Artificial Intelligence 294 (2021) 103431

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

An integrated approach to solving influence diagrams and finite-horizon partially observable decision processes

Eric A. Hansen

Dept. of Computer Science and Eng., Mississippi State University, MS 39762, USA

ARTICLE INFO

Article history:
Received 9 October 2019
Received in revised form 2 August 2020
Accepted 23 November 2020
Available online 2 December 2020

Keywords:
Influence diagram
Variable elimination
Partially observable Markov decision process
Dynamic programming
Decision-theoretic planning

ABSTRACT

We show how to integrate a variable elimination approach to solving influence diagrams with a value iteration approach to solving finite-horizon partially observable Markov decision processes (POMDPs). The integration of these approaches creates a variable elimination algorithm for influence diagrams that has much more relaxed constraints on elimination order, which allows improved scalability in many cases. The new algorithm can also be viewed as a generalization of the value iteration algorithm for POMDPs that solves non-Markovian as well as Markovian problems, in addition to leveraging a factored representation for improved efficiency. The development of a single algorithm that integrates and generalizes both of these classic algorithms, one for influence diagrams and the other for POMDPs, unifies these two approaches to solving Bayesian decision problems in a way that combines their complementary advantages.

1. Introduction

An influence diagram [1–5] is a graphical model of a Bayesian decision problem that leverages problem structure that is represented in the form of a graph, including conditional independence relations among variables and separability of the utility function, in order to represent a decision problem compactly and solve it more efficiently. Influence diagrams have a wide range of successful applications [6].

Another widely-used, and even older model of a Bayesian decision problem is a partially observable Markov decision process (POMDP) [7–11]. The two models are closely related, and many problems can be represented in both ways. Any finite-horizon POMDP can be represented as an influence diagram, and many problems that can be represented as an influence diagram can also be represented as a finite-horizon POMDP. But despite the overlap between these models, very different algorithms have been developed for solving influence diagrams and POMDPs.

Algorithms for solving influence diagrams adopt a dynamic programming approach that enumerates relevant histories, which may be all or part of the full history. By contrast, algorithms for solving POMDPs perform dynamic programming in a space of belief states, where a *belief state* is a probability distribution over an unobserved state space. The POMDP approach scales better for problems with many stages, including infinite-horizon problems. In their classic form, however, algorithms for solving POMDPs do not leverage conditional independence relations among variables, except for the assumption that the state variable at each stage satisfies the Markov property.

Over the past couple of decades, research on the topic of *factored POMDPs* has shown how to more fully leverage conditional independence relations among variables (and the values of variables), as well as separability of the reward function,

E-mail address: hansen@cse.msstate.edu.

https://doi.org/10.1016/j.artint.2020.103431
0004-3702/

in order to improve the scalability of algorithms for POMDPs [e.g., 12–15]. In this work, techniques developed for influence diagrams and related graphical models have been adapted to represent POMDPs more compactly, and solve them more efficiently. This integration of techniques has improved the scalability of algorithms for solving POMDPs. But it has not led to any corresponding improvement of algorithms for solving influence diagrams. This observation provides the starting point for this paper. Our contribution is to develop a complementary approach to integrating these two models that uses techniques originally developed for solving POMDPs in order to improve the effectiveness and scalability of algorithms for solving influence diagrams.

This complementary approach is motivated, in part, by the observation that state-of-the-art algorithms for POMDPs are more effective and scalable than algorithms for influence diagrams in solving problems that can be modeled in both ways, that is, in solving finite-horizon POMDPs. Influence diagrams have the advantage that they represent a broader range of decision problems, including non-Markovian problems. But the best current algorithms for influence diagrams are limited to solving problems with no more than a few decision variables. When the performance of algorithms for these two models is compared in solving finite-horizon POMDPs, algorithms for POMDPs substantially outperform algorithms for influence diagrams as the number of decision variables, which corresponds to the number of stages of the problem, increases.

This difference in performance is related to the fact that the two approaches solve different dynamic programming recurrences. Algorithms for influence diagrams, as mentioned above, solve a dynamic programming recurrence that is based on a discrete state variable that represents the history of the process. Algorithms for POMDPs, by contrast, solve a dynamic programming recurrence that is based on a multi-dimensional continuous state variable that represents a belief state. In the literature on POMDPs, the relative advantages and disadvantages of these two different dynamic programming recurrences are well-understood, and the dynamic programming recurrence based on belief states has long been preferred because it has been found to be more convenient and scalable [e.g., 16, pp. 218–222, 251–270].

In this paper, we develop an approach to solving influence diagrams that leverages the belief-state dynamic programming recurrence associated with POMDP algorithms, allowing larger and more complex influence diagrams to be solved. This approach works especially well in solving influence diagrams that have the same, or similar, structure as finite-horizon POMDPs. But for other problems that can also be represented by influence diagrams, it may not be possible to formulate a purely belief-state dynamic programming recurrence, or, if possible, it may not be the best approach to solving the problem. Therefore, we develop an approach to solving influence diagrams that leverages both dynamic programming recurrences. It can solve the history-based dynamic programming recurrence traditionally used to solve influence diagrams, or it can solve the belief-state dynamic programming recurrence used by POMDP algorithms, or, significantly, it can solve a combination of the two recurrences, that is, it can solve a dynamic programming recurrence that is defined over *both* histories and belief states. The choice of recurrence is made based on how best to solve a given problem. In short, we integrate these two approaches to solving Bayesian decision problems by developing a single framework that generalizes both, and allows existing algorithms for influence diagrams and POMDPs to be viewed as special cases of the same, more general algorithm. The integration of these two approaches does not simply mean that a problem is solved by one approach or the other. The integrated algorithm can solve problems in new ways that are not possible by using either approach alone.

Viewed as a generalization of the traditional variable elimination algorithm for influence diagrams, the improved scalability of the new algorithm results from relaxed constraints on elimination order. The traditional variable elimination algorithm is constrained to eliminate *all* unobserved variables before it eliminates any observed variables, including decision variables. By contrast, the new algorithm can leverage a dynamic programming recurrence that is defined over belief states in order to eliminate decision variables, and other observed variables, before all unobserved variables are eliminated. Relaxing traditional constraints on elimination order improves both the time and memory complexity of the variable elimination approach, often dramatically.

In addition to generalizing the variable elimination approach to solving influence diagrams, the new algorithm can be viewed as a generalization of the value iteration approach to solving POMDPs, in two related ways. First, it leverages a factored representation for improved scalability. Previous work on factored POMDPs also leverages a factored representation, but the new algorithm does so from a different perspective and in new ways. Second, it solves a larger class of finite-horizon partially observable decision processes that includes any problem that can be represented by an influence diagram. This larger class of problems includes problems with delayed action effects, delayed observations, and non-Markovian rewards, that is, it includes a broad range of non-Markovian problems that are not easily modeled in the traditional framework of POMDPs. We argue that the integrated algorithm introduced in this paper offers a promising new approach to solving this large and important class of problems.

The paper is organized as follows. Section 2 reviews relevant background on influence diagrams and POMDPs. In Section 3, the algebra of *potentials* used in the variable elimination approach to solving influence diagrams is generalized to allow decision making under partial observability. This generalization provides the foundation for development of an improved variable elimination algorithm for influence diagrams, described in Section 4, which we call *generalized variable elimination*. Thus Sections 3 and 4 describe the new algorithm. Section 5 considers the new algorithm from the perspective that it also generalizes and improves the value iteration approach to solving POMDPs. Section 6 concludes the paper with a discussion of the significance of the integrated algorithm, including potential applications and extensions. Two appendices provide additional details.

2. Background

In this section, we give an overview of relevant background on influence diagrams and POMDPs. In particular, we review the variable elimination approach to solving influence diagrams and the value iteration approach to solving POMDPs.

2.1. Preliminaries

We begin by introducing notation and concepts related to the variable elimination approach to solving influence diagrams.

2.1.1. Variables, probabilities, and utilities

We use upper-case letters such as X and Y to denote variables and lower-case letters such as x and y to denote values, or states, of variables. For a variable X, we let $sp(X)$ denote its set of states, or state space, and we let $|sp(X)|$ denote the cardinality of the state space. We assume that variables have a finite state space.

We use bold upper-case letters such as $\mathbf{X}$ to represent sets of variables, that is, joint variables. Instantiations (also called "configurations") of $\mathbf{X}$ are denoted by bold lower-case letters, such as $\mathbf{x}$. The state space of a joint variable $\mathbf{X}$ is defined as the Cartesian product of the individual state spaces, denoted $sp(\mathbf{X}) = \times_{X \in \mathbf{X}} sp(X)$, and we let $|sp(\mathbf{X})|$ denote its cardinality.

When $\mathbf{X} \cap \mathbf{Y} = \emptyset$, we let $(\mathbf{X}, \mathbf{Y})$ denote a joint variable, and $(\mathbf{x}, \mathbf{y})$ a state of the joint variable. In the case of an empty set of variables, it is convenient to adopt the convention that its state space consists of a single special state, denoted λ. Thus $sp(\emptyset) = \{\lambda\}$ and $|sp(\emptyset)| = 1$. We also adopt the convention that $(\lambda, \mathbf{y}) = \mathbf{y}$.

If $\mathbf{X}$ and $\mathbf{Y}$ are sets of variables, with $\mathbf{X} \subseteq \mathbf{Y}$, then $\mathbf{y}^{\downarrow \mathbf{X}}$ denotes the *projection* of the state $\mathbf{y}$ onto the state space for $\mathbf{X}$, which means that values of variables in $\mathbf{Y}$ that are not also in $\mathbf{X}$ are ignored. Note that $\mathbf{y}^{\downarrow \mathbf{X}}$ is a state of $\mathbf{X}$. If $\mathbf{X} = \emptyset$, then $\mathbf{y}^{\downarrow \mathbf{X}} = \lambda$.

We let $P(X)$ denote the probability distribution for a variable X, where $P(X = x)$ denotes the probability that variable X has the value x. When the variable is obvious from the context, we let $P(x)$ denote this probability. We let $P(X|Y)$ denote the conditional probability distribution of X given Y. A conditional probability is denoted $P(X = x|Y = y)$, or simply $P(x|y)$ when the variables are obvious from the context. For joint variables, we let $P(\mathbf{X}|\mathbf{Y})$ denote a conditional probability distribution. A conditional probability is denoted $P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = \mathbf{y})$, or simply $P(\mathbf{x}|\mathbf{y})$.

A utility (or reward) function over a set of variables $\mathbf{X}$, denoted $R(\mathbf{X})$, is a mapping $R : sp(\mathbf{X}) \rightarrow \Re$ that expresses the preferences of a decision maker.

2.1.2. Potentials

In variable elimination algorithms, it is common to perform computations using *potentials* in place of probability and utility functions, where a *potential* over a set of variables $\mathbf{X}$ is a function that maps each instantiation $\mathbf{x}$ of $\mathbf{X}$ to a real number.

A *probability potential* over the variables $\mathbf{X}$, denoted $\phi(\mathbf{X})$, is a potential with the further restriction that it is a non-negative function that is not identically zero. The concept of a probability potential generalizes the concept of a probability distribution to contexts where it is not necessarily normalized. Thus a probability distribution $P(\mathbf{X})$ is a special case of a probability potential that sums to 1, and any probability potential can be transformed into an equivalent probability distribution by normalization. A *conditional probability potential* for $\mathbf{X}$ given $\mathbf{Y}$, denoted $\phi(\mathbf{X}|\mathbf{Y})$, is a probability potential over $\mathbf{X}$ conditional on disjoint $\mathbf{Y}$. A conditional probability distribution for $\mathbf{X}$ given $\mathbf{Y}$, denoted $P(\mathbf{X}|\mathbf{Y})$, is a special case of a conditional probability potential that satisfies the condition that for each state $\mathbf{y} \in sp(\mathbf{Y})$, $\sum_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}|\mathbf{Y} = \mathbf{y}) = 1$.

A *utility potential* is a mapping, $\psi : sp(\mathbf{X}) \rightarrow \Re$, that generalizes the concept of expected utility to contexts involving multiplication by a probability potential that is not necessarily normalized.

For convenience, we let the operator *dom* return the variables in the domain (or "scope") of a potential; for example, $dom(\psi(\mathbf{X})) = \mathbf{X}$. Operations on potentials include combination and marginalization operations, which we review next.

2.1.3. Combination operations

A combination operation is a binary operation where the domain of the resulting potential is the union of the domains of the two potentials that are combined. When two potentials, $\psi(\mathbf{X})$ and $\psi'(\mathbf{Y})$, are combined, for example, the domain of the resulting potential, $\psi''(\mathbf{Z})$, is $\mathbf{Z} = \mathbf{X} \cup \mathbf{Y}$. The combination operations used in solving an influence diagram include addition of utility potentials, multiplication and division of probability potentials, and multiplication of a probability potential by a utility potential. The operations are performed element-wise, as follows.

- The *sum* of two utility potentials, $\psi(\mathbf{X})$ and $\psi'(\mathbf{Y})$, is a utility potential, $\psi''(\mathbf{Z}) = \psi(\mathbf{X}) + \psi'(\mathbf{Y})$, defined so that $\psi''(\mathbf{z}) = \psi(\mathbf{z}^{\downarrow \mathbf{X}}) + \psi'(\mathbf{z}^{\downarrow \mathbf{Y}})$, for each $\mathbf{z} \in sp(\mathbf{Z})$.
- The *product* of two probability potentials, $\phi(\mathbf{X})$ and $\phi'(\mathbf{Y})$, is a probability potential, $\phi''(\mathbf{Z}) = \phi(\mathbf{X}) \cdot \phi'(\mathbf{Y})$, defined so that $\phi''(\mathbf{z}) = \phi(\mathbf{z}^{\downarrow \mathbf{X}}) \cdot \phi'(\mathbf{z}^{\downarrow \mathbf{Y}})$, for each $\mathbf{z} \in sp(\mathbf{Z})$. (The dot that represents multiplication is often suppressed.) The product of a utility potential and a probability potential is defined similarly, except the result is a utility potential.
  When two conditional probability potentials are multiplied, as in $\phi''(\mathbf{A}|\mathbf{B}) = \phi(\mathbf{X}|\mathbf{Y}) \cdot \phi'(\mathbf{W}|\mathbf{Z})$, the conditioned and conditioning variables in the product are distinguished in the usual way, so that $\mathbf{A} = \mathbf{X} \cup \mathbf{W}$ and $\mathbf{B} = (\mathbf{Y} \cup \mathbf{Z}) \setminus (\mathbf{X} \cup \mathbf{W})$.

- When a probability potential $\phi(\mathbf{X})$ is divided by another probability potential $\phi'(\mathbf{Y})$, the quotient is a probability potential, $\phi''(\mathbf{Z}) = \phi(\mathbf{X})/\phi'(\mathbf{Y})$, defined as $\phi''(\mathbf{z}) = \phi(\mathbf{z}^{\downarrow \mathbf{X}})/\phi'(\mathbf{z}^{\downarrow \mathbf{Y}})$, for each $\mathbf{z} \in sp(\mathbf{Z})$. If the divisor is zero, we adopt the convention that the result is $+\infty$ if the dividend is positive, $-\infty$ if the dividend is negative, and zero if the dividend is zero.

Algebraic properties. The commutative and associative properties hold for addition of potentials, that is, $\psi_1 + \psi_2 = \psi_2 + \psi_1$ and $(\psi_1 + \psi_2) + \psi_3 = \psi_1 + (\psi_2 + \psi_3)$. The same properties hold for multiplication of potentials, that is, $\phi_1\phi_2 = \phi_2\phi_1$ and $(\phi_1\phi_2)\phi_3 = \phi_1(\phi_2\phi_3)$. The distributive property holds for multiplication over addition, that is: $\phi \cdot (\psi_1 + \psi_2) = \phi \cdot \psi_1 + \phi \cdot \psi_2$. Division is neither commutative nor associative.

A potential is said to be *vacuous* if combining it with a second potential does not change the second potential. In the context of addition, a potential is said to be vacuous if it is identically equal to zero. In the context of multiplication, a potential is vacuous if it is identically equal to 1.

2.1.4. Marginalization operations

Marginalization operations eliminate a variable from the domain of a potential.

Let φ denote a potential that could be either a probability or utility potential. The elimination of a variable Y from the domain of a potential φ by the operation of *sum-marginalization* results in a new potential φ' over the variables $dom(\varphi)\backslash\{Y\}$, defined as $\varphi' = \sum_Y \varphi$. For example, given a potential $\varphi(Y, \mathbf{X})$, elimination of Y by sum-marginalization creates a new potential $\varphi'(\mathbf{X}) = \sum_Y \varphi(Y, \mathbf{X})$, which is defined so that for each $\mathbf{x} \in sp(\mathbf{X})$:

$$\varphi'(\mathbf{x}) = \sum_{y \in sp(Y)} \varphi(y, \mathbf{x}). \tag{1}$$

The elimination of a variable Y from the domain of a potential φ by the operation of *max-marginalization* results in a new potential φ' over the variables $dom(\varphi)\backslash\{Y\}$, defined as $\varphi' = \max_Y \varphi$. For example, given a potential $\varphi(Y, \mathbf{X})$, elimination of the variable Y by max-marginalization creates a new potential $\varphi'(\mathbf{X}) = \max_Y \varphi(Y, \mathbf{X})$, which is defined so that for each $\mathbf{x} \in sp(\mathbf{X})$:

$$\varphi'(\mathbf{x}) = \max_{y \in sp(Y)} \varphi(y, \mathbf{x}). \tag{2}$$

For brevity, we often write $\max_y$ instead of $\max_{y \in sp(Y)}$, and $\sum_y$ instead of $\sum_{y \in sp(Y)}$.

The *restriction* operation instantiates one or more of the variables in the domain of a potential. For example, the restriction $\varphi^{R(Y=y)}$ of a potential $\varphi(Y, \mathbf{X})$ is a potential over $\mathbf{X}$ such that

$$\varphi^{R(Y=y)}(\mathbf{x}) = \varphi(y, \mathbf{x}), \tag{3}$$

for each $\mathbf{x} \in sp(\mathbf{X})$. Since the resulting potential is defined over a smaller set of variables, restriction is a kind of marginalization operation.

Algebraic properties. For sum-marginalization, the commutative property holds, that is: $\sum_Y \sum_Z \varphi = \sum_Z \sum_Y \varphi$. The distributive property also holds, which means that if $Z \notin dom(\varphi_1)$, then $\sum_Z \varphi_1\varphi_2 = \varphi_1 \sum_Z \varphi_2$ and $\sum_Z(\varphi_1 + \varphi_2) = \varphi_1 + \sum_Z \varphi_2$.

For max-marginalization, the commutative property holds, that is: $\max_Y \max_Z \psi = \max_Z \max_Y \psi$. The distributive property also holds, which means that if $Z \notin dom(\psi_1)$, then $\max_Z \psi_1\psi_2 = \psi_1 \max_Z \psi_2$ and $\max_Z(\psi_1 + \psi_2) = \psi_1 + \max_Z \psi_2$.

In general, the sum- and max-marginalization operators are *not* commutative, that is, it does not always hold that $\sum_Y \max_Z \psi = \max_Z \sum_Y \psi$.

For the restriction operation, the commutative property holds, which means that: $(\varphi^{R(Y=y)})^{R(Z=z)} = (\varphi^{R(Z=z)})^{R(Y=y)} = \varphi^{R(Y=y,Z=z)}$.

2.2. Influence diagram

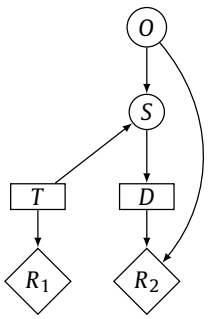
We next review the influence diagram model and the variable elimination algorithm for solving influence diagrams.

2.2.1. Model

Influence diagrams were introduced by Howard and Matheson [1] to provide a more intuitive and compact representation of Bayesian decision problems than is provided by decision trees. We consider a widely-used extension of their original model that allows additive separability of the utility function [4,17].

Definition 1. An influence diagram is a tuple $(G, \mathbf{C}, \mathbf{D}, \mathcal{P}, \mathcal{R})$, where

1. $G = (N, A)$ is a directed acyclic graph with node set N and arc set A. The node set is partitioned into chance nodes, shown graphically as ovals; decision nodes, shown as rectangles; and reward (or value) nodes, shown as diamonds. (See



$P(O) =$

dry	wet	soak
0.5	0.3	0.2

$P(S|O,T) =$

	test			notest		
	dry	wet	soak	dry	wet	soak
closed	0.1	0.3	0.5	1/3	1/3	1/3
open	0.3	0.4	0.4	1/3	1/3	1/3
diffuse	0.6	0.3	0.1	1/3	1/3	1/3

$R_1(T) =$

test	notest
-10	0

$R_2(O,D) =$

	drill	nodrill
dry	-70	0
wet	50	0
soak	200	0

Fig. 1. Influence diagram for oil wildcatter problem with probability and reward tables.

Fig. 1 for an example.) The reward nodes are the leaf nodes of the graph. (If any chance or decision node is a leaf node, it is a "barren node" that can be removed from the influence diagram without affecting an optimal strategy [3].)

2. $\mathbf{V} = \mathbf{C} \cup \mathbf{D}$ is the set of variables of the decision problem, which includes a set of chance (random) variables, $\mathbf{C} = \{C_1, \ldots, C_m\}$, with one variable for each chance node of the graph, and a set of decision variables, $\mathbf{D} = \{D_1, \ldots, D_n\}$, with one variable for each decision node of the graph. Given that variables are identified with their associated nodes in the graph, we let $pa(V)$ denote the set of parent variables of a variable V, where the parent relationship is for the corresponding nodes in the graph.
3. $\mathcal{P} = \{P_1, \ldots, P_m\}$ is a set of conditional probability distributions, one for each chance variable $C_i \in \mathbf{C}$, defined by $P_i = P(C_i|pa(C_i))$. If C_i has no parent variables, it is associated with a marginal probability distribution: $P_i = P(C_i)$.
4. $\mathcal{R} = \{R_1, \ldots, R_q\}$ is a set of reward functions, one for each reward node of the graph, where a reward function is a mapping $R_i : sp(pa(R_i)) \rightarrow \Re$. (Since each reward function corresponds to a reward node in the graph, we let $pa(R_i)$ denote the set of parent variables of the reward function R_i.)

The directed acyclic graph associated with an influence diagram captures structure in the model, including dependence relations and information precedence. Arcs into chance nodes, called *conditional arcs*, represent probabilistic dependence, as in a Bayesian network. Arcs into reward nodes, called *functional arcs*, represent functional dependence by indicating the domain of the associated reward function. Arcs into decision nodes, called *informational arcs*, imply information precedence. That is, an arc from a chance node C to a decision node D indicates that the state of the variable C is known when the decision D is made.

In general, we refer to the values of a chance variable as *states*. In the special case where all of the outgoing arcs from a chance variable are informational arcs, however, we refer to the chance variable as an *observation variable* and we refer to its values as *observations*. We refer to the values of a decision variable as *actions*.

Example: Oil wildcatter. Fig. 1 shows an influence diagram for Raiffa's classic oil wildcatter problem [18,19]. An unobserved chance variable O represents the uncertain presence of oil, with three possible states: *dry* for no oil, *wet* for some oil, and *soak* for a lot of oil. The decision variable T represents the option to perform a seismic *test* or not (*notest*). The observation variable S represents the outcome of the test, which provides imperfect information about the presence of oil: a *closed* reflection pattern suggests a lot of oil, an *open* pattern suggests some oil, and a *diffuse* pattern suggests little oil. The decision variable D represents an option to *drill* for oil or not (*nodrill*). The reward node R_1 represents the cost of performing the seismic test. The reward node R_2 represents the profit from drilling, which depends on how much oil is present. The probability and reward functions are shown by tables alongside the graph in Fig. 1. When no seismic test is performed, the absence of information is modeled by a uniform probability distribution over the possible observations.

Single decision maker with perfect recall. We consider influence diagrams under the classic assumptions of (i) a total ordering of decisions and (ii) no-forgetting.

The first assumption is equivalent to the assumption that there is a directed path in the graph that includes all of the decision nodes, which means the decision variables are ordered in time: $D_1, \ldots, D_n$. Let $\mathbf{Y} \subseteq \mathbf{C}$ denote the subset of chance variables that are observed at some point in the decision problem, and let $\mathbf{X} = \mathbf{C} \backslash \mathbf{Y}$ denote the subset of chance variables that are never observed. Given a total ordering of decision variables, we have a partition of the chance variables, $\{\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n, \mathbf{X}\}$, where $\mathbf{Y} = \mathbf{Y}_1 \cup \mathbf{Y}_2 \cup \ldots \cup \mathbf{Y}_n$. In this partition, $\mathbf{Y}_1$ is the set of chance variables that are observed before the first decision D_1, $\mathbf{Y}_{i+1}$ is the set of chance variables that are observed after the decision D_i and before the decision D_{i+1}, and $\mathbf{X}$ is the set of chance variables that are never observed. Given this partition, there is a partial temporal ordering of variables: $\mathbf{Y}_1 \prec D_1 \prec \mathbf{Y}_2 \prec D_2 \prec \ldots \prec \mathbf{Y}_n \prec D_n \prec \mathbf{X}$. For the oil wildcatter problem, for example, the variables are partitioned as follows: $\mathbf{Y}_1 = \emptyset$, $D_1 = \mathrm{T}$, $\mathbf{Y}_2 = \{\mathrm{S}\}$, $D_2 = \mathrm{D}$, $\mathbf{X} = \{\mathrm{O}\}$.

The second assumption, the *no-forgetting* assumption, means that if the state of a variable V is known when a decision D_i is made, it is known when any posterior decision D_j is made, where $i < j$, even if there is not an explicit arc from V to D_j in the graph. Additional arcs, called *no-forgetting arcs*, are sometimes added to an influence diagram to make the no-forgetting assumption explicit. But to reduce the complexity of the graphical representation of an influence diagram, we follow the convention of not including no-forgetting arcs in the graph. Instead, they are implied. The set of variables whose state is known to the decision maker when a decision D_i is made is called the set of *informational predecessors* of D_i, and is defined as follows:

$$Pred(D_i) = \mathbf{Y}_1 \cup \{D_1\} \cup \mathbf{Y}_2 \cup \ldots \cup \mathbf{Y}_{i-1} \cup \{D_{i-1}\} \cup \mathbf{Y}_i \tag{4}$$

$$= Pred(D_{i-1}) \cup \{D_{i-1}\} \cup \mathbf{Y}_i. \tag{5}$$

An instantiation of the set of informational predecessors for a decision variable D_i is called an *information state* of the decision variable.

This pair of assumptions, a total ordering of decisions and no-forgetting, reflects the perspective that the decision problem is solved by a single decision maker who makes a sequence of decisions based on perfect recall of all past decisions and observations. This perspective is shared by the POMDP model reviewed in Section 2.3.

Strategy representation and optimization. To solve, or evaluate, an influence diagram means to compute an optimal strategy and its corresponding expected utility.

A strategy prescribes an action for each information state of a decision variable, for every decision variable. It is well-known that when a problem is solved by a single decision maker with perfect recall, an optimal deterministic strategy exists. Therefore, a strategy can be represented by a sequence of policies, $\Delta = (\delta_{D_1}, \ldots, \delta_{D_n})$, with one policy, δ_{D_i}, for each decision variable $D_i \in \mathbf{D}$, where a policy is a mapping, $\delta_{D_i} : sp(Pred(D_i)) \rightarrow sp(D_i)$. Sometimes, it is convenient to represent a policy by an equivalent degenerate conditional probability distribution, as follows,

$$P_{\delta_{D_i}}(D_i = d_i | Pred(D_i)) = \begin{cases} 1 \text{ if } d_i = \delta_{D_i}(Pred(D_i)) \\ 0 \text{ otherwise,} \end{cases} \tag{6}$$

in which case a strategy is represented by a sequence, $P_\Delta = (P_{\delta_{D_1}}, \ldots, P_{\delta_{D_n}})$, with one conditional probability distribution, $P_{\delta_{D_i}}$, for each decision variable $D_i \in \mathbf{D}$.

