Join Graph Decomposition Bounds for Influence Diagrams

Junkyu Lee **Alexander Ihler** **Rina Dechter**
Donald Bren School of Information and Computer Sciences
University of California, Irvine
{junkyul, ihler, dechter} @ics.uci.edu

Abstract

We introduce a new decomposition method for bounding the maximum expected utility of influence diagrams. While most current schemes use reductions to the Marginal Map task over a Bayesian Network, our approach is direct, aiming to avoid the large explosion in the model size that often results by such reductions. In this paper, we extend to influence diagrams the principles of decomposition methods that were applied earlier to probabilistic inference, utilizing an algebraic framework called valuation algebra which effectively captures both multiplicative and additive local structures present in influence diagrams. Empirical evaluation on four benchmarks demonstrates the effectiveness of our approach compared to reduction-based approaches and illustrates significant improvements in the upper bounds on maximum expected utility.

1 INTRODUCTION

An influence diagram (ID) [Howard and Matheson, 2005] is a graphical model for sequential decision-making under uncertainty that compactly captures the local structure of the conditional independence of probability functions and the additivity of utility functions. Its structure is captured by a directed acyclic graph (DAG) over nodes representing the variables (decision and chance variables). The standard query on an ID is finding the maximum expected utility (MEU) and an optimal policy, at each decision subject to the history of observations and decisions. Our focus is on computing an upper bound on the MEU in a single agent sequential decision-making scenario when we assume perfect recall. The computation of upper bounds is desirable not only because exact computation is exponential in the number of variables appearing in the history, but also because it can be used as a building block of algorithmic frameworks such as heuristic search and sampling.

Earlier work. One early work that yields bounds on many inference tasks in an anytime manner is the mini-bucket elimination (MBE) scheme that provides upper and lower bounds of graphical model queries by enforcing problem decomposition during the variable elimination process [Dechter and Rish, 2003]. In particular, Dechter [2000] presented an MBE algorithm for influence diagrams. A different principle for generating bounds on the MEU is to relax the constraints imposed on the information available at each stage and for each decision (thus making more observations visible to each decision). This *information relaxation scheme* relaxes the constraints imposed on the information available at each stage and it also permits variable reordering during processing [Nilsson and Hohle, 2001]. In particular, Yuan et al. [2010] presented an AND/OR depth-first branch and bound search algorithm guided by upper bounds generated by such flexible variable orderings.

An alternative set of schemes exploit translations between the maximum a posteriori inference (MMAP) in a Bayesian network (BN) and the MEU inference in an ID [Mauá, 2016]. The idea is to compute the upper bound of the MMAP of the BN translated from an input ID. However, the number of auxiliary variables introduced by the translation is exponential in the size of the history under the perfect recall assumption. If all utility functions were multiplicative, an ID could be treated as an unnormalized distribution and MMAP inference would be applied directly. Liu and Ihler [2012] presented a variational formulation for computing the MEU and message passing algorithms for such IDs where the additive utilities are converted into multiplicative utilities by introducing a latent selector variable. However, such a translation can make it difficult to exploit decompositions present in the additive utility functions.

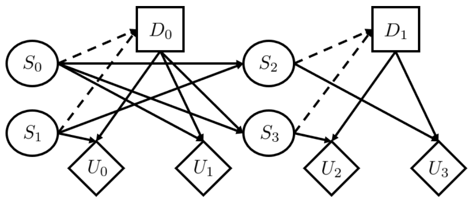
Contributions. We develop a decomposition scheme applied directly to IDs. It extends the decomposition scheme for MMAP [Ping et al., 2015] to IDs to accommodate multiplicative and additive terms. In particular, the upper bound is generated by relaxing the sequential structure of an ID to locally coupled decision subproblems. Subsequently, we present a message passing algorithm derived from the optimization framework that tightens the upper bound over various parameters including the reparameterization of both probability and utility functions. In summary:

- We present a new graphical model decomposition bound specialized for IDs called join graph decomposition bounds for IDs (JGDID), and a message passing algorithm that optimizes the bound.
- The proposed algorithm is compared empirically with current schemes on four benchmarks, showing a significant improvement in the quality of the upper bounds.

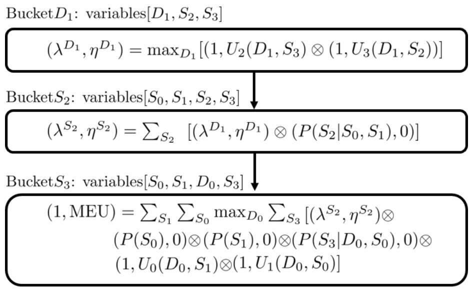
2 PRELIMINARIES

2.1 INFLUENCE DIAGRAMS

An ID is a tuple $\mathcal{M} := \langle \mathbf{C}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$ of a set of discrete random variables $\mathbf{C} = \{C_i | i \in \mathcal{I}_{\mathbf{C}}\}$, a set of discrete decision variables $\mathbf{D} = \{D_i | i \in \mathcal{I}_{\mathbf{D}}\}$, a set of conditional probability functions $\mathbf{P} = \{P_i | Pi \in \mathcal{I}_{\mathbf{P}}\}$, and a set of real-valued additive utility functions $\mathbf{U} = \{U_i | Ui \in \mathcal{I}_{\mathbf{U}}\}$. We use $\mathcal{I}_{\mathbf{S}} = \{0, 1, \cdots, |S| - 1\}$ to denote the set of indices of each element in a set $\mathbf{S}$, where $|S|$ is the cardinality of $\mathbf{S}$. As illustrated in Figure 1(a), an ID can be associated with a DAG of three types of nodes: the chance nodes $\mathbf{C}$ drawn as circles, the decision nodes $\mathbf{D}$ drawn as squares, and the value nodes $\mathbf{U}$ drawn as diamonds. There are also three types of directed edges: edges directed into a chance node C_i from its parents $pa(C_i) \subset \mathbf{C} \cup \mathbf{D}$ representing the conditional probability function $P_i(C_i | pa(C_i))$, edges directed into a value node U_i denoting the utility function $U_i(\mathbf{X}_i)$ from its scope $\mathbf{X}_i \subset \mathbf{C} \cup \mathbf{D}$, and informational arcs (dashed arrows in Figure 1(a)) directed from chance nodes to a decision node. The set of parent nodes associated with a decision node D_i is called the information set I_i, and denotes chance nodes that are assumed to be observed immediately before making decision D_i. The constrained variable ordering $\mathcal{O}$ obeys a partial ordering which alternates between information sets and decision variables $\{\mathbf{I}_0 \prec D_0 \prec \cdots \prec \mathbf{I}_{|\mathbf{D}|-1} \prec D_{|\mathbf{D}|-1} \prec \mathbf{I}_{|\mathbf{D}|}\}$. The partial elimination ordering should ensure the regularity of the ID (a decision can only be preceded by at most one decision), and dictates the available information at each decision D_i so that the *non-forgetting agent* makes decisions in multi-staged manner based on the history available at each stage



