Submodel Decomposition Bounds for Influence Diagrams

Junkyu Lee[1,2], **Radu Marinescu**[2], **Rina Dechter**[1]
[1] University of California Irvine
[2] IBM Research
junkyul@uci.edu, radu.marinescu@ie.ibm.com, dechter@ics.uci.edu

Abstract

Influence diagrams (IDs) are graphical models for representing and reasoning with sequential decision-making problems under uncertainty. Limited memory influence diagrams (LIMIDs) model a decision-maker (DM) who forgets the history in the course of making a sequence of decisions. The standard inference task in IDs and LIMIDs is to compute the maximum expected utility (MEU), which is one of the most challenging tasks in graphical models. We present a model decomposition framework in both IDs and LIMIDs, which we call submodel decomposition that generates a tree of single-stage decision problems through a tree clustering scheme. We also develop a valuation algebra over the submodels that leads to a hierarchical message passing algorithm that propagates conditional expected utility functions over a submodel-tree as external messages. We show that the overall complexity is bounded by the maximum tree-width over the submodels, common in graphical model algorithms. Finally, we present a new method for computing upper bounds over a submodel-tree by first exponentiating the utility functions yielding a standard probabilistic graphical model as an upper bound and then applying standard variational upper bounds for the marginal MAP inference, yielding tighter upper bounds compared with state-of-the-art bounding schemes for the MEU task.

Introduction

Influence diagrams (IDs) (Howard and Matheson 1981) which extend Bayesian networks (BNs) with decision variables and utility functions, are discrete graphical models for single-agent sequential decision making under uncertainty. The standard inference task in IDs is finding the maximum expected utility (MEU) and a set of optimal policy functions that jointly achieve the MEU under given constraints on the available information at each decision.

IDs assume perfect recall, namely that the agent or decision maker (DM) is "no-forgetting" and the sequence of previous decisions and immediate observations are all available when making a decision. The naive evaluation of the MEU leads to algorithms that are space and time exponential in the length of the history (Howard and Matheson 1981), hence, earlier works focused on variable elimination algorithms with the complexity bounded by the treewidth (Bodlaender 1988) derived from tree decomposition methods that are tied to the algorithms for solving for IDs. A valuation algebra (Kohlas and Shenoy 2000) for IDs was introduced by (Jensen, Jensen, and Dittmer 1994) together with a variable elimination algorithm over a corresponding constrained junction tree. More recently, (Dechter 2000) presented an improved variable elimination algorithms with lower width parameters, while (Pralet, Schiex, and Verfaillie 2006) developed a multi-cluster operator DAG (MC-DAG) decomposition to capture the lowest possible treewidth of IDs. Despite this progress, the MEU task remains clearly intractable so current research focuses on developing bounding schemes. One type of bounds uses the principle of information relaxation, thereby relaxing the constrained ordering of observations and decision variables and allowing more observations to be available to the DM (Nilsson and Hohle 2001; Jensen and Gatti 2010; Yuan, Wu, and Hansen 2010). Other approaches are based on translation and exploit the equivalence between the MEU and the Marginal MAP (MMAP) tasks (Mauá 2016), thus applying bounding schemes for MMAP to the translated MEU (Liu and Ihler 2012; Marinescu, Dechter, and Ihler 2014; Mauá and Cozman 2016). Finally, state-of-the-art decomposition bounds (Lee, Ihler, and Dechter 2018; Lee et al. 2019) apply decomposition schemes directly to IDs using methods such as weighted mini-bucket (WMB) (Liu and Ihler 2011; Marinescu et al. 2018), and generalized dual decomposition (GDD) (Ping, Liu, and Ihler 2015). Yet, the quality of those decomposition bounds quickly degrades due to non-convex optimization routines that must be applied.

Relaxing the perfect recall condition leads to limited memory influence diagrams (LIMIDs) that pose additional challenges to the MEU task. A limited memory agent only has access to partial knowledge in the sequence of decisions and observations which means that exact algorithms for LIMIDs must jointly optimize multiple policy functions by exhaustive enumeration (Mauá, de Campos, and Zaffalon 2012). Iterative local policy update algorithms can reduce the complexity by locally improving a subset of policy functions. However, local policy search algorithms assume that the local policy evaluation is easy (Lauritzen and Nilsson 2001; Detwarasiti and Shachter 2005; Mauá and Cozman 2016). To the best of our knowledge, the only upper bounds available in the literature for LIMIDs are the theoretical er-

ror bounds presented in (Mauá and Cozman 2016).

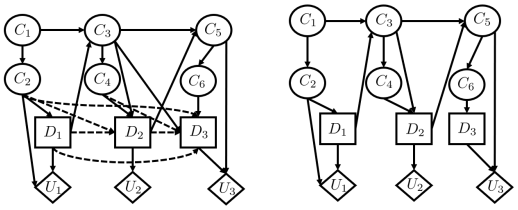
Another line of research related to the decomposition of IDs and LIMIDs studies the structure of DAGs to identify the relevant random variables for optimizing a single policy function. Specifically, (Nielsen and Jensen 1999) introduced d-separation criteria for identifying relevant utility functions and observed variables for a decision, and (Nielsen 2001) presented decomposition of IDs. For LIMIDs, (Zhang and Poole 1992; Lauritzen and Nilsson 2001) defined soluble LIMIDs in which the variable elimination for IDs can apply. (Detwarasiti and Shachter 2005) extended the d-separation criteria for identifying relevant random variables in LIMIDs while updating a single policy function.

Contribution: In this paper, we take a fresh look at IDs and LIMIDs by viewing the underlying DAG as a causal diagram and provide a new unified tree decomposition method, called a *submodel-tree decomposition (STD)*. For IDs, each node in the submodel-tree is a single-time step decision problem for evaluating value functions, and it is equivalent to MC-DAGs (Pralet, Schiex, and Verfaillie 2006). For LIMIDs, each node in the submodel-tree captures multiple decision variables that must be optimized jointly. The STD offers a new tree clustering framework by using d-separation criteria over the causal DAG and it facilitates hierarchical message passing algorithms, whose complexity can be characterized by a graph width parameter applicable to both IDs and LIMIDs. Since exact message passing algorithms over the STD is intractable, we develop convex variational decomposition bounds over the STD, which allow us to reuse MMAP bounds via Jensen's inequality applied to log moment generating functions. Our experimental results show that the new bounding scheme generates bounds that are orders of magnitudes tighter than those obtained by the current methods in IDs and LIMIDs.

Background

Graphical Models

A graphical model $\mathcal{M}$ is a tuple $\langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of discrete random variables, $\mathbf{D} = \{D_1, \ldots, D_n\}$ are their finite domains of values, and $\mathbf{F} = \{F_1(\mathbf{X}_1), \ldots F_r(\mathbf{X}_r)\}$ is a set of non-negative functions with each function $F_k \in \mathbf{F}$ being defined over a subset of variables $\mathbf{X}_k \subseteq \mathbf{X}$ called its scope and denoted by $\mathrm{sc}(F_k)$. The graphical model encodes a joint probability distribution $P(\mathbf{X}) = \frac{1}{Z} \prod_{F_k \in \mathbf{F}} F_k(\mathbf{X}_k)$, where Z is the normalization constant or partition function. The primal graph of $\mathcal{M}$ is an undirected graph in which nodes are associated with variables and two nodes are connected if they appear together in the scope of some function. The join-tree decomposition obtained by triangulating the primal graph offers exact message passing algorithms with the worst-case complexity characterized by the treewidth of the primal graph (Dechter 2013). The join-graph decomposition further decomposes a join-tree into possibly a loopy join-graph $\mathcal{G}_J = \langle \mathcal{C}, \mathcal{S} \rangle$ that satisfies the running intersection property. Each cluster $C \in \mathcal{C}$ is assigned with a subset of variables $\mathbf{X}_C \subseteq \mathbf{X}$ and functions $\mathbf{F}_C \subseteq \mathbf{F}$ such that $\mathrm{sc}(F_k) \subseteq \mathbf{X}_\mathcal{C}$ for all $F_k \in \mathbf{F}_C$, and a separator $S_{ij} \in \mathcal{S}$ between clusters C_i and C_j is labeled by $\mathbf{X}_{S_{ij}} = \mathbf{X}_{C_i} \cap \mathbf{X}_{C_j}$. (Mateescu et al. 2010).



(a) Perfect Recall ID (b) Limited Memory ID

Figure 1. Example of an ID and LIMID

The MMAP inference task eliminates variables either by maximization or summation and, it can be formulated as computing the weighted log partition function,

$$\Phi_{\mathbf{W}}(\theta) = \log \sum_{X_n}^{W_n} \ldots \sum_{X_1}^{W_1} \exp(\theta(\mathbf{X})), \tag{1}$$

where $\theta(\mathbf{X}) = \sum_{F_k \in \mathbf{F}} \log F_k(\mathbf{X}_k)$, and the powered-sum elimination operator $\sum_X^w f(X) = [\sum_X f(X)^{1/w}]^w$ generalizes the maximization ($w = 0^+$) and the summation ($w = 1$) with weights $0 \leq w \leq 1$ for a variable X (Liu and Ihler 2011). When combined with variational inference (Wainwright and Jordan 2008), a decomposition scheme provides variational upper bounds derived from a certain constrained optimization problem with the decomposition graph encoding constraints. The generalized dual decomposition bound (GDD) (Ping, Liu, and Ihler 2015) provides convex upper bounds that are obtained by duplicating all the shared variables between cluster nodes in the join-graph,

$$\Phi_{\mathbf{W}}(\theta) \leq \sum_{C_i \in \mathcal{C}} \log \sum_{\mathbf{X}_{C_i}}^{\mathbf{W}^{C_i}} \exp\Big[\theta_{C_i}(\mathbf{X}_{C_i}) + \sum_{S_{ij} \in \mathcal{S}} \delta_{S_{ij}}(\mathbf{X}_{S_{ij}})\Big], \tag{2}$$

where $\theta_{C_i}(\mathbf{X}_{C_i}) = \sum_{F_k \in \mathbf{F}_C} \log F_k(\mathbf{X}_k)$, the weights $\mathbf{W}^{C_i} = \{W_k^{C_i} | X_k \in \mathbf{X}_{C_i}\}$ are assigned to a variable $\mathbf{X}_k$ such that $W_k = \sum_{C_i \in \mathcal{C}} W_k^{C_i}$, and $\delta_{S_{ij}}(\mathbf{X}_{S_{ij}})$ is the cost-shifting functions over a separator that cancels each other $S_{ij}(\mathbf{X}_{S_{ij}}) = -S_{ji}(\mathbf{X}_{S_{ij}})$. GDD offers a message passing algorithm that tightens the bound by optimizing the weights $\mathbf{W}^{C_i}$ and $\delta_{C_i, C_j}(\mathbf{X}_{C_i} \cap \mathbf{X}_{C_j})$ between clusters. Often, empirical studies report that a simpler weighted mini-bucket with moment-matching (WMB-MM) algorithm that optimizes the cost-shifting functions only once over the decomposition graph with larger cluster size turns out to be effective in difficult problems (Liu and Ihler 2011; Marinescu, Dechter, and Ihler 2014).