When policies are represented in this form, the joint probability distribution over the variables $\mathbf{C} \cup \mathbf{D}$ induced by a strategy Δ can be defined as

$$P_\Delta(\mathbf{C}, \mathbf{D}) = \prod_{C \in \mathbf{C}} P(C | pa(C)) \prod_{D \in \mathbf{D}} P_{\delta_D}(D | Pred(D)). \tag{7}$$

Given the joint utility function of an influence diagram, defined as

$$U(\mathbf{C}, \mathbf{D}) = \sum_{R \in \mathbf{R}} R(pa(R)), \tag{8}$$

the expected utility of a strategy Δ is a scalar that is defined as

$$EU(\Delta) = \sum_{\mathbf{C},\mathbf{D}} P_\Delta(\mathbf{C}, \mathbf{D}) U(\mathbf{C}, \mathbf{D}). \tag{9}$$

In words, the expected utility of a strategy is the sum of the probability of each joint assignment to the variables in $\mathbf{C} \cup \mathbf{D}$ multiplied by the utility of the assignment. An optimal strategy Δ^* is defined as a strategy that satisfies $EU(\Delta^*) \geq EU(\Delta)$, for all strategies Δ, and the maximum expected utility (MEU) is equal to $EU(\Delta^*)$.

The maximum expected utility can also be defined in a more algorithmically useful way, which is given by the following equation [17, pp. 350-2; 20, pp. 157-9]:

$$MEU = \sum_{\mathbf{Y}_1} \max_{D_1} \ldots \sum_{\mathbf{Y}_n} \max_{D_n} \sum_{\mathbf{X}} \prod_{C \in \mathbf{C}} P(C | pa(C)) \left(\sum_{R \in \mathbf{R}} R(pa(R)) \right). \tag{10}$$

Recall that $\mathbf{C} = \mathbf{Y}_1 \cup \mathbf{Y}_2 \cup \ldots \cup \mathbf{Y}_n \cup \mathbf{X}$, and note that the summation operator inside the parentheses denotes a sum in the usual sense, with a term for each reward function R in the set $\mathcal{R}$, while the outer sums represent sum-marginals. This *MEU equation* underlies the variable elimination algorithm we review next.

Algorithm 1: Variable elimination algorithm [17, pp. 353–5, 396].

Input: Influence diagram with variables $\mathbf{V} = \mathbf{C} \cup \mathbf{D}$
Output: Optimal strategy, Δ, and MEU
$\Phi \leftarrow \{P(C|pa(C)) | C \in \mathbf{C}\}$ // initial set of probability potentials
$\Psi \leftarrow \{R(pa(R)) | R \in \mathbf{R}\}$ // initial set of utility potentials
$\Delta \leftarrow \emptyset$ // initial strategy
for $i \leftarrow 1$ **to** $|\mathbf{V}|$ **do** // i is index of elimination step
  Select variable V to eliminate according to some criterion
  // Process probability potentials
  $\Phi_V \leftarrow \{\phi \in \Phi | V \in dom(\phi)\}$ // get relevant probability potentials
  $\phi_V \leftarrow \prod_{\phi \in \Phi_V} \phi$ // multiply probability potentials
  if *V is a chance variable* **then**
    $\phi_i \leftarrow \sum_V \phi_V$ // eliminate V by sum-marginalization
  else if *V is a decision variable* **then**
    $\phi_i \leftarrow \max_V \phi_V$ // eliminate V by max-marginalization
  $\Phi \leftarrow (\Phi \backslash \Phi_V) \cup \{\phi_i\}$ // update set of probability potentials
  // Process utility potentials
  $\Psi_V \leftarrow \{\psi \in \Psi | V \in dom(\psi)\}$ // get relevant utility potentials
  $\psi_V \leftarrow \sum_{\psi \in \Psi_V} \psi$ // add utility potentials
  if *V is a chance variable* **then**
    $\phi_{cond} \leftarrow \phi_V / \phi_i$ // probability of V conditioned on relevant variables
    $\psi_i \leftarrow \sum_V \phi_{cond} \cdot \psi_V$ // expected value (sum-marginalization)
  else if *V is a decision variable* **then**
    $\psi_i \leftarrow \max_V \psi_V$ // maximum value (max-marginalization)
    $\delta_V \leftarrow \arg\max_V \psi_V$ // optimal policy for decision variable
    $\Delta \leftarrow \Delta \cup \{\delta_V\}$ // add policy to strategy
  $\Psi \leftarrow (\Psi \backslash \Psi_V) \cup \{\psi_i\}$ // update set of utility potentials
end
$MEU \leftarrow \sum_{\psi \in \Psi} \psi$ // after variables eliminated, utility potentials are scalars
return (Δ, MEU) // Δ is optimal strategy

2.2.2. Variable elimination algorithm

Several classes of algorithms have been developed for solving influence diagrams. In this paper, we consider the variable elimination approach.

As a foundation for the algorithm developed in this paper, we adopt the simple variable elimination algorithm described by Jensen and Nielsen [17, pp. 353–5, 396]. Variants of the pseudocode for this algorithm can be found in many places [21–23], and we provide our own variant in Algorithm 1. Several closely-related algorithms are described in the literature. Dechter [24] describes a similar algorithm that adopts the *bucket elimination* framework. Shenoy [19] describes a similar *fusion algorithm* for solving *valuation networks*, which are closely related to influence diagrams. The *junction tree* algorithm is an approach to variable elimination that uses an initial clustering step to improve efficiency [25,20]. There are close similarities between the variable elimination approach and the classic approach to solving an influence diagram by performing a sequence of value-preserving node removals and arc reversals [26,3,4], especially when use of potentials makes it unnecessary to perform arc reversals [27]. The variable elimination approach has also been generalized to allow more complex representations of uncertainty and utility [28–30].

MEU equation. A variable elimination algorithm solves an influence diagram by solving Equation (10), which is called the *MEU equation*. To simplify the operations used to solve this equation, it is formulated using *potentials*, which were reviewed in Section 2.1. Obviously, the conditional probability distributions and reward functions given in the initial specification of an influence diagram are themselves potentials.

Algorithm 1 gives pseudocode for the variable elimination algorithm. The first line assigns the probability potentials of the influence diagram to the set Φ, and the second line assigns the utility potentials to the set Ψ. Equation (10) can then be expressed as

$$MEU = \sum_{\mathbf{Y}_1} \max_{D_1} \sum_{\mathbf{Y}_2} \ldots \max_{D_n} \sum_{\mathbf{X}} \prod \Phi \left(\sum \Psi \right), \tag{11}$$

where the expression $\prod \Phi(\sum \Psi)$ is the product of all probability potentials multiplied by the sum of all utility potentials. In this form, the influence diagram is solved by eliminating one variable from the equation in each iteration of the algorithm, and replacing all potentials in Φ and Ψ that mention the variable with equivalent potentials that do not, until all variables are eliminated and the problem is solved.

We adopt the following terminology. A potential is said to be *relevant* in a given elimination step if the variable selected for elimination is in its domain. The variables in the union of the domains of all relevant potentials are called the *relevant variables*.

Elimination order. In the pseudocode, the set of all variables is denoted $\mathbf{V} = \mathbf{C} \cup \mathbf{D}$. Recall that the informational constraints of a problem induce a partition of the chance variables into information sets, $\mathbf{C} = \{\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n, \mathbf{X}\}$, and a corresponding partial temporal ordering of the variables, $\mathbf{Y}_1 \prec D_1 \prec \mathbf{Y}_2 \prec \ldots \prec D_n \prec \mathbf{X}$, where each information set $\mathbf{Y}_i$ contains the chance variables that are informational predecessors of the decision variable D_i, but not of any previous decision variable, and $\mathbf{X}$ is the set of unobserved variables. Variables are eliminated in the reverse of this partial temporal order, which means the algorithm first sum-marginalizes the variables in $\mathbf{X}$, then max-marginalizes D_n, then sum-marginalizes the variables in $\mathbf{Y}_n$, then max-marginalizes D_{n-1}, and so on. Within any set $\mathbf{Y}_i$ or $\mathbf{X}$, the variables can be eliminated in any order, with the order often determined by a heuristic that attempts to optimize efficiency.

Processing probability potentials. Let V denote the variable selected for elimination. After all probability potentials with V in their domain are identified, they are combined by element-wise multiplication. Then the variable V is eliminated from the resulting probability potential ϕ_V, which creates the probability potential ϕ_i. If V is a chance variable, it is eliminated by sum-marginalization. If it is a decision variable, it can be eliminated by max-marginalization, as shown in Line 12 of Algorithm 1, which follows Jensen and Nielsen's description of the algorithm [17, pp. 353–5]. However, this step can be simplified. When a decision variable is eliminated, it is d-separated from its predecessors, and any successors have already been eliminated. As a result, the eliminated decision variable cannot have any effect on the value of a probability potential, even if it is in its domain. (To say that it cannot have an effect means that for every state of the other variables in the domain of the probability potential, the value of the potential is the same regardless of the value of the decision variable.) It follows that a decision variable can be eliminated from a probability potential by simple projection of the potential onto the remaining variables, as pointed out by others [23,21,30].

Processing utility potentials. After all utility potentials with V in their domain are identified, they are combined by element-wise addition, where ψ_V denotes the resulting utility potential. If V is a chance variable, ψ_V is multiplied by the probability potential, $\phi_{cond} = \phi_V / \phi_i$, and then V is eliminated by sum-marginalization. If V is a decision variable, it is eliminated by max-marginalization, and a corresponding policy, δ_V, is computed that records the maximizing action for each instantiation of the variables in the domain of the resulting utility potential ψ_i.

Solution. When the last variable is eliminated, the algorithm returns a utility potential with an empty domain, which is a scalar equal to the MEU value of Equation (11). It also returns an optimal strategy, $\Delta = (\delta_{D_1}, \ldots, \delta_{D_n})$.

Example: Oil wildcatter. To illustrate how the algorithm works, we describe the steps it takes to solve the influence diagram for the oil wildcatter problem.

1. The first step is to initialize the sets of probability and utility potentials with the conditional probability and reward functions of the influence diagram:

$$\Phi \leftarrow \{P(O), P(S|O,T)\} \tag{12}$$

$$\Psi \leftarrow \{R(T), R(O,D)\}. \tag{13}$$

   Given these potentials, the MEU equation for the influence diagram is:

$$MEU = \max_T \sum_S \max_D \sum_O P(S|O,T)P(O)\left(R(T) + R(O,D)\right). \tag{14}$$

2. *Eliminate chance variable O (for Oil)*:
   For this problem, there is only one valid elimination order. The unobserved chance variable is eliminated first. The probability potentials with the variable O in their domain are processed as follows:

$$\phi_O(O,S|T) \leftarrow P(O) \cdot P(S|O,T) \tag{15}$$

$$\phi_1(S|T) \leftarrow \sum_O \phi_O(O,S|T) \tag{16}$$

$$\Phi \leftarrow \{\phi_1(S|T)\}. \tag{17}$$

   The only utility potential with the variable O in its domain is then processed:

$$\phi_{cond}(O|S,T) \leftarrow \frac{\phi_O(O,S|T)}{\phi_1(S|T)} \tag{18}$$

$$\psi_1(S,T,D) \leftarrow \sum_O \phi_{cond}(O|S,T) \cdot R(O,D) \tag{19}$$

$$\Psi \leftarrow \{R(T), \psi_1(S,T,D)\}. \tag{20}$$

After eliminating the variable O, the revised MEU equation is:

$$MEU = \max_T \sum_S \max_D \phi_1(S|T)\,(R(T) + \psi_1(S, T, D))\,. \tag{21}$$

3. *Eliminate decision variable D (for Drill)*:
There are no probability potentials with the variable D in their domain. The single utility potential with D in its domain is processed as follows:

$$\psi_2(S, T) \leftarrow \max_D \psi_1(S, T, D) \tag{22}$$

$$\Psi \leftarrow \{R(T), \psi_2(S, T)\}. \tag{23}$$

The revised equation for the problem is

$$MEU = \max_T \sum_S \phi_1(S|T)\,(R(T) + \psi_2(S, T))\,, \tag{24}$$

and the following policy is saved in Δ:

$$\delta_D(S, T) \leftarrow \arg\max_D \psi_1(S, T, D) \tag{25}$$

4. *Eliminate chance variable S (for Seismic test result)*:
The only probability potential with the variable S in its domain is $\phi_1(S|T)$, and so the variable S is eliminated by sum-marginalization, as follows,

$$\phi_3(T) \leftarrow \sum_S \phi_1(S|T). \tag{26}$$

The resulting probability potential is vacuous, that is, it assigns the value of 1 to each state of T, and thus it does not need to be added to Ψ. (The fact that it is vacuous follows by the *unit potential property* [17, p. 13].) Because it is vacuous, we have $\phi_{cond}(S|T) = \phi_1(S|T)$. The only utility potential with the variable S in its domain, $\psi_2(S, T)$, is processed as follows:

$$\psi_3(T) \leftarrow \sum_S \phi_{cond}(S|T) \cdot \psi_2(S, T) \tag{27}$$

$$\Psi \leftarrow \{R(T), \psi_3(T)\}. \tag{28}$$

The revised equation for the problem is:

$$MEU = \max_T (R(T) + \psi_3(T))\,. \tag{29}$$

5. *Eliminate decision variable T (for Test)*:
No probability potentials have the variable T in their domain, but two utility potentials do. Recall that λ is the unique state of the empty variable, which is the result of eliminating T from the domain of the potential $\psi_T(T)$. So, we have

$$\psi_T(T) \leftarrow R(T) + \psi_3(T) \tag{30}$$

$$\psi_4(\lambda) \leftarrow \max_T \psi_T(T) \tag{31}$$

$$\Psi \leftarrow \{\psi_4(\lambda)\}. \tag{32}$$

The final equation for the problem is

$$MEU = \psi_4(\lambda), \tag{33}$$

and the policy for this decision variable is

$$\delta_T(\lambda) \leftarrow \arg\max_T \psi_T(T). \tag{34}$$

6. *Return solution*:
The optimal strategy, $\Delta = (\delta_T(\lambda), \delta_D(S, T))$, and MEU are returned.

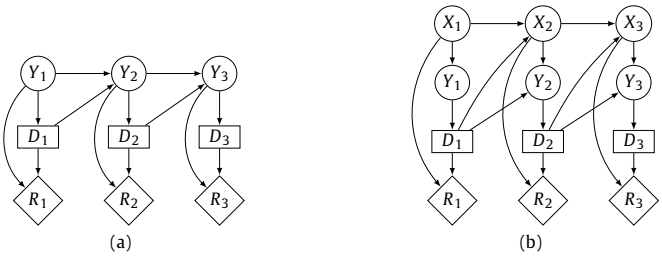


Fig. 2. Influence diagrams for (a) a three-stage completely observable MDP and (b) a three-stage POMDP.

2.3. Finite-horizon partially observable Markov decision process

An alternative model of a single-agent decision-theoretic planning problem under imperfect information is a finite-horizon *partially observable Markov decision process* (POMDP) [7–9,31]. In the rest of this background section, we review this model.

2.3.1. Complete observability

We begin by reviewing an important special case of a POMDP where the state of the process is completely observed.

Definition 2. A finite-horizon completely observable Markov decision process (MDP) is a tuple $(\mathbf{Y}, \mathbf{D}, \mathcal{R}, \mathcal{P})$, where

- the process unfolds over a finite sequence of n stages, indexed by $t = 1, 2, \ldots, n$;
- $\mathbf{Y} = \{Y_t | t = 1, 2, \ldots, n\}$ is a set of random variables, one per stage, that represent the changing state of the process;
- $\mathbf{D} = \{D_t | t = 1, 2, \ldots, n\}$ is a set of decision variables, with one per stage;
- $\mathcal{R} = \{R_t : sp(Y_t) \times sp(D_t) \rightarrow \Re | t = 1, 2, \ldots, n\}$ is a set of reward functions;
- $\mathcal{P} = \{P_t | t = 1, 2, \ldots, n\}$ is a set of probability distributions, one for each random variable. For the first stage, there is an unconditional state probability distribution $P_1(Y_1)$. For each subsequent stage $t = 2 \ldots n$, there is a conditional probability distribution $P_t(Y_t | Y_{t-1}, D_{t-1})$.

At each stage, the state $y_t \in sp(Y_t)$ is observed, an action $d_t \in sp(D_t)$ is performed, and a reward $R_t(y_t, d_t)$ is received. In every stage except the last, the process then makes a transition to a successor state $y_{t+1} \in sp(Y_{t+1})$ with probability $P_t(y_{t+1} | y_t, d_t)$. Fig. 2a shows an influence diagram for a three-stage MDP.

MDPs are often assumed to be *stage-invariant*, which means the variable domains, conditional probability distributions, and reward functions are the same in each stage. But this assumption is not made in general, and the relationship between MDPs and influence diagrams is easier to see when we do not assume the MDP is stage-invariant. To say that a process is *Markovian* means the distribution of future states and rewards is independent of the history of the process given knowledge of the current state.

An MDP is solved by finding a strategy that maximizes expected utility. Because a completely observable MDP is a special case of an influence diagram where the graph of the influence diagram has the form of Fig. 2a, it can be solved by any algorithm that solves influence diagrams. However, an MDP is traditionally solved by a classic value iteration algorithm that is well-described in the literature [16].

2.3.2. Partial observability

Fig. 2b shows an influence diagram that represents a three-stage POMDP. The model is defined as follows.

Definition 3. A finite-horizon partially observable Markov decision process (POMDP) is a tuple $(\mathbf{X}, \mathbf{Y}, \mathbf{D}, \mathcal{R}, \mathcal{P})$, where

- the process unfolds over a finite sequence of n stages, indexed by $t = 1, 2, \ldots, n$;
- $\mathbf{X} = \{X_t | t = 1, 2, \ldots n\}$ is a set of random variables, which we call state variables, that represent the changing state of the process;
- $\mathbf{Y} = \{Y_t | t = 1, 2, \ldots n\}$ is a set of random variables, which we call observation variables because all of their outgoing arcs are informational arcs;
- $\mathbf{D} = \{D_t | t = 1, 2, \ldots n\}$ is a set of decision variables, with one per stage;
- $\mathcal{R} = \{R_t : sp(X_t) \times sp(D_t) \rightarrow \Re | t = 1, 2, \ldots n\}$ is a set of reward functions;
- $\mathcal{P}$ is a set of probability distributions, with one for each random variable in $\mathbf{X} \cup \mathbf{Y}$. For the first stage, the state variable X_1 is associated with an unconditional probability distribution $P(X_1)$, and the observation variable Y_1 is associated with a conditional probability distribution $P(Y_1 | X_1)$. For each subsequent stage $t = 2 \ldots n$, the state variable X_t is associated with a conditional probability distribution $P(X_t | X_{t-1}, D_{t-1})$, and the observation variable Y_t is associated with a conditional probability distribution $P(Y_t | X_t, D_{t-1})$.

In keeping with an assumption already made for influence diagrams, we assume that every variable has a finite number of values. At each stage $t = 1, \ldots, n$ of the process, the decision maker receives an observation $y_t \in sp(Y_t)$ that provides imperfect information about the current state $x_t \in sp(X_t)$. The decision maker then takes an action $d_t \in sp(D_t)$ that results in a reward $R_t(x_t, d_t)$, and, in every stage except the last, a transition to a successor state $x_{t+1} \in sp(X_{t+1})$ with probability $P(x_{t+1}|x_t, d_t)$.

Because a finite-horizon POMDP is a special case of an influence diagram where the graph of the influence diagram has the form of Fig. 2b, it can be solved by any algorithm that solves influence diagrams. But it is traditionally solved by a value iteration algorithm that leverages the special structure of this class of problems [32,33].

2.3.3. Belief state

Although the state of a partially observed process is not directly observed, state probabilities can be computed based on the history of the process. Recall that $y_t \in Y_t$ denotes the observation and $d_t \in D_t$ denotes the action at stage t. Let h_t denote the history of the process at stage t before performing an action, which is defined recursively as,

$$h_1 = \{y_1\} \tag{35}$$

$$h_t = h_{t-1} \cup \{d_{t-1}, y_t\} = \{y_1, \ldots, d_{t-1}, y_t\}, \quad t = 2, 3, \ldots, n, \tag{36}$$

where n is the number of stages of the process. For the first stage, we have by Bayes' rule the following conditional probability that the unobserved state is $x_1 \in sp(X_1)$:

$$P(x_1|h_1) = P(x_1|y_1) = \frac{P(y_1|x_1)P(x_1)}{\sum_{x_1} P(y_1|x_1)P(x_1)}. \tag{37}$$

For the subsequent stages, $t = 2, 3, \ldots, n$, the conditional probabilities are defined recursively as:

$$P(x_t|h_t) = \frac{P(y_t|x_t, d_{t-1}) \sum_{x_{t-1}} P(x_t|x_{t-1}, d_{t-1}) P(x_{t-1}|h_{t-1})}{\sum_{x_t} P(y_t|x_t, d_{t-1}) \sum_{x_{t-1}} P(x_t|x_{t-1}, d_{t-1}) P(x_{t-1}|h_{t-1})}. \tag{38}$$

For convenience, we let $b_t : sp(X_t) \to [0, 1]$ denote a conditional probability distribution where $b_t(x_t) = P(X_t = x_t|h_t)$ is the probability that the process is in state x_t, conditioned on the observed history h_t. In the literature on POMDPs, this conditional probability distribution is called a *belief state* (or information state). Given the belief state b_{t-1} at stage $t-1$, the action d_{t-1} at stage $t-1$, and the observation y_t at stage t, the successor belief state b_t at stage t is given by the deterministic function, $b_t = \tau(b_{t-1}, d_{t-1}, y_t)$, where each component of b_t is defined as

$$b_t(x_t) = P(x_t|b_{t-1}, d_{t-1}, y_t) \tag{39}$$

$$= \frac{P(y_t, x_t|b_{t-1}, d_{t-1})}{P(y_t|b_{t-1}, d_{t-1})} \tag{40}$$

$$= \frac{P(y_t|x_t, d_{t-1}) \sum_{x_{t-1}} P(x_t|x_{t-1}, d_{t-1}) b_{t-1}(x_{t-1})}{\sum_{x_t} P(y_t|x_t, d_{t-1}) \sum_{x_{t-1}} P(x_t|x_{t-1}, d_{t-1}) b_{t-1}(x_{t-1})}. \tag{41}$$

It is well-known that a belief state updated in this way is a *sufficient statistic* for the history of the process, which means that $P(x_t|b_t, h_t) = P(x_t|b_t)$. It follows that an optimal decision can be made based solely on the belief state, without needing to consider the history of the process. This insight provides the key to solving POMDPs efficiently by dynamic programming.

2.3.4. Belief-state MDP and dynamic programming recurrence

The value iteration approach to solving POMDPs is based on the reduction of a POMDP to a completely observable MDP over belief states, called a *belief-state MDP*. For each stage $t = 1, 2, \ldots, n$ of this MDP, the state space is $bsp(X_t)$, which is the multi-dimensional continuous space of belief states over the possible states of the variable X_t. The set of available actions, $sp(D_t)$, is the same as for the original POMDP. The reward function is defined as

$$R_t(b_t, d_t) = \sum_{x_t} b_t(x_t) R_t(x_t, d_t). \tag{42}$$

The (belief) state transition function, which gives the probability of making a transition to belief state $b_{t+1} \in bsp(X_{t+1})$ after taking action $d_t \in sp(D_t)$ in belief state $b_t \in bsp(X_t)$, is defined as

$$P(b_{t+1}|b_t, d_t) = \sum_{y_{t+1}} P(b_{t+1}|b_t, d_t, y_{t+1}) P(y_{t+1}|b_t, d_t), \tag{43}$$

where

$$P(b_{t+1}|b_t, d_t, y_{t+1}) = \begin{cases} 1 & \text{if } b_{t+1} = \tau(b_t, d_t, y_{t+1}) \\ 0 & \text{otherwise,} \end{cases} \tag{44}$$

and

$$P(y_{t+1}|b_t, d_t) = \sum_{x_{t+1}} P(y_{t+1}|x_{t+1}, d_t) \sum_{x_t} P(x_{t+1}|x_t, d_t) b_t(x_t). \tag{45}$$

Equation (43) can be interpreted as follows: the probability of making a transition from belief state b_t to belief state b_{t+1} after action d_t is the sum of the probabilities of all observations y_{t+1} that lead to this belief state.

Once formulated as a belief-state MDP, a finite-horizon POMDP can be solved by solving the following dynamic programming recurrence,

$$V_t(b_t) = \max_{d_t} \left\{ R_t(b_t, d_t) + \sum_{y_{t+1}} P(y_{t+1}|b_t, d_t) V_{t+1}(\tau(b_t, d_t, y_{t+1})) \right\}, \tag{46}$$

for stages $t = 1, 2, \ldots, n-1$, and

$$V_n(b_n) = \max_{d_n} R_n(b_n, d_n), \tag{47}$$

for the last stage of the process. For each stage t of the process, the *value function* $V_t : bsp(X_t) \to \Re$ gives the expected total reward for following an optimal strategy beginning from any belief state $b_t \in bsp(X_t)$.

2.3.5. Value iteration for POMDPs

Solving a finite-horizon POMDP by solving the dynamic programming recurrence of an equivalent belief-state MDP has the advantage that each stage of the POMDP can be solved without considering the previous history of the process. But it presents the challenge that for each stage t, the recurrence needs to be solved for *all* belief states in $bsp(X_t)$, which is a multi-dimensional continuous space. Fortunately, there is an elegant algorithm that gives an exact solution of this dynamic programming recurrence.

Piecewise-linear and concave value function. The key result on which the classic value iteration algorithm for belief-state MDPs depends is due to Smallwood and Sondik [32], who showed that an optimal value function for a finite-horizon belief-state MDP is piecewise linear and concave. (Because Smallwood and Sondik considered cost minimization, they showed, equivalently, that it is piecewise-linear and convex.)

To say that a value function $V_t : bsp(X_t) \to \Re$ is piecewise linear and concave means there is a finite set of linear functions, denoted Γ_t, such that

$$V_t(b_t) = \max_{\gamma \in \Gamma_t} \sum_{x_t} b_t(x_t) \gamma(x_t), \tag{48}$$

where $b_t \in bsp(X_t)$ is a belief state, and each $\gamma \in \Gamma_t$ is said to be a linear function because the value $\sum_{x_t} b_t(x_t)\gamma(x_t)$ is a linear function of the belief state b_t. The classic approach is to represent each linear function (as well as the belief state) by an $|sp(X_t)|$-dimensional vector, where the entries in the vector are mapped to the states of X_t by indexing the states of X_t from 1 through $|sp(X_t)|$. More compact representations of this linear function, including trees [12] and algebraic decision diagrams [13], can be used to leverage problem structure.

Pruning dominated linear functions. For any piecewise linear and concave value function, there is a unique and minimal-size set of linear functions that represents it. The value iteration algorithm for POMDPs performs best if the set of linear functions that represents each piecewise-linear and concave value function is a minimal-size set.

Given a set of linear functions, Γ_t, that represents a value function $V_t : bsp(X_t) \to \Re$, a particular linear function $\gamma' \in \Gamma_t$ is said to be *dominated* by the other linear functions in the set, $\Gamma_t \backslash \{\gamma'\}$, if for all belief states $b \in bsp(X_t)$:

$$\sum_{x_t} b(x_t)\gamma'(x_t) \leq \max_{\gamma \in \Gamma_t \backslash \{\gamma'\}} \sum_{x_t} b(x_t)\gamma(x_t). \tag{49}$$

That is, a linear function $\gamma' \in \Gamma_t$ is dominated if there is no belief state $b \in bsp(X_t)$ for which it provides a better value than the other linear functions in the set Γ_t. This condition can be tested by solving the following linear program:

$$\begin{aligned} &\text{Variables: } \epsilon,\ b(x_t), \forall x_t \in sp(X_t) \\ &\text{Maximize: } \epsilon \\ &\text{Subject to constraints: } \sum_{x_t} \left[b(x_t) \cdot \left(\gamma'(x_t) - \gamma(x_t) \right) \right] \geq \epsilon,\ \forall \gamma \in \Gamma_t \backslash \{\gamma'\} \\ &\qquad\qquad \sum_{x_t} b(x_t) = 1 \text{ and } b(x_t) \geq 0, \forall x_t \in sp(X_t). \end{aligned} \tag{50}$$

If the scalar value ϵ returned by this linear program is non-positive, then γ' is dominated, and can be safely removed from the set Γ_t, that is, it can be "pruned."