(a) Factored MDP as Influence Diagram



BucketD_1: variables$[D_1, S_2, S_3]$

$(\lambda^{D_1}, \eta^{D_1}) = \max_{D_1} [(1, U_2(D_1, S_3) \otimes (1, U_3(D_1, S_2))]$

BucketS_2: variables$[S_0, S_1, S_2, S_3]$

$(\lambda^{S_2}, \eta^{S_2}) = \sum_{S_2} \; [(\lambda^{D_1}, \eta^{D_1}) \otimes (P(S_2|S_0, S_1), 0)]$

BucketS_3: variables$[S_0, S_1, D_0, S_3]$

$(1, \mathrm{MEU}) = \sum_{S_1} \sum_{S_0} \max_{D_0} \sum_{S_3} [(\lambda^{S_2}, \eta^{S_2}) \otimes$
$(P(S_0), 0) \otimes (P(S_1), 0) \otimes (P(S_3|D_0, S_0), 0) \otimes$
$(1, U_0(D_0, S_1) \otimes (1, U_1(D_0, S_0)]$

(b) Computing Maximum Expected Utility

Figure 1: Influence Diagram Example. A 2-step factored MDP is represented by an influence diagram, and the lower figure shows a schematic trace of the variable elimination with the valuation algebra.

i, $H(D_i) := \cup_{k=0}^{i} \{D_k\} \cup \cup_{k=0}^{i} \mathbf{I}_i$. The standard task of solving Influence Diagrams is computing the maximum expected utility $E[\sum_{U_i \in \mathbf{U}} U_i | \mathbf{\Delta}]$ and finding a set of optimal policies $\mathbf{\Delta} = \{\Delta_i | \Delta_i : R(D_i) \mapsto D_i, \forall D_i \in \mathbf{D}\}$, where Δ_i is a deterministic decision rule for D_i and $R(D_i) \subseteq H(D_i)$ is the subset of history called *requisite information* to D_i, namely, the only relevant history for making a decision [Nielsen and Jensen, 1999].

2.2 COMPUTING EXPECTED UTILITY

Unlike probabilistic graphical models, Influence Diagrams hold two type of functions combined by different operators: a product of probability functions, and a summation of utility functions. Jensen et al. [1994] presented a variable elimination algorithm that avoids the complication of dealing with two types of functions by generalizing the combination and marginalization operators such that operators act on a pair of probability and utility functions called a potential. The *valuation algebra* is an algebraic framework for computing the expected utility values, or values for short, based on the combination and marginalization on potentials [Mauá et al., 2012]. Here, we briefly summarize the essence of valuation algebra since it is what we use for developing the decomposition scheme. Let a valuation $\Psi(\mathbf{X})$ be a pair of probability and value functions $(P(\mathbf{X}), V(\mathbf{X}))$ over a set of variables $\mathbf{X}$ called its scope. Occasionally, we will abuse the notation by dropping the scope from a function, e.g., writing $P_1(\mathbf{X}_1)$ as P_1. The combination and marginalization operators are defined as follows.

Definition 1. ***(combination of two valuations)*** *Given two valuations* $\Psi_1(\mathbf{X}_1){:=}(P_1(\mathbf{X}_1), V_1(\mathbf{X}_1))$ *and* $\Psi_2(\mathbf{X}_2){:=}(P_1(\mathbf{X}_2), V_1(\mathbf{X}_2))$*, the combination of the two valuations over* $\mathbf{X}_1 \cup \mathbf{X}_2$ *is defined by*

$$\Psi_1(\mathbf{X}_1) \otimes \Psi_2(\mathbf{X}_2) := (P_1 P_2, P_1 V_2 + P_2 V_1).$$

Following Definition 1, the identity is $(1, 0)$ and the inverse of $(P(\mathbf{X}), V(\mathbf{X}))$ is $(1/P(\mathbf{X}), -V(\mathbf{X})/P^2(\mathbf{X}))$.

Definition 2. ***(marginalization of a valuation)*** *Given a valuation* $\Psi(\mathbf{X}) := (P(\mathbf{X}), V(\mathbf{X}))$*, marginalizing over* $\mathbf{Y} \subseteq \mathbf{X}$ *by summation or maximization are defined by*

$$\sum_{\mathbf{Y}} \Psi(\mathbf{X}) := (\sum_{\mathbf{Y}} P(\mathbf{X}), \sum_{\mathbf{Y}} V(\mathbf{X})),$$
$$\max_{\mathbf{Y}} \Psi(\mathbf{X}) := (\max_{\mathbf{Y}} P(\mathbf{X}), \max_{\mathbf{Y}} V(\mathbf{X})).$$

An ID $\mathcal{M}$ can be compactly represented by the valuation algebra as $\mathcal{M}^{'} := \langle \mathbf{X}, \mathbf{\Psi}, \mathcal{O} \rangle$, where $\mathbf{X} = \mathbf{C} \cup \mathbf{D}$ and $\mathbf{\Psi} = \{(P_i, 0) | P_i \in \mathbf{P}\} \cup \{(1, U_i) | U_i \in \mathbf{U}\}$. The MEU can be written as

$$\sum_{\mathbf{I}_0} \max_{D_0} \cdots \sum_{\mathbf{I}_{|\mathbf{D}|-1}} \max_{D_{|\mathbf{D}|-1}} \sum_{\mathbf{I}_{|\mathbf{D}|}} \bigotimes_{\alpha \in \mathcal{I}_\Psi} \Psi_\alpha(\mathbf{X}_\alpha), \qquad (1)$$

where $\mathbf{X}_\alpha$ denotes the scope of Ψ_α.