Influence Diagrams

An ID $\mathcal{M}$ is a tuple $\langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$ consisting of a set of discrete chance variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, a set of discrete decision variables $\mathbf{D} = \{D_1, \ldots, D_m\}$, a set of conditional probability functions $\mathbf{P} = \{P_1, \ldots, P_n\}$ associated with the chance variables, a set of additive utility functions $\mathbf{U} = \{U_1, \ldots, U_r\}$, and a set of precedence relations $\mathcal{O}$

that we will specify shortly. We denote the finite domain of a variable $X \in \mathbf{X} \cup \mathbf{D}$ by Ω_X, and $\Omega_{\mathbf{S}}$ is the Cartesian product of individual domains Ω_X of variables X in a set $\mathbf{S} \subseteq (\mathbf{X} \cup \mathbf{D})$, i.e., $\Omega_{\mathbf{S}} = \times_{X \in \mathbf{S}} \Omega_X$. Given a node X in a DAG $\mathcal{G}$, we denote the set of parents, children, ancestors, descendants, and family of X by $\mathrm{pa}(X)$, $\mathrm{ch}(X)$, $\mathrm{an}(X)$, $\mathrm{de}(X)$, and $\mathrm{fa}(X) = \mathrm{pa}(X) \cup \{X\}$, respectively. Clearly, the definitions above extend to a set of nodes $\mathbf{X}$ by taking the union of individual sets. Figure 1 illustrates an example of IDs with and without the perfect recall condition which we adapted from the Pigs example (Lauritzen and Nilsson 2001). We associate nodes in a DAG $\mathcal{G}$ with the chance variables drawn as circles, the decision variables drawn as squares, and the utility functions drawn as diamonds. The directed arcs toward any chance node X_i in $\mathcal{G}$ specify the scope of the conditional probability function $P_i\big(X_i|\mathrm{pa}(X_i)\big)$. The parents of a value node U_i define the scope of the utility or value function $U_i\big(\mathrm{pa}(U_i)\big)$. The directed arcs toward decision variables are called informational arcs, and they specify the set of precedence relations $\mathcal{O}_C = \{\mathrm{pa}(D_k) \prec \{D_k\} | D_k \in \mathbf{D}\}$, where $\mathrm{pa}(D_k)$ is the set of variables observed immediately before decision D_k. Under the perfect recall, DM knows the decision order $\mathcal{O}_D = \{\mathbf{D}_1 \prec \mathbf{D}_2 \prec \ldots \prec \mathbf{D}_T\}$ that constitutes the history $\mathbf{h}_t = \mathbf{h}_{t-1} \cup \mathrm{pa}(\mathbf{D}_t)$, where $\mathbf{h}_0 = \emptyset$, for making decisions $\mathbf{D}_t$ at the t-th time step. In case of LIMIDs, DM does not have access to $\mathcal{O}_D$ and each decision D_k is made solely based on immediate observations $\mathrm{pa}(D_k)$. The solid arcs directed toward decision variables in Figure 1a and Figure 1b specify the $\mathcal{O}_C$. The dotted directed arcs in Figure 1a are the explicit informational arcs specifying the history that are omitted in conventional IDs. By explicitly including these informational arcs in the DAG of IDs, we can express $\mathbf{h}_t$ as simply $\mathbf{h}_t = \mathrm{pa}(D_k)$ for $D_k \in \mathbf{D}_t$. In this paper, we encode the perfect recall condition of IDs by explicit informational arcs and IDs will refer to both IDs with or without the perfect recall condition. A deterministic policy function $\Delta_k\big(D_k|\mathrm{pa}(D_k)\big)$ for a decision variable D_k in both IDs and LIMIDs is a mapping $\Delta_k : \Omega_{\mathrm{pa}(D_k)} \mapsto \Omega_{D_k}$, and the set of all policy functions $\mathbf{\Delta} = \{\Delta_k\big(D_k|\mathrm{pa}(D_k)\big) | D_k \in \mathbf{D}\}$ is called a strategy. The MEU task is to compute the MEU and the global maximum (optimal) strategy $\mathbf{\Delta}^*$. Namely,

$$\mathrm{MEU} := \max_{\mathbf{\Delta}} \mathbb{E}\Big[\sum_{U_i \in \mathbf{U}} U_i\Big] \tag{3}$$

$$= \max_{\mathbf{\Delta}} \sum_{\mathbf{X} \cup \mathbf{D}} \Big(\prod_{P_k \in \mathbf{P}} P_k\Big) \cdot \Big(\prod_{\Delta_k \in \mathbf{\Delta}} \Delta_k\Big) \cdot \Big(\sum_{U_k \in \mathbf{U}} U_k\Big), \tag{4}$$

$$\mathbf{\Delta}^* = \operatorname*{argmax}_{\mathbf{\Delta}} \mathbb{E}[\sum_{U_i \in \mathbf{U}} U_i]. \tag{5}$$

Submodel-Tree Decomposition

A DM controls the decision variables in an ID $\mathcal{M}$. Deliberate choices by the DM based on some strategy $\tilde{\mathbf{\Delta}}$ result in the expected utility $\mathbb{E}_{\tilde{\mathbf{\Delta}}}[\sum_{U_i \in \mathbf{U}} U_i]$. Therefore, we view the DAG $\mathcal{G}$ of $\mathcal{M}$ as a causal diagram and make explicit use of graph concepts from causal graphs (Pearl 2009). In this section, we identify a relevant subset of $\mathcal{M}$ for computing the local MEU in the partial evaluation of $\mathcal{M}$ denoted by $\mathrm{LMEU}_{\mathcal{M}(\mathbf{D}'\mathbf{U}')}$, which is a local maximum conditional expected utility obtained by maximizing the sum of expected utilities over a subset of utility functions $\mathbf{U}' \subseteq \mathbf{U}$ on policy functions for a subset decision variables $\mathbf{D}' \subseteq \mathbf{D}$. We call the relevant subset of $\mathcal{M}$ for computing $\mathrm{LMEU}_{\mathcal{M}(\mathbf{D}'\mathbf{U}')}$ *submodel* $\mathcal{M}'(\mathbf{D}', \mathbf{U}')$, and show that it can be identified by the backdoor and front door criteria (Pearl 2009). Next, we define graph-based combination and marginalization operations over the submodels and develop a valuation algebra over the system of submodels in an ID. It is known that the valuation algebra satisfies the axioms of local computation (Shenoy and Shafer 1990), so we can obtain a cluster tree decomposition, which we call submodel-tree decomposition, that assigns submodels to each cluster. We subsequently develop a hierarchical message passing algorithm over the submodel-tree. The complexity of exact algorithms over this tree can be characterized by the maximum width of the individual submodel clusters, which we call submodel width w_s.

Submodels in the MEU Task

Given $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$, we define a partially specified strategy for a subset of decision variables $\mathbf{D}' \subseteq \mathbf{D}$ by $(\mathbf{\Delta}_{\mathbf{D}'} \cup \tilde{\mathbf{\Delta}}_{\mathbf{D} \setminus \mathbf{D}'})$, where $\mathbf{\Delta}_{\mathbf{D}'} = \{\Delta\big(D_k|\mathrm{pa}(D_k)\big) | D_k \in \mathbf{D}'\}$ and $\tilde{\mathbf{\Delta}}_{\mathbf{D} \setminus \mathbf{D}'}$ is a set of arbitrary policy functions for $\mathbf{D} \setminus \mathbf{D}'$. The partial evaluation of $\mathcal{M}$ over $(\mathbf{D}', \mathbf{U}')$ computes a local maximum conditional expected utility and finds the local maximum strategy $(\mathbf{\Delta}^*_{\mathbf{D}'} \cup \tilde{\mathbf{\Delta}}_{\mathbf{D} \setminus \mathbf{D}'})$ as follows.

Definition 1. *The local maximum conditional expected utility evaluated on $\mathbf{D}' \subseteq \mathbf{D}$ and $\mathbf{U}' \subseteq \mathbf{U}$ in $\mathcal{M}$ is defined by*

$$\mathit{LMEU}_{\mathcal{M}(\mathbf{D}', \mathbf{U}')} := \max_{\mathbf{\Delta}_{\mathbf{D}'}} \mathbb{E}\Big[\sum_{U_i \in \mathbf{U}'} U_i | \mathit{pa}(\mathbf{D}')\Big], \tag{6}$$

where the conditional expectation is taken over the conditional probability $P(\mathbf{X} \cup \mathbf{D})/P\big(\mathit{pa}(\mathbf{D}')\big)$ and the local maximum conditional expected utility is obtained by a local maximum strategy $(\mathbf{\Delta}^*_{\mathbf{D}'} \cup \tilde{\mathbf{\Delta}}_{\mathbf{D} \setminus \mathbf{D}'})$.