In the rest of the paper, we let $Prune(\Gamma)$ denote an operator that takes a set of linear functions Γ and prunes all of its dominated linear functions. A naive way to implement this operator is to test each linear function using the above linear program. A much more efficient algorithm, due to Lark and White [9], is well-described in the literature [33,34], including recent improvements [35,36].

Incremental pruning. Among several approaches to value iteration for POMDPs, the most widely-used is the *incremental pruning* algorithm [33]. It is based, first of all, on the observation that the definition of the value function V_t in Equation (46) can be decomposed into simpler combinations of other value functions, as follows,

$$V_t(b_t) = \max_{d_t} V_t^{d_t}(b_t) \tag{51}$$

$$V_t^{d_t}(b_t) = \sum_{y_{t+1}} V_t^{d_t, y_{t+1}}(b_t) \tag{52}$$

$$V_t^{d_t, y_{t+1}}(b_t) = \frac{R_t(b_t, d_t)}{|sp(Y_{t+1})|} + P(y_{t+1}|b_t, d_t) V_{t+1}(\tau(b_t, d_t, y_{t+1})), \tag{53}$$

where there is a value function $V_t^{d_t}$ for each action $d_t \in sp(D_t)$, and a value function $V_t^{d_t, y_{t+1}}$ for each pair of action $d_t \in sp(D_t)$ and observation $y_{t+1} \in sp(Y_{t+1})$.

Assuming the value function V_{t+1} is piecewise-linear and concave, and is represented by a set Γ_{t+1} of linear functions, then each of these three value functions is also piecewise-linear and concave, and can be represented by a set of linear functions, denoted by Γ_t, $\Gamma_t^{d_t}$, and $\Gamma_t^{d_t, y_{t+1}}$. For each stage t of a POMDP, the sets of linear functions that represent each of these value functions can be generated by performing the following three steps.

The first step, called *backprojection*, takes as input the set Γ_{t+1} of linear functions that represents the stage-$(t+1)$ value function V_{t+1}. For each pair of action d_t and observation y_{t+1}, it generates a set of linear functions,

$$\Gamma_t^{d_t, y_{t+1}} = Prune\left(\{\gamma_t^i | i = 1, \ldots, |\Gamma_{t+1}|\} \right), \tag{54}$$

where, before pruning, there is one linear function $\gamma_t^i \in \Gamma_t^{d_t, y_{t+1}}$ for each linear function $\gamma_{t+1}^i \in \Gamma_{t+1}$. For each state $x_t \in sp(X_t)$, the value of the linear function γ_t^i is

$$\gamma_t^i(x_t) = \frac{R(x_t, d_t)}{|sp(Y_{t+1})|} + \sum_{x_{t+1}} P(x_{t+1}, y_{t+1}|x_t, d_t) \gamma_{t+1}^i(x_{t+1}). \tag{55}$$

The second step of the algorithm is called the *cross sum* step. Given two sets of linear functions, A and B, their cross sum is defined as the set of all pairwise additions of linear functions from these two sets, which is

$$A \oplus B = \{a + b | a \in A, b \in B\}, \tag{56}$$

where this operation extends to more than two sets of linear functions in the obvious way. For each action $d_t \in D_t$, the following set of linear functions is computed:

$$\Gamma_t^{d_t} = Prune\left(\oplus_{y_{t+1} \in sp(Y_{t+1})} \Gamma_t^{d_t, y_{t+1}} \right). \tag{57}$$

The key insight of the incremental pruning algorithm (for which the algorithm is named) is that this set can be computed more efficiently by interleaving the cross-sum operation with the pruning of intermediate sets of linear functions, as follows:

$$\Gamma_t^{d_t} = Prune(\ldots Prune(Prune(\Gamma_t^{d_t, y_{t+1}^1} \oplus \Gamma_t^{d_t, y_{t+1}^2}) \oplus \Gamma_t^{d_t, y_{t+1}^3}) \ldots \Gamma_t^{d_t, y_{t+1}^3})). \tag{58}$$

The third step of the algorithm, called *maximization*, computes the set of linear functions,

$$\Gamma_t = Prune\left(\cup_{d_t \in sp(D_t)} \Gamma_t^{d_t} \right), \tag{59}$$

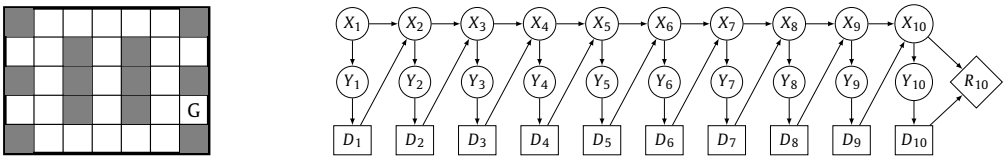


Fig. 3. (a) Maze for ten-stage POMDP and (b) corresponding influence diagram.

which represents the piecewise-linear and concave value function V_t.

To use incremental pruning to solve a finite-horizon POMDP by value iteration, a set Γ_n of linear functions that represents the piecewise-linear and concave value function V_n for the last stage of the problem must first be created, as the base step of the value iteration recursion. It contains one linear function for each action $d_n \in sp(D_n)$. Let $i = 1 \ldots |D_n|$ denote the index of the linear function. The value of each component of the linear function ψ_n^i is defined as

$$\gamma_n^i(x_n) = R_n(x_n, d_n^i), \tag{60}$$

where i is also the index of the corresponding action.

Strategy representation. A strategy for a finite-horizon POMDP is a sequence of policies, $\Delta = \{\delta_1, \ldots, \delta_n\}$, one for each decision variable, where a policy, $\delta_t : bsp(X_t) \rightarrow sp(D_t)$, maps each belief state $b(X_t) \in bsp(X_t)$ to an action d_t that optimizes the corresponding value function V_t in the sequence of value functions, $\mathcal{V} = \{V_1, V_2, \ldots, V_n\}$, which is defined by the dynamic programming recurrence of Equations (46) and (47).

The representation of a policy defined in this way is complicated by the fact that its domain is a multi-dimensional continuous space. Although difficult to represent explicitly, such a policy can be represented implicitly with the help of the set Γ_t of linear functions that represents the piecewise-linear and concave value function V_t for the corresponding stage t of the POMDP. Let $d(\gamma) \in sp(D_t)$ denote the action associated with generation of a linear function $\gamma \in \Gamma_t$. We represent the policy for stage t as follows,

$$\delta_t(b_t) = d\left(\arg\max_{\gamma \in \Gamma_t} \sum_{x_t} b_t(x_t)\gamma(x_t)\right), \tag{61}$$

where $b_t \in bsp(X_t)$ is a belief state over the states of the unobserved variable X_t.

Example: Maze navigation. To motivate the integrated approach to problem solving developed in the rest of this paper, we compare the performance of the variable elimination and incremental pruning algorithms in solving an influence diagram that represents a simple ten-stage POMDP.

The POMDP is a robot navigation problem introduced in previous work on limited-memory influence diagrams [37]. Fig. 3a shows a maze with 23 states. Each white cell represents a state, with an absorbing goal state marked by the letter G. Shaded cells and outside borders represent walls through which the robot cannot pass. The robot can take one of four possible actions in each stage; it can move a single step in any of the four compass directions. It successfully moves in its intended direction with probability 0.89. It moves sideways with respect to its intended direction with probability 0.02 (0.01 for each side), it moves backward with probability 0.001, and it fails to move with probability 0.089. If movement in some direction would take it into a wall, the robot remains in its current location. The robot can accurately sense whether the neighboring cell in each direction of the compass is a wall. For this maze, there are 13 possible observations, including perfect observation of the goal state. Because the same observation can be received in different states, the problem is partially observable.

The problem begins with the robot placed in a (uniformly) random non-goal state, so that it does not know its initial location. It then performs an action in each of a sequence of ten stages. If it reaches the absorbing goal state by the final stage, it receives a reward of 1; otherwise, it receives a reward of 0. Thus the objective is to maximize the probability of reaching the goal state within ten stages.

Because there are 13 possible observations and four possible actions in each stage, there are 52^{10} possible histories over ten stages! The traditional variable elimination algorithm must eliminate all unobserved chance variables before it can eliminate a decision variable. But doing so for this POMDP creates a single probability potential that includes all of the decision and observed chance variables of the influence diagram in its domain, and gives the probability of each of the 52^{10} possible histories over ten stages. It also creates a utility potential with the same domain that gives the utility for all 52^{10} possible histories. When the last decision variable is eliminated, these two potentials are used to compute a utility potential that maps each of the $13 \cdot 52^9$ possible histories before the last action to the expected utility of the optimal action. The corresponding policy has the same dimensions. Given the *immense* size of these potentials, the traditional variable elimination algorithm cannot solve this problem, or even the last stage of this problem, after many hours of CPU time.

By contrast, the incremental pruning algorithm finds an optimal solution for this problem in less than one second of CPU time! In fact, computing an optimal policy and value function for the last decision variable is the easiest part of the

problem to solve. For the last stage of the problem, the incremental pruning algorithm computes a piecewise-linear and concave value function that has just four linear functions, one for each possible action. In this form, it gives the optimal value for any belief state, and an optimal policy is represented in a similarly compact form. The total number of linear functions needed to represent all ten piecewise-linear and concave value functions, one for each stage of the problem, is just 123, where each linear function is a vector of dimension 23. Moreover, only 45 of these linear functions are needed to represent an optimal strategy for the initial belief state, since the other linear functions are unreachable from the initial belief state under an optimal strategy. (See Fig. 9.)

Certainly, not every ten-stage POMDP can be solved so easily by value iteration. In the worst case, the number of linear functions needed to represent a value function grows at a doubly-exponential rate in the number of stages! But in practice, it often grows more slowly, or hardly at all, as in this case. The importance of this example is that it shows that the value iteration algorithm for POMDPs leverages problem structure that is not considered by the traditional variable elimination algorithm. The dramatic difference between the performance of variable elimination and value iteration in solving the same problem motivates the approach developed in the rest of this paper: we show how to improve the performance of the variable elimination approach to solving influence diagrams by integrating it with techniques for solving POMDPs.

3. Piecewise-linear and concave potentials and associated operations

In this section, we show that traditional constraints on elimination order when solving an influence diagram by variable elimination can be relaxed by allowing utility potentials to be represented by piecewise-linear and concave functions, and by using POMDP techniques to generalize the operations on utility potentials. This generalization provides the foundation for development of a more scalable variable elimination algorithm for influence diagrams, which is described in Section 4.

3.1. Constraints on elimination order based on the representation of potentials

The standard model of potentials reviewed in Section 2.1.2, and used by the traditional variable elimination algorithm, unnecessarily constrains the order in which variables can be eliminated when solving an influence diagram.

Recall that the informational constraints of a decision problem induce a partial temporal ordering of the variables of an influence diagram, $\mathbf{Y}_1 \prec D_1 \prec \mathbf{Y}_2 \prec \ldots \prec D_n \prec \mathbf{X}$, where each information set $\mathbf{Y}_i$ contains the chance variables that are informational predecessors of the decision variable D_i, but not of any previous decision variable, and $\mathbf{X}$ is the set of unobserved variables. The traditional variable elimination algorithm eliminates variables in the reverse of this partial temporal order because the standard max-marginalization operator it relies on to eliminate a decision variable from a utility potential assumes that the state of every variable in the domain of the utility potential is known before the corresponding action is taken, that is, it assumes that all variables in the domain of the utility potential are informational predecessors of the eliminated decision variable. If this condition is not met, the utility potential generated by eliminating the decision variable by max-marginalization, and the policy generated at the same time, are not guaranteed to be optimal. In fact, a policy that is represented as a mapping from the states of the variables in its domain to actions cannot even be executed if the states are not known before the action is taken.

It follows that to relax constraints on elimination order, we must generalize the definition of a utility potential, and the max-marginalization operator used to eliminate a decision variable from a utility potential, so that they model decision making under partial observability. We must also similarly generalize the definition of a policy.

3.2. Generalized representation of potentials

When a decision variable D is eliminated by the traditional variable elimination algorithm, the utility potential generated is a mapping,

$$\psi : sp(\mathbf{H}) \rightarrow \Re, \tag{62}$$

where $\mathbf{H} \subseteq Pred(D)$ denotes the variables in the domain of the utility potential. We use the letter "H" because all unobserved variables are eliminated by traditional variable elimination before any decision variable D is eliminated, and so $\mathbf{H}$ represents the relevant *history* of the process for this decision.

It is interesting to compare this representation of a utility potential to the representation of a value function for a POMDP, which is a mapping,

$$V : bsp(U) \rightarrow \Re, \tag{63}$$

where U is an unobserved state variable, and $bsp(U)$ denotes the set of all belief states, or probability distributions, over the possible states of U. An important property of this value function is that it is piecewise-linear and concave, as discussed in Section 2.3.5, which means it is represented by a finite set of linear functions, denoted Γ, such that

$$V(b) = \max_{\gamma \in \Gamma} \sum_u b(u)\gamma(u), \tag{64}$$

where $b \in bsp(U)$ is a belief state, and $b(u)$ is the probability that U is in the state $u \in sp(U)$.

The generalized variable elimination algorithm developed in the rest of this paper combines these two different representations by adopting the following more general representation of a utility potential,

$$\psi : sp(\mathbf{H}) \times bsp(\mathbf{U}) \rightarrow \Re, \tag{65}$$

where $\mathbf{H}$ denotes a (possibly empty) set of observed variables, $\mathbf{U}$ denotes a (possibly empty) set of unobserved variables, and $\times$ denotes the Cartesian product. This utility potential is piecewise-linear and concave, where we define this property in a way that also takes into account the relevant history of the process.

Definition 4. A piecewise-linear and concave potential, $\psi : sp(\mathbf{H}) \times bsp(\mathbf{U}) \rightarrow \Re$, is a potential that is represented by an indexed family of sets of ordinary potentials, $\{\Gamma_{\mathbf{h}}\}_{\mathbf{h} \in sp(\mathbf{H})}$, where each ordinary potential $\gamma \in \Gamma_{\mathbf{h}}$ is a mapping $\gamma : sp(\mathbf{U}) \rightarrow \Re$, and, for a given history $\mathbf{h} \in sp(\mathbf{H})$ and belief state $b(\mathbf{U}) \in bsp(\mathbf{U})$, its value is

$$\psi(\mathbf{h}, b(\mathbf{U})) = \max_{\gamma \in \Gamma_{\mathbf{h}}} \sum_{\mathbf{u}} b(\mathbf{u}) \gamma(\mathbf{u}). \tag{66}$$

From now on, we refer to the ordinary potentials in each set $\Gamma_{\mathbf{h}}$ as *linear potentials*, both to indicate their role in representing a piecewise-linear and concave potential, and to distinguish them from the ordinary probability and utility potentials used in the rest of the algorithm. But it is important to note that linear potentials are still ordinary potentials, although they are used in a different role. They are represented in the same way as ordinary potentials, and the same operations apply to them.

3.2.1. Belief variables

It is a convention in the literature on influence diagrams that $\psi(\mathbf{X})$ denotes a potential defined as $\psi : sp(\mathbf{X}) \rightarrow \Re$. In keeping with our generalization of the concept of a potential, we extend this notation by adopting the convention that $\psi(B(\mathbf{X}))$ denotes a piecewise-linear and concave potential defined as $\psi : bsp(\mathbf{X}) \rightarrow \Re$, and, similarly, $\psi(\mathbf{H}, B(\mathbf{U}))$ denotes a piecewise-linear and concave potential defined as $\psi : sp(\mathbf{H}) \times bsp(\mathbf{U}) \rightarrow \Re$. In this context, we say that $B(\mathbf{X})$ is a joint *belief variable*. An instantiation of a joint belief variable $B(\mathbf{X})$ is a belief state $b(\mathbf{X}) \in bsp(\mathbf{X})$, just as an instantiation of a joint variable $\mathbf{X}$ is a state $\mathbf{x} \in sp(\mathbf{X})$. Obviously, for a single variable X, we simply call $B(X)$ a belief variable.

3.2.2. Ordinary potentials as a special case of piecewise-linear and concave potentials

In the rest of this section, we generalize the operations on ordinary potentials so that they apply to piecewise-linear and concave potentials. The following result ensures that operations on piecewise-linear and concave potentials are also valid for ordinary potentials.

Lemma 1. *An ordinary potential, $\psi : sp(\mathbf{H}, \mathbf{U}) \rightarrow \Re$, is a special case of a piecewise-linear and concave potential, $\psi : sp(\mathbf{H}) \times bsp(\mathbf{U}) \rightarrow \Re$.*

Proof. The domain of an ordinary potential, $\psi(\mathbf{H}, \mathbf{U})$, can be extended so that it is defined for any belief state $b(\mathbf{U})$ over the possible states of $\mathbf{U}$, as follows:

$$\psi(\mathbf{h}, b(\mathbf{U})) = \sum_{\mathbf{u}} b(\mathbf{u}) \psi(\mathbf{h}, \mathbf{u}) \tag{67}$$

$$= \sum_{\mathbf{u}} b(\mathbf{u}) \psi^{R(\mathbf{H}=\mathbf{h})}(\mathbf{u}). \tag{68}$$

The extended potential, $\psi(\mathbf{H}, B(\mathbf{U}))$, is piecewise-linear and concave because it can be represented by an indexed family of sets of linear potentials, $\{\Gamma_{\mathbf{h}}\}_{\mathbf{h} \in sp(\mathbf{H})}$, where each set $\Gamma_{\mathbf{h}}$ contains just the one potential $\psi^{R(\mathbf{H}=\mathbf{h})}$. □

Although the generalized operations on piecewise-linear and concave potentials that we define in the rest of this section work correctly for ordinary potentials, we prefer to perform ordinary operations on ordinary potentials, whenever possible, because it allows potentials to be represented more simply.

3.3. Subproblem decomposition by optimizing over belief states

To illustrate the key idea of our approach, we begin by describing the operation that generates the first piecewise-linear and concave utility potential that is generated when our new variable elimination algorithm eliminates variables in a non-traditional order.

3.3.1. Initial piecewise-linear and concave utility potential

The new algorithm generates a piecewise-linear and concave utility potential if and only if it eliminates the last decision variable before all unobserved chance variables have been eliminated. The following theorem describes a generalization of the max-marginalization operation that eliminates a decision variable from an ordinary utility potential that includes unobserved chance variables in its domain.

Theorem 1. *Elimination by max-marginalization of a decision variable D from the domain of an ordinary potential, $\psi(\mathbf{H}, D, \mathbf{U})$, where $\mathbf{H}$ denotes a set of observed variables and $\mathbf{U}$ denotes a non-empty set of unobserved variables, creates a piecewise-linear and concave potential, $\psi'(\mathbf{H}, B(\mathbf{U}))$, which is represented by an indexed family of sets of linear potentials, $\{\Gamma'_{\mathbf{h}}\}_{\mathbf{h}\in sp(\mathbf{H})}$, where each set $\Gamma'_{\mathbf{h}}$ is defined as follows,*

$$\Gamma'_{\mathbf{h}} = Prune\left(\left\{\psi^{R(\mathbf{H}=\mathbf{h}, D=d)} | d \in sp(D)\right\}\right), \tag{69}$$

and the domain of each linear potential in $\Gamma'_{\mathbf{h}}$ is $\mathbf{U}$.

Proof. For any inputs $\mathbf{h}$ and $b(\mathbf{U})$, we have the equivalences:

$$\max_d \psi(\mathbf{h}, d, b(\mathbf{U})) = \max_d \sum_{\mathbf{u}} b(\mathbf{u}) \psi^{R(\mathbf{H}=\mathbf{h}, D=d)}(\mathbf{u}) \tag{70}$$

$$= \max_{\gamma' \in \Gamma'_{\mathbf{h}}} \sum_{\mathbf{u}} b(\mathbf{u}) \gamma'(\mathbf{u}) \tag{71}$$

$$= \psi'(\mathbf{h}, b(\mathbf{U})). \tag{72}$$

Equation (70) follows from Lemma 1. Equation (71) follows from the definition of $\Gamma'_{\mathbf{h}}$ given by Equation (69). Equation (72) follows from the representation of ψ' by the indexed family of sets of linear potentials, $\{\Gamma'_{\mathbf{h}}\}_{\mathbf{h}\in sp(\mathbf{H})}$, as defined in the theorem. □

The *Prune* operator in Equation (69), and in similar equations in the rest of the paper, removes dominated linear potentials from a set, as described in Section 2.3.5. It is the same operator used by exact dynamic programming algorithms for POMDPs. It improves efficiency because the complexity of operations on sets of potentials increases with the size of the sets. Of course. pruning dominated potentials from a set of linear potentials does not change the piecewise-linear and concave potential that it represents.

3.3.2. Example: Maze navigation revisited

Recall the maze POMDP described in Section 2.3.5. When solved by traditional variable elimination, the unobserved chance variables for all ten stages of the problem must be eliminated before any other variable is eliminated, including the last decision variable. As discussed earlier, the initial elimination of all unobserved variables creates both a probability potential and a utility potential with all of the decision and observed chance variables in their domain, giving both the probability and the utility of each of the 52^{10} possible histories over ten stages. The immense size of these potentials explains why the traditional variable elimination algorithm cannot solve this problem, or even eliminate the last decision variable, even after hours of CPU time.

By contrast, the generalized max-marginalization operator described by Theorem 1 can eliminate the last decision variable before any other variable is eliminated, and generates an optimal utility potential and policy for the last decision variable almost instantly. The utility potential and policy are also represented much more compactly. When the last decision variable D_{10} of the influence diagram in Fig. 3b is eliminated first, there are no relevant probability potentials, and the only relevant utility potential is the reward function $R(X_{10}, D_{10})$, which includes the unobserved chance variable X_{10} in its domain. By Theorem 1, eliminating D_{10} from $R(X_{10}, D_{10})$ by max-marginalization creates a piecewise-linear and concave utility potential, $\psi_1(B(X_{10}))$, represented by a set Γ_1 of linear potentials, with one linear potential, $R^{R(D_{10}=d_{10})}(X_{10})$, for each action $d_{10} \in sp(D_{10})$. For any belief state $b(X_{10}) \in bsp(X_{10})$, we have

$$\psi_1(b(X_{10})) = \max_{\gamma \in \Gamma_1} \sum_{x_{10}} b(x_{10}) \gamma(x_{10}) \tag{73}$$

$$= \max_{d_{10}} \sum_{x_{10}} b(x_{10}) R^{R(D_{10}=d_{10})}(x_{10}), \tag{74}$$

where $b(x_{10})$ is the belief, or probability, that the unobserved state of X_{10} is x_{10}.

This example illustrates the key idea of the POMDP approach we adopt. The utility potential ψ_1 is generated by optimizing for all belief states over the possible states of X_{10}. By optimizing for all belief states, the optimization problem for the last decision variable is decoupled from the rest of the optimization problem. That is, it is decoupled from the history of the process, allowing this subproblem to be solved independently, and easily. By contrast, the traditional variable elimination

algorithm conditions the last decision on the entire history of the process, which makes the optimization problem for the last decision variable almost prohibitively difficult to solve.

3.4. Generalized operations on potentials

Once an initial piecewise-linear and concave utility potential is generated, the other operations of the variable elimination algorithm need to be able to process it. We next show how to extend the definitions of all of the operations on potentials given in Section 2.1 so that they apply to piecewise-linear and concave potentials. First we consider the combination operations of addition and multiplication. (The new algorithm does not divide piecewise-linear and concave potentials.) Then we consider the max-marginalization and sum-marginalization operations.

3.4.1. Generalized combination operations: Addition and multiplication

When two potentials are combined by addition or multiplication, the domain of the new potential is the union of the domains of the combined potentials. For piecewise-linear and concave potentials, we also need to distinguish between the observed and unobserved variables in the domain of a potential, since this distinction affects the representation of the potential. Let $\circ$ denote a combination operator for potentials. The following theorem applies to both addition and multiplication.

Theorem 2. *Consider the combination (by addition or multiplication) of a piecewise-linear and concave potential, $\psi(\mathbf{H}, B(\mathbf{U}))$, represented by an indexed family of sets of linear potentials, $\{\Gamma_{\mathbf{h}}\}_{\mathbf{h}\in sp(\mathbf{H})}$, and a piecewise-linear and concave potential, $\psi'(\mathbf{H}', B(\mathbf{U}'))$, represented by an indexed family of sets of linear potentials, $\{\Gamma'_{\mathbf{h}'}\}_{\mathbf{h}'\in sp(\mathbf{H}')}$. The result is a piecewise-linear and concave potential, $\psi''(\mathbf{H}'', B(\mathbf{U}''))$, represented by an indexed family of sets of linear potentials, $\{\Gamma''_{\mathbf{h}''}\}_{\mathbf{h}''\in sp(\mathbf{H}'')}$, where $\mathbf{H}'' = \mathbf{H}\cup\mathbf{H}'$; for each instantiation $\mathbf{h}''$ of $\mathbf{H}''$, the set $\Gamma''_{\mathbf{h}''}$ is defined as*

$$\Gamma''_{\mathbf{h}''} = Prune\left(\left\{\gamma\circ\gamma' | \gamma\in\Gamma_{\mathbf{h}}, \gamma'\in\Gamma'_{\mathbf{h}'}\right\}\right); \tag{75}$$

and the linear potentials in each set $\Gamma''_{\mathbf{h}''}$ have the domain $\mathbf{U}'' = \mathbf{U}\cup\mathbf{U}'$.

Proof. The result follows from the equivalences given below by Equations (76) through (79), which hold for any combination: $\psi''(\mathbf{h}'', b''(\mathbf{U}'')) = \psi(\mathbf{h}, b(\mathbf{U}))\circ\psi'(\mathbf{h}', b'(\mathbf{U}'))$. In these equivalences, $b(\mathbf{U}) = \sum_{\mathbf{U}''\setminus\mathbf{U}} b''(\mathbf{U}'')$ and $b'(\mathbf{U}') = \sum_{\mathbf{U}''\setminus\mathbf{U}'} b''(\mathbf{U}'')$. That is, b is the same belief state as b'', but with the variables $\mathbf{U}''\setminus\mathbf{U}$ marginalized out. The belief state b' is defined similarly. The equivalences are:

$$\begin{aligned}
\psi''(\mathbf{h}'', b''(\mathbf{U}'')) &= \psi(\mathbf{h}, b(\mathbf{U}))\circ\psi'(\mathbf{h}', b'(\mathbf{U}')) && (76)\\
&= \left(\max_{\gamma\in\Gamma_{\mathbf{h}}}\sum_{\mathbf{u}} b(\mathbf{u})\gamma(\mathbf{u})\right)\circ\left(\max_{\gamma'\in\Gamma'_{\mathbf{h}'}}\sum_{\mathbf{u}'} b'(\mathbf{u}')\gamma'(\mathbf{u}')\right) && (77)\\
&= \max_{\gamma\in\Gamma_{\mathbf{h}},\gamma'\in\Gamma'_{\mathbf{h}}}\sum_{\mathbf{u}''} b''(\mathbf{u}'')\left(\gamma(\mathbf{u})\circ\gamma'(\mathbf{u}')\right) && (78)\\
&= \max_{\gamma''\in\Gamma''_{\mathbf{h}}}\sum_{\mathbf{u}''} b''(\mathbf{u}'')\gamma''(\mathbf{u}''). && (79)
\end{aligned}$$

Equation (76) simply states the operation to be performed. Equation (77) follows from the representation of the piecewise-linear and concave potentials ψ and ψ' by corresponding indexed families of sets of linear potentials. Equation (78) expresses the insight that when the indices for two maximization operators are different, the combination of all maximums is equal to the maximum of all combinations. Equation (79) follows since $\gamma''(\mathbf{U}'') = \gamma(\mathbf{U})\circ\gamma'(\mathbf{U}')$, and the piecewise-linear and concave potential ψ'' is represented by the indexed family of sets of linear potentials defined in the theorem. □

Although Theorem 2 applies to both addition and multiplication, the algorithm we develop only uses it to add two piecewise-linear and concave utility potentials. We state the theorem in this more general form, however, because it implies the following corollary, which is used to multiply an ordinary probability potential by a piecewise-linear and concave utility potential, as well as to add an ordinary utility potential to a piecewise-linear and concave utility potential.