The following example illustrates the variable elimination algorithm with the valuation algebra.

Example 1. *Figure 1(b) shows a schematic trace of the variable elimination algorithm [Dechter, 1999] using the valuation algebra. We use* $\mathcal{O} : \{D_1, S_2, S_3, D_0, S_0, S_1\}$ *as a legal elimination ordering. The first eliminated variable is* D_1*, so the variable elimination algorithm collects all valuations whose scope includes* D_1 *in Bucket* D_1*. Then it generates the outgoing message* $(\lambda^{D_1}, \eta^{D_1})$ *and sends it to Bucket* S_2*. Bucket* S_2 *combines the preallocated valuations and the incoming message, and generates the outgoing message* $(\lambda^{S_2}, \eta^{S_2})$*. This elimination process continues until we obtain the MEU. We refer to Mauá et al. [2012] for more details.*

2.3 JOIN GRAPH DECOMPOSITION

A probabilistic graphical model $\mathcal{G} := \langle \mathbf{X}, \mathbf{F} \rangle$ is a tuple of a set of discrete variables $\mathbf{X}$ and a set of non-negative real valued functions $\mathbf{F} = \{F_\alpha(\mathbf{X}_\alpha) | \alpha \in \mathcal{I}_\mathbf{F}\}$, where $\mathbf{X}_\alpha \subset \mathbf{X}$ is the scope of F_α. Graphical model inference tasks can be viewed as eliminating a set of variables from the joint function by either summation or maximization. The MMAP task computes the mode of the marginal of the joint function by

$$\max\nolimits_{\mathbf{X}_M} \sum\nolimits_{\mathbf{X}_S} \prod\nolimits_{\alpha \in \mathcal{I}_\mathbf{F}} F_\alpha(\mathbf{X}_\alpha), \qquad (2)$$

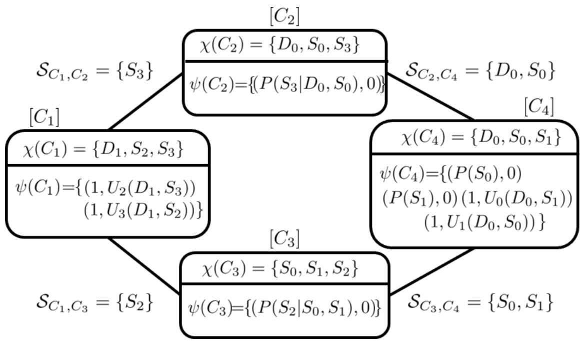

$[C_2]$
$\chi(C_2) = \{D_0, S_0, S_3\}$
$\psi(C_2)=\{(P(S_3|D_0, S_0), 0)\}$
$\mathcal{S}_{C_1,C_2} = \{S_3\}$
$\mathcal{S}_{C_2,C_4} = \{D_0, S_0\}$
$[C_1]$
$\chi(C_1) = \{D_1, S_2, S_3\}$
$\psi(C_1)=\{(1, U_2(D_1, S_3))$
$(1, U_3(D_1, S_2))\}$
$[C_4]$
$\chi(C_4) = \{D_0, S_0, S_1\}$
$\psi(C_4)=\{(P(S_0), 0)$
$(P(S_1), 0)\,(1, U_0(D_0, S_1))$
$(1, U_1(D_0, S_0))\}$
$[C_3]$
$\chi(C_3) = \{S_0, S_1, S_2\}$
$\psi(C_3)=\{(P(S_2|S_0, S_1), 0)\}$
$\mathcal{S}_{C_1,C_3} = \{S_2\}$
$\mathcal{S}_{C_3,C_4} = \{S_0, S_1\}$

Figure 2: Join Graph Decomposition Example. Figure shows a join graph of the influence diagram in Figure 1(a). The join graph is generated by limiting the maximum cluster size from 4 to 3. The scope from a node labeling function $\chi(C_i)$ is shown inside each node C_i and functions (valuations) are also allocated by $\psi(C_i)$. Separators $\mathcal{S}_{C_i,C_j}$ are shown next to edge (C_i, C_j).

where $\mathbf{X}_M$ denotes the maximization variables and $\mathbf{X}_S$ denotes the summation variables. The relevance relation between variables is captured by an undirected graph $\mathcal{G}_p = (V, E)$ called primal graph, where the set of nodes V represents variables, and an edge $e \in E$ connects two nodes if both variables associated with those nodes appear in the scope of some function. A tree decomposition of $\mathcal{G}_p$ produces a tree of clusters that captures the underlying structure of non-serial dynamic programming subject to the sequence of variable elimination operations. Namely, each cluster collects a set of functions that should be processed together. The worst-case space and time complexity of an inference query is exponential in the maximum cluster size called induced width w of $\mathcal{G}_p$.

Join graph decomposition [Mateescu et al., 2010] is an approximation scheme that further decomposes clusters in a join tree into a possibly loopy graph of finer grained clusters and, hence, it can control the complexity by limiting the maximum number of variables that are allowed to appear in cluster nodes.

