showing that the lemma is true when σ and σ' differ only on $\mathcal{E}$, and agree on $\mathcal{C}_a$.

We proceed by induction on the number of decision nodes in $\mathcal{C}_a$. If $|\mathcal{C}_a| = 1$, then the lemma reduces to Lemma 6.2. For larger SCCs, we cannot simply use Lemma 6.2, as we have no guarantee that the strategy profile within $\mathcal{C}_a$ is fully mixed. We therefore need a somewhat more elaborate inductive proof.

As an inductive hypothesis, assume the lemma holds for $|\mathcal{C}_a| = m$, and we will prove that it holds for $|\mathcal{C}_a| = m + 1$. Because the relevance graph restricted to $\mathcal{D}_a$ is acyclic, it has a topological ordering; let $D_1, \ldots, D_{m+1}$ be the restriction of this topological ordering to $\mathcal{C}_a$. We must show that for any strategy τ' over $D_1, \ldots, D_{m+1}$:

$$\mathrm{EU}_a(\sigma'_{-\mathcal{C}_a}, \tau) \geqslant \mathrm{EU}_a\big(\sigma'_{-\mathcal{C}_a}, \tau'\big). \tag{9}$$

Consider the decision rule δ_1 that τ assigns to D_1. Since τ is optimal for σ, we know δ_1 is optimal for $(\sigma_{-\mathcal{C}_a}, \tau)$. Note that $(\sigma'_{-\mathcal{C}_a}, \tau')$ may differ from $(\sigma_{-\mathcal{C}_a}, \tau)$ on $\mathcal{E}$ and on any of $D_1, \ldots, D_{m+1}$. But because $D_1, \ldots, D_{m+1}$ is part of a topological ordering of the relevance graph, none of the subsequent decision nodes $D_2, ..., D_{m+1}$ are s-reachable from D_1. Also, by hypothesis, no node in $\mathcal{E}$ is s-reachable from D_1. So by Lemma 6.1, there is some decision rule δ_1^* for D_1 that agrees with δ_1 on all $\mathbf{pa} \in \mathit{dom}(\mathit{Pa}(D_1))$ where $P_{\mathcal{M}[(\sigma_{-\mathcal{C}_a}, \tau)]}(\mathbf{pa}) > 0$, and this δ_1^* is optimal for $(\sigma'_{-\mathcal{C}_a}, \tau')$.

Let $\widetilde{\mathcal{C}}_a = \{D_2, \ldots, D_{m+1}\}$. Let $\widetilde{\tau}$ and $\widetilde{\tau}'$ be the restrictions of τ and τ', respectively, to $\widetilde{\mathcal{C}}_a$. Since δ_1^* differs from δ_1 only on parent instantiations that have zero probability in $\mathcal{M}[\sigma_{-\mathcal{C}_a}, \tau]$, it follows that:

$$\mathrm{EU}_a\big(\sigma_{-\mathcal{C}_a}, \widetilde{\tau}, \delta_1^*\big) = \mathrm{EU}_a(\sigma_{-\mathcal{C}_a}, \widetilde{\tau}, \delta_1) = \mathrm{EU}_a(\sigma_{-\mathcal{C}_a}, \tau). \tag{10}$$

Also, since σ is fully mixed on all nodes where it differs from σ', the set of parent instantiations of D_1 that have nonzero probability in $\mathcal{M}[\sigma_{-\mathcal{C}_a}, \tau]$ is a superset of the set of parent instantiations that have nonzero probability in $\mathcal{M}[\sigma'_{-\mathcal{C}_a}, \tau]$. So δ_1^* agrees with δ_1 on all parent instantiations that have nonzero probability in $\mathcal{M}[\sigma'_{-\mathcal{C}_a}, \tau]$. Therefore:

$$\mathrm{EU}_a\big(\sigma'_{-\mathcal{C}_a}, \tau\big) = \mathrm{EU}_a\big(\sigma'_{-\mathcal{C}_a}, \widetilde{\tau}, \delta_1\big) = \mathrm{EU}_a\big(\sigma'_{-\mathcal{C}_a}, \widetilde{\tau}, \delta_1^*\big). \tag{11}$$

This is the first step toward proving (9).