Corollary 1. *Consider the combination (by addition or multiplication) of an ordinary potential, $\psi(\mathbf{H}, \mathbf{U})$, and a piecewise-linear and concave potential, $\psi'(\mathbf{H}', B(\mathbf{U}'))$, represented by an indexed family of sets of linear potentials, $\{\Gamma'_{\mathbf{h}'}\}_{\mathbf{h}'\in sp(\mathbf{H}')}$. The result is a piecewise-linear and concave potential, $\psi''(\mathbf{H}'', B(\mathbf{U}''))$, represented by an indexed family of sets of linear potentials, $\{\Gamma''_{\mathbf{h}''}\}_{\mathbf{h}''\in sp(\mathbf{H}'')}$, where $\mathbf{H}'' = \mathbf{H}\cup\mathbf{H}'$; for each instantiation $\mathbf{h}''$ of $\mathbf{H}''$, the set $\Gamma''_{\mathbf{h}''}$ is defined as*

$$\Gamma''_{\mathbf{h}''} = Prune\left(\left\{\psi^{R(\mathbf{H}=\mathbf{h}''^{\downarrow\mathbf{H}})}\circ\gamma' | \gamma'\in\Gamma'_{\mathbf{h}'}\right\}\right); \tag{80}$$

and the linear potentials in each set $\Gamma''_{\mathbf{h}''}$ have domain $\mathbf{U}'' = \mathbf{U} \cup \mathbf{U}'$. (Note that the expression $\psi^{R(\mathbf{H}=\mathbf{h}''^{\downarrow \mathbf{H}})}$ in Equation (80) simply means that the potential $\psi(\mathbf{H}, \mathbf{U})$ is restricted to a potential with domain $\boldsymbol{U}$ by instantiating $\boldsymbol{H}$ to $\mathbf{h}''^{\downarrow \mathbf{H}}$.)

Proof. The result follows from Theorem 2 and Lemma 1, which justifies treating an ordinary potential as a special case of a piecewise-linear and concave potential. □

Example. Let C denote an unobserved chance variable with three possible states. Let the vector, $(0.2, 0.3, 0.5)$, represent the probability potential, $\phi(C)$, and let the set of vectors, $\Gamma = \{(4, 6, 7), (5, 2, 3), (3, 7, 1)\}$, represent the set of linear potentials that represents a piecewise-linear and concave utility potential, $\psi(B(C))$. The product of $\phi(C)$ and $\psi(B(C))$ is a piecewise-linear and concave utility potential, $\psi'(B(C))$, represented by the set of vectors: $\Gamma' = \{(0.8, 1.8, 3.5), (1.0, 0.6, 1.5), (0.6, 2.1, 0.5)\}$.

Criteria for selecting combination operator. We could use Theorem 2 to combine a utility potential with another potential in every case, if we represented every utility potential as piecewise-linear and concave. However, we prefer to use ordinary potentials whenever possible, and so we adopt the following criteria to determine which operator to use to combine a utility potential with another potential: (i) to add two ordinary utility potentials, or multiply an ordinary utility potential by a probability potential, we use the traditional addition and multiplication operators; (ii) to add two piecewise-linear and concave utility potentials, we use the operator described by Theorem 2; and (iii) to add a piecewise-linear and concave utility potential and an ordinary utility potential, or multiply a piecewise-linear and concave utility potential by a probability potential, we use the operator described by Corollary 1.

3.4.2. Generalized max-marginalization

We next show how to generalize the max-marginalization operator to eliminate a decision variable from a piecewise-linear and concave potential. The following result complements and generalizes Theorem 1, which considers how to eliminate a decision variable from an ordinary potential that has unobserved chance variables in its domain.

Theorem 3. *Elimination of a decision variable D by max-marginalization from a piecewise-linear and concave potential $\psi(\mathbf{H}, D, B(\mathbf{U}))$, represented by an indexed family of sets of linear potentials, $\{\Gamma_{(\mathbf{h},d)}\}_{(\mathbf{h},d)\in sp(\mathbf{H},D)}$, creates a piecewise-linear and concave potential, $\psi'(\mathbf{H}, B(\mathbf{U}))$, represented by an indexed family of sets of linear potentials, $\{\Gamma'_{\mathbf{h}}\}_{\mathbf{h}\in sp(\mathbf{H})}$, where for each instantiation $\boldsymbol{h}$ of $\boldsymbol{H}$, we have*

$$\Gamma'_{\mathbf{h}} = Prune\left(\cup_{d\in sp(D)}\Gamma_{(\boldsymbol{h},d)}\right), \tag{81}$$

and the domain of the linear potentials in $\Gamma'_{\mathbf{h}}$ is $\boldsymbol{U}$.

Proof. For any inputs $\mathbf{h}$ and $b(\mathbf{U})$, the result follows from the equivalences:

$$\max_d \psi(\mathbf{h}, d, b(\mathbf{U})) = \max_d \max_{\gamma\in\Gamma_{\mathbf{h},d}} \sum_{\mathbf{u}} b(\mathbf{u})\gamma(\mathbf{u}) \tag{82}$$

$$= \max_{\gamma'\in\Gamma'_{\mathbf{h}}} \sum_{\mathbf{u}} b(\mathbf{u})\gamma'(\mathbf{u}) \tag{83}$$

$$= \psi'(\mathbf{h}, b(\mathbf{U})). \tag{84}$$

Equation (82) follows from representation of $\psi(\mathbf{H}, D, B(\mathbf{U}))$ by the indexed family of sets of linear potentials, $\{\Gamma_{(\mathbf{h},d)}\}_{(\mathbf{h},d)\in sp(\mathbf{H},D)}$. Equation (83) follows from the observation that each set $\Gamma'_{\mathbf{h}}$ contains all undominated linear potentials in the union of sets, $\cup_{d\in sp(D)}\Gamma_{(\mathbf{h},d)}$, based on Equation (81), and so, for any belief state $b(\mathbf{U})$, the maximizing linear potential in $\Gamma'_{\mathbf{h}}$ must be the same as the maximizing linear potential in the union of sets, $\cup_{d\in sp(D)}\Gamma_{(\mathbf{h},d)}$. Equation (84) follows from representation of the piecewise-linear and concave potential $\psi'(\mathbf{H}, B(\mathbf{U}))$ by the indexed family of sets of linear potentials, $\{\Gamma'_{\mathbf{h}}\}_{\mathbf{h}\in sp(\mathbf{H})}$, defined by the theorem. □

By Lemma 1, an ordinary potential is a special case of a piecewise-linear and concave potential. Therefore, this theorem also applies to the elimination of a decision variable from an ordinary utility potential, $\psi(\mathbf{H}, D)$, in which case it gives the same result as Theorem 1.

Generalized policy representation. When variable elimination eliminates a decision variable from a utility potential, it not only computes a new utility potential. It also computes a policy with the same domain as the new utility potential that records the maximizing decision for each instantiation of the variables in the domain.

We generalize the representation of a policy in a similar way to how we generalize the representation of a utility potential. Instead of representing a policy as a mapping, $\delta_D : sp(\mathbf{H}) \rightarrow sp(D)$, where $\mathbf{H} \subseteq Pred(D)$ is a subset of decision and observed chance variables that represents the relevant history for this decision, we represent a policy in a more general way, as follows,

$$\delta_D : sp(\mathbf{H}) \times bsp(\mathbf{U}) \rightarrow sp(D), \tag{85}$$

where $\mathbf{U}$ is a subset of unobserved state variables that represents the relevant unobserved state for the decision.

When $\mathbf{U}$ is empty, the two representations of a policy coincide. When $\mathbf{U}$ is not empty, the domain of a policy includes a continuous, multi-dimensional space of belief states, and a policy for a decision variable D is represented implicitly by associating an action $d \in sp(D)$ with each linear potential used to represent the corresponding piecewise-linear and concave utility potential, so that

$$\delta_D(\mathbf{h}, b(\mathbf{U})) = d\left(\arg\max_{\gamma \in \Gamma_{\mathbf{h}}} \sum_{\mathbf{u}} b(\mathbf{u})\gamma(\mathbf{u})\right), \tag{86}$$

where $d(\gamma)$ denotes the action associated with the linear potential γ. The association of an action with each linear potential generalizes the similar representation of a policy for a POMDP, given by Equation (61).

Example. Consider a piecewise-linear and concave utility potential $\psi(D, B(C))$, where D is a Boolean decision variable and C is an unobserved Boolean chance variable. This utility potential is represented by two sets of linear potentials with domain C. Let $\Gamma_{D=0} = \{(0, 6), (3, 2)\}$ be one set, and let $\Gamma_{D=1} = \{(4, 2), (5, 1), (2, 3)\}$ be the other. Eliminating the decision variable D from this utility potential by max-marginalization creates a piecewise-linear and concave utility potential, $\psi'(B(C))$, that is represented by a single set of linear potentials with domain C, as follows: $\Gamma' = Prune(\Gamma_{D=0} \cup \Gamma_{D=1}) = \{(0, 6), (5, 1)\}$. Note that the union of these two sets originally has five linear potentials, but three are pruned because they are dominated by the other two. The corresponding policy is represented by associating the action $D = 0$ with the linear potentials from the set $\Gamma_{D=0}$, the action $D = 1$ with the linear potentials from the set $\Gamma_{D=1}$, and using Equation (86) to map any belief state to an action. In this case, of course, the vector $(0, 6)$ is associated with the action $D = 0$, and the vector $(5, 1)$ is associated with the action $D = 1$.

Criteria for selecting max-marginalization operator. It is possible to represent all utility potentials as piecewise-linear and concave, and use Theorem 3 to eliminate a decision variable from a utility potential in every case. However, we prefer to represent utility potentials as ordinary potentials whenever possible. Therefore, we use the following criteria to determine how to eliminate a decision variable from a utility potential: (i) for an ordinary utility potential with no unobserved variables in its domain, we use the traditional max-marginalization operator; (ii) for an ordinary utility potential with one or more unobserved variables in its domain, we use the max-marginalization operator of Theorem 1; and (iii) for a piecewise-linear and concave utility potential, we use the max-marginalization operator of Theorem 3. In summary, we use the least-general version of the operation that applies.

3.4.3. Generalized sum-marginalization

It remains to consider how to eliminate a chance variable from a piecewise-linear and concave utility potential. The sum-marginalization operator works very differently depending on whether the variable being eliminated is observed or not, and so we consider the two cases separately.

Elimination of an observed variable from a piecewise-linear and concave potential. First we consider the case where sum-marginalization is used to eliminate an observed chance variable.

Theorem 4. *Elimination by sum-marginalization of an observed chance variable C from a piecewise-linear and concave potential, $\psi(\mathbf{H}, C, B(\mathbf{U}))$, represented by an indexed family of sets of linear potentials, $\{\Gamma_{(\mathbf{h},c)}\}_{(\mathbf{h},c)\in sp(\mathbf{H},C)}$, with domain $\mathbf{U}$, creates a piecewise-linear and concave potential, $\psi'(\mathbf{H}, B(\mathbf{U}))$, which is represented by an indexed family of sets of linear potentials, $\{\Gamma'_{\mathbf{h}}\}_{\mathbf{h}\in sp(\mathbf{H})}$, with domain $\mathbf{U}$, defined as follows: for each instantiation $\mathbf{h}$ of $\mathbf{H}$,*

$$\Gamma'_{\mathbf{h}} = Prune\left(\left\{\oplus_{c\in sp(C)}\Gamma_{(\mathbf{h},c)}\right\}\right), \tag{87}$$

where $\oplus$ denotes the cross sum *operator defined by Equation (56).*

Proof. The result follows from the following equivalences, which hold for any inputs $\mathbf{h}$ and $b(\mathbf{U})$ for the potentials:

$$\sum_c \psi(\mathbf{h}, c, b(\mathbf{U})) = \sum_c \max_{\gamma\in\Gamma_{\mathbf{h},c}} \sum_{\mathbf{u}} b(\mathbf{u})\gamma(\mathbf{u}) \tag{88}$$

$$= \max_{\gamma'\in\Gamma'_{\mathbf{h}}} \sum_{\mathbf{u}} b(\mathbf{u})\gamma'(\mathbf{u}) \tag{89}$$

$$= \psi'(\mathbf{h}, b(\mathbf{U})). \tag{90}$$

Equation (88) follows by the definition of ψ given in the theorem. Equation (89) expresses the insight that each linear potential $\gamma' \in \Gamma'_{\mathbf{h}}$ is the sum of $|sp(C)|$ linear potentials, one from each set $\Gamma_{\mathbf{h},c}$, for $c \in sp(C)$, and the maximum of all sums

in Equation (89) is equal to the sum of all maximums in Equation (88). Equation (90) follows by the definition of ψ' in the theorem. □

Of course, the most efficient way of performing the computation indicated by Equation (87) is by interleaving the cross-sum and pruning operations, as in Equation (58).

Example. Let C denote an observed chance variable with three possible values, and let U denote an unobserved Boolean chance variable. Consider a piecewise-linear and concave utility potential, $\psi(C, B(U))$, represented by three sets of linear potentials: $\Gamma_{C=0} = \{(1,3),(2,0)\}$, $\Gamma_{C=1} = \{(3,2),(1,4)\}$, and $\Gamma_{C=2} = \{(5,1),(4,2),(3,3)\}$. Elimination of the observed chance variable C by sum-marginalization creates a new piecewise-linear and concave utility potential, $\psi'(B(U))$, which is represented by a single set of linear potentials, constructed as follows:

$$\Gamma' = Prune(Prune(\Gamma_{C=0} \oplus \Gamma_{C=1}) \oplus \Gamma_{C=2}) = \{(10,3),(5,10),(9,6)\}. \tag{91}$$

Without pruning, the resulting set would have included twelve linear potentials, instead of three. However, nine of the twelve linear potentials were dominated.

Elimination of an unobserved variable from a piecewise-linear and concave potential. Finally, we describe how to eliminate an unobserved chance variable from a piecewise-linear and concave utility potential.

We begin by making two observations about this operation. First, when a variable C is unobserved, the expression $\sum_C \psi(\mathbf{H}, B(C, \mathbf{U}))$ is not meaningful because the summation operator for an unobserved variable must occur on the right-hand side of any maximization operator in the MEU equation, and there is a maximization operator in the definition of a piecewise-linear and concave utility potential. Recall that for any input $(\mathbf{h}, b(C, \mathbf{U}))$:

$$\psi(\mathbf{h}, b(C, \mathbf{U})) = \max_{\gamma \in \Gamma_{\mathbf{h}}} \sum_{c,\mathbf{u}} b(c, \mathbf{u}) \gamma(c, \mathbf{u}). \tag{92}$$

Moreover, this equation *already* includes a summation operator for C on the right-hand side of the maximization operator.

A second observation is that before a variable is eliminated from a utility potential by the variable elimination algorithm, it must be eliminated from all probability potentials in the MEU equation. Therefore, when an unobserved chance variable C is eliminated from a piecewise-linear and concave utility potential, $\psi(\mathbf{H}, B(C, \mathbf{U}))$, by sum-marginalization, we can assume that the variable is no longer in the domain of any probability potential in the MEU equation. It follows that there is no reason for the utility potential to be defined for belief states over the possible states of C.

Based on these two observations, we introduce the following symbol to denote elimination of an unobserved chance variable from a piecewise-linear and concave potential by sum-marginalization. We use this symbol as a notational convenience only, since the ordinary summation operator is not meaningful in this case.

Definition 5. Let the symbol, $\sum_{unobserved(C)}$, when it appears to the immediate left of a piecewise-linear and concave potential, denote the elimination of an unobserved variable C from the potential, which is defined so that

$$\sum_{unobserved(C)} \psi(\mathbf{h}, b(C, \mathbf{U})) = \max_{\gamma \in \Gamma_{\mathbf{h}}} \sum_{c,\mathbf{u}} \left(\sum_{c} b(c, \mathbf{u}) \right) \gamma(c, \mathbf{u}). \tag{93}$$

By this definition, a variable C is eliminated from a piecewise-linear and concave utility potential in two steps. First, it is eliminated from the belief state, $b(C, \mathbf{U})$, by sum-marginalization. Then it is straightforward to eliminate C by sum-marginalization from each of the linear potentials in the representation of the piecewise-linear and concave utility potential, as shown in the proof of the following theorem.

Theorem 5. *Consider the elimination of an unobserved chance variable C by sum-marginalization from a piecewise-linear and concave potential, $\psi(\mathbf{H}, B(C, \mathbf{U}))$, which is represented by an indexed family of sets of linear potentials, $\{\Gamma_{\mathbf{h}}\}_{\mathbf{h} \in sp(\mathbf{H})}$, where the domain of the linear potentials is $(C, \mathbf{U})$, and $\mathbf{U}$ may or may not be empty.*

If $\mathbf{U}$ is non-empty, the result is a piecewise-linear and concave potential, denoted $\psi'(\mathbf{H}, B(\mathbf{U}))$, which is represented by an indexed family of sets of linear potentials, $\{\Gamma'_{\mathbf{h}}\}_{\mathbf{h} \in sp(\mathbf{H})}$, with domain $\mathbf{U}$, where for each instantiation $\boldsymbol{h}$ of $\boldsymbol{H}$, we have

$$\Gamma'_{\mathbf{h}} = Prune\left(\left\{ \sum_{C} \gamma \,|\, \gamma \in \Gamma_{\mathbf{h}} \right\} \right). \tag{94}$$

If $\mathbf{U}$ is empty, that is, if C is the only unobserved variable in the domain of ψ, the result is an ordinary utility potential $\psi'(\mathbf{H})$, where for each instantiation $\boldsymbol{h}$ of $\boldsymbol{H}$:

$$\psi'(\mathbf{h}) = \max_{\gamma \in \Gamma_{\mathbf{h}}} \sum_{c \in sp(C)} \gamma(c). \tag{95}$$

Proof. First we consider the case where C is not the only unobserved variable in the domain of ψ. For any belief state $b(C, \mathbf{U})$, we can define a related belief state: $b'(\mathbf{U}) = \sum_C b(C, \mathbf{U})$. The following equivalences hold for any input $(\mathbf{h}, b(C, \mathbf{U}))$ for ψ:

$$\sum_{unobserved(C)} \psi(\mathbf{h}, b(C, \mathbf{U})) = \max_{\gamma \in \Gamma_{\mathbf{h}}} \sum_{c,\mathbf{u}} \left(\sum_c b(c, \mathbf{u}) \right) \gamma(c, \mathbf{u}) \tag{96}$$

$$= \max_{\gamma \in \Gamma_h} \sum_{c,\mathbf{u}} b'(\mathbf{u}) \gamma(c, \mathbf{u}) \tag{97}$$

$$= \max_{\gamma' \in \Gamma'_h} \sum_{\mathbf{u}} b'(\mathbf{u}) \gamma'(\mathbf{u}) \tag{98}$$

$$= \psi'(\mathbf{h}, b'(\mathbf{U})). \tag{99}$$

Equation (96) simply restates Definition 5. Equation (97) follows by definition of the belief state b'. Equation (98) follows from the definition of $\Gamma'_{\mathbf{h}}$ given by Equation (94). Equation (99) follows by the definition given in the theorem of the piecewise-linear and concave utility potential $\psi'(\mathbf{H}, B(\mathbf{U}))$.

Next we consider the case where C is the only unobserved variable in the domain of ψ. For any belief state $b(C)$, we have $\sum_C b(C) = 1$, of course, and so the following equivalences hold for any input $(\mathbf{h}, b(C))$ for ψ:

$$\sum_{unobserved(C)} \psi(\mathbf{h}, b(C)) = \max_{\gamma \in \Gamma_{\mathbf{h}}} \sum_c \left(\sum_c b(c) \right) \gamma(c) \tag{100}$$

$$= \max_{\gamma \in \Gamma_h} \sum_c \gamma(c). \quad \square \tag{101}$$

Example. Consider a piecewise-linear and concave utility potential, $\psi(D, B(C))$, where D is a Boolean decision variable and C is an unobserved Boolean chance variable. Let the two sets of linear potentials that represent this utility potential be: $\Gamma_{D=0} = \{(6, 7), (8, 2)\}$ and $\Gamma_{D=1} = \{(3, 5), (1, 6)\}$. The result of eliminating C by sum-marginalization is an ordinary utility potential, $\psi'(D)$, represented by the vector: $(13, 8)$. That is, the ordinary potential $\psi'(D)$ has the value 13 when $D = 0$, and 8 when $D = 1$.

Criteria for selecting sum-marginalization operator. In summary, we use the following criteria to determine how to eliminate a chance variable from a utility potential: (i) for an ordinary utility potential, we use the traditional sum-marginalization operator; (ii) for a piecewise-linear and concave utility potential where the chance variable to be eliminated is observed, we use the sum-marginalization operator of Theorem 4; and (iii) for a piecewise-linear and concave utility potential where the chance variable to be eliminated is unobserved, we use the sum-marginalization operator of Theorem 5.

4. Generalized variable elimination

In this section, we introduce a generalization of the variable elimination algorithm for influence diagrams that uses piecewise-linear and concave utility potentials, and the generalized operations on potentials defined in Section 3, to relax traditional constraints on elimination order. Because the algorithm adopts this more general representation of potentials, we call it *generalized variable elimination*.

It is important to note that the new algorithm performs exactly the same steps as the traditional variable elimination algorithm, and represents potentials in exactly the same way, when it eliminates variables in an order that is allowed by the traditional algorithm. All of the differences between generalized and traditional variable elimination relate to how utility potentials are represented and processed when variables are eliminated in an order that is not allowed by the traditional algorithm.

We describe the relaxed constraints on elimination order in Section 4.1. They are not shown in the high-level pseudocode of Algorithm 2, just as the traditional constraints on elimination order are not shown in the pseudocode of Algorithm 1. Although the pseudocode of Algorithm 2 is almost the same as the pseudocode of Algorithm 1, there are significant differences in the implementation of the algorithm that are not shown in the pseudocode. In particular, the pseudocode of Algorithm 2 uses the same notation for utility potentials, and operations on utility potentials, regardless of whether they are ordinary or piecewise-linear and concave. It relies on the implementation of the algorithm to distinguish between these two cases, and handle them appropriately. Recall from Section 3.4.2 that a policy can also be represented in two ways, in keeping with the two different ways that a utility potential can be represented. The pseudocode uses the same notation for a policy δ_V for an eliminated decision variable V, regardless of how the policy is represented, and it relies on the algorithm's implementation to choose the appropriate policy representation.

The only differences shown in the revised pseudocode are in lines 15 through 18, where the differences are highlighted. These added lines identify the piecewise-linear and concave utility potentials that need to be replaced when a variable is

Algorithm 2: Generalized variable elimination algorithm.

Input: Influence diagram with variables $\mathbf{V} = \mathbf{C} \cup \mathbf{D}$
Output: Optimal strategy, Δ, and MEU

$\Phi \leftarrow \{P(C|pa(C)) | C \in \mathbf{C}\}$ // initial set of probability potentials
$\Psi \leftarrow \{R(pa(R)) | R \in \mathbf{R}\}$ // initial set of utility potentials
$\Delta \leftarrow \emptyset$ // initial strategy
for $i \leftarrow 1$ **to** $|\mathbf{V}|$ **do** // i is index of elimination step
  Select variable V to eliminate according to some criterion
  // Process probability potentials
  $\Phi_V \leftarrow \{\phi \in \Phi | V \in dom(\phi)\}$ // get relevant probability potentials
  $\phi_V \leftarrow \prod_{\phi \in \Phi_V} \phi$ // multiply probability potentials
  if *V is a chance variable* **then**
    $\phi_i \leftarrow \sum_V \phi_V$ // eliminate V by sum-marginalization
  else if *V is a decision variable* **then**
    $\phi_i \leftarrow \max_V \phi_V$ // eliminate V by max-marginalization
  $\Phi \leftarrow (\Phi \backslash \Phi_V) \cup \{\phi_i\}$ // update set of probability potentials
  // Process utility potentials
  $\Psi_V \leftarrow \{\psi \in \Psi | (V \in dom(\psi)$ **or** // get relevant utility potentials
      $((\psi$ is piecewise-linear and concave) **and**
      $((V$ is an observed chance variable) **and**
      (V is d-connected to an unobserved chance variable in $dom(\psi))))\}$
  $\psi_V \leftarrow \sum_{\psi \in \Psi_V} \psi$ // add utility potentials
  if *V is a chance variable* **then**
    $\phi_{cond} \leftarrow \phi_V / \phi_i$ // conditional probability of V
    $\psi_i \leftarrow \sum_V \phi_{cond} \cdot \psi_V$ // multiply, then sum-marginalize
  else if *V is a decision variable* **then**
    $\psi_i \leftarrow \max_V \psi_V$ // eliminate V by max-marginalization
    $\delta_V \leftarrow \arg\max_V \psi_V$ // optimal policy for decision variable
    $\Delta \leftarrow \Delta \cup \{\delta_V\}$ // add policy to strategy
  $\Psi \leftarrow (\Psi \backslash \Psi_V) \cup \{\psi_i\}$ // update set of utility potentials
end
$MEU \leftarrow \sum_{\psi \in \Psi} \psi$ // MEU is sum of final utility potentials
return (Δ, MEU) // Δ is optimal strategy

eliminated. We explain this addition to the algorithm in Section 4.2. In Appendix B, we describe an optimization of the algorithm, called *interleaving operations* on piecewise-linear and concave potentials. There, to help explain this optimization, we give more detailed, low-level pseudocode.

4.1. Relaxed constraints on elimination order

We first describe how the new algorithm relaxes constraints on elimination order.

Ordering constraints based on information precedence. As explained in Section 2.2.2, the traditional variable elimination algorithm eliminates variables in an order that is constrained by the partial order $\prec$ of information precedence,

$$\mathbf{Y}_1 \prec D_1 \prec \mathbf{Y}_2 \prec \ldots \prec D_{n-1} \prec \mathbf{Y}_n \prec D_n \prec \mathbf{X}, \tag{102}$$

where each decision variable D_i, or set of chance variables $\mathbf{Y}_i$, is instantiated before all subsequent variables in the partial order, and $\mathbf{X}$ is the set of unobserved chance variables. The traditional algorithm eliminates each set of chance variables $\mathbf{Y}_i$, and each decision variable D_i, before it eliminates their predecessors in the partial order, that is, it eliminates variables in backwards order of instantiation. It also eliminates the unobserved variables in $\mathbf{X}$ before it eliminates any other variable.

Similarly, the generalized algorithm must eliminate variables in an order that is constrained by the partial order of information precedence, but with a key difference. It is not required to eliminate all unobserved variables before it eliminates any other variable. That is, the new algorithm eliminates variables in an order that respects the following partial order of information precedence,

$$\mathbf{Y}_1 \prec D_1 \prec \mathbf{Y}_2 \prec \ldots \prec D_{n-1} \prec \mathbf{Y}_n \prec D_n, \tag{103}$$

where this partial order no longer includes the set $\mathbf{X}$ of unobserved variables.

Dropping the constraint that all unobserved variables must be eliminated before any other variables are eliminated does not contradict the requirement that an elimination order must reflect the partial order of information precedence because the variables in $\mathbf{X}$ are not observed, and thus they provide no information. It follows that the order in which they are eliminated is not subject to informational constraints in the same way as the other variables.

Ordering constraints based on causal precedence. Although we do not require that all unobserved chance variables be eliminated first, there are still constraints on how long their elimination can be postponed. To describe these constraints, we introduce the following definition, which is analogous to the definition of the set of informational predecessors of a decision variable given by Equations (4) and (5).

Definition 6. The set of causal successors of a decision variable D, denoted $Succ(D)$, is the set of variables that are descendants of the decision variable D via some directed path of conditional arcs in the graph of the influence diagram.

Only conditional arcs are used to define causal precedence, just as only informational arcs are used to define information precedence. Elimination-ordering constraints based on causal precedence take the form: *any variable that is a causal successor of a decision variable must be eliminated before the decision variable is eliminated*. Without this constraint, a decision variable could be eliminated from a utility potential before it is eliminated from all probability potentials. In that case, there could be effects of the decision that are not taken into consideration when computing a policy for the eliminated decision variable, and the policy and corresponding utility potential would no longer be guaranteed to be optimal. For exactly this reason, both the traditional and generalized variable elimination algorithms are constrained to eliminate a decision variable from all probability potentials before eliminating it from a utility potential.