Definition 3. ***(join graph decomposition)*** *A join graph decomposition of a graphical model* $\mathcal{G}$ *is a tuple* $\mathcal{D} := \langle \mathcal{G}_J, \chi, \psi \rangle$*, where* $\mathcal{G}_J = (\mathcal{C}, \mathcal{S})$ *is a graph with nodes* $\mathcal{C}$ *and edges* $\mathcal{S}$*, and* χ *and* ψ *are labeling function that* χ *maps a node* $C \in \mathcal{C}$ *to a set of variables* $\chi(C_i) = \mathbf{X}_C$*, and* ψ *allocates each function* F_α *exclusively to a node* $C \in \mathcal{C}$ *such that* $\mathbf{X}_\alpha \subseteq \mathbf{X}_C$*. An edge* $(C_i, C_j) \in \mathcal{S}$ *is associated with a subset of variables shared between the two clusters* $\chi(C_i) \cap \chi(C_j)$*, called separator* $\mathcal{S}_{C_i,C_j}$*. The labeling function should ensure the running intersection property; for each variable* $X_i \in \mathbf{X}$*, the set* $\{C \in \mathcal{C} | X_i \in \psi(C)\}$ *induces a connect subgraph.*

A valid join graph can be systematically structured from a mini bucket tree produced by the MBE algorithm with the bounding parameter i-bound that controls the maximum cluster size [Dechter and Rish, 2003]. The following example illustrates a join graph decomposition of the ID shown in Figure 1(a).

Example 2. *Figure 2 shows a join graph decomposition $\mathcal{D} : \langle \mathcal{G}_J, \chi, \psi \rangle$ of the ID in Figure 1(a). The primal $\mathcal{G}_p$ of an ID can be obtained by removing all informational arcs before moralization and then removing the value nodes. From the $\mathcal{G}_p$ and a legal variable elimination ordering compatible with the MEU query, a join graph $\mathcal{G}_J$ can be generated by standard methods. The labeling functions χ and ψ are displayed inside nodes and separators $\mathcal{S}_{C_i,C_j}$ are displayed next to edges. Compared with the join tree shown in Figure 1(b), we see that the additional cluster node C_2 contains a function $P(S_3|D_0, S_0)$ that was included in Bucket S_3 in the join tree.*

2.4 DECOMPOSITION BOUNDS

Decomposition methods for bounding graphical model inference queries are based on two techniques. Namely, the graphical model decomposition with some auxiliary parameters and the optimization procedure that optimizes the parameters to improve the bound. For example, dual decomposition for MAP optimizes the dual variables, corresponding to Lagrange multipliers enforcing a set of local consistency constraints defined on the factor graph [Sontag et al., 2011]. Various decomposition bounds are available in the literature depending on decomposition schemes, methods of parameterization, and optimization frameworks. The common characteristic of such decomposition bounds is that they decompose the original graphical model to a relaxed lower complexity model, compute the global bound from decomposed local bounds and optimize the bound by additional local computations.

We review the generalized dual decomposition (GDD) bound for MMAP [Ping et al., 2015] that our bounding scheme is built upon. First, define a powered-sum elimination operator $\sum_X^w$ by

$$\textstyle\sum_X^w f(X) = [\sum_X |f(X)|^{1/w}]^w, \tag{3}$$

which generalizes maximization and summation with a weight $0 \leqslant w \leqslant 1$ for the variable X. Note that the powered-sum elimination operator reduces to maximization and summation when $w{\to}0$ and $w{=}1$, respectively. Given a graphical model $\mathcal{G} : \langle \mathbf{X}, \mathbf{F} \rangle$, the MMAP task in Eq. (2) can be expressed by the powered-sum elimination operator as,

$$\textstyle\sum_{\mathbf{X}}^{\mathbf{w}} \prod_{\alpha \in \mathcal{I}_{\mathbf{F}}} F_\alpha(\mathbf{X}_\alpha), \tag{4}$$

where each weight $w_i \in \mathbf{w}$ of a variable $X_i \in \mathbf{X}$ is assigned 0 for the maximization variables and 1 for the summation variables. The bounding scheme of GDD is based on the generalization of the Hölder's inequality,

$$\textstyle\sum_{\mathbf{X}}^{\mathbf{w}} \prod_{\alpha \in \mathcal{I}_{\mathbf{F}}} F_\alpha(\mathbf{X}_\alpha) \leqslant \prod_{\alpha \in \mathcal{I}_{\mathbf{F}}} \sum_{\mathbf{X}_\alpha}^{\mathbf{w}^\alpha} F_\alpha(\mathbf{X}_\alpha), \tag{5}$$

where $\mathcal{I}_{\mathbf{F}}$ is the index set of functions $\mathbf{F}$, $\mathbf{w}$ is the set of weights that are either 0 or 1, $\mathbf{w}^\alpha$ is the set of non-negative weights distributed to $\mathbf{X}_\alpha$ such that $w_i = \sum_{\alpha \in \mathcal{I}_{\mathbf{F}}} w_i^\alpha$. Note that the upper bound on the right-hand side of Eq. (5) combines local MMAP values only computed from a subset of variables $\mathbf{X}_\alpha$. The upper bounds of the MMAP in Eq. (5) can be formulated as an optimization problem by introducing cost-shifting parameters defined over a join graph decomposition $\langle \mathcal{G}_J(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ by the following equation,

$$\prod_{C_i \in \mathcal{C}} \sum_{\mathbf{X}_{C_i}}^{\mathbf{w}^{C_i}} [\prod_{\alpha \in \psi(C_i)} F_\alpha(\mathbf{X}_{C_i})][\prod_{(C_i,C_j) \in \mathcal{S}} \delta_{C_i,C_j}(\mathcal{S}_{C_i,C_j})], \tag{6}$$

where each cluster node C_i in the join graph $\mathcal{G}_J$ collects a set of functions mapped by $\chi(C_i)$, and the cost-shifting parameters $\delta_{C_j,C_i}(\mathcal{S}_{C_i,C_j})$ and $\delta_{C_i,C_j}(\mathcal{S}_{C_i,C_j})$ are introduced on the separators $\mathcal{S}_{C_i,C_j} \in \mathcal{S}$ such that the costs on the both sides cancel each other. The complexity of computing the upper bound is bounded by the maximum number of variables appearing in the cluster nodes. The optimization framework takes Eq. (6) as an objective function with weights $\mathbf{w}^{C_i}$ and cost-shifting functions $\delta_{C_i,C_j}(\mathcal{S}_{C_i,C_j})$ as optimization parameters. Since the objective function is convex after taking $\log$ on Eq. (6), efficient optimization procedures are available for tightening the upper bound.