Definition 2. *The set of relevant variables for evaluating $\mathit{LMEU}_{\mathcal{M}(\mathbf{D}', \mathbf{U}')}$ with an arbitrary fixed $\tilde{\mathbf{\Delta}}_{\mathbf{D} \setminus \mathbf{D}'}$ is the subset of observed variables $\mathit{REL}_O(\mathbf{D}', \mathbf{U}') \subseteq \mathit{pa}(\mathbf{D}')$ and the subset of hidden variables $\mathit{REL}_H(\mathbf{D}', \mathbf{U}') \subseteq \big((\mathbf{X} \cup \mathbf{D}) \setminus \mathit{pa}(\mathbf{D}')\big)$ so that Eq. (6) can be evaluated by*

$$\mathit{LMEU}_{\mathcal{M}(\mathbf{D}', \mathbf{U}')} = \max_{\mathbf{\Delta}_{\mathbf{D}'}} \mathbb{E}\Big[\sum_{U_i \in \mathbf{U}'} U_i \big| \mathit{REL}_O(\mathbf{D}', \mathbf{U}')\Big], \tag{7}$$

where the conditional expectation is taken over the relevant subset of variables $P\big(\mathit{REL}_H(\mathbf{D}', \mathbf{U}') | \mathit{REL}_O(\mathbf{D}', \mathbf{U}')\big)$. The local maximum strategy remains the same under the relevant observed variables $\mathit{REL}_O(\mathbf{D}', \mathbf{U}')$.

Definition 3. *A submodel $\mathcal{M}'(\mathbf{D}', \mathbf{U}')$ of an ID $\mathcal{M}$ relevant to $\mathit{LMEU}_{\mathcal{M}(\mathbf{D}', \mathbf{U}')}$ is a tuple $\langle \mathbf{X}', \mathbf{D}', \mathbf{P}', \mathbf{U}', \mathcal{O}' \rangle$ such that $\mathbf{X}' = \mathbf{X} \cap \big(\mathit{REL}_O(\mathbf{D}', \mathbf{U}') \cup \mathit{REL}_H(\mathbf{D}', \mathbf{U}')\big)$, $\mathbf{P}' = \{P_k\big(X_k|\mathit{pa}(X_k)\big) | X_k \in \mathbf{X}', \mathit{sc}(P_k) \in (\mathbf{X}' \cup \mathbf{D}')\}$, and $\mathcal{O}' = \big\{\mathit{REL}_O(\mathbf{D}', \mathbf{U}') \prec \mathbf{D}' \prec \mathit{REL}_H(\mathbf{D}', \mathbf{U}')\big\}$.*

The relevant observed variables and hidden variables can be found by graph separation criteria. In the following, the backdoor set for a pair $(\mathbf{X}, \mathbf{Y})$ is denoted by $\text{BD}(\mathbf{X}, \mathbf{Y})$, and the front door set is denoted by $\text{FD}(\mathbf{X}, \mathbf{Y})$ (Pearl 2009).

Proposition 1. *For computing $\text{LMEU}_{\mathcal{M}(\mathbf{D}', \mathbf{U}')}$, the relevant observed variables $\text{REL}_O(\mathbf{D}', \mathbf{U}')$ is a subset of $pa(\mathbf{D}')$ that forms $BD(\mathbf{D}', \mathbf{U}')$, and the relevant hidden variables $\text{REL}_H(\mathbf{D}', \mathbf{U}')$ is a subset of $\big(\mathbf{X} \cup \mathbf{D} \setminus pa(D')\big)$, where each variable in $\text{REL}_H(\mathbf{D}', \mathbf{U}')$ is a member of any $FD(pa(\mathbf{D}'), ch(\mathbf{U}'))$, where $ch(\mathbf{U}')$ is an auxiliary child node for the value nodes $\mathbf{U}'$.*

We say that a relevant set is minimal if we cannot reduce the set further. Intuitively, $\text{REL}_O(\mathbf{D}', \mathbf{U}')$ is a subset of observed variables that shields all the influence between the decision nodes $\mathbf{D}'$ and the value nodes $\mathbf{U}'$, and $\text{REL}_H(\mathbf{D}', \mathbf{U}')$ is a subset of hidden variables that mediates influence between the decision nodes $\mathbf{D}'$ and value nodes $\mathbf{U}'$. Next, we introduce a utility random variable and show lemmas for proving the above proposition.

Definition 4 (Utility Random Variables). *Given a utility function $U_k \in \mathbf{U}$ in an ID $\mathcal{M}$, we define W_k as a random variable with its domain defined by*

$$\Omega_{W_k} = range(U_k) \quad s.t.\ U_k : sc(U_k) \to range(U_k), \tag{8}$$

where $range(U_k)$ is the set of outcomes of the utility function U_k. The probability associated with W_k can be defined by

$$P\big(W_k{=}w_k\big){=}\sum_{pa(U_k)} P\big(pa(U_k)\big) \cdot \mathbb{I}\big(U_k(pa(U_k)) = w_k\big), \tag{9}$$

where $w_k \in range(U_k)$, $\mathbb{I}\big(U_k(pa(U_k)) = w_k\big)$ is the indicator function, and $P\big(pa(U_k)\big)$ is the marginal probability over the set of random variables $pa(U_k)$ in $\mathcal{M}$.

By using the utility random variable, we can rewrite the expected utility by $\mathbb{E}\Big[U_k\Big] = \sum_{w_k \in \Omega_{W_k}} w_k \cdot P(W_k = w_k)$. In the presence of multiple utility functions $\mathbf{U}'$, we first add a new child value node to all value nodes $\mathbf{U}'$, and define a utility random variable for the newly added value node.

Lemma 1 (Identifying Relevant Observed Nodes). *Let $\langle B_k \rangle_{k \in \mathbb{N}}$ be a nested sequence of subsets of $pa(\mathbf{D}')$ such that (1) $B_0 = pa(\mathbf{D}')$, and (2) $B_k \subseteq B_{k-1}$, where each B_k is a $BD(\mathbf{D}', \mathbf{U}')$. Then, for a set $\text{REL}_O(\mathbf{D}', \mathbf{U}')$, there exists a sequence $\langle B_k \rangle_{k \in \mathbb{N}}$ that reaches to the minimal $\text{REL}_O(\mathbf{D}', \mathbf{U}')$.*

Proof. By induction, let's assume that B_k is a $\text{BD}(\mathbf{D}', \mathbf{U}')$ and $\mathbf{Y} = \text{pa}(\mathbf{D}') \backslash B_k$. Since B_k is a backdoor set, B_k blocks every path between $\mathbf{D}'$ and $\mathbf{U}'$ that contains an arrow into $\mathbf{D}'$. The $\text{LMEU}_{\mathcal{M}(\mathbf{D}', \mathbf{U}')}$ can be written by

$$\max_{\Delta\left(\mathbf{D}'|\text{pa}(\mathbf{D}')\right)} \mathbb{E}\Big[\sum_{U_i \in \mathbf{U}'} U_i \big|\text{pa}(\mathbf{D}')\Big] \tag{10}$$

$$= \max_{\Delta\left(\mathbf{D}'|\text{pa}(\mathbf{D}')\right)} \sum_{\mathbf{D}'} \Delta\big(\mathbf{D}'|\text{pa}(\mathbf{D}')\big) \sum_{W} W{\cdot}P(W|\text{fa}(\mathbf{D}')) \tag{11}$$

$$= \max_{\Delta\left(\mathbf{D}'|B_k\right)} \sum_{\mathbf{X}'_D} \Delta(\mathbf{D}'|B_k) \sum_{W} W \cdot P(W|\mathbf{D}', B_k). \tag{12}$$

Since B_k is a $\text{BD}(\mathbf{D}', \mathbf{U}')$, the joint policy function $\Delta\big(\mathbf{D}'|\text{pa}(\mathbf{D}')\big)$ over $\mathbf{D}'$ only depends on B_k, i.e., $\Delta\big(\mathbf{D}'|\text{pa}(\mathbf{D}')\big) = \Delta(\mathbf{D}'|B_k)$, and $P(W|\mathbf{Y}, B_k) = P(W|B_k)$. The input ID has a finite number of nodes, and by removing irrelevant observed nodes we eventually reach to the minimal $\text{REL}_O(\mathbf{D}', \mathbf{U}')$. □

Lemma 2 (Identifying Relevant Hidden Nodes). *Let $\langle F_k \rangle_{k \in \mathbb{N}}$ be a nested sequence of subsets of $(\mathbf{X} \cup \mathbf{D}) \setminus (fa(\mathbf{D}'))$ such that $F_0 = \emptyset$ and $F_{k-1} \subseteq F_k$ where each F_k is generated by adding a node $X \in FD(fa(\mathbf{D}'), ch(\mathbf{U}'))$, where $ch(\mathbf{U}')$ is an auxiliary utility node added as a child of all utility nodes $\mathbf{U}'$. Then, for a set $\text{REL}_H(\mathbf{D}', \mathbf{U}')$, there exists a sequence $\langle F_k \rangle_{k \in \mathbb{N}}$ that reaches to the minimal $\text{REL}_H(\mathbf{D}', \mathbf{U}')$.*

Proof. Suppose a set of hidden variables $\mathbf{Z}$ satisfies $\mathbf{Z} \subseteq \text{FD}\big(\text{fa}(\mathbf{D}'), \text{ch}(\mathbf{U}')\big)$. Introducing a new utility random variable W for the node $\text{ch}(\mathbf{U}')$, the $\text{LMEU}_{\mathcal{M}(\mathbf{D}', \mathbf{U}')}$ can be written by

$$\max_{\Delta\left(\mathbf{D}'|\text{pa}(\mathbf{D}')\right)} \sum_{W} W \cdot P\big(W|\text{pa}(\mathbf{D}')\big) \tag{13}$$

$$= \max_{\Delta(\mathbf{D}'|\text{pa}(\mathbf{D}'))} \sum_{W} W \sum_{\mathbf{D}'} \Delta\big(\mathbf{D}'|\text{pa}(\mathbf{D}')\big){\cdot}P\big(W|\text{fa}(\mathbf{D}')\big) \tag{14}$$

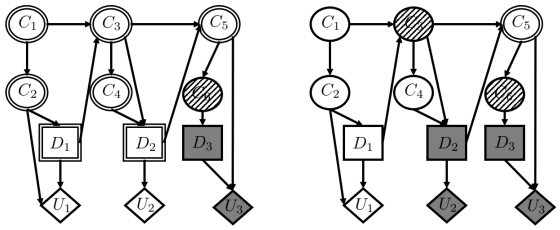
$$= \max_{\Delta\left(\mathbf{D}'|\text{pa}(\mathbf{D}')\right)} \sum_{W} W \sum_{\mathbf{D}'} \Delta\big(\mathbf{D}'|\text{pa}(\mathbf{D}')\big) \cdot \Big[\sum_{\mathbf{Z}} P\big(\mathbf{Z}|\text{fa}(\mathbf{D}')\big) \cdot P\big(W|\text{fa}(\mathbf{D}'), \mathbf{Z}\big)\Big]. \tag{15}$$