Elimination-ordering constraints based on causal precedence complement the revised ordering constraints based on information precedence given by Equation (103). We combine these constraints by requiring that variables be eliminated in an order that is *consistent* with an influence diagram, which we define as follows.

Definition 7. A variable elimination order is *consistent* with an influence diagram if each decision variable is eliminated (i) after all of its causal successors are eliminated and (ii) before any of its informational predecessors are eliminated.

Example. Consider the influence diagram for a three-stage POMDP shown in Fig. 2b. The traditional variable elimination algorithm eliminates the unobserved chance variables, X_1, X_2, and X_3, before it eliminates any other variables. By contrast, the new algorithm can postpone the elimination of the unobserved variables while one or more decision and observed chance variables are eliminated. However, the unobserved state variable X_3 must be eliminated before the decision variable D_2, since is a causal successor of D_2, and the unobserved state variable X_2 must be eliminated before the decision variable D_1, since it is a causal successor of D_1.

4.2. Relevant piecewise-linear and concave utility potentials

When a variable is eliminated from an MEU equation by variable elimination, all potentials with a value that depends on, or may depend on, the eliminated variable, are replaced by equivalent potentials that do not depend on the variable.

We call the potentials that need to be replaced when a variable is selected for elimination the *relevant potentials*. For the traditional variable elimination algorithm, a potential is relevant if and only if the variable selected for elimination is in the domain of the potential. For the generalized algorithm, ordinary potentials are relevant under the same condition. But for piecewise-linear and concave potentials created by eliminating variables in a non-traditional order, the condition for relevance is more complex.

Before describing this more complex condition for relevance, we review the concept of *d-connection* in an influence diagram, which is a well-known graphical criterion for conditional dependence between variables.

Definition 8 *(d-connection in an influence diagram [38,39,17]).* The variables X and Y in an influence diagram are d-connected given a disjoint set of instantiated variables, $\mathbf{Z}$, if there is an undirected path between X and Y such that

- there are no informational arcs or reward nodes on the path;
- in every triple on the path of the form $A \rightarrow B \rightarrow C$ or $A \leftarrow B \leftarrow C$, which represent "serial connections" (the arrows indicate the directions of the conditional arcs), the variable B is not in $\mathbf{Z}$;
- in every triple on the path of the form $A \leftarrow B \rightarrow C$, which represents a "diverging connection," the variable B is not in $\mathbf{Z}$;
- in every triple on the path of the form $A \rightarrow B \leftarrow C$, which represents a "converging connection," the variable B is either in $\mathbf{Z}$ or it is an ancestor of a variable in $\mathbf{Z}$.

The significance of this criterion is that if two variables, X and Y, are d-connected given a disjoint set of instantiated variables, $\mathbf{Z}$, they are conditionally dependent given $\mathbf{Z}$. In that case, observation of the state of one of the variables provides information about the state of the other variable, if it is unobserved. If the two variables are not d-connected given $\mathbf{Z}$, they are said to be *d-separated* by $\mathbf{Z}$, which means they are conditionally independent. There is a well-known linear-time algorithm for testing whether two variables are d-connected or d-separated [39].

The following theorem states the revised conditions for the relevance of a utility potential when a variable V is selected for elimination by generalized variable elimination.

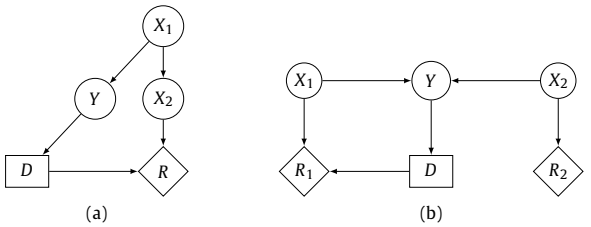


Fig. 4. Simple influence diagrams used to illustrate the conditions under which a utility potential is relevant when a variable is eliminated.

Theorem 6. *A utility potential ψ is relevant when a variable V is selected for elimination by generalized variable elimination if and only if:*

- *V is in the domain of ψ, or*
- *ψ is piecewise-linear and concave, and V is an observed chance variable that is d-connected to an unobserved chance variable C that is in the domain of ψ.*

Proof. A potential ψ is obviously relevant if a variable V in its domain is selected for elimination, and so we consider the case where V is not in the domain of ψ.

If ψ is piecewise-linear and concave, its value depends on a belief state over the unobserved chance variables in its domain, and so information that influences the belief state influences the value of ψ. If V is an observed chance variable that is d-connected to an unobserved chance variable C in the domain of ψ, then observation of the state of V provides information that influences the belief state upon which the value of ψ depends, and ψ is relevant when V is selected for elimination.

The value of a piecewise-linear and concave potential ψ cannot be affected by an unobserved chance variable that is not in its domain because an unobserved variable provides no information, and thus cannot affect a belief state.The value of ψ cannot be affected by a decision variable that is not in its domain by the following reasoning: when a decision variable is selected for elimination, its descendants have already been eliminated, and it is d-separated from its predecessors. Therefore, it cannot be d-connected to an unobserved chance variable in the domain of ψ. □

In summary, an observed chance variable can influence the value of a piecewise-linear and concave potential even if it is not in its domain, if it influences a belief about the state of an unobserved chance variable that is in its domain. To illustrate this condition for the relevance of a utility potential, we consider two examples.

Examples. Consider the influence diagram shown in Fig. 4a. If the first variable eliminated is the decision variable D, the reward potential $R(X_2, D)$ is replaced by a piecewise-linear and concave utility potential $\psi_1(B(X_2))$. If the next variable eliminated is the observed chance variable Y, then the piecewise-linear and concave utility potential $\psi_1(B(X_2))$ is relevant even though Y is not in its domain, because Y is d-connected to the unobserved chance variable X_2, which *is* in its domain.

Next consider the influence diagram shown in Fig. 4b. If the decision variable D is eliminated first, the utility potential $R_1(X_1, D)$ is replaced by a piecewise-linear and concave utility potential $\psi_1(B(X_1))$. Let the next variable eliminated be the observed chance variable Y, and note that Y is not in the domain of either the piecewise-linear and concave utility potential $\psi_1(B(X_1))$ or the ordinary utility potential $R(X_2)$. Yet it is d-connected to the unobserved chance variable X_1 in the domain of $\psi_1(B(X_1))$, and it is d-connected to the unobserved chance variable X_2 in the domain of $R(X_2)$. In this case, $\psi_1(B(X_1))$ is relevant because it is piecewise-linear and concave, while $R(X_2)$ is *not* relevant because it is an ordinary utility potential.

4.3. Oil wildcatter example revisited

To illustrate how generalized variable elimination uses piecewise-linear and concave utility potentials in solving an influence diagram, we consider the steps it takes to solve the same oil wildcatter problem used as an example in Section 2.2.2. Instead of eliminating the unobserved chance variable first, however, we show how the new algorithm solves this problem when the unobserved chance variable is eliminated last.

The initialization step is the same as described in Section 2.2.2 for the traditional algorithm, and so it is enough to show the initial MEU equation:

$$MEU = \max_T \sum_S \max_D \sum_O P(S|O,T)P(O)\left(R(T) + R(O,D)\right). \tag{104}$$

Eliminate decision variable D (for Drill). We eliminate the decision variable D first. There are no probability potentials with D in their domain. But the utility potential $R(O, D)$ has D in its domain. Since it is an ordinary utility potential that also has

an unobserved chance variable in its domain, we use Theorem 1 to eliminate D. The result is creation of a piecewise-linear and concave utility potential, $\psi_1(B(O))$, which is represented by a set Γ_1 of linear potentials, with one linear potential in the set for the *drill* decision, and one for the decision not to drill (*nodrill*).

When the problem is solved for the parameters specified in Fig. 1, we have $\Gamma_1 = \{(0,0,0), (-70,50,200)\}$, where the vector $(0,0,0)$ represents the linear potential associated with the decision not to drill, and the vector $(-70,50,200)$ represents the linear potential associated with the decision to drill. Note that the three elements of these vectors correspond to the three possible states of the unobserved chance variable O for oil, in the order: *dry*, *wet*, and *soak*. For any belief state $b(O) \in bsp(O)$, the value of the piecewise-linear and concave utility potential is given as follows:

$$\psi_1(b(O)) = \max_{\gamma \in \Gamma_1} \sum_{o \in sp(O)} b(o)\gamma(o). \tag{105}$$

After this elimination step, we have the potentials,

$$\Phi \leftarrow \{P(O), P(S|O,T)\} \tag{106}$$

$$\Psi \leftarrow \{R(T), \psi_1(B(O))\}, \tag{107}$$

and the revised MEU equation is

$$MEU = \max_T \sum_S \max_{\gamma \in \Gamma_1} \sum_O P(S|O,T)P(O)\left(R(T) + \gamma(O)\right), \tag{108}$$

where the part of the equation that represents a piecewise-linear and concave utility potential is highlighted.

A policy $\delta_D(B(O))$ for the decision variable D is represented implicitly by associating the corresponding action $d \in sp(D)$ with each linear potential in the set Γ_1, which represents the piecewise-linear and concave utility function $\psi_1(B(O))$. This representation of a policy is defined by Equation (86).

Eliminate observed chance variable S (for Seismic test result). The next variable eliminated is the observed chance variable S. The only probability potential with S in its domain is $P(S|O,T)$. Eliminating S from $P(S|O,T)$ by sum-marginalization gives

$$\phi_2(O,T) = \sum_S P(S|O,T), \tag{109}$$

which is not added to the set Φ because it is vacuous, that is, it assigns the value of 1 to every instantiation of (O,T). Thus we also have $\phi_{cond}(S|O,T) = P(S|O,T)$.

The observed chance variable S is not in the domain of any utility potential. But it is d-connected to the unobserved chance variable O, which is in the domain of the piecewise-linear and concave utility potential $\psi_1(B(O))$. Therefore, $\psi_1(B(O))$ is relevant in this elimination step, and it is replaced by the piecewise-linear and concave utility potential,

$$\psi_2(T, B(O)) = \sum_S \phi_{cond}(S|O,T) \cdot \psi_1(B(O)), \tag{110}$$

which is represented by an indexed family of sets of linear potentials, $\{\Gamma_{2,t}\}_{t \in sp(T)}$, created by the cross-sum operator described in Theorem 4. For any relevant history $t \in sp(T)$ and belief state $b(O) \in bsp(O)$, we have

$$\psi_2(t, b(O)) = \max_{\gamma \in \Gamma_{2,t}} \sum_{o \in sp(O)} b(o)\gamma(o). \tag{111}$$

When the problem is solved for the parameters specified in Fig. 1, we have the set $\Gamma_{2,notest} = \{(0,0,0), (-70,50,200)\}$ for the case where no seismic test is performed. In this set, the vector $(0,0,0)$ represents the linear potential associated with the decision not to drill after not testing, and the vector $(-70,50,200)$ represents the linear potential associated with the decision to drill after not testing. We also have the set $\Gamma_{2,test} = \{(0,0,0), (-70,50,200), (-28,35,180), (-7,15,100)\}$ for the case where a seismic test is performed. In this set, the vector $(0,0,0)$ represents the linear potential associated with the decision not to drill regardless of the test result, the vector $(-70,50,200)$ represents the linear potential associated with the decision to drill regardless of the test result, the vector $(-28,35,180)$ represents the linear potential associated with the decision to drill if the test result is *closed* or *open*, but not *diffuse*, and the vector $(-7,15,100)$ represents the linear potential associated with the decision to drill if and only if the test result is *closed*. Note that before pruning, the set $\Gamma_{2,test}$ has eight vectors. The four pruned vectors correspond to conditional plans that are not optimal for any belief state. For example, the vector that represents the linear potential corresponding to the decision to drill if the result of the seismic test is *diffuse*, but not otherwise, is pruned because it is never the best thing to do.

After this elimination step, we have the potentials,

$$\Phi \leftarrow \{P(O)\} \tag{112}$$

$$\Psi \leftarrow \{R(T), \psi_2(T, B(O))\}, \tag{113}$$

and the revised MEU equation is

$$MEU = \max_T \max_{\gamma \in \Gamma_{2,T}} \sum_O P(O) \Big(R(T) + \gamma(O) \Big). \tag{114}$$

Eliminate decision variable T (for Test). No probability potentials have the variable T in their domain, but two utility potentials do: the ordinary utility potential $R(T)$ and the piecewise-linear and concave utility potential $\psi_2(T, B(O))$. Therefore, a new piecewise-linear and concave utility potential is created, as follows,

$$\psi_3(B(O)) = \max_T \big(R(T) + \psi_2(T, B(O)) \big), \tag{115}$$

which is represented by the set of linear potentials, Γ_3, created by the maximization operator described in Theorem 3.

When the problem is solved for the parameters specified in Fig. 1, we have $\Gamma_3 = \{(0, 0, 0), (-70, 50, 200), (-38, 25, 170), (-17, 5, 90)\}$, where the vector $(0, 0, 0)$ represents the linear potential associated with the decision not to test or drill, the vector $(-70, 50, 200)$ represents the linear potential associated with the decision to drill without testing first, the vector $(-38, 25, 170)$ represents the linear potential associated with the decision to test and then drill if the result of the test is *open* or *closed*, and the vector $(-17, 5, 90)$ represents the linear potential associated with the decision to test and then drill if the result of the test is *closed*. (Before pruning, the set has six vectors, since it is the union of the sets $\Gamma_{2,notest}$ and $\Gamma_{2,test}$.) As before, for any belief state $b(O) \in bsp(O)$, the value of the piecewise-linear and concave utility potential is:

$$\psi_3(b(O)) = \max_{\gamma \in \Gamma_3} \sum_{o \in sp(O)} b(o)\gamma(o). \tag{116}$$

After this elimination step, we have the potentials,

$$\Phi \leftarrow \{P(O)\} \tag{117}$$

$$\Psi \leftarrow \{\psi_3(B(O))\}, \tag{118}$$

and the revised MEU equation is:

$$MEU = \max_{\gamma \in \Gamma_3} \sum_O P(O)\, \gamma(O). \tag{119}$$

A policy $\delta_T(B(O))$ for the decision variable T is represented implicitly by associating an action $t \in sp(T)$ with each linear potential in the set Γ_3 that represents the piecewise-linear and concave utility potential $\psi_3(B(O))$, as defined by Equation (86).

Eliminate unobserved chance variable O (for Oil). The unobserved chance variable O is eliminated last, as follows,

$$\psi_4(\lambda) = \sum_O P(O)\psi_3(B(O)), \tag{120}$$

where λ is the unique state of the empty set of variables in the domain of ψ_4. For this problem, the prior probability distribution for the unobserved chance variable O for oil is $(0.5, 0.3, 0.2)$, where the probabilities for the possible states of the variable are in the order: *dry*, *wet*, and *soak*. The linear potential in Γ_3 that optimizes this belief state is represented by the vector $(-38, 25, 170)$, which corresponds to the strategy of testing, and then drilling if the result of the test is open or closed. The scalar value of this strategy is 22.5, which is the maximum expected utility (MEU) for the problem.

Return solution. In the optimal strategy, $\Delta = (\delta_T(B(O)), \delta_D(B(O))$, the optimal policies for the decision variables T and D are represented implicitly by way of the piecewise-linear and concave utility potentials generated when they are eliminated.

4.4. Generalized influence diagram

Our generalization of the variable elimination algorithm also suggests a potentially useful generalization of the definition of an influence diagram.

In Definition 2 of a completely observable MDP, and in Definition 3 of a POMDP, every chance variable is associated with a probability distribution. It is associated with a conditional probability distribution if it has one or more parent

variables, and with an unconditional probability distribution otherwise. Associating every chance variable with a probability distribution ensures that the definitions of an MDP and a POMDP are consistent with the definition of an influence diagram given by Definition 1.

In the literature on Markov decision processes, however, it is often the case that a chance variable that belongs to the first stage of the process, and does not have a parent variable, is *not* associated with an unconditional (that is, prior) probability distribution. Instead, the value iteration algorithm solves the problem for *all* possible initial states. In the completely observable case, value iteration solves the problem for all possible states of the observed chance variable that represents the state of the process, for each stage of the problem, including the first stage. In the partially observable case, value iteration solves the problem for all possible probability distributions (or belief states) over the possible states of the unobserved chance variable that represents the state of the process, for each stage of the process, including the first.

Because generalized variable elimination generalizes the value iteration approach, it can similarly solve an influence diagram when no probability distribution is associated with one or more chance variables, as long as the chance variables do not have a parent variable. In a sense, any chance variable that does not have a parent variable belongs to the "first stage" of the problem represented by an influence diagram. Thus we can define an influence diagram in a slightly more general way.

Definition 9. A generalized influence diagram is an influence diagram that is defined the same as in Definition 1, except that the association of an unconditional probability distribution with a chance variable that has no parent variable is optional.

A chance variable that is not associated with a probability distribution cannot be eliminated by variable elimination. But in that case, once all the variables have been eliminated that can be eliminated, the value (or MEU) of the influence diagram is given by a utility potential that has the remaining chance variables in its domain, which are the chance variables that are *not* associated with a probability distribution. If all of these chance variables are observed, the final utility potential is an ordinary potential. Otherwise, it is piecewise-linear and concave. In either case, the final utility potential gives the MEU for *every* possible initial state and initial belief state of the problem.

For example, consider a modified influence diagram for the oil wildcatter problem, where the only difference is that no prior probability distribution is associated with the unobserved chance variable O for the presence of oil. In this case, the variable for oil is never eliminated by generalized variable elimination, and the final utility potential includes it in its domain, and maps every possible prior probability distribution over the possible states of this variable, that is, every possible belief state, to a value.

In Section 4.3, we considered the steps taken by generalized variable elimination in solving the oil wildcatter problem when the unobserved chance variable O for oil is eliminated last. The same steps are taken by generalized variable elimination when no prior probability distribution is associated with O, except that the variable O is never eliminated. In this case, we can interpret the steps of the algorithm a little differently, as follows. When the symbol MEU appears on the left-hand side of equations in Section 4.3, it always denotes a scalar value. But if the unobserved chance variable O for oil has no unconditional probability distribution associated with it, the left-hand side of the same equations becomes $MEU(B(O))$, and we are solving for a utility potential that gives the maximum expected utility for *any* initial belief state $b(O)$ over the possible states of the unobserved chance variable for oil, and not just for a particular prior probability distribution $P(O)$. When the problem is solved in this more general form, the same sequence of elimination steps transforms the MEU equation as follows,

$$MEU(B(O)) = \max_T \sum_S \max_D \sum_O P(S|O,T)b(O)\,(R(T) + R(O,D)) \tag{121}$$

$$= \max_T \sum_S \max_{\gamma \in \Gamma_1} \sum_O P(S|O,T)b(O)\,(R(T) + \gamma(O)) \tag{122}$$

$$= \max_T \max_{\gamma \in \Gamma_{2,T}} \sum_O b(O)\,(R(T) + \gamma(O)) \tag{123}$$

$$= \max_{\gamma \in \Gamma_3} \sum_O b(O)\gamma(O), \tag{124}$$

where the variable O is never eliminated.

For some influence diagrams, this generalization of the optimization problem may be more difficult to solve. But it can also be useful to have a more general solution, and the revised definition of an influence diagram, and the generalized variable elimination algorithm, allow the problem to be solved in this more general way.

4.5. Correctness

From the results and analysis presented so far, the correctness of the generalized variable elimination algorithm follows in a straightforward way.

Theorem 7. *Generalized variable elimination finds an optimal strategy and value for an influence diagram.*

Proof. When variables are eliminated in a traditional order, the generalized algorithm performs exactly the same steps as the traditional algorithm. Therefore, we only need to establish correctness for the case where variables are eliminated in a non-traditional, but consistent order, using piecewise-linear and concave utility potentials.

Theorem 6 ensures that any piecewise-linear and concave utility potential that is affected by the value of an eliminated variable, and needs to be replaced during the elimination step, is identified. Theorems 2 through 5 establish that the operations on piecewise-linear and concave utility potentials are value-preserving. It follows that any elimination step involving piecewise-linear and concave utility potentials is a value-preserving transformation, which means the value of the equation for MEU is the same before and after the variable is eliminated.

The algorithm terminates after a number of steps that is less than or equal to the number of variables. By the value-preserving property of the operations on potentials, the value of the MEU equation after the last variable is eliminated is the value of an optimal strategy for the influence diagram, and the policy associated with each decision variable is optimal. □

5. Generalized value iteration

The generalized variable elimination algorithm introduced in Section 4 generalizes the traditional variable elimination algorithm for influence diagrams. But it can also be viewed as a generalization of the value iteration algorithm for finite-horizon POMDPs. In this section, we consider the new algorithm from this alternative perspective.

Of course, there is a straightforward way in which the new algorithm can be viewed as a generalization of the value iteration algorithm for POMDPs: it solves any influence diagram, and not just influence diagrams that represent finite-horizon POMDPs. In this section, we show in a more precise way how the new algorithm generalizes the value iteration approach. In Section 5.1, we show that generalized variable elimination solves any influence diagram by reducing it to an equivalent finite-horizon POMDP that it solves by value iteration. In Section 5.2, we show that generalized variable elimination enhances the value iteration approach by leveraging problem structure that can be represented in an influence diagram, but not in the traditional model of a POMDP.

5.1. Variable elimination as generalized value iteration

We first show that generalized variable elimination reduces the decision problem represented by an influence diagram to an equivalent finite-horizon POMDP with special structure, called a mixed-observable MDP, that it solves by value iteration.

5.1.1. Mixed-observable Markov decision process

A mixed-observable Markov decision process [40,41] is a special type of factored POMDP where the state of the process at each stage is factored into two state variables, one observed and the other unobserved, and imperfect information about the state of the unobserved variable is provided by an additional observation variable. (Recall that an *observation variable* is a chance variable that is distinguished from a state variable by the fact that all of its outgoing arcs are informational.)

Definition 10. A finite-horizon mixed-observable Markov decision process (MOMDP) is a tuple $(\mathbf{Y}, \mathbf{X}, \mathbf{Y}^X, \mathbf{D}, \mathcal{R}, \mathcal{P})$, where

- the process unfolds over a finite sequence of n stages, indexed by $t = 1, 2, \ldots, n$;
- $\mathbf{Y} = \{Y_t | t = 1, 2, \ldots, n\}$ is a set of observed chance variables that represent the observed component of the state of the process at each stage;
- $\mathbf{X} = \{X_t | t = 1, 2, \ldots, n\}$ is a set of unobserved chance variables that represent the unobserved component of the state of the process at each stage;
- $\mathbf{Y}^X = \{Y_t^X | t = 1, 2, \ldots, n\}$ is a set of observation variables that provide imperfect information about the state of the unobserved variables $\mathbf{X}$;
- $\mathbf{D} = \{D_t | t = 1, 2, \ldots, n\}$ is a set of decision variables, with one for each stage;
- $\mathcal{R} = \{R_t : sp(Y_t) \times sp(X_t) \times sp(D_t) \rightarrow \Re | t = 1, 2, \ldots, n\}$ is a set of reward functions, with one for each stage; and
- $\mathcal{P}$ is a set of probability distributions, with one for each chance variable. For the first stage, there is an unconditional probability distribution $P(Y_1)$, an unconditional probability distribution $P(X_1)$, and a conditional probability distribution $P(Y_1^X | X_1)$. For each subsequent stage $t = 2, \ldots, n$, there is a conditional probability distribution $P(Y_t | Y_{t-1}, X_{t-1}, D_{t-1})$, a conditional probability distribution $P(X_t | Y_{t-1}, X_{t-1}, D_{t-1})$, and a conditional probability distribution $P(Y_t^X | X_t, D_{t-1})$.

At each stage $t = 1, \ldots, n$, the state of the process, (y_t, x_t), is the state of a joint variable, (Y_t, X_t), where $y_t \in sp(Y_t)$ is the observed state and $x_t \in sp(X_t)$ is the unobserved state. After an action $d_t \in sp(D_t)$ is taken, a reward $R_t(y_t, x_t, d_t)$ is received. In every stage except the last, the process then makes a transition to an observed state $y_{t+1} \in sp(Y_{t+1})$ with probability $P(y_{t+1} | y_t, x_t, d_t)$, and it makes a transition to an unobserved state $x_{t+1} \in sp(X_{t+1})$ with probability $P(x_{t+1} | y_t, x_t, d_t)$. Fig. 5a shows an influence diagram for a three-stage mixed-observable MDP.

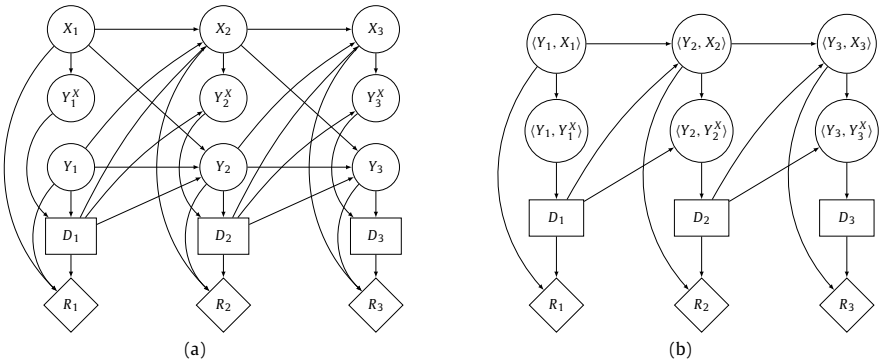


Fig. 5. Influence diagrams for (a) a three-stage mixed-observable MDP and (b) an equivalent POMDP.

5.1.2. Mixed-observable MDP as a POMDP

Like any factored POMDP, a mixed-observable MDP can be represented by an equivalent "flat" POMDP that is defined in accordance with Definition 3. Fig. 5b shows the influence diagram for a three-stage POMDP that is equivalent to the three-stage mixed-observable MDP shown in Fig. 5a. The state at stage t is an instantiation (y_t, x_t) of a joint variable (Y_t, X_t); the decision at stage t is $d_t \in sp(D_t)$; and the observation at stage-$(t+1)$ is an instantiation (y_{t+1}, y_{t+1}^X) of a joint variable (Y_{t+1}, Y_{t+1}^X). The same chance variable Y_t is part of both the state and the observation at stage t because it represents a component of the state that is perfectly observed.

The conditional probabilities and rewards for the equivalent POMDP are easy to calculate from those for the mixed-observable model given by Definition 10. The transition probabilities for the state variable (Y_{t+1}, X_{t+1}) are defined as

$$P((y_{t+1}, x_{t+1})|(y_t, x_t), d_t) = P(y_{t+1}|y_t, x_t, d_t) \cdot P(x_{t+1}|y_t, x_t, d_t), \tag{125}$$

the observation probabilities for the observation variable (Y_{t+1}, Y_{t+1}^X) are defined as

$$P((y_{t+1}, y_{t+1}^X)|(y_{t+1}, x_{t+1}), d_t) = P(y_{t+1}^X|x_{t+1}, d_t), \tag{126}$$

and the stage-t reward function is defined as

$$R((y_t, x_t), d_t) = R(y_t, x_t, d_t). \tag{127}$$

From this perspective, a mixed-observable MDP is just a POMDP with a special kind of structure. We next show that this special structure can be leveraged to improve the efficiency of both the belief state update and the value iteration algorithm.