For the next step, we must show that $\widetilde{\tau}$ is optimal for $(\sigma_{-D_1}, \delta_1^*)$. To see this, recall that τ is optimal for σ, which implies:

$$\mathrm{EU}_a(\sigma_{-\mathcal{C}_a}, \tau) \geqslant \mathrm{EU}_a\big(\sigma_{-\mathcal{C}_a}, \widetilde{\tau}'', \delta_1^*\big)$$

for any strategy $\widetilde{\tau}''$ over $\widetilde{\mathcal{C}}_a$. Using the equality in (10), we find that for any $\widetilde{\tau}''$:

$$\mathrm{EU}_a\big(\sigma_{-\mathcal{C}_a}, \widetilde{\tau}, \delta_1^*\big) \geqslant \mathrm{EU}_a\big(\sigma_{-\mathcal{C}_a}, \widetilde{\tau}'', \delta_1^*\big).$$

So $\widetilde{\tau}$ is indeed optimal for $(\sigma_{-D_1}, \delta_1^*)$.

Note that $\widetilde{\mathcal{C}}_a$ contains only m variables. Hence, by the inductive hypothesis, $\widetilde{\tau}$ is also optimal for $(\sigma'_{-D_1}, \delta_1^*)$. In particular, $\widetilde{\tau}$ provides no worse an expected utility for a in $\mathcal{M}[\sigma'_{-D_1}, \delta_1^*]$ than $\widetilde{\tau}'$. That is:

$$\mathrm{EU}_a\big(\sigma'_{-\mathcal{C}_a}, \widetilde{\tau}, \delta_1^*\big) \geqslant \mathrm{EU}_a\big(\sigma'_{-\mathcal{C}_a}, \widetilde{\tau}', \delta_1^*\big).$$

Now we finally use the fact that δ_1^* is optimal for $(\sigma'_{-\mathcal{C}_a}, \tau')$, which tells us:

$$\mathrm{EU}_a\big(\sigma'_{-\mathcal{C}_a}, \widetilde{\tau}', \delta_1^*\big) \geqslant \mathrm{EU}_a\big(\sigma'_{-\mathcal{C}_a}, \tau'\big).$$

Applying transitivity to the last two inequalities, we can conclude that:

$$\mathrm{EU}_a\big(\sigma'_{-\mathcal{C}_a}, \widetilde{\tau}, \delta_1^*\big) \geqslant \mathrm{EU}_a\big(\sigma'_{-\mathcal{C}_a}, \tau'\big).$$

Then applying (11) to the left-hand side of this inequality yields the desired inequality in (9). □

Given this lemma, we can prove the correctness of Algorithm 6.2. The proof follows the same lines as the proof of Theorem 6.1.

Theorem 6.2. *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* 6.2 *with $\mathcal{M}$ and $\mathcal{C}_1, \ldots, \mathcal{C}_m$ as inputs is a Nash equilibrium for $\mathcal{M}$.*

Proof. We must show that for any agent a and any alternative strategy σ'_a for $\mathcal{D}_a$, $\mathrm{EU}_a(\sigma^m) \geqslant \mathrm{EU}_a(\sigma^m_{-a}, \sigma'_a)$. The difference between what we must prove here and what we proved in Lemma 6.4 is that here an agent could deviate on several SCCs at once. We proceed by induction on the number of SCCs where σ^m_a and σ'_a differ. The base case is where they differ on zero SCCs; then $\sigma^m_a = \sigma'_a$ and $\mathrm{EU}_a(\sigma^m) = \mathrm{EU}_a(\sigma^m, \sigma'_a)$.