Since $\mathbf{Z} \subseteq \text{FD}\big(\text{fa}(\mathbf{D}'), W\big)$, Eq. (15) shows that the hidden variables $\mathbf{Z}$ are relevant to $\text{LMEU}_{\mathcal{M}(\mathbf{D}', \mathbf{U}')}$. Now suppose that a hidden node T is not a member of any $\text{FD}\big(\text{fa}(\mathbf{D}'), W\big)$. Following the definition of the front door criteria, there are two possible cases in IDs. First, the node T is not an ancestor of W. Namely, T is a barren node for evaluating $\text{LMEU}_{\mathcal{M}(\mathbf{D}', \mathbf{U}')}$. Second, the node T is an ancestor of W, but it is d-separated from W given $\text{pa}(\mathbf{D}')$. In both cases, the hidden random variable T is not relevant to $\text{LMEU}_{\mathcal{M}(\mathbf{D}', \mathbf{U}')}$ since the variable T can be marginalized out without inducing any change to $\text{LMEU}_{\mathcal{M}(\mathbf{D}', \mathbf{U}')}$. The input ID has a finite number of nodes, and by removing all irrelevant hidden nodes from the set of all hidden nodes, we obtain the minimal $\text{REL}_H(\mathbf{D}', \mathbf{U}')$. □

The MEU task in a submodel $\mathcal{M}'(\mathbf{D}', \mathbf{U}')$ is subject to change due to the changes in $\mathbf{U}'$ and $\tilde{\mathbf{\Delta}}_{\mathbf{D} \backslash \mathbf{D}'}$. Therefore, we want to capture a submodel $\mathcal{M}'(\mathbf{D}', \mathbf{U}')$ such that the evaluation of the local maximum conditional expected utility $\text{LMEU}_{\mathcal{M}(\mathbf{D}', \mathbf{U}')}$ and the local maximum policy functions $\mathbf{\Delta}^*_{\mathbf{D}'}$ remains stable regardless of the changes in the utility and policy functions outside of the submodel.

Definition 5. *We say a submodel $\mathcal{M}'(\mathbf{D}', \mathbf{U}')$ is stable if the set of relevant hidden variables does not contain any unobserved decision variable.*

Given a fixed set of policy functions $\tilde{\mathbf{\Delta}}_{\mathbf{D} \backslash \mathbf{D}'}$, irrelevant utility functions do not change the local maximum policy



(a) $\mathcal{M}'(\{D_3\},\{U_3\})$ (b) $\mathcal{M}'(\{D_2,D_3\},\{U_2,U_3\})$

Figure 2. Example of submodels. See Example 1.

functions $\mathbf{\Delta}^*_{\mathbf{D}'}$. The relevant utility functions can be identified from $\mathcal{G}$ by $\text{REL}_U(\mathbf{D}') = \text{de}(\mathbf{D}') \cup \mathbf{U}$ (Nielsen and Jensen 1999).

Proposition 2. *If a submodel $\mathcal{M}'(\mathbf{D}', \textit{REL}_U(\mathbf{D}'))$ is stable, the MEU task in $\mathcal{M}'$ is independent to policy functions outside of the submodel $\tilde{\mathbf{\Delta}}_{\mathbf{D}\setminus\mathbf{D}'}$, and the local maximum policy functions $\mathbf{\Delta}^*_{\mathbf{D}'}$ extend to the global maximum strategy $\mathbf{\Delta}^*$.*

Example 1. *Figure 2 shows submodels in the LIMID from Figure 1b. In both examples, the decision nodes and value nodes of the submodel are shaded, the relevant observed nodes are marked with stripes, and the relevant hidden nodes are marked with a double-lined boarder. Figure 2a depicts $\mathcal{M}'(\{D_3\},\{U_3\})$, where $\textit{REL}_O(\{D_3\},\{U_3\}) = \{C_6\}$ since removing C_6 from $pa(D_3)$ opens a backdoor path between D_3 and U_3. $\textit{REL}_H(\{D_3\},\{U_3\})$ contains all hidden variables $\{C_1, C_2, D_1, C_3, C_4, D_2, C_5\}$ since influence between $pa(D_3)$ and $ch(U_3)$ propagates through C_5. The submodel $\mathcal{M}'(\{D_3\},\{U_3\})$ is not stable since $\textit{REL}_H(\{D_3\},\{U_3\})$ contains unobserved decision variables D_1 and D_2 that may change the local maximum policy function $\Delta^*(D_3|C_3)$. On the other hand, Figure 2b shows a stable submodel $\mathcal{M}'(\{D_2,D_3\},\{U_2,U_3\})$. We can see that C_4 can be removed from $pa(\{D_2,D_3\})$ because C_4 does not open any backdoor path, and the relevant observed variables are $\{C_3, C_6\}$. $\textit{REL}_H(\{D_2,D_3\},\{U_2,U_3\})$ is $\{C_5\}$ since other unobserved variables C_1, C_2, and D_1 are not in any $FD(\{D_2,D_3\},\{U_2,U_3\})$. The utility functions $\{U_2,U_3\}$ in the submodel is $\textit{REL}_U(\{D_2,D_3\})$.*

Valuation Algebra Over Submodels

A valuation algebra is a system of potentials with combination and marginalization operations (Kohlas and Shenoy 2000). In this section, we develop a valuation algebra for solving IDs that uses submodels as potentials. We introduce the structural concepts and necessary definitions for the valuation algebra over submodels.

Proposition 3. *Given a submodel $\mathcal{M}'$, the probability and policy functions can be factorized as*

$$P(\mathbf{X}'_{H,up}|\mathbf{X}'_{O,out})P(\mathbf{X}'_{O,in}|\mathbf{X}'_{O,out},\mathbf{X}'_{H,up})\Delta(\mathbf{D}'|\mathbf{X}'_{O,in},\mathbf{X}'_{O,out})$$
$$P(\mathbf{X}'_{H,down}|\mathbf{X}'_{O,out},\mathbf{X}'_{H,up},\mathbf{X}'_{O,in},\mathbf{D}'), \quad (16)$$

where $\mathbf{X}'_{O,in}$ is a subset of relevant observed variables whose parents are included in submodel, $\mathbf{X}'_{O,out}$ is the rest of the relevant observed variables, $\mathbf{X}'_{H,up}$ is the relevant hidden variables that are ancestors of relevant observed variables, and $\mathbf{X}'_{H,down}$ is the rest of the relevant hidden variables. The joint policy function $\Delta(\mathbf{D}'|\mathbf{X}'_{O,in},\mathbf{X}'_{O,out})$ is factorized as $\prod_{\Delta_k\in\mathbf{\Delta}_{\mathbf{D}'}}\Delta_k\big(D_k|pa(D_k)\cap(\mathbf{X}'_{O,in}\cup\mathbf{X}'_{O,out})\big)$.

In Figure 2b, we can inspect $\mathbf{X}'_{\text{O,in}} = \{C_6\}$, $\mathbf{X}'_{\text{O,out}} = \{C_3\}$, and $\mathbf{X}'_{\text{H,up}} = \{C_5\}$. By using the factorization in Eq. (16), we can compute the conditional expected utility from the submodel $\mathcal{M}'$ by

$$\mathbb{E}[\sum_{U_i\in\mathbf{U}'} U_i|\mathbf{X}'_{\text{O,out}}] \quad (17)$$

and avoid unnecessary marginalization involved in Eq. (7). Since the conditional expected utility is a function over $\mathbf{X}'_{\text{O,out}}$, we will call it a set of interface variables.

Definition 6. *The domain of a submodel $\Omega_{\mathcal{M}'}$ is the set of all variables $\mathbf{X}'\cup\mathbf{D}'$ in the submodel.*

Definition 7. *The combination operation $\otimes$ is a binary operation, $\mathcal{M}'(\mathbf{D}',\mathbf{U}') = \mathcal{M}'_1(\mathbf{D}'_1,\mathbf{U}'_1)\otimes\mathcal{M}'_2(\mathbf{D}'_2,\mathbf{U}'_2)$ such that $\mathbf{D}' = \mathbf{D}'_1\cup\mathbf{D}'_2$ and $\mathbf{U}' = \mathbf{U}'_1\cup\mathbf{U}'_2$.*

The marginalization operation is defined by a projection operation $\Downarrow_{\mathbf{Y}}$ that marginalizes variables $(\mathbf{X}'\cup\mathbf{D}')\setminus\mathbf{Y}$.

Definition 8. *We denote the projection operation of a submodel $\mathcal{M}'(\mathbf{D}',\mathbf{U}')$ to $\mathcal{M}''(\mathbf{D}'',\mathbf{U}'')$ by*

$$\mathcal{M}''(\mathbf{D}'',\mathbf{U}'') = \Downarrow_{\mathbf{Y}}\mathcal{M}'(\mathbf{D}',\mathbf{U}'),$$

where $\mathcal{M}''$ is a tuple $\langle\mathbf{X}'',\mathbf{D}'',\mathbf{P}'',\mathbf{U}'',\mathcal{O}''\rangle$ with variables $\mathbf{X}''$ and $\mathbf{D}''$ projected on $\mathbf{Y}$. For a probability function $P_k\in\mathbf{P}''$, if $sc(P_k)\subseteq\mathbf{Y}$, then P_k remains the same in $\mathbf{P}''$. Otherwise, we marginalize $sc(P_k)\setminus\mathbf{Y}$ from P_k by

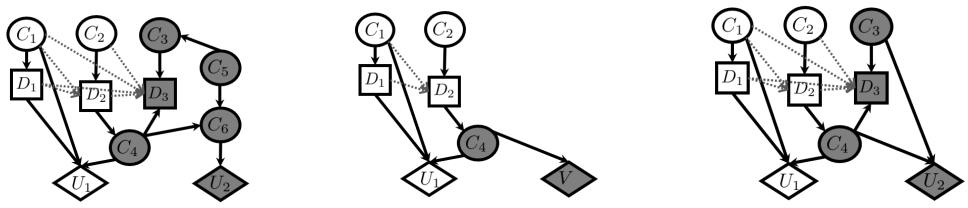
$$\frac{\sum_{sc(P_k)\setminus\mathbf{Y}}P_k\big(X_k,pa(X_k)\big)}{\sum_{sc(P_k)\setminus\mathbf{Y}}P_k\big(pa(X_k)\big)}. \quad (18)$$

For a utility function $U_k\in\mathbf{U}''$, if $sc(P_k)\not\subseteq\mathbf{Y}$, then we marginalize $sc(P_k)\setminus\mathbf{Y}$ from U_k by

$$\sum_{sc(P_k)\setminus\mathbf{Y}}U_kP\Big(an(U_k)\setminus\mathbf{Y}|pa\big((an(U_k)\cup\{U_k\})\setminus\mathbf{Y}\big)\Big). \quad (19)$$

Although the definition of the marginalization operation is lengthy, the graph manipulation is intuitive as shown in Figure 3.