Hybrid state and belief update. We first explain the more efficient belief update. Since the state of a mixed-observable MDP at stage t is a joint state (y_t, x_t), where y_t is observed and x_t is not, a Bayesian belief update only needs to maintain a belief state over the possible states of the unobserved state variable X_t. We define the *hybrid state* of a mixed-observable process at stage t as a tuple $(y_t, b(X_t))$, where $b(X_t)$ is a belief state over the possible states of the unobserved state variable X_t. For a hybrid state $(y_{t-1}, b(X_{t-1}))$ and action d_{t-1}, the stochastic outcome is a hybrid state $(y_t, b(X_t))$. The successor belief state, $b_t = b(X_t)$, of this hybrid state is given by a deterministic function, $b_t = \tau(b_{t-1}, y_{t-1}, d_{t-1}, y_t^X)$, where each component of b_t is defined as:

$$b_t(x_t) = P(x_t|b_{t-1}, y_{t-1}, d_{t-1}, y_t^X) \tag{128}$$

$$= \frac{P(y_t^X, x_t|b_{t-1}, y_{t-1}, d_{t-1})}{P(y_t^X|b_{t-1}, y_{t-1}, d_{t-1})} \tag{129}$$

$$= \frac{P(y_t^X|x_t, d_{t-1}) \sum_{x_{t-1}} P(x_t|x_{t-1}, y_{t-1}, d_{t-1}) b_{t-1}(x_{t-1})}{\sum_{y_t^X} P(y_t^X|x_t, d_{t-1}) \sum_{x_{t-1}} P(x_t|x_{t-1}, y_{t-1}, d_{t-1}) b_{t-1}(x_{t-1})}. \tag{130}$$

Dynamic programming recurrence. The value iteration algorithm for mixed-observable MDPs is based on the reduction of a mixed-observable MDP to a completely observable MDP over hybrid states, and follows a similar logic as the reduction of a POMDP to a completely observable MDP over belief states. For stages $t = 1, 2, \ldots, n-1$ of the process, the dynamic programming recurrence takes the form,

$$V_t(y_t, b_t) = \max_{d_t} \left\{ R_t((y_t, b_t), d_t) + \sum_{y_{t+1}, y_{t+1}^X} P(y_{t+1}|(y_t, b_t), d_t) V_{t+1}(y_{t+1}, b_{t+1}) \right\}, \tag{131}$$

where $b_{t+1} = \tau((y_t, b_t), d_t, y^X_{t+1})$. For the last stage of the process, which is the base case of the dynamic programming recurrence, we have

$$V_n(y_n, b_n) = \max_{d_n} R_n((y_n, b_n), d_n). \tag{132}$$

Appendix A.1 describes a value iteration algorithm for mixed-observable MDPs that improves the efficiency of the incremental pruning algorithm for POMDPs by taking advantage of the special structure of this dynamic programming recurrence.

5.1.3. Reduction of an influence diagram to an equivalent mixed-observable MDP

There is an important and obvious similarity between the dynamic programming recurrence for a mixed-observable MDP and the dynamic programming recurrence solved by generalized variable elimination: each of these recurrences is defined for both the state of an observed variable and a belief state over the possible states of an unobserved variable. The following theorem, proved in Appendix A.2, leverages this similarity to establish a fundamental relationship between the influence diagram solved by generalized variable elimination and the mixed-observable model.

Theorem 8. *The generalized variable elimination algorithm reduces any influence diagram to an equivalent mixed-observable MDP that it solves by value iteration.*

Also proved in Appendix A.2 is the following special case of this result.

Corollary 2. *The traditional variable elimination algorithm reduces any influence diagram to an equivalent completely observable MDP that it solves by value iteration.*

Although both results reduce variable elimination to a form of value iteration, the difference between the two reductions explains why generalized variable elimination can be much more effective than traditional variable elimination in solving partially observable decision problems. It can leverage a dynamic programming recurrence that is defined for belief states, or for both belief states and history, whereas the recurrence solved by traditional variable elimination is defined only for observed states, that is, for history. By optimizing utility for all belief states over the possible states of an unobserved variable, generalized variable elimination can decouple later stages of a sequential decision problem from earlier stages, allowing the problem to be solved by the same kind of stagewise problem decomposition that is leveraged by the dynamic programming approach to solving POMDPs and mixed-observable MDPs. As a result, generalized variable elimination can be more scalable than traditional variable elimination in solving partially observable decision problems.

5.2. Variable elimination as enhanced value iteration

Theorem 8 and Corollary 2 show that both traditional and generalized variable elimination reduce an influence diagram to an equivalent MDP that is solved by value iteration. From this perspective, the variable elimination approach to solving influence diagrams can be viewed as a value iteration algorithm that is enhanced by leveraging problem structure that can be represented in an influence diagram, but is not represented in the traditional model of an MDP. In particular, variable elimination enhances the value iteration approach in the following ways.

- Variable elimination can compute probabilities and expected utilities more efficiently than traditional value iteration by leveraging conditional independence relations among variables, and additive separability of the utility function, as represented in the graph of an influence diagram.
- Variable elimination can solve non-Markovian decision problems by using state augmentation to convert an influence diagram that represents a non-Markovian problem to an equivalent MDP that it solves by value iteration. As already shown, generalized variable elimination converts an influence diagram to an equivalent mixed-observable MDP, and traditional variable elimination converts an influence diagram to an equivalent completely observable MDP.
- Variable elimination can decompose a problem into independent subproblems that can be solved separately before combining their solutions, where subproblem independence is represented in the graph of an influence diagram.

The first of these enhancements is well-understood, and has already been extensively explored in related work on factored MDPs [e.g., 42–44] and factored POMDPs [e.g., 12,13]. The second and third enhancements leverage more complex forms of problem structure, including non-Markovian dependencies, and these enhancements have received much less study. Therefore we focus on these enhancements in the rest of our discussion.

5.2.1. Automatic state augmentation for non-Markovian problems

We begin by considering non-Markovian decision problems that can be represented by an influence diagram. It is well-known that any finite-horizon non-Markovian decision process can be converted to an equivalent finite-horizon MDP by *state augmentation*, which means the state at a given stage of the process is made Markovian by augmenting it with part

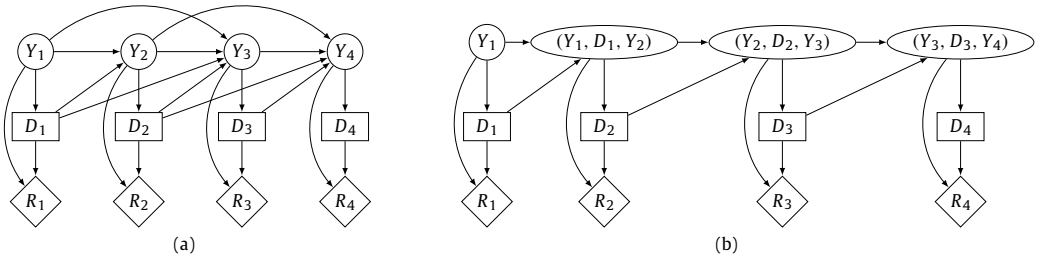


Fig. 6. Influence diagrams for (a) a four-stage completely observable decision process with a one-stage time lag and (b) an equivalent four-stage completely observable MDP. Variable elimination solves the influence diagram on the left by converting it to the MDP on the right, which it solves by value iteration.

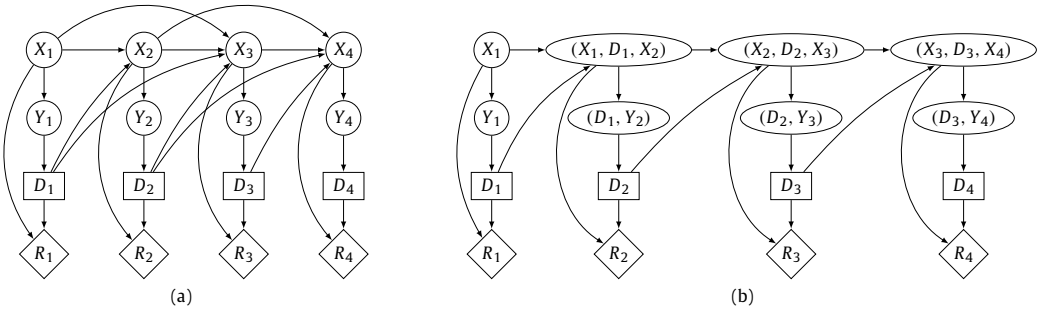


Fig. 7. Influence diagrams for (a) a four-stage *partially observable* decision process with a one-stage time lag, and (b) an equivalent four-stage mixed-observable MDP that is created when variables are eliminated in the order: $D_4, X_4, Y_4, D_3, X_3, Y_3$, etc. Generalized variable elimination solves the influence diagram on the left by converting it to the mixed-observable MDP on the right, which it solves by value iteration.

or all of the history of the process. However, this conversion often requires human involvement and insight to reformulate the problem appropriately. Modeling a non-Markovian decision problem as an influence diagram has the advantage that the variable elimination algorithm itself *automatically* converts the problem to an equivalent MDP, as established by Theorem 8 and its corollary.

For example, consider the influence diagram shown in Fig. 6a, which is adapted from an example given by Tatman and Shachter to make a similar point about the advantages of modeling non-Markovian decision problems as influence diagrams in the completely observable case [4, p. 377]. The influence diagram shown in Fig. 6a represents a completely observed decision process where the state transition probabilities depend not only on the current state and action, but also on the previous state and action. Tatman and Shachter call it a decision process with a *one-stage time lag*. For this non-Markovian problem, note that there is only one valid elimination order. The variable elimination algorithm (like Shachter and Tatman's node reduction algorithm) solves this problem by automatically using state augmentation to create an equivalent completely observable MDP, where the state of the equivalent MDP at stage t is a tuple of the current state Y_t, previous state Y_{t-1}, and previous decision D_{t-1}. The influence diagram shown in Fig. 6b represents the equivalent completely observable MDP. For stages 2, 3, and 4 of this new influence diagram, the state variables are labeled by a subset of state variables from the original influence diagram shown in Fig. 6a. (Of course, when variable elimination converts the influence diagram shown in Fig. 6a to the equivalent finite-horizon MDP shown in Fig. 6b, it also computes the transition probabilities and reward functions of the equivalent MDP. For details, see the proof of Theorem 8 in Appendix A.2.)

Fig. 7a shows an influence diagram for a four-stage partially observable process with a one-stage time lag. It is the same as Tatman and Shachter's example in Fig. 6a except the state of the process is partially observed. For this problem, the traditional variable elimination algorithm would use state augmentation to create an equivalent completely observed MDP. But the augmented state for each stage would consist of the entire history of the process! Unlike traditional variable elimination, generalized variable elimination can eliminate the variables of this problem in the order: D_4, X_4, Y_4, D_3, X_3, Y_3, D_2, X_2, Y_2, D_1, X_1, Y_1. When it does so, it uses state augmentation to create an equivalent mixed-observed MDP where the augmented state for each stage t consists of the unobserved variables, X_t and X_{t-1}, and the previous decision, D_{t-1}. The part of the state that corresponds to the previous decision D_{t-1} is observed directly, and the observed variable Y_t provides imperfect information about the state of the unobserved state variables X_t and X_{t-1}. Thus the value function for each stage t of this mixed-observable MDP is piecewise-linear and concave, and it is represented by an indexed family of sets of linear potentials, $\{\Gamma_d^t\}_{d \in sp(D_{t-1})}$, where the domain of the linear potentials in each set Γ_d^t consists of the unobserved variables X_t and X_{t-1}. In many cases, and especially as the number of stages of the process increases, this mixed-observable MDP can be solved more efficiently by generalized variable elimination than traditional variable elimination can solve the equivalent completely observable MDP over the entire history of the process.

When the augmented state of an equivalent mixed-observable MDP contains variables from just the k most recent stages of the process, as it does in this case (where $k = 2$), it is called a *k-order mixed-observable MDP*. The class of k-order mixed-observable MDPs includes many problems of practical importance, including problems with delayed observations, delayed action effects, and delayed rewards, where the delay is bounded by k. There has been considerable work on how to use state augmentation to reformulate k-order decision processes as equivalent MDPs in the completely observable case [e.g., 45,46]. There has been some, though much less, exploration of this topic in the partially observable case [47]. Our integrated approach to solving influence diagrams and POMDPs offers a promising direction for further exploration of how best to represent and solve k-order partially observable decision problems.

Example: Maze navigation with one-stage time lag. As an example of how generalized variable elimination can use state augmentation to convert an influence diagram that represents a non-Markovian problem to an equivalent mixed-observable MDP that can be solved by value iteration, we consider a modified version of the ten-stage maze POMDP introduced in Section 2.3.5 The modified problem differs from the original problem in just one way: if the robot bumps into a wall, it cannot move in the next stage of the problem. With this simple change, the transition probabilities for the problem depend not only on the current state and action. They also depend on the previous state and action, since they depend on whether the robot bumped into a wall in the previous stage. Therefore, the problem can be represented by a ten-stage version of the influence diagram shown in Fig. 7a, which has a one-stage time lag.

The traditional variable elimination algorithm solves this problem in the same way that it solves a POMDP. Before it eliminates the last decision variable, it eliminates all unobserved chance variables. The effect is to convert the problem to an equivalent completely observable MDP over the full history of the process. By contrast, generalized variable elimination can eliminate the last decision variable before eliminating any other variables. Because the problem has a one-stage time lag, the piecewise-linear and concave utility potential created when the last decision variable D_{10} is eliminated has the decision variable D_9, and the unobserved chance variables X_9 and X_{10}, in its domain. Therefore, it is represented by $|sp(D_9)| = 4$ sets of $|sp(D_{10})| = 4$ linear potentials, where each linear potential is a function of $|sp(X_9)| \cdot |sp(X_9)| = 23^2$ unobserved states. If we define the size of a potential as the number of scalars used to represent it, the size of the utility potential created when the last decision variable is eliminated by generalized variable elimination before eliminating any other variable is $4^2 \cdot 23^2$. By contrast, the size of the utility potential created when the last decision variable is eliminated by traditional variable elimination, after eliminating all unobserved chance variables, is 52^{10}, which is several orders of magnitude larger!

Of course, the maze problem with a one-stage time lag is more difficult to solve by generalized variable elimination than the original maze problem, which is Markovian. When the original problem is solved, the utility potential created when the last decision variable is eliminated only has size $|sp(D_{10})| \cdot |sp(X_{10}) = 4 \cdot 23$, which is much smaller than the $|sp(D_9)| \cdot |sp(D_{10})| \cdot |sp(X_9)| \cdot |sp(X_{10})| = (4 \cdot 23)^2$ size of the utility potential created when the problem with a one-stage time lag is solved. Nevertheless, for the time-lagged problem, the optimal utility potential and policy for the last decision variable are computed *much* more efficiently (and represented more compactly) by generalized variable elimination than by traditional variable elimination.

As additional decision variables and observed chance variables are eliminated by generalized variable elimination in solving this problem, the size of the piecewise-wise linear and concave potentials may grow larger, and potentially *much* larger. But note that generalized variable elimination can always reduce their size by eliminating unobserved chance variables. Once it eliminates all remaining unobserved chance variables, it performs identically to traditional variable elimination from that point forward.

Because generalized variable elimination can perform *much* better than traditional variable elimination in eliminating the last decision variable (and possibly the last several decision and observed chance variables), and can then eliminate the remaining variables in the same order as traditional variable elimination, it can always perform at least as well as traditional variable elimination, and potentially better. How much better depends on both the elimination order and the effect of pruning on the size of the piecewise-linear and concave utility potentials.

5.2.2. Subproblem independence and factored representation of value function

We next consider another form of problem structure that is leveraged by generalized variable elimination, but is not leveraged by the traditional value iteration approach.

Consider the two influence diagrams shown in Figs. 8a and 8b. One represents a completely observable decision process and the other a partially observable process. At first glance, the influence diagrams appear to have the same non-Markovian structure considered in Section 5.2.1. The stage-1 nodes have outgoing arcs to both the stage-2 nodes and the stage-3 nodes, as in the case of a one-stage time lag. However, there is a key difference: there is no arc from any node in stage 2 to any node in stage 3, except for an arc from decision node D_2 to decision node D_3, which is included only to ensure a total ordering of decision nodes, and otherwise has no effect. In fact, the subproblem corresponding to stage 2 and the subproblem corresponding to stage 3 are *independent subproblems* of the problem corresponding to stage 1.

Variable elimination can leverage subproblem independence to further simplify the dynamic programming recurrence it solves. To help show this, we consider a slightly-modified version of the variable elimination algorithm that more closely resembles the value iteration approach. The pseudocode for the revised algorithm is shown in Algorithm 3, with the differences from the pseudocode of Algorithms 1 and 2 highlighted. All of the differences relate to a single modification: the utility potential ψ_{i-1} generated in elimination step $i - 1$ is always included in the set of relevant utility potentials in elim-

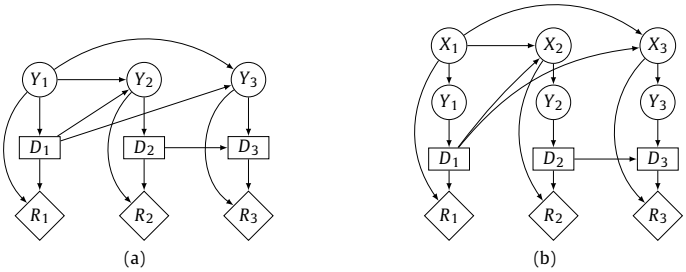


Fig. 8. Influence diagrams for (a) a three-stage completely observable decision process with independent subproblems and (b) a three-stage partially observable decision process with independent subproblems.

Algorithm 3: Variable elimination modified to resemble value iteration.

Input: Influence diagram with variables $\mathbf{V} = \mathbf{C} \cup \mathbf{D}$
Output: Optimal strategy, Δ, and MEU

1 $\Phi \leftarrow \{P(C|pa(C))|C \in \mathbf{C}\}$ // initial set of probability potentials
2 $\Psi \leftarrow \{R(pa(R)|R \in \mathbf{R}\}$ // initial set of utility potentials
3 $\Delta \leftarrow \emptyset$ // initial set of policies
4 $\psi_0 \leftarrow 0$ // initial utility potential
5 // Each iteration of for-loop eliminates a variable
6 **for** $i \leftarrow 1$ **to** $|\mathbf{V}|$ **do** // i is index of elimination step
7 *Select variable V to eliminate according to some criterion*
8 // Process probability potentials
9 $\Phi_V \leftarrow \{\phi \in \Phi | V \in dom(\phi)\}$ // get relevant probability potentials
10 $\phi_V \leftarrow \prod_{\phi \in \Phi_V} \phi$ // multiply probability potentials
11 **if** *V is a chance variable* **then**
12 $\phi_i \leftarrow \sum_V \phi_v$ // eliminate V by sum-marginalization
13 **else if** *V is a decision variable* **then**
14 $\phi_i \leftarrow \max_V \phi_v$ // eliminate V by max-marginalization
15 $\Phi \leftarrow (\Phi \backslash \Phi_V) \cup \{\phi_i\}$ // update set of probability potentials
16 // Process utility potentials
17 $\Psi_V \leftarrow \{\psi \in \Psi | V \in dom(\psi)\}$ // get newly-relevant utility potentials
18 $\psi_V \leftarrow \left(\sum_{\psi \in \Psi_V} \psi\right) + \psi_{i-1}$ // add utility potentials
19 **if** *V is a chance variable* **then** // eliminate V by sum-marginalization
20 $\phi_{cond} \leftarrow \phi_V / \phi_i$ // conditional probability of V
21 $\psi_i \leftarrow \sum_V \phi_{cond} \cdot \psi_V$ // multiply, then sum-marginalize
22 **else if** *V is a decision variable* **then**// eliminate V by max-marginalization
23 $\psi_i \leftarrow \max_V \psi_V$ // eliminate V by max-marginalization
24 $\delta_V \leftarrow \arg\max_V \psi_V$ // optimal policy for decision variable
25 $\Delta \leftarrow \Delta \cup \{\delta_V\}$ // add policy to strategy
26 $\Psi \leftarrow (\Psi \backslash \Psi_V)$ // update set of utility potentials
27 **end**
28 **return** $(\Delta, \psi_{|\mathbf{V}|})$ // Δ is optimal strategy, final utility potential is MEU

ination step i. (As a result, there is no need to add the utility potential ψ_i created in elimination step i to the set Ψ in line 26.) To distinguish this algorithm from the generalized variable elimination algorithm of Algorithm 2, we call it *generalized value iteration*. (Of course, generalized variable elimination is itself a generalization of the value iteration algorithm for POMDPs. However, Algorithm 3 more closely adopts the traditional value iteration approach.)

In the generalized value iteration algorithm of Algorithm 3, the utility potential ψ_i that is created in each elimination step more clearly plays the role of the cumulative value function that is updated in each iteration of value iteration. Always including the utility potential ψ_{i-1} generated in elimination step $i-1$ in the set of relevant utility potentials in step i may lead to inefficiency in cases where the utility potential ψ_{i-1} is irrelevant for elimination of the variable selected for elimination in that step. However, processing an irrelevant utility potential together with relevant utility potentials does not affect the correctness of the algorithm.

The generalized value iteration algorithm of Algorithm 3 is an interesting algorithm in its own right. It also has the advantage that it is easier to implement than Algorithm 2 because it does not need to check if an observed chance variable is d-connected to an unobserved chance variable in the domain of a piecewise-linear and concave utility potential when the observed chance variable is eliminated. Nevertheless, the generalized variable elimination algorithm of Algorithm 2 leverages problem structure that is not considered by the value iteration approach of Algorithm 3. In particular, the variable elimination approach leverages subproblem independence. For example, consider the steps involved in solving the influence diagram of Fig. 8a. The last stage of the problem is solved by eliminating variables D_3 and Y_3, which creates the utility potential $\psi_2(Y_1, D_1)$. The next variable eliminated is D_2. If D_2 is eliminated by Algorithm 3, the resulting utility potential

takes the form $\psi_3(Y_1, D_1, Y_2)$, which represents the cumulative utility. If D_2 is eliminated by Algorithm 2, however, the resulting utility potential takes the simpler form $\psi_3(Y_2)$, and the previously-generated utility potential, $\psi_2(Y_1, D_1)$, remains in the set Ψ, since it is not relevant in this elimination step. After Y_2 is eliminated by Algorithm 2, the set Ψ contains two utility potentials, one each for the second and third stages of the problem, which represent the solutions of independent subproblems. Fig. 8b shows an influence diagram with the same structure, but with partial observability. In this case also, generalized variable elimination can solve the two subproblems independently, while generalized value iteration cannot.

Like any value iteration algorithm, both Algorithms 2 and 3 leverage stagewise decomposition of a problem into subproblems. But the generalized variable elimination approach of Algorithm 2 has the advantage that it also leverages a form of hierarchical problem decomposition by representing the cumulative value function as a set of utility potentials instead of a single utility potential. That is, it represents the value function in factored form. By allowing independent subproblems to be solved independently, the variable elimination approach of Algorithm 2 is potentially more efficient than Algorithm 3.

6. Discussion

The algorithm introduced in this paper generalizes both the traditional variable elimination algorithm for influence diagrams and the value iteration algorithm for finite-horizon POMDPs, and includes both as special cases. Interestingly, it provides a perspective from which these two seemingly different algorithms can be viewed as the same algorithm, with just a different elimination order.

The integration of these algorithms leads to an improved understanding of the relationship, and synergy, between influence diagrams and POMDPs. In many cases, it also allows improved scalability in solving influence diagrams, as the result of relaxed constraints on elimination order and more effective use of dynamic programming.

6.1. Dynamic programming and influence diagrams

Our generalization of the variable elimination algorithm for influence diagrams can be helpfully viewed as an extension of the classic work of Tatman and Shachter [4,48] on how to adapt dynamic programming techniques for Markov decision processes in order to solve influence diagrams more efficiently.

Tatman wrote in his dissertation that the work he did under Shachter's supervision "began with the attempt to perform dynamic programming on a Markov decision process (MDP) in an influence diagram framework" [48, p. 26]. The original definition of an influence diagram allowed a single reward node [1,3]. However, Tatman and Shachter noticed that when a completely observable MDP with stage-dependent rewards is represented by an equivalent influence diagram with a single reward node, Shachter's node reduction algorithm [3] solves the problem much less efficiently than it is solved by the value iteration algorithm for completely observable MDPs, which leverages an additive decomposition of the utility function into multiple reward functions, one for each stage of the problem. This observation led Tatman and Shachter to generalize the definition of an influence diagram to allow multiple reward nodes, which is now the standard model. They also revised the node reduction algorithm so that it can solve influence diagrams that take this more general form. When an influence diagram that is defined in this more general way is used to represent a finite-horizon completely observable MDP, they showed that their revised node reduction algorithm performs the same computational steps as the value iteration algorithm for completely observable MDPs, and is just as efficient [4].

Tatman and Shachter also considered the relationship between influence diagrams and POMDPs. But they did not see how to integrate the belief-state dynamic programming recurrence solved by the value iteration algorithm for POMDPs in an algorithm for solving influence diagrams. In the alternative, history-based approach to solving influence diagrams, however, the entire history of the process is relevant for each decision variable of an influence diagram that represents a POMDP, and the POMDP cannot be decomposed into smaller subproblems, one for each stage, in a way that allows it to be solved more efficiently by dynamic programming. Recognizing this limitation, Tatman [48, p. 120] concluded that "it is impractical to solve almost any reasonable POMDP with this technique," where he used the word "technique" to refer to the node reduction algorithm for influence diagrams. As we have seen, the traditional variable elimination algorithm for influence diagrams is limited in the same way, as illustrated by its dismal performance in solving the simple maze POMDP of Section 2.3.5, compared to the more efficient performance of value iteration in solving the same problem.

In this paper, we have shown how to overcome this limitation. That is, we have shown how to solve an influence diagram for a partially observable decision problem more efficiently by solving a dynamic programming recurrence that is defined over belief states as well as histories. Thus we have the following parallel. Tatman and Shachter generalized the node reduction algorithm for influence diagrams so that it includes the value iteration algorithm for completely observable MDPs as a special case. We have generalized the variable elimination algorithm for influence diagrams so that it includes the value iteration algorithm for POMDPs as a special case. (In an earlier paper, we explored a similar generalization of the node reduction algorithm [49].)

It is important to note that our new algorithm does not simply perform *either* value iteration *or* traditional variable elimination. It solves problems in new ways that are not possible for either traditional algorithm. To see that, it helps to recognize that the order in which variables are eliminated by generalized variable elimination defines the dynamic programming recurrence that it solves. When the generalized algorithm eliminates variables in an order that is allowed by traditional variable elimination, it solves a dynamic programing recurrence that is defined over histories only, and it solves

the recurrence in the same way that it is solved by the traditional variable elimination algorithm. When the generalized algorithm eliminates variables in the order in which they are eliminated by the value iteration algorithm for POMDPs (for an influence diagram that represents a POMDP), it solves a dynamic programming recurrence that is defined over belief states, and it solves this recurrence in the way it is solved by the value iteration algorithm for POMDPs. However, generalized variable elimination can also eliminate variables in an order that is not allowed by the traditional variable elimination algorithm, and does not correspond to the order in which variables are eliminated by the value iteration algorithm for POMDPs. In that case, it solves a dynamic programming recurrence that is defined over *both* histories and belief states. Because it can solve dynamic programming recurrences that take this more complex form, which neither traditional algorithm can solve, generalized variable elimination has the potential to outperform *both* of the two algorithms it combines.

6.2. Promising applications

We believe that generalized variable elimination is most likely to outperform both traditional variable elimination and value iteration in solving partially observable decision processes with k-order Markovian structure, as discussed in Section 5.2.1. This class of problems, which traditional value iteration cannot solve (without reformulation and state augmentation), includes problems with delayed observations, delayed action effects, and non-Markovian rewards. Tatman and Shachter also emphasized the potential advantage of using influence diagrams to solve k-order Markovian problems. In their words: "dynamic programs with nonstandard problem structures can be naturally solved within the influence diagram framework, without reformulation and state augmentation" [4, p. 366]. However, they could only show that representing a k-order Markovian problem as an influence diagram offers a computational advantage in the completely observable case. Our development of the generalized variable elimination algorithm shows that representing a k-order Markovian problem as an influence diagram offers a computational advantage in the partially observable case as well. For this class of problems, generalized variable elimination can find an optimal solution by solving a dynamic programming recurrence that is defined over both belief states and history, where the belief state allows the length of the relevant history to be bounded by k [47]. The potential effectiveness of generalized variable elimination in solving partially observable k-order decision processes is especially significant given that there is currently a lack of good algorithms for solving this important class of problems.

Although we believe generalized variable elimination is most likely to be useful in solving k-order Markovian partially observable problems, the new algorithm may also help in solving other problems, including standard Markovian problems. For example, it may allow some POMDPs to be solved more efficiently by eliminating variables in a different order than they are eliminated by the traditional value iteration algorithm. (The work of Zhang and Zhang [50] may be interpreted in this way.) The more fine-grained control of elimination order that is possible in the variable elimination approach may also allow performance to be improved in solving factored POMDPs. In addition, being able to solve a dynamic programming recurrence that is defined over both histories and belief states makes it possible for generalized variable elimination to switch, in the course of problem solving, from solving a recurrence over belief states to solving a recurrence over histories. Solving a belief-state dynamic programming recurrence is much easier when eliminating variables near the end of a problem, while solving a history-based dynamic programming recurrence is easier when eliminating variables near the beginning of a problem, where histories are short. A good elimination-ordering heuristic can take advantage of this flexibility, as discussed further in Section 6.3.1.