3 DECOMPOSITION BOUNDS FOR INFLUENCE DIAGRAMS

3.1 DECOMPOSING INFLUENCE DIAGRAMS

In this section, we develop a decomposition bound for IDs based on the valuation algebra. First, we generalize the powered-sum elimination operator in Eq. (3) to the valuation algebra.

Definition 4. ***(powered-sum elimination for a valuation)*** *The powered-sum elimination operator for a valuation $\Psi(X) := (P(X), V(X))$ is defined by*

$$\sum_X^{(w,A)} \Psi(X) {:=} (\sum_X^w P(X), \sum_X^w h(P(X), V(X), A)) \otimes (1, {-}A) \tag{7}$$

with the activation function h that adds an arbitrary utility constant A to the normalized expected utility value $\frac{V(X)}{P(X)}$ and clips negative expected utility value as

$$h(P(X),V(X),A) {=} \begin{cases} P(X)(\frac{V(X)}{P(X)} + A) & \textit{if } \frac{V(X)}{P(X)} + A > 0 \\ 0, & \textit{otherwise.} \end{cases} \tag{8}$$

Since the powered-sum elimination applies to the absolute values of a function, we introduce the activation function h so that the powered-sum elimination on the value component converges to the usual sum-elimination with a constant shift by A when weights $\mathbf{w}$ are close to 1 and the value $V(\mathbf{X})$ is negative. Namely, $[\sum_X^w h(P(X),V(X))] \rightarrow \sum_X V(x)+A$ when $w \rightarrow 1$ and $A+\min(\frac{V(X)}{P(X)}) \geqslant 0$.

Next, we define the comparison operator for the valuation algebra as a partial order as follows.

Definition 5. ***(comparison of two valuations)*** *Given two valuations $\Psi_1 := (P_1, V_1)$ and $\Psi_2 : (P_2, V_2)$, we define inequality $\Psi_1 \leqslant \Psi_2$ iff. $P_1 \leqslant P_2$ and $V_1 \leqslant V_2$.*

Equipped with the powered-sum elimination and comparison operator for the valuation algebra, we state the decomposition bounds for IDs as follows.

Theorem 1. ***(decomposition bounds for IDs)*** *Given an ID $\mathcal{M}' := \langle \mathbf{X}, \mathbf{\Psi}, \mathcal{O} \rangle$, the MEU can be bounded by*

$$\sum_{\mathcal{O}}^{\mathbf{w}} \otimes_{\alpha \in \mathcal{I}_{\mathbf{\Psi}}} \Psi_\alpha(\mathbf{X}_\alpha) \leqslant \otimes_{\alpha \in \mathcal{I}_{\mathbf{\Psi}}} \sum_{\mathcal{O}}^{(\mathbf{w}^\alpha,\, \mathbf{A})} \Psi_\alpha(\mathbf{X}_\alpha). \tag{9}$$

The left-hand side is the MEU in Eq. (1), by rewriting the sequence of elimination operators as the powered-sum elimination operators $\sum_{\mathbf{I}_0}^{\mathbf{w}^{I_0}} \sum_{D_0}^{w_{D_0}} \cdots \sum_{\mathbf{I}_{|\mathbf{D}|}}^{\mathbf{w}^{\mathbf{I}_{|\mathbf{D}|}}}$ following the partial ordering $\mathcal{O}$, where the weights $\mathbf{w}^{I_k}$ are 1 for the summation variables, and $\mathbf{w}^{D_k}$ are 0 for the maximization variables. The right-hand side switches the order of the elimination and combination operators, hence it combines fully eliminated local valuations to the global valuation with weights $\mathbf{w}^\alpha$ that are distributed from $\mathbf{w}$ such that $w_i = \sum_{\alpha \in \mathcal{I}_\Psi} w_i^\alpha$, where w_i is the weight of $X_i \in \mathbf{X}$ and w_i^α is the weight of X_i at Ψ_α.

Proof. The decomposition bound can be obtained by applying Minkowski's inequality in Eq. (10) and Hölder's inequality in Eq. (11) to the probability and value functions of the valuations.

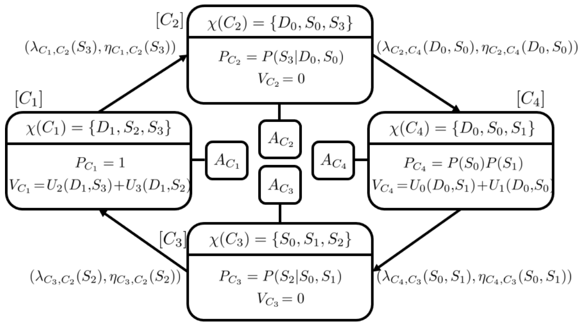
$$\textstyle\sum_X^w f(X)+g(X) \leqslant \sum_X^w f(x)+\sum_X^w g(X) \tag{10}$$

$$\textstyle\sum_X^w f(X)g(X) \leqslant \sum_X^{w_1} f(x) \sum_X^{w-w_1} g(X) \tag{11}$$

The probability component in the left-hand side of Eq. (9) can be bounded by applying Hölder's inequality as shown in Eq. (12).

$$\textstyle\sum_{\mathcal{O}}^{\mathbf{w}} \prod_{i \in \mathcal{I}_{\mathbf{\Psi}}} P_i \leqslant \prod_{i \in \mathcal{I}_{\mathbf{\Psi}}} \sum_{\mathcal{O}}^{\mathbf{w}^i} P_i. \tag{12}$$

The value component can be bounded by the following steps. We rewrite the MEU by introducing constant utilities A_i as shown in Eq. (13), and bound the MEU by the activation function h_i defined in Eq. (8) as shown in Eq. (14). The non-constant term in Eq. (14) can be further bounded by applying Minkowski's inequality and