As an inductive hypothesis, suppose that whenever σ'_a differs from σ^m_a on k SCCs, $\mathrm{EU}_a(\sigma^m) \geqslant \mathrm{EU}_a(\sigma^m_{-a}, \sigma'_a)$. Now suppose σ'_a differs from σ^m_a on $k+1$ SCCs. Let $\mathcal{C}_j$ be the first SCC in the ordering where σ'_a and σ^m_a are different. Let τ be the restriction of σ^m to $\mathcal{C}_j$. This partial strategy profile τ for $\mathcal{C}_j$ is chosen in Algorithm 6.2 to be a Nash equilibrium in $\mathcal{M}[\sigma^{j-1}_{-\mathcal{C}_j}]$. Let $\mathcal{C}_{j,a} = \mathcal{C}_j \cap \mathcal{D}_a$ be the set of agent a's decisions in $\mathcal{C}_j$, and let τ_a be agent a's strategy in $\mathcal{C}_{j,a}$, i.e., the restriction of τ to $\mathcal{C}_{j,a}$. It follows by Definition 3.5 that τ_a is optimal for $(\sigma^{j-1}_{-\mathcal{C}_j}, \tau)$. Since the next step in the algorithm is to let $\sigma^j = (\sigma^{j-1}_{-\mathcal{C}_j}, \tau)$, we know τ_a is optimal for σ^j.

By construction in the algorithm, σ^m agrees with σ^j on $\mathcal{C}_1, \ldots, \mathcal{C}_j$. Also, because $\mathcal{C}_j$ is the first SCC where σ'_a differs from σ_a, we know that $(\sigma^m_{-a}, \sigma'_a)$ agrees with σ^m on $\mathcal{C}_1, \ldots, \mathcal{C}_{j-1}$. Although $(\sigma^m_{-a}, \sigma'_a)$ differs from σ^m on $\mathcal{C}_{j,a}$, it agrees with σ^m on the rest of $\mathcal{C}_j$ (which consists of nodes controlled by other agents). Let $\mathcal{E} = \mathcal{C}_{j+1}, \ldots, \mathcal{C}_m$, so that

$(\sigma^m_{-a}, \sigma'_a)$ differs from σ^j only on $\mathcal{E}$ and $\mathcal{C}_{j,a}$. By the properties of the topological ordering of SCCs in the relevance graph, none of the nodes in $\mathcal{E}$ are s-reachable from any node in $\mathcal{C}_{j,a}$. Also, σ^j agrees with σ^0 on $\mathcal{E}$, so it is fully mixed on these nodes. Thus, by Lemmas 6.2 and 6.4, τ_a is also optimal for $(\sigma^m_{-a}, \sigma'_a)$.

In particular, τ_a yields at least as much expected utility as the restriction of σ'_a to $\mathcal{C}_{j,a}$ in $\mathcal{M}[\sigma^m_{-a}, (\sigma'_a)_{-\mathcal{C}_{j,a}}]$. So:

$$\mathrm{EU}_a\big(\sigma^m_{-a}, (\sigma'_a)_{-\mathcal{C}_{j,a}}, \tau_a\big) \geqslant \mathrm{EU}_a\big(\sigma^m_{-a}, \sigma'_a\big).$$

But, as τ_a is the restriction of σ^m_a to $\mathcal{C}_j$, the strategy $((\sigma'_a)_{-\mathcal{C}_{j,a}}, \tau_a)$ differs from σ^m_a on only k SCCs. Hence, by the inductive hypothesis:

$$\mathrm{EU}_a(\sigma^m) \geqslant \mathrm{EU}_a\big(\sigma^m_{-a}, (\sigma'_a)_{-\mathcal{C}_{j,a}}, \tau_a\big).$$

So by transitivity:

$$\mathrm{EU}_a(\sigma^m) \geqslant \mathrm{EU}_a\big(\sigma^m_{-a}, \sigma'_a\big)$$

which is the desired result. □

Thus, we have shown that we can find a Nash equilibrium for a complex game by breaking it up into a set of interacting smaller games. Each subgame can be solved in sequence, relying on the solution to the previous games.

Our algorithm finds a single Nash equilibrium; if the game-solving subroutine finds multiple Nash equilibria for an induced MAID $\mathcal{M}[\sigma^{i-1}_{-\mathcal{C}_i}]$, our algorithm selects one arbitrarily. If it selected a different equilibrium for this small game, we would obtain a global equilibrium with different decision rules for $\mathcal{C}_i$ and (possibly) for all subsequent SCCs. We can visualize the choices of equilibria for the induced MAIDs as generating a tree of possible executions of the algorithm, with global equilibria at the leaves. Of course, since a game may have an infinite number of equilibria, we generally cannot construct this tree in practice.