Interestingly, the marginal maximum *a posteriori* probability (MAP) problem for Bayesian networks is closely related to the problem of solving an influence diagram [51]. It is also equivalent to an unobservable POMDP, which is a POMDP with a single observation that provides no information. The variable elimination algorithm for the marginal MAP problem is limited in the same way as the traditional variable elimination algorithm for influence diagrams: it must eliminate all unobserved variables before maximizing any controllable variables [52, pp. 554–561]. It follows that the approach developed in this paper may be used to similarly generalize and improve the variable elimination algorithm for the marginal MAP problem.

We leave it for future work to explore these possibilities further, and to potentially identify other classes of problems where an integrated approach to problem solving may offer an advantage.

6.3. Extensions

We conclude by discussing two important extensions of the generalized variable elimination algorithm that we also leave for future work. The first is development of an elimination-ordering heuristic that ensures good performance. The second is representation of a strategy as a graph. Of course, other extensions are possible.

6.3.1. Elimination-ordering heuristic

It is well-known that the performance of a variable elimination algorithm depends to a great extent on the order in which variables are eliminated. In the variable elimination approach to solving Bayesian networks, the problem of finding an elimination order that optimizes performance is NP-hard, and heuristics are used to quickly find good elimination orders that are not necessarily optimal [53]. Elimination-ordering heuristics that are used in solving Bayesian networks have been adapted for use by the traditional variable elimination algorithm for influence diagrams, with adjustment for the additional constraints on elimination order [54]. However, these heuristics cannot be used by generalized variable elimination, in part because the new algorithm has different constraints on elimination order, but also because the effect of elimination order on the size of piecewise-linear and concave utility potentials is difficult to predict.

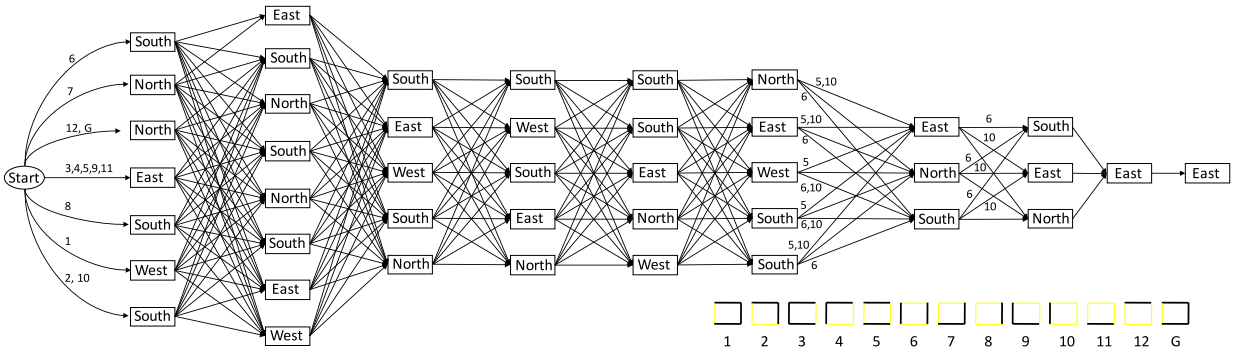


Fig. 9. Optimal policy graph for the maze of Fig. 3a. For space reasons, only some of the edge labels are shown. The numbers refer to the observations shown in the lower right. The start node represents the starting belief state, which is a uniform random probability distribution over the non-goal states.

Dynamic heuristic. Elimination-ordering heuristics for variable elimination algorithms typically try to limit the size of the potentials that are generated, which in turn limits both the time and memory complexity of the algorithm. For ordinary potentials, the size of a potential under a given elimination order is relatively easy to predict before the start of the algorithm because it depends only on the variables in the domain of the potential. Thus elimination-ordering heuristics for traditional variable elimination can find a good elimination order before the start of the algorithm.

For generalized variable elimination, the size of a piecewise-linear and concave utility potential depends not only on the variables in its domain; it also depends on the number of linear potentials used to represent it. However, the uncertain effect of pruning makes it difficult to predict the number of linear potentials in the representation of a piecewise-linear and concave utility potential. Therefore, an effective elimination-ordering heuristic for generalized variable elimination likely needs to be a *dynamic* heuristic that monitors the size of piecewise-linear and concave utility potentials as they are generated during problem solving, and chooses the next variable to eliminate based on the size of the potentials created so far.

Simple heuristic. We sketch a simple dynamic heuristic. As discussed in Section 3.3.2, postponing the elimination of unobserved chance variables in order to eliminate a decision or observed chance variable works best for variables near the end of a problem, especially when the last decision variable (and its corresponding policy) would otherwise be conditioned on a long history. This observation suggests a heuristic that delays the elimination of a given unobserved variable until either (i) it must be eliminated because of causal precedence constraints, or (ii) it should be eliminated because it is in the domain of a piecewise-linear and concave utility potential that has become too large. Recall that the size of a piecewise-linear and concave utility potential can always be reduced by eliminating one or more unobserved chance variables in its domain. We leave it for future work to develop the details of such an elimination-ordering heuristic.

6.3.2. Strategy representation

As described at the end of Section 3.4.2, generalized variable elimination can represent a strategy as a sequence of policies, one for each decision variable $D \in \mathbf{D}$. Each policy is a mapping,

$$\delta_D : sp(\mathbf{H}) \times bsp(\mathbf{U}) \rightarrow sp(D), \tag{133}$$

where $\mathbf{H}$ and $\mathbf{U}$ denote the relevant observed and unobserved variables, respectively, for the decision D.

When the set of relevant unobserved variables, $\mathbf{U}$, is not empty, a policy is not explicitly represented in this approach. Instead, it is represented implicitly by an indexed family of sets of linear potentials, $\{\Gamma_{\mathbf{h}}\}_{\mathbf{h} \in sp(\mathbf{H})}$, that represents the piecewise-linear and concave utility potential that is computed when the decision variable D is eliminated. The corresponding policy δ_D is represented as follows,

$$\delta_D(\mathbf{h}, b(\mathbf{U})) = d\left(\arg\max_{\gamma \in \Gamma_{\mathbf{h}}} \sum_{\mathbf{u}} b(\mathbf{u}) \cdot \gamma(\mathbf{u})\right), \tag{134}$$

where $d(\gamma) \in sp(D)$ denotes the action associated with generation of the linear potential γ. A drawback of this approach to policy representation is that it requires updating a belief state during strategy execution, since a decision is made based not only on the relevant observed history, $\mathbf{h}$, but on a belief, $b(\mathbf{U})$, about the relevant unobserved state.

There is an alternative approach to policy representation for POMDPs that is also widely-used, but does not require a belief state to be updated during strategy execution. In this approach, a strategy is represented explicitly by an acyclic graph, called a *policy graph* [31]. For the ten-stage maze POMDP described in Section 2.3.5, Fig. 9 shows an optimal policy graph constructed by value iteration when a strategy is represented in this way. Each node of the graph (except for the start node) is associated with an action, and each outgoing edge corresponds to an observation. Besides allowing a strategy to be executed without belief updates, this representation of a strategy is *much* more compact than the representation of a strategy that is traditionally used for influence diagrams, and it can also be easier to interpret.

Representing a strategy for an influence diagram in a similar way will require a generalization of the policy graph representation for POMDPs so that it applies to influence diagrams. For example, a strategy for an influence diagram will need to be represented by a graph with two kinds of nodes, one for decisions and one for observations, and some additional processing may be needed to compress the graph so that it is as compact as possible, especially for non-Markovian problems. We leave the details of this generalization for future work. Here it is enough to point out that the possibility of representing a strategy for an influence diagram in the compact form of a graph, similar to a policy graph, is one of a number of promising extensions of an integrated approach to problem solving.

Acknowledgements

Support for this research was provided by The National Science Foundation, in the form of Award IIS:RI #1718384. The author is especially grateful to Jinchuan Shi, his former PhD student, for many helpful discussions and insights about how to combine influence diagrams and POMDPs. The author is also grateful to the anonymous reviewers for several helpful suggestions that have improved this paper.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Additional results related to mixed-observable MDPs

In this first appendix, we fill in some of the theoretical details left out of Section 5.1. We summarize the value iteration algorithm for mixed-observable MDPs, and then prove Theorem 8 and Corollary 2 from Section 5.1.3.

A.1. Value iteration for mixed-observable MDPs

The value iteration algorithm for mixed-observable MDPs solves the dynamic programming recurrence of Equations (131) and (132). It is an extension of the value iteration algorithm for POMDPs that achieves improved efficiency by leveraging the special structure of mixed-observable problems. Since the relevant literature on this algorithm is very limited [40], we briefly summarize the algorithm here.

Piecewise-linear and concave value function. For each stage $t = 1, 2, \ldots, n$ of a mixed-observable MDP, the value function, $V_t : sp(Y_t) \times bsp(X_t) \rightarrow \Re$, is piecewise-linear and concave, and is represented by an indexed family of sets of linear functions, $\{\Gamma_t^{y_t}\}_{y_t \in sp(Y_t)}$, where

$$V_t(y_t, b_t) = \max_{\gamma \in \Gamma_t^{y_t}} \sum_{x_t \in sp(X_t)} b_t(x_t)\gamma(x_t). \tag{A.1}$$

This value function is partitioned into $|sp(Y_t)|$ sets of linear functions with domain X_t, and it is indexed by both the state y_t of an observed variable Y_t, and a belief state b_t over the possible states of an unobserved variable X_t.

Incremental pruning. The incremental pruning algorithm for POMDPs, reviewed in Section 2.3.5, is modified to solve mixed-observable MDPs, as follows.

For the last stage of the process, the value function V_n is represented by $|Y_n|$ sets of linear functions with domain X_n, where each set $\Gamma_n^{y_n}$ contains a linear function for each action. Let $i = 1 \ldots |sp(D_n)|$ denote the index of both the action $d_n^i \in sp(D_n)$ and the corresponding linear function $\gamma_n^i \in \Gamma_n^{y_n}$. The value of the linear function for a given state $x_n \in X_n$ is

$$\gamma_n^i(x_n) = R_n(y_n, x_n, d_n^i). \tag{A.2}$$

Next we consider the recursive step of the algorithm. Given a stage-$(t+1)$ piecewise-linear and concave value function that is represented by an indexed family of sets of linear functions over X_{t+1}, $\{\Gamma_{t+1}^{y_{t+1}}\}_{y_{t+1} \in sp(Y_{t+1})}$, a stage-$t$ piecewise-linear and concave value function is constructed that is represented by an indexed family of sets of linear functions over X_t, $\{\Gamma_t^{y_t}\}_{y_t \in sp(Y_t)}$, by performing the following three steps.

First, for each quadruple of observed state y_t, decision d_t, subsequent observation y_{t+1}^X, and subsequent observed state y_{t+1}, the *backprojection* step generates a set of linear functions:

$$\Gamma_t^{y_t, d_t, y_{t+1}^X, y_{t+1}} = Prune\left(\{\gamma_t^i | i = 1, \ldots, |\Gamma_{t+1}^{y_{t+1}}|\}\right). \tag{A.3}$$

Before pruning, there is one linear function γ_t^i in $\Gamma_t^{y_t, d_t, y_{t+1}^X, y_{t+1}}$ for each linear function γ_{t+1}^i in $\Gamma_{t+1}^{y_{t+1}}$, where the value of γ_t^i for a given state $x_t \in X_t$ is defined as

$$\gamma_t^i(x_t, y_t) = \frac{R(y_t, x_t, d_t)}{|sp(Y_{t+1}^X)|} + \sum_{x_{t+1}} P(x_{t+1}, y_{t+1}^X | x_t, y_t, d_t)\gamma_{t+1}^i(x_{t+1}, y_{t+1}). \tag{A.4}$$

Second, for each pair of observed state y_t and decision d_t, the *cross sum* step generates a set of linear functions,

$$\Gamma_t^{y_t,d_t} = Prune\left(\oplus_{(y_{t+1}^X, y_{t+1})} \Gamma_t^{y_t,d_t,y_{t+1}^X,y_{t+1}}\right), \tag{A.5}$$

where the cross-sum step should be performed incrementally, as in Equation (58).

Finally, for each observed state y_t, the *maximization* step generates a set of linear functions, as follows:

$$\Gamma_t^{y_t} = Prune\left(\cup_{d_t} \Gamma_t^{y_t,d_t}\right). \tag{A.6}$$

The resulting indexed family of sets of linear functions, $\{\Gamma_t^{y_t}\}_{y_t \in sp(Y_t)}$, represents the piecewise-linear and concave value function V_t of Equation (A.1).

A.2. Reduction of an influence diagram to a mixed-observable MDP

To prove Theorem 8 and Corollary 2, we first prove the following lemma, which provides the foundation for both results. To simplify the proof, the lemma considers the modified version of generalized variable elimination summarized by the pseudocode of Algorithm 3, which more closely resembles the value iteration algorithm.

Lemma 2. *Algorithm 3 reduces any influence diagram to an equivalent mixed-observable MDP that it solves by value iteration.*

Proof. The proof strategy is to show that each elimination step corresponds to a stage of an equivalent mixed-observable MDP that is solved by value iteration.

To distinguish between observed and unobserved variables in the domain of a piecewise-linear and concave potential, we let $domH(\psi)$ denote the set of observed variables in the domain of a potential ψ, and we let $domU(\psi)$ denote the set of unobserved variables, so that $dom(\psi) = domH(\psi) \cup domU(\psi)$.

State. For each elimination step i of Algorithm 3, let V denote the variable selected for elimination. Once the relevant probability and utility potentials are identified, let $\mathbf{H}_i$ denote the set of relevant observed variables, let $\mathbf{U}_i$ denote the set of relevant unobserved variables, and note that V is not included in $\mathbf{H}_i$ or $\mathbf{U}_i$. The state of the mixed-observable MDP at stage i is a joint state, $(\mathbf{h}_i, \mathbf{u}_i) \in sp(\mathbf{H}_i, \mathbf{U}_i)$.

Action. If the variable V selected for elimination is a decision variable, it is the decision variable for the corresponding stage of the mixed-observable MDP. If V is a chance variable, the decision variable for this stage of the mixed-observable MDP has a single value, λ, which means there is a single available action, and thus no choice.

State transition probabilities. If the eliminated variable V is a decision variable, then for each instantiation of the relevant variables, $\mathbf{H}_i$, $\mathbf{U}_i$, and V, the state transition is deterministic, with probability,

$$P\left(\left((\mathbf{h}_i, v)^{\downarrow domH(\psi_{i-1})}, \mathbf{u}_i^{\downarrow domU(\psi_{i-1})}\right) \middle| (\mathbf{h}_i, \mathbf{u}_i), v\right) = 1, \tag{A.7}$$

where $(\mathbf{h}_i, \mathbf{u}_i)$ is the state at stage i, the action at stage i is v, and the successor state at stage $(i-1)$ is $((\mathbf{h}_i, v)^{\downarrow domH(\psi_{i-1})}, \mathbf{u}_i^{\downarrow domU(\psi_{i-1})})$. Use of the projection operation ensures the successor state is in the state set of the successor stage of the MDP.

If the eliminated variable V is an observed chance variable, the single available action λ has a stochastic outcome, which is the observation $v \in sp(V)$. For each instantiation of the relevant variables, the state transition probability is

$$P\left(\left((\mathbf{h}_i, v)^{\downarrow domH(\psi_{i-1})}, \mathbf{u}_i^{\downarrow domU(\psi_{i-1})}\right) \middle| (\mathbf{h}_i, \mathbf{u}_i), \lambda\right) = \phi_{cond}(v|\mathbf{h}_i, \mathbf{u}_i), \tag{A.8}$$

where ϕ_{cond} is defined by line 20 of Algorithm 3, so that

$$\phi_{cond}(v|\mathbf{h}_i, \mathbf{u}_i) = \phi_V(v, \mathbf{h}_i, \mathbf{u}_i)/\phi_i(\mathbf{h}_i, \mathbf{u}_i). \tag{A.9}$$

If the eliminated variable V is an unobserved chance variable, the single available action λ has a stochastic effect on the unobserved state. For each instantiation of the relevant variables, the state transition probability is:

$$P\left(\left(\mathbf{h}_i^{\downarrow domH(\psi_{i-1})}, (\mathbf{u}_i, v)^{\downarrow domU(\psi_{i-1})}\right) \middle| (\mathbf{h}_i, \mathbf{u}_i), \lambda\right) = \phi_{cond}(v|\mathbf{h}_i, \mathbf{u}_i). \tag{A.10}$$

Observation probabilities. If the variable V selected for elimination is a decision variable, then for each instantiation of the relevant variables, we have the observation probability

$$P\left(\left((\mathbf{h}_i, v)^{\downarrow domH(\psi_{i-1})}, \lambda\right) \middle| \left((\mathbf{h}_i, v)^{\downarrow domH(\psi_{i-1})}, \mathbf{u}_i^{\downarrow domU(\psi_{i-1})}\right), v\right) = 1, \tag{A.11}$$

which means that when the action $v \in sp(V)$ is followed by a transition to the state $((\mathbf{h}_i, v)^{\downarrow domH(\psi_{i-1})}, \mathbf{u}_i^{\downarrow domU(\psi_{i-1})})$, the observation is a pair of the observed state $(\mathbf{h}_i, v)^{\downarrow domH(\psi_{i-1})}$ and λ, where λ provides no information about the unobserved state.

If the eliminated variable V is an observed chance variable, then for each instantiation of the relevant variables, we have the observation probability

$$P\left(\left((\mathbf{h}_i, v)^{\downarrow domH(\psi_{i-1})}, v\right) \middle| ((\mathbf{h}_i, v)^{\downarrow domH(\psi_{i-1})}, \mathbf{u}_i^{\downarrow domU(\psi_{i-1})}), \lambda\right) = \phi_{cond}(v|\mathbf{h}_i, \mathbf{u}_i), \tag{A.12}$$

which means that when a transition to state $((\mathbf{h}_i^{\downarrow domH(\psi_{i-1})}, v), \mathbf{u}_i^{\downarrow domU(\psi_{i-1})})$ follows the action λ, the resulting observation is a pair of observed state $(\mathbf{h}_i, v)^{\downarrow domH(\psi_{i-1})}$ and $v \in sp(V)$, where v provides (possibly imperfect) information about the unobserved state $\mathbf{u}_i$.

If the eliminated variable V is an unobserved chance variable, then for each instantiation of the relevant variables, we have the observation probability

$$P\left(\left(\mathbf{h}_i^{\downarrow domH(\psi_{i-1})}, \lambda\right) \middle| \left(\mathbf{h}_i^{\downarrow domH(\psi_{i-1})}, (\mathbf{u}_i, v)^{\downarrow domU(\psi_{i-1})}\right), \lambda\right) = 1, \tag{A.13}$$

which means that when a transition to state $(\mathbf{h}_i^{\downarrow domH(\psi_{i-1})}, (\mathbf{u}_i, v)^{\downarrow domU(\psi_{i-1})})$ follows the action λ, the resulting observation is a pair of observed state $\mathbf{h}_i^{\downarrow domH(\psi_{i-1})}$ and λ, where λ provides no information about the unobserved state $\mathbf{u}_i$.

Reward. If the variable V selected for elimination is a decision variable, the reward for this stage of the mixed-observable MDP is

$$R\left((\mathbf{h}_i, \mathbf{u}_i), v\right) = \psi_V^R(\mathbf{h}_i, \mathbf{u}_i, v), \tag{A.14}$$

where $\psi_V^R = \sum_{\psi \in \Psi \text{such that } V \in dom(\psi)} \psi$, that is, ψ_V^R is the sum of all newly-relevant reward potentials for this elimination step. Note that ψ_V^R does not include ψ_{i-1}, which represents the "next-stage value function."

If the eliminated variable V is a chance variable, observed or unobserved, then ψ_V^R is defined the same way, and the reward for this stage of the mixed-observable MDP is

$$R((\mathbf{h}_i, \mathbf{u}_i), \lambda) = \sum_{v \in sp(V)} \phi_{cond}(v|\mathbf{h}_i, \mathbf{u}_i)\psi_V^R(\mathbf{h}_i, \mathbf{u}_i, v), \tag{A.15}$$

which is the expected reward averaged over all possible outcomes.

Markov property. Since all variables in the domains of the conditional probability and reward functions are part of the current state set, current action set, or successor state set, the mixed-observable model satisfies the Markov property, and is an MDP.

Value iteration. Each stage of the mixed-observable MDP is solved by value iteration, as follows. If the eliminated variable V is a decision variable, then $\psi_i = \max_V \psi_V$ is the value function for this stage of the MDP, where $\psi_V = \psi_V^R + \psi_{i-1}$, and δ_V is the policy that maximizes ψ_i. If V is a chance variable, then $\psi_i = \sum_V \psi_V$ is the value function for this stage of the MDP, and the policy is irrelevant, since there is a single available action, and thus no choice. □

The only difference between Algorithms 2 and 3 is that Algorithm 3 represents the cumulative value function computed by value iteration as a single utility potential ψ_i, whereas Algorithm 2 represents the same cumulative value function as a set of utility potentials, with the advantage that only the potentials in this set that are relevant in a given elimination step need to be updated. Therefore, the analysis of Algorithm 3 also applies to the generalized variable elimination algorithm of Algorithm 2, and we have the following proof of Theorem 8.

Theorem 8. *The generalized variable elimination algorithm reduces any influence diagram to an equivalent mixed-observable MDP that it solves by value iteration.*

Proof. At every step in the execution of Algorithm 2, let Ψ^R denote the subset of utility potentials in Ψ that are part of the original influence diagrams, and let Ψ^V denote the subset of utility potentials in Ψ generated in some previous elimination step, where $\Psi = \Psi^R \cup \Psi^V$. When a variable V is selected for elimination, let ψ_V^R denote the sum of the relevant utility potentials in Ψ^R and note that it is equal to the utility potential ψ_V^R computed in Algorithm 3. Note also that the quantity $\sum_{\psi \in \Psi^V} \psi$ in Algorithm 2 is equal to the utility potential ψ_{i-1} in Algorithm 3. In every other respect, the two algorithms are identical. Since they compute the same quantities, it follows from the reduction in the proof of Lemma 2 that Algorithm 2 also reduces an influence diagram to an equivalent mixed-observable MDP that it solves by value iteration. □

Finally, we prove the following special case of this result.

Corollary 2. *The traditional variable elimination algorithm reduces any influence diagram to an equivalent completely observable MDP that it solves by value iteration.*

Proof. Traditional variable elimination eliminates variables in an order that ensures that all of the variables in the domain of a utility potential that is created when a decision variable is eliminated are observable. If follows that the mixed-observable MDP to which the traditional variable elimination algorithm reduces an influence diagram is a completely observable MDP. □

Appendix B. Interleaving operations on piecewise-linear and concave potentials

In this appendix, we further consider the relationship between generalized variable elimination and the incremental pruning algorithm for POMDPs, which it generalizes. In particular, we show how to improve the efficiency of generalized variable elimination by making it even more similar to incremental pruning.

Recall that the operations on piecewise-linear and concave utility potentials defined in Section 3 are performed by iterating over one or more sets of linear potentials. When a sequence of these operations is performed, one at a time, on an initial piecewise-linear and concave utility potential, a sequence of intermediate piecewise-linear and concave utility potentials is created before the final result. In this appendix, we show that it is not necessary to create these intermediate piecewise-linear and concave utility potentials. Instead, the sequence of operations can be performed all at once inside a single iteration over the sets of linear potentials that represent the initial piecewise-linear and concave utility potential. We call this optimization *interleaving* operations on piecewise-linear and concave utility potentials.

B.1. Generalized variable elimination algorithm with interleaved operations

Algorithm 4 gives pseudocode for a version of generalized variable elimination that includes this optimization. To show the optimization more clearly, the pseudocode includes low-level detail that is not included in the pseudocode of Algorithms 2. Algorithm 4 invokes three subroutines that correspond to the tasks of eliminating an unobserved chance variable, eliminating an observed chance variable, and eliminating a decision variable. These subroutines are named after the three steps of the incremental pruning algorithm, which they generalize: *BackProject*, *CrossSum*, and *Maximize*.

To support this optimization, the pseudocode of Algorithm 4 distinguishes between the sum of relevant ordinary utility potentials when V is eliminated, denoted ψ_V, and the sum of relevant piecewise-linear and concave utility potentials, denoted $\overline{\psi}_V$. The subroutines take both ψ_V and $\overline{\psi}_V$ as arguments so that they can interleave computation of the sum, $\psi_V + \overline{\psi}_V$, with subsequent operations performed on these utility potentials.

B.2. Generalized backprojection

Algorithm 5 gives pseudocode for the *Backproject* subroutine, which processes utility potentials when an unobserved chance variable C is eliminated. It generates the utility potential $\psi_i = \sum_C \phi_{cond} \cdot (\psi_C + \overline{\psi}_C)$, which requires three operations. First, the two utility potentials, ψ_V and $\overline{\psi}_V$, are added. Then their sum is multiplied by the probability potential ϕ_{cond}. Finally, the unobserved chance variable C is eliminated from the product by sum-marginalization.

If there is no piecewise-linear and concave utility potential $\overline{\psi}_C$, that is, if its domain is empty, then the three operations can be performed on ordinary potentials in the traditional way. Otherwise, the operations on piecewise-linear and concave potentials are *interleaved*. Note that in the pseudocode, $domH(\psi)$ denotes the set of observed variables in the domain of a potential ψ. For each instantiation $\mathbf{h}_i$ of $\mathbf{H}_i$, where $\mathbf{H}_i$ is the set of observed variables in the domain of the new piecewise-linear and concave utility potential ψ_i, a set of linear potentials, $\Gamma_{i,\mathbf{h}_i}$, is created that contains (before pruning) one linear potential γ' for each linear potential γ in the set $\Gamma_{C,\mathbf{h}_i^{\downarrow \mathbf{H}_C}}$, where $\mathbf{H}_C$ is the set of observed variables in the domain of the piecewise-linear and concave utility potential $\overline{\psi}_C$. The new linear potential is defined as

$$\gamma' = \sum_C \left(\phi_{cond}^{R(\mathbf{H}_i = \mathbf{h}_i)} \cdot \left(\psi_C^{R(\mathbf{H}_i = \mathbf{h}_i)} + \gamma \right) \right). \tag{B.1}$$

Algorithm 4: Generalized variable elimination with interleaved operations on piecewise-linear and concave (PWLC) utility potentials.