$[C_1]$
$\chi(C_1)=\{D_1,S_2,S_3\}$
$P_{C_1}=1$
$V_{C_1}=U_2(D_1,S_3)+U_3(D_1,S_2)$
$[C_2]$
$\chi(C_2)=\{D_0,S_0,S_3\}$
$P_{C_2}=P(S_3|D_0,S_0)$
$V_{C_2}=0$
$[C_3]$
$\chi(C_3)=\{S_0,S_1,S_2\}$
$P_{C_3}=P(S_2|S_0,S_1)$
$V_{C_3}=0$
$[C_4]$
$\chi(C_4)=\{D_0,S_0,S_1\}$
$P_{C_4}=P(S_0)P(S_1)$
$V_{C_4}=U_0(D_0,S_1)+U_1(D_0,S_0)$
A_{C_1}
A_{C_2}
A_{C_3}
A_{C_4}
$(\lambda_{C_1,C_2}(S_3),\eta_{C_1,C_2}(S_3))$
$(\lambda_{C_2,C_4}(D_0,S_0),\eta_{C_2,C_4}(D_0,S_0))$
$(\lambda_{C_3,C_2}(S_2),\eta_{C_3,C_2}(S_2))$
$(\lambda_{C_4,C_3}(S_0,S_1),\eta_{C_4,C_3}(S_0,S_1))$

Figure 3: Optimization Parameters for Join Graph Decomposition Bounds. Figure shows the optimization parameters introduced in Proposition 1. The cost-shifting valuations are displayed next to the separators $\mathcal{S}_{C_i,C_j}$ as $(\lambda_{C_i,C_j}, \eta_{C_i,C_j})$, and utility constants A_{C_i} are attached next to each cluster node C_i.

Hölder's inequality as shown in Eq. (15) and (16), respectively.

$$\textstyle\sum_{\mathcal{O}}^{\mathbf{w}} \sum_{i \in \mathcal{I}_{\mathbf{\Psi}}} [V_i + P_i(A_i - A_i)][\prod_{j \neq i} P_j] \tag{13}$$

$$\leqslant \sum_{\mathcal{O}}^{\mathbf{w}} \sum_{i \in \mathcal{I}_{\mathbf{\Psi}}} h_i(P_i, V_i, A_i)[\prod_{j \neq i} P_j] - \sum_{i \in \mathcal{I}_{\mathbf{\Psi}}} A_i \tag{14}$$

$$\leqslant \sum_{i \in \mathcal{I}_{\mathbf{\Psi}}} [\sum_{\mathcal{O}}^{\mathbf{w}} h_i(P_i, V_i, A_i)][\prod_{j \neq i} P_j] - \sum_{i \in \mathcal{I}_{\mathbf{\Psi}}} A_i \tag{15}$$

$$\leqslant \sum_{i \in \mathcal{I}_{\mathbf{\Psi}}} [\sum_{\mathcal{O}}^{\mathbf{w}^i} h_i(P_i, V_i, A_i)][\prod_{j \neq i} \sum_{\mathcal{O}}^{\mathbf{w}^j} P_j] - \sum_{i \in \mathcal{I}_{\mathbf{\Psi}}} A_i \tag{16}$$

Note that the weights $\mathbf{w}^i$ assigned to each valuation Ψ_i in Eq. (12) and Eq. (16) are the same. The final result can be obtained by combining the upper bound on the probability component in Eq. (12) and the upper bound on the value component in Eq. (16) as a valuation with the powered-sum elimination operator for a valuation. □

3.2 OPTIMIZING THE UPPER BOUNDS

The following Proposition 1 presents the parameterized decomposition bounds based on the Theorem 1 by introducing optimization parameters relative to a join graph decomposition. Then, the partial derivatives of the parameterized decomposition bounds are derived to be used in the first order optimization framework.

Proposition 1. ***(parameterized decomposition bounds for IDs)*** *Given an ID $\mathcal{M}' := \langle \mathbf{X}, \mathbf{\Psi}, \mathcal{O} \rangle$ and its join graph decomposition $\mathcal{D} := \langle \mathcal{G}_J(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$, the decomposition bounds in Theorem 1 can be parameterized relative to a join graph decomposition $\mathcal{G}_J$ as a pair of upper bounds on the probability component and the value component (L_{MMAP}, L_{MEU}) as follows.*

$$L_{MMAP} = \textstyle\prod_{C_i \in \mathcal{C}} \sum_{\mathcal{O}}^{\mathbf{w}^{C_i}} P'_{C_i}, \tag{17}$$

$$L_{MEU} = \sum_{C_i \in \mathcal{C}} [\sum_{\mathcal{O}}^{\mathbf{w}^{C_i}} h_{C_i}(P'_{C_i}, V'_{C_i})][\prod_{C_j \neq C_i} \sum_{\mathcal{O}}^{\mathbf{w}^{C_j}} P'_{C_j}] - A_{C_i}. \tag{18}$$

In Proposition 1, the P'_{C_i} and V'_{C_i} are probability and value functions after incorporating cost-shifting parameters that can be written as

$$P'_{C_i} = P_{C_i} \prod_{(C_i,C_j)\in\mathcal{S}} \lambda_{C_i,C_j}, \tag{19}$$

$$V'_{C_i} = P'_{C_i}[\tfrac{V_{C_i}}{P_{C_i}} + \sum_{(C_i,C_j)\in\mathcal{S}} \tfrac{\eta_{C_i,C_j}}{\lambda_{C_i,C_j}}]. \tag{20}$$

Each node $C_i \in \mathcal{C}$ of the $\mathcal{G}_J$ collects the probability and value functions by the labeling function ψ, and each edge $(C_i, C_j)\in\mathcal{S}$ introduces a cost-shifting parameters $\delta_{C_i,C_j}=(\lambda_{C_i,C_j}, \eta_{C_i,C_j})$ over the variables $\mathcal{S}_{C_i,C_j}$ such that $\delta_{C_i,C_j} \otimes \delta_{C_j,C_i}=(1, 0)$. The utility constant parameters A_{C_i} is introduced through the activation function h_{C_i}, and the weight parameters $\mathbf{w}^{C_i}$ are distributed from $\mathbf{w}$ such that $w_X = \sum_{C_i\in\mathcal{C}} w_X^{C_i}$ for all $X \in \chi(C_i)$. Note that the reparameterized decomposition bound for IDs subsumes the GDD bound for MMAP in Eq. (6) at the probability component, L_{MMAP}, and the new parameterized upper bound for the MEU at the value component, L_{MEU}. Note that the L_{MEU} in Eq. (20) is non-convex.