Example 2. *Figure 3 illustrates the marginalization operation applied to the ID shown in Figure 3a. We will consider a submodel $\mathcal{M}'(\{D_3\},\{U_2\})$ highlighted by the shaded nodes. Figure 3b shows the case where we are marginalizing out all but $\mathbf{X}_{O,\,out} = \{X_4\}$. We can see that all the nodes in $\mathcal{M}'(\{D_3\},\{U_2\})$ are removed and the new value node V is added according to Eq.* (19)*. Figure 3c shows another case where we are marginalizing out $\{C_5, C_6\}$. We can see that both hidden variables are removed and U_2 is also replaced by a new value function $V(C_3, C_4)$. Although the diagram does not show the changes in the parameters of the probability function, $P(C_3)$ is also updated according to Eq.* (18)*.*



(a) ID and $\mathcal{M}'(\{D_3\},\{U_2\})$ (b) $\Downarrow_{\{C_4\}} \mathcal{M}'(\{D_3\},\{U_2\})$ (c) $\Downarrow_{\{C_3,C_4,D_3\}} \mathcal{M}'(\{D_3\},\{U_2\})$

Figure 3. Example of the marginalization operation for a submodel. See Example 2 for the details.

While solving LIMIDs, we can refine the decision order $\mathcal{O}_D$ from the reverse topological order of decision nodes in $\mathcal{G}$. Note that the decision order $\mathcal{O}_D$ is already specified in $\mathcal{O}$ in IDs with perfect recall. We next define a decision order $\mathcal{O}_D$ relative to $\mathcal{G}$ as follows.

Definition 9. *Given an $\mathcal{M}$, $\mathcal{O}_D = \{\mathbf{D}_1 \prec \ldots \prec \mathbf{D}_T\}$ is a sequence of disjoint subsets of decision variables, found along with a sequence of submodels $\langle \mathcal{M}_t \rangle_{t=1}^{T}$ and DAGs $\langle \mathcal{G}_t \rangle_{t=1}^{T}$ such that:*

1. *$\mathcal{G}_T = \mathcal{G}$,*
2. *$\mathcal{M}_t$ is a stable submodel $\mathcal{M}'\big(\mathbf{D}_t, \mathit{REL}_U(\mathbf{D}_t)\big)$ in $\mathcal{G}_t$, where $\mathbf{D}_t$ is a set of decision variables in $\mathcal{G}_t$ chosen from the last to the first decision nodes in the reverse topological order in $\mathcal{G}_t$,*
3. *$\mathcal{G}_{t-1}$ is generated from $\mathcal{G}_t$ by deleting $\mathbf{D}_t$ and $\mathit{REL}_U(\mathbf{D}_t)$ and adding a new value node $V(\mathbf{X}'_{O,\,out})$ from $\mathcal{M}_t$.*

Following the decision order $\mathcal{O}_D$ relative to $\mathcal{G}$, we can obtain a set of stable submodels $\mathbf{M}_{\mathcal{O}_\mathbf{D}}$ in $\mathcal{M}$. To define the valuation algebra, let $\mathbb{M}_{\mathcal{O}_D}$ denote a closure of submodels in $\mathbf{M}_{\mathcal{O}_\mathbf{D}}$, and $\mathbb{D}_{\mathcal{O}_D}$ denote the set of domains of submodels in $\mathbb{M}_{\mathcal{O}_D}$.

Theorem 1. *Given an ID $\mathcal{M}$ and the decision ordering $\mathcal{O}_D$ relative to $\mathcal{G}$, a tuple $\Upsilon_\mathcal{M} = \langle \mathbb{M}_{\mathcal{O}_D}, \mathbb{D}_{\mathcal{O}_D}, \otimes, \Downarrow \rangle$ is a valuation algebra.*

Next, we prove that the system of stable submodels in an ID $\mathcal{M}$ relative to the decision order $\mathcal{O}_D$, $\Upsilon_\mathcal{M}$, satisfies the axioms of valuation algebra (Kohlas and Shenoy 2000).

Definition 10 (Neutral Submodels). *Given a submodel $\mathcal{M}' := \langle \mathbf{X}', \mathbf{D}', \mathbf{P}', \mathbf{U}', \mathcal{O}' \rangle$ of an ID $\mathcal{M}$, we say $\mathcal{M}'$ is a neutral submodel of $\mathcal{M}$ if $\mathbf{P}' = \emptyset$ and $\mathbf{U}' = \{\mathbf{0}(X) | X \in \mathbf{X}'\}$, where $\mathbf{0}(X)$ is a constant function $\mathbf{0}(X) : X \to 0$. We denote a neutral submodel over a variable X by $\mathcal{M}'_{\mathbf{0}(X)}$, a neutral submodel over a set of variables $\mathbf{X}$ by $\mathcal{M}'_{\mathbf{0}(\mathbf{X})} = \otimes_{X \in \mathbf{X}} \mathcal{M}'_{\mathbf{0}(X)}$, and a set of neutral submodels over each variable $X \in \mathbf{X}$ by $\mathbf{M}_{\mathbf{0}(\mathbf{X})} = \{\mathcal{M}'_{\mathbf{0}(X)} | X \in \mathbf{X}\}$.*

Lemma 3 (Neutral Elements). *For $\mathbf{X}, \mathbf{Y} \in \mathbb{D}_{\mathcal{O}_D}$,*

$$\mathcal{M}'_{\mathbf{0}(\mathbf{X})} \otimes \mathcal{M}'_{\mathbf{0}(\mathbf{Y})} = \mathcal{M}'_{\mathbf{0}(\mathbf{X} \cup \mathbf{Y})}.$$

Proof. The set of utility functions in $\mathcal{M}'_{\mathbf{0}(\mathbf{X})}$ and $\mathcal{M}'_{\mathbf{0}(\mathbf{Y})}$ is $\{\mathbf{0}(X_i) | X_i \in \mathbf{X}\}$ and $\{\mathbf{0}(X_i) | X_i \in \mathbf{Y}\}$, respectively. It is immediate to see that the set of utility functions in $\mathcal{M}'_{\mathbf{0}(\mathbf{X})} \otimes \mathcal{M}'_{\mathbf{0}(\mathbf{Y})}$ and $\mathcal{M}'_{\mathbf{0}(\mathbf{X} \cup \mathbf{Y})}$ are $\{\mathbf{0}(X_i) | X_i \in \mathbf{X} \cup \mathbf{Y}\}$. □

Lemma 4 (Semigroup). *$\mathbb{M}_{\mathcal{O}_D}$ is a semigroup with the combination operation over submodels.*

Proof. The combination operation is closed in $\mathbb{M}_{\mathcal{O}_D}$. Given $\mathcal{M}_1, \mathcal{M}_2 \in \mathbb{M}_{\mathcal{O}_D}$, the combination of submodels $\mathcal{M}_1 \otimes \mathcal{M}_2$ is defined as a component-wise union of the sets in submodels. Therefore, the combination operator is commutative and associative, showing that $\mathbb{M}_{\mathcal{O}_D}$ is a semigroup. □

Lemma 5 (Domain of Combination). *For $\mathcal{M}'_1, \mathcal{M}'_2 \in \mathbb{M}_{\mathcal{O}_D}$, $\Omega_{\mathcal{M}'_1 \otimes \mathcal{M}'_2} = \Omega_{\mathcal{M}'_1} \cup \Omega_{\mathcal{M}'_2}$.*

Proof. Since $\mathcal{M}'_1(\mathbf{D}'_1, \mathbf{U}'_1)$ and $\mathcal{M}'_2(\mathbf{D}'_2, \mathbf{U}'_2)$ are stable submodels, each domain contains all relevant variables for computing the $\mathrm{LMEU}_{\mathcal{M}(\mathbf{D}'_1, \mathbf{U}'_1)}$ and $\mathrm{LMEU}_{\mathcal{M}(\mathbf{D}'_2, \mathbf{U}'_2)}$, respectively. Without loss of generality, we assume that $\mathbf{D}'_1 \cap \mathbf{D}'_2 = \emptyset$ and $\mathbf{U}'_1 \cap \mathbf{U}'_2 = \emptyset$. Any non-neutral submodel in $\mathbb{M}_{\mathcal{O}_D}$ can be reduce to the combination of submodels in $\mathbf{M}_{\mathcal{O}_D}$. Let $\mathbf{D}'_1 \prec \mathbf{D}'_2$ in $\mathcal{O}_D$. Then, $\mathbf{U}'_1$ is not relevant to $\mathbf{D}'_2$ by construction of $\mathbf{M}_{\mathcal{O}_D}$. Therefore $\mathrm{REL}_O(\mathbf{D}'_2, \mathbf{U}'_1) = \emptyset$ and $\mathrm{REL}_H(\mathbf{D}'_2, \mathbf{U}'_1) = \emptyset$. If $\mathbf{D}'_2 \notin \mathrm{de}(\mathbf{D}'_1)$, then $\mathrm{REL}_O(\mathbf{D}'_1, \mathbf{U}'_2) = \emptyset$ and $\mathrm{REL}_H(\mathbf{D}'_1, \mathbf{U}'_2) = \emptyset$. If $\mathbf{D}'_2 \in \mathrm{de}(\mathbf{D}'_1)$, then $\mathrm{REL}_O(\mathbf{D}'_1 \cup \mathbf{D}'_2, \mathbf{U}'_2) = \mathrm{REL}_O(\mathbf{D}'_2, \mathbf{U}'_2)$ and $\mathrm{REL}_H(\mathbf{D}'_1 \cup \mathbf{D}'_2, \mathbf{U}'_2) = \mathrm{REL}_H(\mathbf{D}'_2, \mathbf{U}'_2)$. Therefore, $\mathrm{REL}_O(\mathbf{D}'_1 \cup \mathbf{D}'_2, \mathbf{U}'_1 \cup \mathbf{U}'_2) = \mathrm{REL}_O(\mathbf{D}'_1, \mathbf{U}'_1) \cup \mathrm{REL}_O(\mathbf{D}'_2, \mathbf{U}'_2)$, and $\mathrm{REL}_H(\mathbf{D}'_1 \cup \mathbf{D}'_2, \mathbf{U}'_1 \cup \mathbf{U}'_2) = \mathrm{REL}_H(\mathbf{D}'_1, \mathbf{U}'_1) \cup \mathrm{REL}_H(\mathbf{D}'_2, \mathbf{U}'_2)$ □