Input: Influence diagram with variables $\mathbf{V} = \mathbf{C} \cup \mathbf{D}$
Output: Optimal strategy, Δ, and MEU

$\Phi \leftarrow \{P(C|pa(C))|C \in \mathbf{C}\}$ // set of probability potentials
$\Psi \leftarrow \{R(pa(R)|R \in \mathbf{R}\}$ // set of ordinary utility potentials
$\overline{\Psi} \leftarrow \emptyset$ // set of PWLC utility potentials
$\Delta \leftarrow \emptyset$ // strategy
for $i \leftarrow 1$ **to** $|\mathbf{V}|$ **do** // i is index of elimination step
  Select variable V to eliminate according to some criterion
  // Process probability potentials
  $\Phi_V \leftarrow \{\phi \in \Phi | V \in dom(\phi)\}$
  $\phi_V \leftarrow \prod_{\phi \in \Phi_V} \phi$ // product of relevant probability potentials
  if *V is a chance variable* **then**
    $\phi_i \leftarrow \sum_V \phi_V$ // eliminate V by sum-marginalization
    $\phi_{cond} \leftarrow \phi_V / \phi_i$ // used to process utility potentials
  else if *V is a decision variable* **then**
    $\phi_i \leftarrow \max_V \phi_V$ // eliminate V by max-marginalization
  $\Phi \leftarrow (\Phi \backslash \Phi_V) \cup \{\phi_i\}$ // update set of probability potentials
  // Process utility potentials
  $\Psi_V \leftarrow \{\psi \in \Psi | V \in dom(\psi)\}$ // relevant ordinary utility potentials
  $\Psi \leftarrow \Psi \backslash \Psi_V$
  $\psi_V \leftarrow 0 + \sum_{\psi \in \Psi_V} \psi$ // sum of relevant ordinary utility potentials
  $\overline{\Psi}_V \leftarrow \{\psi \in \overline{\Psi} | (V \in dom(\psi))$ **or**
      ((V is an observed chance variable) **and**
       (V is d-connected to an unobserved variable in $dom(\psi)$))$\}$
      // relevant PWLC utility potentials
  $\overline{\Psi} \leftarrow \overline{\Psi} \backslash \overline{\Psi}_V$
  $\overline{\psi}_V \leftarrow 0 + \sum_{\psi \in \overline{\Psi}_V} \psi$ // sum of relevant PWLC utility potentials
  if *V is an unobserved chance variable* **then**
    $\psi_i \leftarrow$ **BackProject** $\left(V, \phi_{cond}, \psi_V, \overline{\psi}_V\right)$
  else if *V is an observed chance variable* **then**
    $\psi_i \leftarrow$ **CrossSum** $\left(V, \phi_{cond}, \psi_V, \overline{\psi}_V\right)$
  else if *V is a decision variable* **then**
    $(\psi_i, \delta_V) \leftarrow$ **Maximize** $\left(V, \psi_V, \overline{\psi}_V\right)$
    $\Delta \leftarrow \Delta \cup \{\delta_V\}$ // generalized representation of policy
  if *ψ_i is an ordinary utility potential* **then** $\Psi \leftarrow \Psi \cup \{\psi_i\}$
  else $\overline{\Psi} \leftarrow \overline{\Psi} \cup \{\psi_i\}$
end
$MEU \leftarrow \sum_{\psi \in \Psi} \psi$ // MEU is sum of final utility potentials
return (Δ, MEU)

Algorithm 5: Generalized backprojection subroutine.

Function *BackProject*$(C, \phi_{cond}, \psi_C, \overline{\psi}_C)$
  // Efficiently compute $\psi_i \leftarrow \sum_C \phi_{cond} \cdot (\psi_C + \overline{\psi}_C)$, where:
  // C is an unobserved chance variable to be eliminated
  // ϕ_{cond} is a probability potential
  // ψ_C is an ordinary utility potential
  // $\overline{\psi}_C$ is a PWLC utility potential represented by $\{\Gamma_{\mathbf{h}_C}\}_{\mathbf{h}_C \in sp(\mathbf{H}_C)}$,
  // where $\mathbf{H}_C$ is the set of observed variables in the domain of $\overline{\psi}_C$
  if $dom(\overline{\psi}_C) = \emptyset$ **then** // new utility potential ψ_i is ordinary potential
    $\psi_i \leftarrow \sum_C \phi_{cond} \cdot \psi_C$ // process just like traditional algorithm
  else // new utility potential ψ_i is PWLC
    $\mathbf{H}_i \leftarrow \left(domH(\psi_C) \cup domH(\overline{\psi}_C)\right) \backslash \{C\}$
    $\mathbf{H}_C \leftarrow domH(\overline{\psi}_C)$
    foreach $\mathbf{h}_i \in sp(\mathbf{H}_i)$ **do**
      $\Gamma_{i,\mathbf{h}_i} \leftarrow \emptyset$ // initialize set of linear potentials for history $\mathbf{h}_i$
      foreach $\gamma \in \Gamma_{C,\mathbf{h}_i^{\downarrow \mathbf{H}_C}}$ **do** // for each old linear potential ...
        $\gamma' \leftarrow \sum_C \left(\phi_{cond}^{R(\mathbf{H}_i = \mathbf{h}_i)} \cdot \left(\psi_C^{R(\mathbf{H}_i = \mathbf{h}_i)} + \gamma\right)\right)$
        $\Gamma_{i,\mathbf{h}_i} \leftarrow \Gamma_{i,\mathbf{h}_i} \cup \{\gamma'\}$ // add new linear potential to set
      end
      $\Gamma_{i,\mathbf{h}_i} \leftarrow Prune(\Gamma_{i,\mathbf{h}_i})$ // prune dominated linear potentials
    end
  end
  return (ψ_i) // return new utility potential

Algorithm 6: Generalized cross-sum subroutine.

```
1  Function CrossSum(C, φ_cond, ψ_C, ψ̄_C)
2      // Efficiently compute ψ_i ← Σ_C φ_cond · (ψ_C + ψ̄_C), where:
3      //   C is an observed chance variable to be eliminated
4      //   φ_cond is a probability potential
5      //   ψ_C is an ordinary utility potential
6      //   ψ̄_C is a PWLC utility potential represented by {Γ_{h_C}}_{h_C ∈ sp(H_C)},
7      //       where H_C is the set of observed variables in the domain of ψ̄_C
8      if dom(ψ̄_C) = ∅ then // new utility potential ψ_i is ordinary potential
9          ψ_i ← Σ_C φ_cond · ψ_C // process just like traditional algorithm
10     else // new utility potential ψ_i is PWLC
11         H_i ← (domH(ψ_C) ∪ domH(ψ̄_C)) \ {C}
12         H_C ← domH(ψ̄_C)
13         foreach h_i ∈ sp(H_i) do
14             Γ_{i,h_i} ← { φ_cond^{R(H_i=h_i)} · ψ_C^{R(H_i=h_i)} } // initialize linear potentials
15             foreach c ∈ sp(C) do // observation c
16                 // compute cross-sum, Γ_xsum, of sets Γ_{i,h_i} and Γ_{C,(h_i,c)↓H_C}
17                 Γ_xsum ← ∅
18                 foreach γ ∈ Γ_{i,h_i} do
19                     foreach γ' ∈ Γ_{C,(h_i↓H_C, c)} do
20                         Γ_xsum ← Γ_xsum ∪ {γ + φ_cond^{R(H_i=h_i)} · γ'}
21                     end
22                 end
23                 Γ_{i,h_i} ← Prune(Γ_xsum) // eliminate dominated linear potentials
24             end
25         end
26     end
27     return (ψ_i) // return new utility potential
```

All three operations are performed to create each linear potential γ' in $\Gamma_{i,\mathbf{h}_i}$. As a result, the new piecewise-linear and concave utility potential ψ_i is created all at once, without creating any intermediate piecewise-linear and concave utility potentials.

B.3. Generalized cross-sum

Algorithm 6 shows pseudocode for the *CrossSum* subroutine, which is invoked to process utility potentials when an observed chance variable is eliminated. Like the *BackProject* subroutine, it generates the utility potential $\psi_i = \sum_C \phi_{cond} \cdot (\psi_C + \overline{\psi}_C)$, which requires the operations of addition, multiplication, and sum-marginalization. If there is no piecewise-linear and concave utility potential $\overline{\psi}_C$, that is, if $dom(\overline{\psi}_C) = \emptyset$, these operations are performed on ordinary potentials. Otherwise, they are performed on a piecewise-linear and concave utility potential. The *CrossSum* subroutine differs from the *BackProject* subroutine because the sum-marginalization operation is performed differently when an *observed* chance variable is eliminated from a piecewise-linear and concave utility potential than when an unobserved variable is eliminated.

In the *CrossSum* subroutine, the operations on piecewise-linear and concave potentials are *interleaved*, as follows. The sum, $\psi_C + \overline{\psi}_C$, is interleaved with the cross-sum operation that eliminates C by sum marginalization, as follows,

$$\{\psi_C\} \oplus \left(\oplus_{c \in sp(C)} \Gamma_{C,(\mathbf{h}_i^{\downarrow \mathbf{H}_C}, c)} \right), \tag{B.2}$$

and the multiplication operation is interleaved with the other operations by multiplying each linear potential added to the cross sum by the probability potential ϕ_{cond}, as shown in line 20 of the pseudocode. For improved efficiency, dominated potentials are pruned from each set created by the cross-sum operation.

B.4. Generalized maximization

Algorithm 7 shows pseudocode for the *Maximize* subroutine, which processes utility potentials when a decision variable is eliminated. It generates the utility potential $\psi_i = \max_D (\psi_D + \overline{\psi}_D)$, by two operations: addition and max-marginalization.

If the domain of either ψ_D or $\overline{\psi}_D$ includes an unobserved chance variable, which is tested for in line 7, the utility potential ψ_i generated by this subroutine is piecewise-linear and concave. Otherwise, it is an ordinary utility potential and the decision variable is eliminated in the same way as in traditional variable elimination.

If the domain of $\overline{\psi}_D$ is empty, and ψ_D has an unobserved variable in its domain, the subroutine creates a new piecewise-linear and concave utility potential ψ_i that is represented by an indexed family of sets of linear potentials, $\{\Gamma_{i,\mathbf{h}_i}\}_{\mathbf{h}_i \in sp(\mathbf{H}_i)}$, where $\mathbf{H}_i$ is the set of relevant observed variables. For each instantiation $\mathbf{h}_i$ of $\mathbf{H}_i$, the set $\Gamma_{i,\mathbf{h}_i}$ contains (before pruning) one linear potential for each action $d \in sp(D)$.

Algorithm 7: Generalized maximize subroutine.

```
1  Function Maximize(D, ψ_D, ψ̄_D)
2      // Efficiently compute ψ_i ← max_D(ψ_D + ψ̄_D), where:
3      //   D is the decision variable to be eliminated
4      //   ψ_D is an ordinary utility potential
5      //   ψ̄_D is a PWLC utility potential represented by {Γ_{h_D}}_{h_D ∈ sp(H_D)},
6      //        where H_D is the set of observed variables in the domain of ψ̄_D
7      if (domU(ψ̄_D) = ∅) and (domU(ψ_D) = ∅) then ψ_i is ordinary potential
8          ψ_i ← max_D ψ_D
9          δ_D ← arg max_D ψ_D
10     else // new utility potential ψ_i is PWLC
11         H_i ← (domH(ψ_D) ∪ domH(ψ̄_D)) \ {D}
12         H_D ← domH(ψ̄_D)
13         foreach h_i ∈ sp(H_i) do
14             Γ_{i,h_i} ← ∅ // initialize set of linear potentials for history h_i
15             if (dom(ψ̄_D) = ∅) then // one linear potential per decision
16                 foreach d ∈ sp(D) do
17                     γ' ← ψ_D^{R(H_i = h_i, D = d)} // new linear potential
18                     d(γ') ← d // save best decision with utility potential
19                     Γ_{i,h_i} ← Γ_{i,h_i} ∪ {γ'}
20                 end
21             else // union of sets
22                 foreach d ∈ sp(D) do
23                     foreach γ ∈ Γ_{(h_i^{↓H_D}, d)} do
24                         γ' ← ψ_D^{R(H_i = h_i, D = d)} + γ // new linear potential
25                         d(γ') ← d // save best decision with linear potential
26                         Γ_{i,h_i} ← Γ_{i,h_i} ∪ {γ'}
27                     end
28                 end
29             end
30             Γ_{i,h_i} ← Prune(Γ_{i,h_i})
31         end
32         δ_D ← {Γ_{i,h_i}}_{h_i ∈ sp(H_i)} // implicit policy representation
33     end
34     return (ψ_i, δ_D) // return new utility potential and corresponding policy
```

If the domain of $\overline{\psi}_D$ is not empty, the subroutine interleaves the addition of ψ_D and $\overline{\psi}_D$ with the set union, $\cup_{d \in sp(D)} \Gamma_{(\mathbf{h}_i^{\downarrow \mathbf{H}_D}, d)}$, by adding $\psi_D^{R(\mathbf{H}_i = \mathbf{h}_i, D = d)}$ to each linear potential in the set union, as shown in the pseudocode.

B.5. Comparison to incremental pruning

As suggested by the names of the three subroutines in the pseudocode given above, generalized variable elimination generalizes the incremental pruning algorithm [33] so that it solves any influence diagram, and not just finite-horizon POMDPs.

From this perspective, it is interesting to compare the steps taken by generalized variable elimination and incremental pruning in solving the same problem. Recall the ten-stage maze POMDP described in Section 2.3.5. In Section 3.3.1, we described how to eliminate the last decision variable before eliminating any other variables, which creates a piecewise-linear and concave utility potential, $\psi_1(B(X_{10}))$, represented by a set of linear potentials, Γ_1, with domain X_{10}. In this section, we describe how to subsequently eliminate the variables X_{10}, Y_{10}, and D_9, and we compare these three elimination steps to a single iteration of the incremental pruning algorithm, which performs the corresponding three steps of backprojection, cross-sum, and maximization.

Eliminate unobserved chance variable X_{10}. When X_{10} is selected for elimination, there are two relevant probability potentials, $P(X_{10}|X_9, D_9)$ and $P(Y_{10}|X_{10}, D_9)$. Eliminating X_{10} from their product by sum-marginalization creates the probability potential:

$$\phi_2(Y_{10}|X_9, D_9) = \sum_{X_{10}} P(X_{10}|X_9, D_9) P(Y_{10}|X_{10}, D_9). \tag{B.3}$$

The following conditional probability potential is also computed for use in processing utility potentials:

$$\phi_{cond}(X_{10}|X_9, D_9, Y_{10}) = \frac{P(X_{10}|X_9, D_9) P(Y_{10}|X_{10}, D_9)}{\phi_2(Y_{10}|X_9, D_9)}. \tag{B.4}$$

The only relevant utility potential is $\psi_1(B(X_{10}))$, which is piecewise-linear and concave. Eliminating the variable X_{10} from $\psi_1(B(X_{10}))$ by sum-marginalization creates a piecewise-linear and concave utility potential, $\psi_{2,(D_9,Y_{10})}(B(X_9))$, represented by an indexed family of sets of linear potentials, $\{\Gamma_{2,(d_9,y_{10})}\}_{(d_9,y_{10})\in sp(D_9,Y_{10})}$, where each set is defined as

$$\Gamma_{2,(d_9,y_{10})} = Prune\left(\gamma^i | i = 1,\ldots,|\Gamma_1|\right), \tag{B.5}$$

and each linear potential $\gamma_2^i \in \Gamma_{2,(d_9,y_{10})}$ is created by *backprojection* from a corresponding linear potential $\gamma_1^i \in \Gamma_1$, as follows. For each state $x_9 \in sp(X_9)$, the value of the linear potential γ_2^i is defined as

$$\gamma^i_{2,(d_9,y_{10})}(x_9) = \sum_{x_{10}\in sp(X_{10})} P(x_{10}|x_9,d_9,y_{10})\gamma_1^i(x_{10}), \tag{B.6}$$

where $P(x_{10}|x_9,d_9,y_{10}) = \phi_{cond}(x_{10}|x_9,d_9,y_{10})$ is a normalized probability.

Eliminate observed chance variable Y_{10}. The only relevant probability potential is $\phi_2(Y_{10}|X_9,D_9)$, and eliminating Y_{10} from this potential by sum-marginalization creates a vacuous probability potential. The only relevant utility potential is the piecewise-linear and concave utility potential, $\psi_2(D_9,Y_{10},B(X_9))$, which is represented by the indexed family of sets of linear potentials, $\{\Gamma_{2,(d_9,y_{10})}\}_{(d_9,y_{10})\in sp(D_9,Y_{10})}$. Multiplying this piecewise-linear and concave utility potential by $\phi_2(Y_{10}|X_9,D_9)$ creates a new piecewise-linear and concave utility potential $\psi_2'(D_9,Y_{10},B(X_9))$, represented by the indexed family of sets of linear potentials, $\{\Gamma'_{2,(d_9,y_{10})}\}_{(d_9,y_{10})\in sp(D_9,Y_{10})}$. Eliminating Y_{10} from $\psi_2'(D_9,Y_{10},B(X_9))$ by sum-marginalization creates a new piecewise-linear and concave utility potential, $\psi_3(D_9,B(X_9))$, represented by an indexed family of sets of linear potentials, $\{\Gamma_{3,d_9}\}_{d_9\in sp(D_9)}$, where each set is defined as

$$\Gamma_{3,d_9} = Prune\left(\left\{\oplus_{y_{10}\in sp(Y_{10})}\Gamma'_{2,(d_9,y_{10})}\right\}\right). \tag{B.7}$$

This step can be performed more efficiently by interleaving the operations on piecewise-linear and concave utility potentials, which makes it unnecessary to create the intermediate piecewise-linear and concave utility potential, $\psi_2'(D_9,Y_{10},B(X_9))$.

Eliminate decision variable D_9. When D_9 is selected for elimination, there is no relevant probability potential and only one relevant utility potential: the piecewise-linear and concave utility potential $\psi_3(D_9,B(X_9))$. Eliminating D_9 by max-marginalization creates the piecewise-linear and concave utility potential, $\psi_4(B(X_9))$, represented by a single set of linear potentials over the unobserved variable X_9, defined as

$$\Gamma_4 = Prune\left(\cup_{d_9\in sp(D_9)}\Gamma_{3,d_9}\right). \tag{B.8}$$

Comparison. The steps performed by generalized variable elimination in eliminating these three variables are equivalent to the steps performed by incremental pruning, but not identical. Equation (B.6) differs from Equation (55) in two minor ways.

First, the probability $P(x_{10}|x_9,d_9,y_{10})$ in Equation (B.6) is different from the probability $P(x_{10},y_{10}|x_9,d_9)$ in Equation (55) for incremental pruning. However, when the observed chance variable Y_{10} is eliminated in the next step, the result is multiplied by the probability $P(y_{10}|x_9,d_9)$, and so, since

$$P(x_{10},y_{10}|x_9,d_9) = P(x_{10}|x_9,d_9,y_{10})\cdot P(y_{10}|x_9,d_9), \tag{B.9}$$

the two computations are equivalent.

Second, the reward term $R(x_t,d_t)/|sp(Y_{t+1})|$ in Equation (55) is missing from Equation (B.6). For the maze problem, there is no reward function associated with the decision variable D_9. But if there were, the reward would be added to the piecewise-linear and concave utility potential $\psi_4(B(X_9))$ when eliminating the decision variable D_9 by max-marginalization, which reflects the following alternative equations for the value functions computed by incremental pruning:

$$V_t(b_t) = \max_{d_t\in sp(D_t)}\left(R_t(b_t,d_t) + V_t^{d_t}(b_t)\right) \tag{B.10}$$

$$V_t^{d_t}(b_t) = \sum_{z_{t+1}} V_t^{d_t,y_{t+1}}(b_t) \tag{B.11}$$

$$V_t^{d_t,y_{t+1}}(b_t) = P(y_{t+1}|b_t,d_t)V_{t+1}(\tau(b_t,d_t,y_{t+1})). \tag{B.12}$$

Obviously, these equations are equivalent to Equations (51), (52) and (53) for incremental pruning, with the difference that the reward is included in Equation (B.10) instead of Equation (B.12).

References

[1] R. Howard, J. Matheson, Influence diagrams, in: R. Howard, J. Matheson (Eds.), Readings on the Principles and Applications of Decision Analysis, vol. II, Strategic Decisions Group, Menlo Park, CA, 1981, pp. 719–762.
[2] A. Miller, M. Merkhofer, R. Howard, J. Matheson, T. Rice, Development of automated aids for decision analysis, Tech. Report 3309, SRI International, Menlo Park, CA, 1976.
[3] R. Shachter, Evaluating influence diagrams, Oper. Res. 34 (6) (1986) 871–882.
[4] J. Tatman, R. Shachter, Dynamic programming and influence diagrams, IEEE Trans. Syst. Man Cybern. 20 (2) (1990) 365–379.
[5] F. Jensen, T. Nielsen, Probabilistic decision graphs for optimization under uncertainty, Ann. Oper. Res. 204 (1) (2013) 223–248.
[6] M. Gómez, Real-world applications of influence diagrams, in: J. Gámez, S. Moral, A. Salmerón (Eds.), Advances in Bayesian Networks: Studies in Fuzziness and Soft Computing, vol. 146, Springer-Verlag, 2004, pp. 161–180.
[7] G. Monahan, A survey of partially observable Markov decision processes: theory, models, and algorithms, Manag. Sci. 28 (1982) 1–16.
[8] W. Lovejoy, A survey of algorithmic methods for partially observed Markov decision processes, Ann. Oper. Res. 28 (1991) 47–66.
[9] C. White III, A survey of solution techniques for the partially observed Markov decision process, Ann. Oper. Res. 32 (1991) 215–230.
[10] M. Spaan, Partially observable Markov decision processes, in: M. Wiering, M. van Otterlo (Eds.), Reinforcement Learning: State of the Art, Springer-Verlag, 2012, pp. 387–414.
[11] A. Dutech, B. Scherrer, Partially observable Markov decision processes, in: Markov Decision Processes in Artificial Intelligence, John Wiley & Sons, Inc., 2010, pp. 185–228, Ch. 7.
[12] C. Boutilier, D. Poole, Computing optimal policies for partially observable Markov decision processes using compact representations, in: Proc. of 13th National Conf. on Artificial Intelligence, 1996, pp. 1168–1175.
[13] E. Hansen, Z. Feng, Dynamic programming for POMDPs using a factored state representation, in: Proc. of 5th International Conf. on AI Planning Systems, 2000, pp. 130–139.
[14] T. Veiga, M. Spaan, P. Lima, Point-based POMDP solving with factored value function approximation, in: Proc. of the 28th AAAI Conf. on Artificial Intelligence, 2014, pp. 2512–2518.
[15] G. Shani, Task-based decomposition of factored POMDPs, IEEE Trans. Cybern. 44 (2) (2014) 208–216.
[16] D. Bertsekas, Dynamic Programming and Optimal Control, vol. 1, 3rd edition, Athena Scientific, Belmont, MA, 2005.
[17] F. Jensen, T. Nielsen, Bayesian Networks and Decision Graphs, 2nd edition, Springer-Verlag, New York, 2007.
[18] H. Raiffa, Decision Analysis, Addison-Wesley, Reading, MA, 1968.
[19] P. Shenoy, Valuation based systems for Bayesian decision analysis, Oper. Res. 40 (1992) 463–484.
[20] R. Cowell, A. Dawid, S. Lauritzen, D. Spiegelhalter, Probabilistic Networks and Expert Systems, Springer-Verlag, New York, 1999.
[21] M. Arias, F. Díez, Cost-effectiveness analysis with influence diagrams, Methods Inf. Med. 54 (2015) 353–358.
[22] R. Cabañas, A. Cano, M. Gómez-Olmedo, A. Madsen, Improvements to variable elimination and symbolic probabilistic inference for evaluating influence diagrams, Int. J. Approx. Reason. 70 (2016) 13–35.
[23] M. Luque, M. Arias, F. Díez, Synthesis of strategies in influence diagrams, in: Proc. of 33rd Conf. on Uncertainty in Artificial Intelligence, 2017.
[24] R. Dechter, A new perspective on algorithms for optimizing policies under uncertainty, in: Proc. of 5th International Conf. on Artificial Intelligence Planning Systems, 2000, pp. 72–81.
[25] F. Jensen, F. Jensen, S. Dittmer, From influence diagrams to junction trees, in: Proc. of 10th Conf. on Uncertainty in Artificial Intelligence, 1994, pp. 367–373.
[26] S. Olmsted, On representing and solving decision problems, Ph.D. thesis, Stanford University, 1983.
[27] P. Ndilikilikesha, Potential influence diagrams, Int. J. Approx. Reason. 10 (3) (1994) 251–285.
[28] M. Luque, F. Díez, Variable elimination for influence diagrams with super-value nodes, Int. J. Approx. Reason. 51 (6) (2010) 615–631.
[29] R. Marinescu, A. Razak, N. Wilson, Multi-objective influence diagrams, in: Proc. of 28th Conf. on Uncertainty in Artificial Intelligence, 2012, pp. 574–583.
[30] R. Cabañas, A. Antonucci, A. Cano, M. Gómez-Olmedo, Evaluating interval-valued influence diagrams, Int. J. Approx. Reason. 80 (2017) 393–411.
[31] L. Kaelbling, M. Littman, A. Cassandra, Planning and acting in partially observable stochastic domains, Artif. Intell. 101 (1998) 99–134.
[32] R. Smallwood, E. Sondik, The optimal control of partially observable Markov processes over a finite horizon, Oper. Res. 21 (1973) 1071–1088.
[33] A. Cassandra, M. Littman, N. Zhang, Incremental pruning: a simple, fast, exact method for partially observable Markov decision processes, in: Proc. of 13th Conf. on Uncertainty in Artificial Intelligence, 1997, pp. 54–61.
[34] M. Littman, The Witness Algorithm: Solving partially observable Markov decision processes, Tech. Rep. CS-94-40, Dept. of Computer Science, Brown University, 1994.
[35] E. Walraven, M. Spaan, Accelerated vector pruning for optimal POMDP solvers, in: Proc. of 31st AAAI Conf. on Artificial Intelligence, 2017, pp. 3672–3678.
[36] E. Hansen, T. Bowman, Improved vector pruning in exact algorithms for solving POMDPs, in: Proc. of 36th Conf. on Uncertainty in Artificial Intelligence, vol. 124, PMLR, 2020.
[37] D. Nilsson, M. Hohle, Computing bounds on expected utilities for optimal policies based on limited information, Tech. Rep. 94, Danish Informatics Network in the Agricultural Sciences, 2001.
[38] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.
[39] R.D. Shachter, Bayes-ball: the rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams), in: Proc. of 14th Conf. in Uncertainty in Artificial Intelligence, 1998, pp. 480–487.
[40] M. Araya-Lopez, V. Thomas, O. Buffet, F. Charpillet, A closer look at MOMDPs, in: Proc. of 22nd International Conf. on Tools with Artificial Intelligence, 2010, pp. 197–204.
[41] S. Ong, S. Png, D. Hsu, W.S. Lee, Planning under uncertainty for robotic tasks with mixed observability, Int. J. Robot. Res. 29 (8) (2010) 1053–1068.
[42] C. Boutilier, R. Dearden, M. Goldszmidt, Stochastic dynamic programming with factored representations, Artif. Intell. 121 (1) (2000) 49–107.
[43] J. Hoey, R. St-Aubin, A. Hu, C. Boutilier, SPUDD: stochastic planning using decision diagrams, in: Proc. of 15th Conf. on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, 1999, pp. 279–288.
[44] T. Degris, O. Sigaud, Factored Markov decision processes, in: O. Sigaud, O. Buffet (Eds.), Markov Decision Processes in Artificial Intelligence, John Wiley & Sons, Inc., 2010, pp. 113–139, Ch. 4.
[45] K. Katsikopoulos, S. Engelbrecht, Markov decision processes with delays and asynchronous cost collection, IEEE Trans. Autom. Control 48 (4) (2003) 568–574.
[46] S. Thiébaux, C. Gretton, J. Slaney, D. Price, F. Kabanza, Decision-theoretic planning with non-Markovian rewards, J. Artif. Intell. Res. 25 (2006) 17–74.
[47] J. Bander, C. White, Markov decision processes with noise-corrupted and delayed state observations, J. Oper. Res. Soc. 50 (6) (1999) 660–668.
[48] J. Tatman, Decision processes in influence diagrams: Formulation and analysis, Ph.D. thesis, Stanford University, 1986.
[49] E. Hansen, J. Shi, A. Khaled, A POMDP approach to influence diagram evaluation, in: Proc. of 25th International Joint Conf. on Artificial Intelligence, AAAI Press, 2016, pp. 3124–3132.
[50] W. Zhang, N. Zhang, Restricted value iteration: theory and algorithms, J. Artif. Intell. Res. 23 (1) (2005) 123–165.

[51] D. Mauá, Equivalences between maximum a posteriori inference in Bayesian networks and maximum expected utility computation in influence diagrams, Int. J. Approx. Reason. 68 (2016) 211–229.
[52] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.
[53] U. Kjærulff, Triangulation of graphs–algorithms giving small total state space, Tech. Rep. R-90-09, Univ. of Aalborg, Dept. of Math. and Comp. Sci, Denmark, 1990.
[54] R. Cabañas, A. Cano, M. Gómez-Olmedo, A. Madsen, Heuristics for determining the elimination ordering in the influence diagram evaluation with binary trees, in: Proc. of the 12th Scandinavian Conf. on Artificial Intelligence, in: Frontiers in Artificial Intelligence and Applications, vol. 257, IOS Press, 2013, pp. 65–74.