The following example illustrates the optimization parameters for the ID shown in Figure 1(a).

Example 3. *Figure 3 illustrates the optimization parameters introduced by the join graph decomposition $\mathcal{D} := \langle \mathcal{G}_J(\mathcal{C}, \mathcal{S}), \chi, \psi\rangle$ of Example 2. The valuations at each node $C_i \in \mathcal{C}$ is displayed inside each node and the utility constant A_{C_i} is attached next to the node. The cost-shifting valuation $(\lambda_{C_i,C_j}, \eta_{C_i,C_j})$ is shown next to the directed edges from C_i to C_j implying that the cost is moving from C_i to C_j, and hence δ_{C_i,C_j} and δ_{C_j,C_i} cancel each other.*

Next, we present the first-order optimization procedures for tightening the parameterized decomposition bounds, L_{MEU} in Eq. (18). For efficiency and simplicity, we apply a block coordinate method that cycles through a subset of optimization parameters associated with the nodes and edges of the join graph $\mathcal{G}_J$, which we call the inner optimization problems. To optimize cost-shifting parameters $\{\delta_{C_i,C_j} | (C_i, C_j) \in \mathcal{S}\}$ and utility constants $\{A_{C_i} | C_i \in \mathcal{C}\}$, we use gradient descent with line search [Wright and Nocedal, 1999] by

$$\mathbf{x}^{t+1} = \mathbf{x}^t - s \cdot [\nabla f(\mathbf{x}^t)], \tag{21}$$

where f is the objective function, s is the step size, $\mathbf{x}^t$ is the optimization parameter at the t-th iteration. The weights $\{\mathbf{w}^{C_i} | C_i \in \mathcal{C}\}$ are updated by exponentiated gradient descent [Kivinen and Warmuth, 1997] by

$$\mathbf{w}^{C_i,t+1} = \frac{\mathbf{w}^{C_i,t}\exp[-s\cdot[\nabla_{\mathbf{w}^{C_i}} L_{\text{MEU}}(\mathbf{w}^{C_i,t})]]}{\sum_{C_i\in\mathcal{C}} \mathbf{w}^{C_i,t}\exp[-s\cdot[\nabla_{\mathbf{w}^{C_i}} L_{\text{MEU}}(\mathbf{w}^{C_i,t})]]}. \tag{22}$$

Now, we define pseudo marginals and some useful expressions before deriving the gradients of each subset of the optimization parameters.

Definition 6. ***(pseudo marginals)*** *Given a non-negative real-valued function $Z_0(\mathbf{X}_{1:n})$ over a set of variables $\mathbf{X}_{1:n}=\{X_1, \cdots, X_n\}$, and the weights $\mathbf{w} = \{w_1, \cdots, w_n\}$ associated with $\mathbf{X}_{1:n}$, we define a partial powered-sum elimination of variables $\mathbf{X}_{1:i}$ from $Z_0(\mathbf{X}_{1:n})$ by*

$$Z_i(\mathbf{X}_{i+1:n}) = \sum\nolimits_{x_i}^{w_i} Z_{i-1}(\mathbf{X}_{i:n}).$$

The pseudo marginal of $Z_0(\mathbf{X}_{1:n})$ is defined by

$$\Lambda(Z_0(\mathbf{X}_{1:n})) = [\tfrac{Z_{n-1}(X_n)}{Z_n}]^{1/w_n} \cdots [\tfrac{Z_0(\mathbf{X}_{1:n})}{Z_1(\mathbf{X}_{2:n})}]^{1/w_1}.$$

Note that $\Lambda(Z_0(\mathbf{X}_{1:n}))$ is a normalized distribution over $\mathbf{X}_{1:n}$, and each $[\frac{Z_{i-1}(\mathbf{X}_{i:n})}{Z_i(\mathbf{X}_{i+1:n})}]^{1/w_i}$ is a conditional distribution over X_i given $\mathbf{X}_{i+1:n}$.

Let's define a selector $F_{C_j|C_i}$ that selects a probability or a value component depending on the indices i and j by

$$F_{C_j|C_i} = \begin{cases} h_{C_j}(P^{C_j}, V^{C_j}), & \text{if } j = i \\ P^{C_j}, & \text{otherwise.} \end{cases} \tag{23}$$

By using Eq. (23), the upper bound of the expected utility value at C_i can be expressed by

$$\Theta_{C_i} = \prod_{C_j\in\mathcal{C}} \sum\nolimits_{\mathcal{O}}^{\mathbf{w}^{C_j}} F_{C_j|C_i}. \tag{24}$$

In the following, we summarize the gradients of L_{MEU}, ∇L_{MEU} with respect to subsets of parameters.

$$\nabla L_{\text{MEU}}(w_k^{C_i}) = \sum_{C_j\in\mathcal{C}} \rho_{C_j} H(X_k | \mathbf{X}_{i+1:|\mathcal{O}|}; F_{C_j|C_i}) \tag{25}$$

$$\nabla L_{\text{MEU}}(A_{C_i}) = \Theta_{C_i} \sum_{\mathbf{x}_{C_i}\setminus \mathcal{S}_{C_i,C_j}} \frac{P^{C_i}\Lambda(h_{C_i}(P^{C_i}, V^{C_i}))}{h_{C_i}(P^{C_i}, V^{C_i})} - 1 \tag{26}$$

$$\nabla L_{\text{MEU}}(\lambda_{C_i,C_j}) = \sum_{C_k\in\mathcal{C}} \Theta_{C_k}[\sum_{\mathbf{x}_{C_j}\setminus \mathcal{S}_{C_i,C_j}} \Lambda(F_{C_i|C_k}) - \sum_{\mathbf{x}_{C_i}\setminus \mathcal{S}_{C_i,C_j}} \Lambda(F_{C_j|C_k})] \tag{27}$$

$$\nabla L_{\text{MEU}}(\eta_{C_i,C_j}) = \Theta_{C_j} \sum_{\mathbf{x}_{C_j}\setminus \mathcal{S}_{C_i,C_j}} \frac{P^{C_j}\Lambda(h_{C_j}(P^{C_i}, V^{C_j}))}{h_{C_j}(P^{C_i}, V^{C_j})} - \Theta_{C_i} \sum_{\mathbf{x}_{C_i}\setminus \mathcal{S}_{C_i,C_j}} \frac{P^{C_i}\Lambda(h_{C_i}(P^{C_i}, V^{C_i})}{h_{C_i}(V^{C_i})} \tag{28}$$