Lemma 6 (Marginalization). *For any stable submodel $\mathcal{M}' \in \mathbb{M}_{\mathcal{O}_D}$ and $\mathbf{X} \in \mathbb{D}_{\mathcal{O}_D}$, the marginalization satisfies $\Downarrow_\mathbf{X} \mathcal{M}' = \Downarrow_{\mathbf{X} \cap \Omega_{\mathcal{M}'}} \mathcal{M}', \Omega_{\Downarrow_\mathbf{X} \mathcal{M}'} = \mathbf{X} \cap \Omega_{\mathcal{M}'}$, and $\Downarrow_{\Omega_{\mathcal{M}'}} \mathcal{M}' = \mathcal{M}'$.*

Proof. Two projections $\mathcal{M}''_1 := \Downarrow_\mathbf{X} \mathcal{M}'$ and $\mathcal{M}''_2 := \Downarrow_{\mathbf{X} \cap \Omega_{\mathcal{M}'}} \mathcal{M}'$ are equivalent by definition given a stable submodel $\mathcal{M}' := \langle \mathbf{X}', \mathbf{D}', \mathbf{P}', \mathbf{U}', \mathcal{O}' \rangle$. Since $\Omega_\mathbf{X}$ is $\mathbf{X}$, $\Omega_{\Downarrow_X \mathcal{M}'}$ is $\mathbf{X} \cap (\mathbf{X}' \cup \mathbf{D}')$. Finally, projecting a submodel to its domain leaves the submodel unchanged. □

Lemma 7 (Transitivity of Marginalization). *For a stable submodel $\mathcal{M}' \in \mathbb{M}_{\mathcal{O}_D}$ and $\mathbf{X} \in \mathbb{D}_{\mathcal{O}_D}$, $\Downarrow_\mathbf{X} (\Downarrow_\mathbf{Y} \mathcal{M}') = \Downarrow_{\mathbf{X} \cap \mathbf{Y}} \mathcal{M}'$.*

Proof. Let a stable submodel $\mathcal{M}'$ in $\mathbb{M}_{\mathcal{O}_D}$ be denoted by $\mathcal{M}' := \langle \mathbf{X}', \mathbf{D}', \mathbf{P}', \mathbf{U}', \mathcal{O}' \rangle$ and the submodels after

marginalization by

$$\mathcal{M}''^{L} := \langle \mathbf{X}''^{L}, \mathbf{D}''^{L}, \mathbf{P}''^{L}, \mathbf{U}''^{L}, \mathcal{O}''^{L} \rangle = \Downarrow_{\mathbf{X}} (\Downarrow_{\mathbf{Y}} \mathcal{M}'),$$
$$\mathcal{M}''^{R} := \langle \mathbf{X}''^{R}, \mathbf{D}''^{R}, \mathbf{P}''^{R}, \mathbf{U}''^{R}, \mathcal{O}''^{R} \rangle = \Downarrow_{\mathbf{X}\cap\mathbf{Y}} \mathcal{M}'.$$

Two submodels are equivalent if both have the same sets of variables, domains, functions, and constrained ordering. By definition of marginalization operation, $\mathbf{X}''^{L} = (\mathbf{X}' \cap \mathbf{Y}) \cap \mathbf{X} = \mathbf{X}' \cap (\mathbf{X} \cap \mathbf{Y}) = \mathbf{X}''^{R}$. For the probability functions, consider a conditional probability function $P\big(X_i'|\mathrm{pa}(X_i')\big)$. There are 3 possible cases when marginalizing out $\mathbf{Y}$ and $\mathbf{X}$ in sequence. First, the probability function remains the same in the marginalized submodel $\mathcal{M}''^{L}$ if $\mathrm{sc}(P\big(X_i'|\mathrm{pa}(X_i')\big)) \subset \mathbf{Y}$ and $\mathrm{sc}(P\big(X_i'|\mathrm{pa}(X_i')\big)) \subset \mathbf{X}$. Second, a probability function $P\big(X_i'|\mathrm{pa}(X_i')\big)$ is removed from $\mathcal{M}''^{L}$ if $X_i' \in \mathbf{X}' \setminus \mathbf{Y}$ or $X_i' \in (\mathbf{X}' \cap \mathbf{Y}) \setminus \mathbf{X}$, which is equivalent to the condition $X_i' \in \mathbf{X}' \setminus (\mathbf{X} \cap \mathbf{Y})$. The last case is marginalizing the subset of $\mathrm{pa}\big(X_i'\big)$, resulting in eliminating all but $\big(\mathrm{pa}\big(X_i'\big) \cap (\mathbf{X}' \cap \mathbf{Y})\big) \cap (\mathbf{Y} \cap \mathbf{X})$, which is equivalent to $\mathrm{pa}\big(X_i'\big) \cap (\mathbf{X}' \cap (\mathbf{X} \cap \mathbf{Y}))$. From the above cases, we can see that the probability functions in $\mathcal{M}''^{L}$ and $\mathcal{M}''^{R}$ are the same. For a utility function U_i, if U_i remains in $\mathcal{M}''^{L}$, it will also remain in $\mathcal{M}''^{R}$ for the similar reasoning as shown in the probability functions. Lastly, $\mathbf{D}''^{L} = \mathbf{D}''^{R}$ is immediate due to $\mathbf{X}''^{L} = \mathbf{X}''^{R}$, and $\mathcal{O}''^{L} = \mathcal{O}''^{R}$ since the relevant sets are the same in $\mathcal{M}''^{L}$ and $\mathcal{M}''^{R}$. □

Lemma 8 (Distributivity of Marginalization). *For a pair of stable submodels $\mathcal{M}_1', \mathcal{M}_2' \in \mathbb{M}_{\mathcal{O}_D}$ with $\Omega_{\mathcal{M}_1'} = \mathbf{X}$, $\Downarrow_{\mathbf{X}} (\mathcal{M}_1' \otimes \mathcal{M}_2') = \mathcal{M}_1' \otimes (\Downarrow_{\mathbf{X}} \mathcal{M}_2')$.*

Proof. We show the equivalence by comparing two submodels,

$$\mathcal{M}''^{L} := \langle \mathbf{X}''^{L}, \mathbf{D}''^{L}, \mathbf{P}''^{L}, \mathbf{U}''^{L}, \mathcal{O}''^{L} \rangle = \Downarrow_{\mathbf{X}} \mathcal{M}_1' \otimes \mathcal{M}_2',$$
$$\mathcal{M}''^{R} := \langle \mathbf{X}''^{R}, \mathbf{D}''^{R}, \mathbf{P}''^{R}, \mathbf{U}''^{R}, \mathcal{O}''^{R} \rangle = \mathcal{M}_1' \otimes \big(\Downarrow_{\mathbf{X}} \mathcal{M}_2'\big).$$

$\mathbf{X}''^{L} = (\mathbf{X} \cup \mathbf{X}_2') \cap \mathbf{X} = \mathbf{X} \cup (\mathbf{X}_2' \cap \mathbf{X}) = \mathbf{X}''^{R}$. The projection operation applies to each element in the set of functions. Since $\Omega_{\mathcal{M}_1'} = \mathbf{X}$, the functions in $\mathcal{M}_1'$ remain the same in $\mathcal{M}''^{L}$. The rest of the functions in $\mathcal{M}''^{L}$ can be obtained by $\Downarrow_{\mathbf{X}} \mathcal{M}_2'$. Therefore, $\mathbf{P}''^{L} = \mathbf{P}''^{R}$ and $\mathbf{U}''^{L} = \mathbf{U}''^{R}$. The equivalence of $\mathbf{D}''^{L}$ and $\mathbf{D}''^{R}$ is immediate due to $\mathbf{X}''^{L} = \mathbf{X}''^{R}$. Lastly, $\mathcal{O}''^{L} = \mathcal{O}''^{R}$ since the relevant sets of the observed variables and the hidden variables in both $\mathbf{X}''^{L}$ and $\mathbf{X}''^{R}$ are the same. □

Any valuation algebra satisfies the axiom for local computation. Therefore, the MEU task in IDs can also be solved by local computations using the valuation algebra over the stable submodels relative to $\mathcal{O}_D$.

Submodel-Tree Clustering

The tree decomposition framework for inference tasks over graphical models offers the generic architecture for local computation (Shenoy 1997; Kask et al. 2005). We now present the submodel-tree decomposition for IDs.

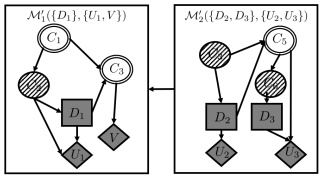


Figure 4. Submodel-tree decomposition $\mathcal{T}_{ST}$

Definition 11. *Given an ID $\mathcal{M}$, and the set of stable submodels $\mathbf{M}_{\mathcal{O}_D}$ relative to the decision ordering $\mathcal{O}_D$, the submodel-tree decomposition is a tuple $\mathcal{T}_{ST} := \langle T, \chi, \psi \rangle$, where $T = (\mathcal{C}, \mathcal{S})$ is a tree of cluster nodes $\mathcal{C}$ and separator edges $\mathcal{S}$, χ is a labelling function that maps a node $C \in \mathcal{C}$ to a set of chance and decision variables $\chi(C) = \mathbf{X}^C \cup \mathbf{D}^C$, and ψ maps a node $C \in \mathcal{C}$ to a set of stable submodels $\psi(C) \subset \mathbf{M}_{\mathcal{O}_D}$. In addition, $T = (\mathcal{C}, \mathcal{S})$ should satisfy the running intersection property. Namely, for any variable X, a set of cluster nodes including the variable induces a connected subtree of T.*

Now, we use an example to illustrate the submodel-tree decomposition $\mathcal{T}_{\mathrm{ST}}$.