Also, a MAID may have some equilibria that our algorithm cannot produce, no matter how it chooses equilibria for the induced MAIDs. For instance, the leader-follower MAID in Example 5.1 has an equilibrium in which Alice goes north with probability 1, and Bob goes north with probability 1 regardless of where he sees Alice go. Applying Algorithm 6.2 to this MAID, we find that $\mathcal{C}_1$ consists of Bob's decision, and $\mathcal{C}_2$ consists of Alice's decision (we could also solve this MAID with Algorithm 6.1). The algorithm starts with a fully-mixed strategy profile: in particular, σ^0 assigns Alice a fully-mixed decision rule. So in $\mathcal{M}[\sigma^0_{-\mathcal{C}_1}]$, the only Nash equilibrium (optimal strategy) is for Bob to go north if Alice goes north, and south if she goes south. Thus, our algorithm cannot find the nonsubgame-perfect equilibrium where Bob goes north regardless of Alice's action.

6.3. Experimental results

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. 4: 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. 8(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 strategic-form game matrix with about 4.7×10^{27} entries (one for each pure strategy profile).

Solving the Road game either in its extensive form or in the normal form is infeasible even for $n = 4$. By contrast, Fig. 9 shows the computational cost of our divide-and-conquer algorithm as n grows: the time required grows approximately linearly with n. 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. Our algorithm ends up generating a sequence of $n/2$ small games, each with two decision variables. We convert each of the induced MAIDs into a small game tree, and use the game solver Gambit (2000) to solve it. Computing the payoffs in one of the small game trees requires Bayesian network inference, but the observed linear performance of our algorithm implies that in the Road MAID, the time needed to construct a single game tree remains constant as n increases.

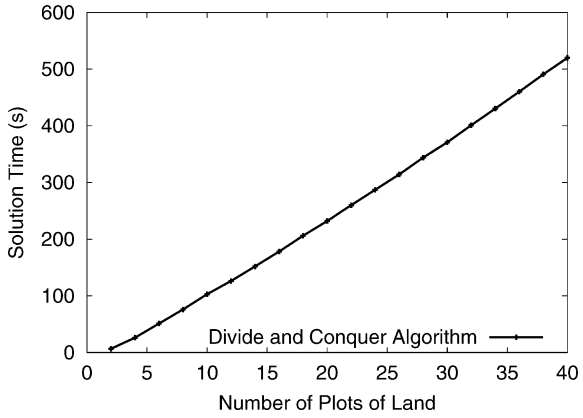

Fig. 9. Performance results for the Road example.

7. Related work

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. Brown (1999) suggests the use of influence diagrams for modeling various game-theoretic and economic scenarios, but does not discuss either the formal foundations or the algorithmic issues. Zhang et al. (1992, 1994) and Nilsson and Lauritzen (2000, 2001) discuss IDs where the perfect recall assumption does not hold, although there is only one agent (or all the agents share a common utility function). Both series of papers propose graphical criteria for determining whether the optimal decision rule for a given node depends on the decision rules for any other nodes—in our terminology, whether the given node strategically relies on any other nodes. These criteria are similar to s-reachability, but are not complete in the sense of Theorem 5.2. The papers then derive algorithms similar to our Algorithm 6.1. However, they do not deal with multiple competing agents, nor with cyclic relevance graphs.

Several other papers use ID-like representations for multi-agent scenarios. Poole's (1997) independent choice logic can represent the same conditional independence assumptions as a MAID, and also allows compact representations of CPDs and decision rules. Suryadi and Gmytrasiewicz (1999) use influence diagrams as a framework for learning in multi-agent systems. The models used by Milch and Koller (2000) for reasoning about agents' beliefs and decisions are really MAIDs, although they do not use that term. Penalva et al. (2002) introduce influence opportunity diagrams (IODs), which are MAIDs without utility nodes. They show that if the IODs for two games satisfy a certain graphical equivalence condition, then the two games are *empirically equivalent* in the sense that it is impossible to tell which game is being played just by observing the probability distribution over outcomes. However, none of these papers deal with algorithms for finding Nash equilibria.