Example 3. *Figure 4 shows the $\mathcal{T}_{ST}$ of the LIMID shown in Figure 1b. The submodel-tree decomposition starts by identifying $\mathcal{O}_D$ and a set of stable submodels $\mathbf{M}_{\mathcal{O}_D}$. Beginning with $\mathcal{M}'(D_3, U_3)$, we see that $\mathit{REL}_H(D_3, U_3)$ contains unobserved decision variables, so we combine $\mathcal{M}'(\{D_3\},\{U_3\})$ and $\mathcal{M}'(\{D_2\},\{U_2,U_3\})$ to find a stable submodel $\mathcal{M}_2'(\{D_2,D_3\},\{U_2,U_3\})$ and $\mathbf{D}_2' = \{D_2, D_3\}$. After adding $\mathcal{M}_2'$ to the submodel-tree, we delete $\mathbf{D}_2'$ and $\{U_2, U_3\}$ and add a new value node $V(C_3)$, which is the conditional expected utility computed from $\mathcal{M}_2'$. We can also find $\mathbf{D}_1' = \{D_1\}$ and $\mathcal{M}_1'(\mathbf{D}_1, \{U_1, V\})$ in the same way and complete the submodel-tree decomposition of $\mathcal{M}$.*

Evaluating IDs over Submodel-Trees

The submodel-tree decomposition $\mathcal{T}_{\mathrm{ST}}$ facilitates the MEU task in IDs by identifying independent single-stage decision problems, where each submodel cluster defines a single time step decision problem in perfect recall IDs. In case of LIMIDs, a submodel cluster may span decision variables over multiple time steps and it is required to perform joint optimization over the policy functions in the submodel. If a LIMID is soluble (Lauritzen and Nilsson 2001), each cluster only contains a single decision variable that allows applying the exact variable elimination algorithm.

Hierarchical Message Passing

Algorithm 1 presents a hierarchical message passing scheme over a submodel-tree decomposition. The message propagation over the submodel-tree is a single pass procedure starting from the last decision stage to the first. The evaluation of each submodel is performed by a subroutine $Eval(\mathcal{M}^C)$, which is any exact algorithm for solving IDs or LIMIDs. This step is internal message passing for computing the conditional expected utility V^C at cluster $\mathcal{M}^C$. The V^C propagates from the leaf nodes to the root node and the algorithm

Algorithm 1 Hierarchical Message Passing over $\mathcal{T}_{\text{ST}}$

Require: $\mathcal{T}_{\text{ST}} := \langle T(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$
Ensure: Submodel-tree augmented with messages sent out from clusters, MEU

```
MEU ← 0
for each cluster C from the leaves to the root in T do
    Pull messages from incoming edges
    Update submodel M^C at C
    V^C ← Eval(M^C)
    if C is the root node or V^C is a constant then
        MEU ← MEU + V^C
    else
        Push V^C to the outgoing edge
return MEU
```

returns the MEU at the root. The message at each cluster C can be computed by

$$V^C(\mathbf{X}^C_{\text{O,out}}) = \max_{\boldsymbol{\Delta}_{\mathbf{D}^C}} \mathbb{E}[\sum_{U_i \in \mathbf{U}^C} U_i \big| \mathbf{X}^C_{\text{O, out}}], \tag{20}$$

where $\mathbf{D}^C$ and $\mathbf{U}^C$ denote the decision variables and utility functions at cluster C and $\mathbf{X}^C_{\text{O,out}}$ is the interface variables at cluster C.

Theorem 2. *The time and space complexity for solving IDs over the submodel-tree decomposition is exponential in the submodel-tree width $w_s := \max_{C \in \mathcal{C}} w_c(C)$, which is the maximum of the constrained tree width $w_c(C)$ of individual submodels.*

Variational Decomposition Bounds

Exact evaluation of individual submodels is still infeasible for practical problems and recent works focus on developing bounding schemes having an anytime behavior that can trade computational resources for the quality of the bounds. The main obstacle for developing efficient variational bounds for the MEU task is due to the additive utility functions, which hinders formulating a dual representation of the inference task. Therefore, we propose to use Jensen's inequality (Jensen 1906) applied to the log moment generating function, $\mathbb{E}[X] \leq \log \mathbb{E}[e^X]$, which exponentiates the utility functions in IDs to bound the MEU by the log partition function of a probabilistic graphical model.

Proposition 4. *The MEU task in a submodel can be bounded by the Marginal MAP task after exponentiating the utility functions,*

$$V^C(\mathbf{X}^C_{O,out}) \leq \max_{\boldsymbol{\Delta}_{\mathbf{D}^C}} \log \mathbb{E}[e^{\sum_{U_i \in \mathbf{U}^C} U_i} \big| \mathbf{X}^C_{O,\,out}] \tag{21}$$

$$\leq \log \sum_{\mathbf{X}^C_{O,in}} \max_{\mathbf{D}^C} \sum_{\mathbf{X}^C_H} P(\mathbf{X}^C | \mathbf{X}^C_{O,out}) \cdot \exp\Big[\sum_{U_i \in \mathbf{U}^C} U_i + \sum_{(C',C) \in \mathcal{S}} V_{C'}(\mathbf{X}^C)\Big] \tag{22}$$

The first inequality in Eq. (21) is obtained by the Jensen's inequality, and the second inequality in Eq. (22) is the decomposition bounds for the MMAP task (Liu 2014; Ping, Liu, and Ihler 2015).

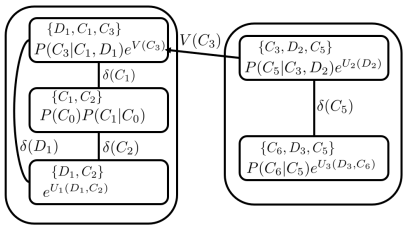


Figure 5. ST-GDD(i): decomposition bounds over $\mathcal{T}_{\text{ST}}$

We present briefly two new algorithms based on the variational decomposition bounds for MMAP: (1) submodel-tree decomposition with generalized dual decomposition (ST-GDD), and (2) submodel-tree decomposition with weighted mini-bucket with moment matching (ST-WMBMM). Both GDD and WMBMM algorithms provide state-of-the-art upper bounds for the MMAP task and they use the i-bound parameter for controlling the space and time complexity in an anytime manner. Next, we illustrate ST-GDD(i) by using an example in Figure 5.

Example 4. *For the 2-stage submodel-tree in Figure 4, $Eval(\mathcal{M}^C)$ bounds two submodels $\mathcal{M}'_1$ and $\mathcal{M}'_2$. The internal GDD message passing algorithm decomposes each submodel into a join-graph using some i-bound. For $\mathcal{M}'_2$, the internal message passing tightens the upper bounds, and sends an approximate message from the internal clusters by marginalizing out all the variables except for the interface variable C_3. The MEU can be computed by adding all the values from the internal clusters at the root submodel, $\mathcal{M}'_1$.*

Experiments

We compare the upper-bounds from the proposed algorithms ST-GDD(i) and ST-WMBMM(i) with the state-of-the-art methods in the following three experiments: (1) synthetic IDs with perfect recall comparing against the state-of-the-art decomposition bounds JGDID(i) (Lee, Ihler, and Dechter 2018) and WMBEID(i) (Lee et al. 2019), (2) upper bounds of the LIMIDs comparing against the error-bounds presented by (Mauá 2016), and (3) a case study that evaluates the upper bounds on large scale problems adopted from an online-planning domain.

Benchmarks Domains. The synthetic benchmark domains are generated as follows: (1) Factored FH-MDP instances are generated from two-stage factored MDP templates by varying the number of state and action variables, and the time horizon; (2) Factored FH-POMDP instances are generated similarly to FH-MDP instances, using factored POMDP templates; (3) Random influence diagram instances (RAND) are generated by generating DAGs in random for influence diagrams; (4) BN instances are existing Bayesian networks used in the UAI-2006 probabilistic inference competitions which we converted to IDs; (5) The LIMID benchmark converted the above four benchmark domains into LIMIDs by removing the temporal constraints; and (6) System administrator MDP/POMDP instances are generated by translat-

Domain	n	w_c	w_s	ST-GDD(i=1)	ST-GDD(i=5)	ST-WMB(i=10)	JGDID(i=1)	WMBEID(i=10)
ID-BN	84.6	30.2	21.8	0.19	0.15	**0.13**	0.33	0.74
IDBN14w57d12	115	57	42	103.89	96.24	95.37	1420	2.2E+4
FH-MDP	105.7	25.5	25.4	**0.06**	0.07	0.18	0.16	0.44
mdp9-32-3-8-3	99	43	43	18.92	19.71	25.31	23.09	111.81
FH-POMDP	55.9	28.1	28.1	0.31	0.22	**0.06**	0.56	0.72
pomdp8-14-9-3-12-14	96	47	46	73.53	76.37	67.18	5.E+08	5.E+09
RAND	56.2	20.5	17.9	**0.22**	0.24	0.24	0.23	0.46
rand-c70d21o1	84	32	34	1309.89	1791.93	1752.47	1743.6	2.E+04

Table 1. Quality of upper bounds on ID benchmarks. n is the number of variables, w_c is the constrained induced width by JGDID or WMBEID, w_s is the submodel induced width by ST-GDD or ST-WMB. Each domain shows the average of the statistics and the average gap $\frac{U-U_{\min}}{U}$, and the bounds from the hardest instance.

ing RDDL instances into 2-stage influence diagrams and unrolling them to the desired time-horizons (up to 10).

Results from the ID domains. Table 1 shows a summary of the experiments comparing the quality of the upper bounds for IDs. We can see that the submodel induced width w_s is lower than the constrained induced width w_c from a strong junction tree. The average gaps (the lower, the better) from all four benchmarks show that the proposed ST-GDD and ST-WMBMM generate higher quality upper bounds than others. From the results of ST-GDD(i=5) and ST-WMBMM(i=10), we can see that a simple single-pass internal message-passing algorithm can generate comparable upper bounds with higher i-bounds.

Results from the LIMID domains. Table 2 shows the results from the LIMIDs benchmark domain. To the best of our knowledge, there's no practical upper bounding scheme available for LIMIDs, so we compare against the theoretical bounds given in (Mauá and Cozman 2016). We can see that the upper bounds from ST-WMB with i=10 are orders of magnitudes tighter than the theoretical kpu-UB.

A Case Study on SysAdmin Domain. The SysAdmin domain (Guestrin et al. 2003) is one of the problems used in the international planning competitions. We compare the upper bounds from ST-WMB(i=20) and the expected value returned by the online MDP/POMDP planner (Cui and Khardon 2016, 2019). This is because other schemes JGDID, WMBEID, and kpu-UB could not generate any meaningful bounds at this scale. Although the values from the online planner and ST-WMB are not directly comparable as the online planner computes the Monte-Carlo estimate of the suboptimal online strategy while the ST-WMB computes the deterministic upper bounds of the optimal offline strategy, we can see that the upper bounds from the finite-horizon MDP and POMDP instances are close to the lower bounds, demonstrating the potential for applying our bounding schemes to planning under uncertainty.

Domain	n	w_s	ST-WMB(i=10)	kpu-UB
IDBN14w42d6	115	33	45.3	1.22E+8
IDBN14w57d12	115	44	93.73	4.70E+8
mdp9-32-3-8-3	99	43	25.33	1.68E+11
pomdp9-14-8-3-10-4	92	44	41.35	4.27E+8
rand-c50d15o1	84	27	1250	1.13E+6
rand-c70d7o1	91	21	658	4.38E+5

Table 2. Upper bounds on LIMID benchmarks. w_s is the submodel induced width of the LIMID instances, kpu-UB is the analytical bound (Mauá and Cozman 2016).

Instance	w_s	t=4 (OP)	t=4 (UB)	t=10 (OP)	t=10 (UB)
mdp6	68	110.51	130.63	239.00	328.25
mdp7	88	147.94	172.62	312.81	433.28
mdp8	92	146.62	174.22	300.06	439.45
mdp9	109	184.25	218.21	385.26	547.76
mdp10	113	183.65	217.52	364.57	549.87
pomdp6	360	110.55	155.24	229.13	420.17
pomdp7	480	147.50	195.35	304.58	526.01
pomdp8	480	147.90	207.96	298.83	568.49
pomdp9	701	182.78	244.42	372.75	666.88
pomdp10	300	184.93	253.04	375.90	707.23

Table 3. Upper bounds for SysAdmin domain. w_s is the submodel induced width, OP denotes the value from online planners, UB denotes the upper bounds from ST-WMB(20), and t is the number of stages.

Conclusion

This paper presents a graph-based decomposition scheme, submodel-tree decomposition for solving influence diagrams. For the perfect recall IDs, the submodel-tree decomposition identifies the minimal relevant sets for computing the MEU. For LIMIDs, it identifies the subset of decision variables that should be optimized jointly under the imperfect recall. Since exact algorithms are intractable, we also present a variational decomposition bounding scheme for the MEU task which is built on top of the proposed submodel-tree decomposition and decomposition bounds for MMAP inference, achieving tighter upper bounds compared with the state-of-the-art approaches. In the future work, the presented decomposition methods can be extended to more general influence diagrams such as multi-agent influence diagrams. Since submodel decomposition bounds provide a cost-to-go function over a submodel-tree, we can utilize the bounding scheme for the heuristic search.

Acknowledgments

We thank the reviewers for their valuable feedback. This work was supported in part by NSF grants IIS-2008516.

Broader Impact

In this paper, we present a new method for solving influence diagrams that can also apply to more general limited memory influence diagrams. The main contribution is two folds. First, we presented a graph-based method for reducing influence diagrams. The proposed method can also apply to a more general case than earlier works; decision making under the limited memory (decision-maker does not have access to the information about the history). Second, we also provide a simple yet efficient way to generate upper bounds of the optimal solution. For the potential impact of the current work, the proposed method can improve other approaches such as heuristic search of sampling methods for decision-making. More importantly, the ideas shown in the current work can provide a new perspective on sequential decision making by relaxing the restriction of underlying models from MDP or POMDP to more general structures. Concerning the limitation of our work, care must be taken when applying our work to a real-world application setting. Our proposed method provides upper bounds of the value of the optimal solution and suboptimal policies that may not achieve the optimal value. Besides, the input problem has to be provided in a structured format that requires knowledge engineering or machine learning.

References

Bodlaender, H. L. 1988. Dynamic programming on graphs with bounded treewidth. In *International Colloquium on Automata, Languages, and Programming*, 105–118. Springer.

Cui, H.; and Khardon, R. 2016. Online Symbolic Gradient-Based Optimization for Factored Action MDPs. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, 3075–3081.

Cui, H. J.; and Khardon, R. 2019. Sampling Networks and Aggregate Simulation for Online POMDP Planning. In *Advances in Neural Information Processing Systems*, 9218–9228.

Dechter, R. 2000. A New Perspective on Algorithms for Optimizing Policies under Uncertainty. In *Proceedings of the 5th Conference on Artificial Intelligence and Planning Systems*, 72–81.

Dechter, R. 2013. Reasoning with probabilistic and deterministic graphical models: Exact algorithms. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 7(3): 1–191.

Detwarasiti, A.; and Shachter, R. D. 2005. Influence diagrams for team decision analysis. *Decision Analysis* 2(4): 207–228.

Guestrin, C.; Koller, D.; Parr, R.; and Venkataraman, S. 2003. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research* 19: 399–468.

Howard, R. A.; and Matheson, J. E. 1981. Influence Diagrams. *Readings on the Principles and Applications of Decision Analysis* 721–762.

Jensen, F.; Jensen, F. V.; and Dittmer, S. L. 1994. From Influence Diagrams to Junction Trees. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 367–373.

Jensen, F. V.; and Gatti, E. 2010. Information enhancement for approximate representation of optimal strategies from influence diagrams. *on Probabilistic Graphical Models* 161.

Jensen, J. L. W. V. 1906. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Math.* 30: 175–193. doi:10.1007/BF02418571. URL https://doi.org/10.1007/BF02418571.

Kask, K.; Dechter, R.; Larrosa, J.; and Dechter, A. 2005. Unifying tree decompositions for reasoning in graphical models. *Artificial Intelligence* 166(1-2): 165–193.

Kohlas, J.; and Shenoy, P. P. 2000. Computation in valuation algebras. In *Handbook of defeasible reasoning and uncertainty management systems*, 5–39. Springer.

Lauritzen, S. L.; and Nilsson, D. 2001. Representing and Solving Decision Problems with Limited Information. *Management Science* 47(9): 1235–1251.

Lee, J.; Ihler, A.; and Dechter, R. 2018. Join Graph Decomposition Bounds for Influence Diagrams. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*, 1053–1062.

Lee, J.; Marinescu, R.; Ihler, A.; and Dechter, R. 2019. A Weighted Mini-Bucket Bound for Solving Influence Diagrams. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Liu, Q. 2014. *Reasoning and Decisions in Probabilistic Graphical Models–A Unified Framework*. University of California, Irvine.

Liu, Q.; and Ihler, A. 2011. Bounding the Partition Function using Hölder's Inequality. In *Proceedings of the International Conference on Machine Learning*, 849–856.

Liu, Q.; and Ihler, A. 2012. Belief Propagation for Structured Decision Making. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, 523–532.

Marinescu, R.; Dechter, R.; and Ihler, A. 2014. AND/OR Search for Marginal MAP. In *Proceeding of the International Conference on Uncertainty in Artificial Intelligence*, 563–572. Quebec City, Canada.

Marinescu, R.; Lee, J.; Dechter, R.; and Ihler, A. 2018. And/or Search for Marginal Map. *Journal of Artificial Intelligence Research* 63: 875–921.

Mateescu, R.; Kask, K.; Gogate, V.; and Dechter, R. 2010. Join-graph propagation algorithms. *Journal of Artificial Intelligence Research* 37: 279–328.

Mauá, D. D. 2016. Equivalences Between Maximum a Posteriori Inference in Bayesian Networks and Maximum Expected Utility Computation in Influence Diagrams. *Int. J. Approx. Reasoning* 68(C): 211–229.

Mauá, D. D.; and Cozman, F. G. 2016. Fast Local Search Methods for Solving Limited Memory Influence Diagrams. *Int. J. Approx. Reasoning* 68(C): 230–245.

Mauá, D. D.; de Campos, C. P.; and Zaffalon, M. 2012. Solving Limited Memory Influence Diagrams. *Journal of Artificial Intelligence Research* 44: 97–140.

Nielsen, T. D. 2001. Decomposition of influence diagrams. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, 144–155. Springer.

Nielsen, T. D.; and Jensen, F. V. 1999. Well-defined Decision Scenarios. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 502–511.

Nilsson, D.; and Hohle, M. 2001. Computing Bounds on Expected Utilties for Optimal Policies Based on Limited Information. *Dinar Research Report* .

Pearl, J. 2009. Causality. 78–85. Cambridge University Press.

Ping, W.; Liu, Q.; and Ihler, A. T. 2015. Decomposition Bounds for Marginal MAP. In *Proceedings of Advances in Neural Information Processing Systems 28*, 3267–3275.

Pralet, C.; Schiex, T.; and Verfaillie, G. 2006. From Influence Diagrams to Multi-operator Cluster DAGs. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 393–400.

Shenoy, P. P. 1997. Binary join trees for computing marginals in the Shenoy-Shafer architecture. *International Journal of approximate reasoning* 17(2-3): 239–263.

Shenoy, P. P.; and Shafer, G. 1990. Axioms for Probability and Belief-function propagations. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 169–198.

Wainwright, M. J.; and Jordan, M. I. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning* 1(1-2): 1–305.

Yuan, C.; Wu, X.; and Hansen, E. A. 2010. Solving Multistage Influence Diagrams Using Branch-and-bound Search. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 691–700.

Zhang, N. L.; and Poole, D. L. 1992. Stepwise-Decomposable Influence Diagrams. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 141–152.