LaMura has proposed such algorithms in his work on game networks (LaMura, 2000), which are like MAIDs except that they represent multiplicative rather than additive decompositions of the agents' utility functions. LaMura 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. Also, it does not allow a backward induction process for decisions that are not mutually independent. An interesting aspect of LaMura's algorithm is that, to find equilibria for a part of the game, it searches for solutions to equations constructed directly from the game network. We should be able to apply a similar technique to the induced MAIDs constructed in Algorithm 6.2, avoiding the use of game trees as an intermediate representation.

Other work on finding Nash equilibria efficiently uses the *graphical games* model introduced by Kearns et al. (2001) and Littman et al. (2001). They deal with games where each agent makes only a single binary-valued decision, and there are no observations. The game is represented as an undirected graph with a node for each agent: an agent's utility is determined by his own decision and the decisions of his neighbors in the graph. Graphical games are essentially MAIDs with a single decision and utility node for each agent, and no information edges. Each agent's utility depends on his own decision, and on that of his

neighbors in the graph. The relevance graph for such a scenario has two edges (one in each direction) between each pair of neighboring decisions, and thus consists of one large SCC. So our divide-and-conquer algorithm is not helpful in such scenarios. However, Kearns et al. (2001) propose several exact and approximate algorithms for finding Nash equilibria efficiently when the graphical game is a tree, and Vickrey and Koller (2002) describe a set of approximate algorithms for general graphical games. It is an open question whether these algorithms can be extended to more general MAIDs, where agents have observations and make multiple decisions.

8. 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.

This work opens the door to a variety of possible extensions. One obvious direction is to relate MAIDs to existing concepts in game theory, particularly equilibrium refinements. It is fairly straightforward to show that the solution found by our algorithm in the case of acyclic relevance graphs is a *perfect Bayesian equilibrium* (Fudenberg and Tirole, 1991); it would be interesting to show an analogous result for the more general case.

Another direction relates to additional structure that is revealed by the notion of strategic relevance. In particular, even if the relevance graph is cyclic, 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 computation gain.

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). In this case, we can use d-separation to identify irrelevant parents of a given decision node in time linear

in the number of variables. We can use a similar technique in MAIDs, allowing us to eliminate some parents of the decision node. At the level of the associated game tree, this process results in collapsing of several information sets into one (a process called *deflation* in (Dalkey, 1953; Okada, 1987)). The structure of the MAID allows us to detect cases when this process can be executed without any loss to the agents.

However, the multi-agent case also raises subtleties that are absent in the single-agent case. In this case, an observed variable that does not directly influence one agent's payoff might nevertheless be relevant, if another agent conditions his behavior on this variable. Maskin and Tirole (1997) provide a definition of payoff-relevant events in the multi-agent setting, and define a *Markov perfect equilibrium* as a perfect equilibrium in which each agent's decision rules are conditioned only on payoff-relevant events. Maskin and Tirole also provide an algorithm for determining whether an event is payoff-relevant. This algorithm involves comparing the utilities of all outcomes in two sub-trees of a game tree, to determine whether the utility functions in the two sub-trees are equivalent. Thus, in a symmetric tree, it requires examining an exponential number of outcomes. We have preliminary results indicating that we can determine sets of payoff-relevant events (variables) in a MAID in polynomial time, using an extension of the standard algorithms for influence diagrams. We plan to pursue this issue in future work.

Acknowledgments

This work was supported by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the Office of Naval Research under Grant N00014-00-1-0637, and by Air Force contract F30602-00-2-0598 under DARPA's TASK program. The second author was also supported by a University of California Microelectronics Fellowship.

References

Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D., 1996. Context-specific independence in Bayesian networks. In: Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96), pp. 115–123.

Brown, P.C., 1999. Influence diagram to model and classify game theoretic problems. Mimeo, also presented at the 11th International Conference on Game Theory, 2000.

Cooper, G.F., 1988. A method for using belief networks as influence diagrams. In: Proceedings of the Fourth Workshop on Uncertainty in Artificial Intelligence (UAI), pp. 55–63.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., 1990. Introduction to Algorithms. MIT Press.

Dalkey, N., 1953. Equivalence of information patterns and essentially determinate games. Ann. Math. Stud. 28, 217–243.

Fudenberg, D., Tirole, J., 1991. Game Theory. MIT Press.

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

Geiger, D., Pearl, J., 1990. On the logic of causal models. In: Uncertainty in Artificial Intelligence 4, pp. 136–147.

Geiger, D., Verma, T., Pearl, J., 1990. Identifying independence in Bayesian networks. Networks 20, 507–534.

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

Jensen, F., Jensen, F.V., Dittmer, S.L., 1994. From influence diagrams to junction trees. In: Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94), pp. 367–373.

Kearns, M., Littman, M.L., Singh, S., 2001. Graphical models for game theory. In: Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI 2001), pp. 253–260.

Keeney, R.L., Raiffa, H., 1976. Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Wiley.

Koller, D., Megiddo, N., von Stengel, B., 1994. Fast algorithms for finding randomized strategies in game trees. In: Proceedings of the 26th ACM Symposium on Theory of Computing (STOC-94), pp. 750–759.

Koller, D., Milch, B., 2001. Multi-agent influence diagrams for representing and solving games. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01), pp. 1027–1034.

LaMura, P., 2000. Game networks. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI 2000), pp. 335–342.

Lauritzen, S.L., Nilsson, D., 2001. Representing and solving decision problems with limited information. Management Sci. 47 (9), 1235–1251.

Lauritzen, S.L., Spiegelhalter, D.J., 1988. Local computations with probabilities on graphical structures and their application to expert systems. J. Roy. Statist. Soc. B 50 (2), 157–224.

Littman, M.L., Kearns, M., Singh, S., 2001. An efficient exact algorithm for singly connected graphical games. In: Advances in Neural Information Processing Systems (NIPS) 14.

Maskin, E., Tirole, J., 1997. Markov perfect equilibrium I: Observable actions. Working Paper 1799, Harvard Institute for Economic Research (HIER).

McKelvey, R.D., McLennan, A., 1996. Computation of equilibria in finite games. In: Handbook of Computational Economics, Vol. 1. Elsevier, Amsterdam, pp. 87–142.

Meek, C., 1995. Strong completeness and faithfulness in Bayesian networks. In: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-95), pp. 411–418.

Milch, B., Koller, D., 2000. Probabilistic models for agents' beliefs and decisions. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI 2000).

Nash, J., 1950. Equilibrium points in n-person games. Proc. Nat. Acad. Sci. USA 36, 48–49.

Nilsson, D., Lauritzen, S.L., 2000. Evaluating influence diagrams with LIMIDs. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI 2000), pp. 436–445.

Okada, A., 1987. Complete inflation and perfect recall in extensive games. Int. J. Game Theory 16 (2), 85–91.

Pearl, J., 1988. Probabilistic Reasoning in Intelligent Systems. Kaufmann, San Francisco.

Penalva-Zuasti, J., Ryall, M.D., 2002. Causal assessment in finite extensive-form games. Games Econ. Behav. In press.

Poole, D., 1997. The independent choice logic for modelling multiple agents under uncertainty. Artificial Intelligence 94 (1–2), 5–56.

Romanovskii, I.V., 1962. Reduction of a game with complete memory to a matrix game. Soviet Math. 3, 678–681.

Shachter, R.D., 1986. Evaluating influence diagrams. Oper. Res. 34, 871–882.

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

Shachter, R.D., 1998. Bayes-ball: The rational pastime. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98), pp. 480–487.

Shenoy, P.P., 1992. Valuation-based systems for Bayesian decision analysis. Oper. Res. 40, 463–484.

Suryadi, D., Gmytrasiewicz, P.J., 1999. Learning models of other agents using influence diagrams. In: Proceedings of the Seventh International Conference on User Modeling (UM-99), pp. 223–232.

Verma, T., Pearl, J., 1990. Causal networks: semantics and expressiveness. In: Uncertainty in Artificial Intelligence 4, pp. 69–76.

Vickrey, D., Koller, D., 2002. Multi-agent algorithms for solving graphical games. In: Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02), pp. 345–351.

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

Zhang, N.L., Poole, D., 1992. Stepwise-decomposable influence diagrams. In: Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning (KR-92), pp. 141–152.

Zhang, N.L., Qi, R., Poole, D., 1994. A computational theory of decision networks. Int. J. Approx. Reason. 11 (2), 83–158.