The term $H(X_k | \mathbf{X}_{i+1:|\mathcal{O}|}; F_{C_j|C_i})$ in Eq. (25) is the conditional entropy $H(X_k | \mathbf{X}_{i+1:|\mathcal{O}|})$ of the pseudo marginal of the function selected by $F_{C_j|C_i}$.

3.3 MESSAGE PASSING ALGORITHM

Applying the parameterized decomposition bounds for IDs and the first order optimization procedures described in Section 3.2, we develop an iterative message passing algorithm that updates the optimization parameters defined relative to a join graph decomposition.

Algorithm 1 Join Graph Decomposition for IDs (JGDID)

```
Require: Influence diagram $\mathcal{M}' = \langle \mathbf{X}, \mathbf{\Psi}, \mathcal{O} \rangle$, initial weights $w_i$ associated with a variable $X_i \in \mathbf{X}$, $i$-bound, total iteration limit $M_1$, iteration limit $M_2$ for updating weights and costs before updating utility constants.
Ensure: an upper bound of the MEU, $L_{\text{MEU}}$,
 1: generate a join graph decomposition $\mathcal{D} = \langle \mathcal{G}_J(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ by MBE with $i$-bound and assign valuations to nodes by labeling function $\psi$.
 2: execute single pass cost-shifting by messages generated by MBE algorithm based on the valuation algebra (MBE-VA)
 3: initialize weights $\mathbf{w}^{C_i}, \forall C_i \in \mathcal{C}$ by uniform weights.
 4: $iter$=0, $L_{\text{MEU}} = \inf$
 5: while $iter < M_1$ or $L_{\text{MEU}}$ is not converged do
 6:   for each variable $X_i \in \mathbf{X}$ do
 7:     $L_{\text{MEU}} \leftarrow \min(L_{\text{MEU}}$, UPDATE-WEIGHTS$(\mathcal{G}_J, X_i))$
 8:   end for
 9:   for each edge $(C_i, C_j) \in \mathcal{S}$ do
10:     $L_{\text{MEU}} \leftarrow \min(L_{\text{MEU}}$, UPDATE-COSTS$(\mathcal{G}_J, \{C_i, C_j\}))$
11:   end for
12:   if $iter > M_2$ then
13:     for each node $C_i \in \mathcal{C}$ do
14:       $L_{\text{MEU}} \leftarrow \min(L_{\text{MEU}}$, UPDATE-UTIL-CONST$(\mathcal{G}_J, C_i))$
15:     end for
16:   end if
17:   $iter = iter + 1$
18: end while
```

Algorithm 1 outlines the procedure for updating the parameters, the Join Graph Decomposition Bound for IDs (JGDID). Given an input ID $\mathcal{M}' := \langle \mathbf{X}, \mathbf{\Psi}, \mathcal{O} \rangle$, we first generate a join graph decomposition $\mathcal{D} = \langle \mathcal{G}_J(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ by executing the MBE algorithm with an input i-bound. For details on how to structure a join graph decomposition, see Mateescu et al. [2010]. Then, we assign valuations to the nodes in $\mathcal{C}$ by labeling function ψ and run a single pass cost-shifting over the join graph using the messages generated by the MBE algorithm [Dechter and Rish, 2003] based on valuation algebra (MBE-VA). In our empirical evaluation, this preliminary step significantly improves the speed of convergence and the quality of the upper bound. The initial weights $\mathbf{w}^{C_i}$ at each node $C_i \in \mathcal{C}$ for the summation variables are uniform, and a small constant $\epsilon \approx 10^{-6}$ initializes the weights for the maximization variables.

The block coordinate method updates the subset of the parameters by solving inner optimization problems following the structure of $\mathcal{G}_J$. The `UPDATE-WEIGHTS` routine in line 7 updates the weights $w_i^{C_j}$ for a variable X_i over $\forall C_j \in \mathcal{C}$ by Eq. (22) with the gradient in Eq. (25), the `UPDATE-COSTS` routine in line 10 updates cost-shifting valuations δ_{C_i,C_j} over each edge $(C_i, C_j) \in \mathcal{S}$ by the gradient descent with the gradients in Eq. (27) and in Eq. (28), and the `UPDATE-UTIL-CONST` routine in line 14 updates the utility constants A_{C_i} for each node $C_i \in \mathcal{C}$ by the gradient descent with the gradient in Eq. (26). Since the optimization objective function L_{MEU} is non-convex, the block coordinate method with our first-order optimization procedures is not guaranteed to provide a globally minimum bound, yet often performs well in practice.

Table 1: Benchmark statistics. Table shows the minimum, median, and maximum instance statistics from 10 instances. n is the number of chance and decision variables, f is the number of probability and utility functions, k is the maximum domain size, s is the maximum scope size, and w is the induced width.

Domain	n	f	k	s	w
FH-MDP	25, 110, 170	30, 143, 170	2, 3, 5	4, 7, 9	5, 25, 43
FH-POMDP	15, 51, 96	18, 61, 140	2, 2, 3	3, 5, 9	10, 27, 47
RAND	22, 57, 91	22, 57, 91	2, 2, 2	3, 3, 3	6, 18, 41
BN	54, 100, 115	54, 100, 115	2, 2, 2	6, 8, 10	18, 28, 45

In our empirical evaluation, we set the hyperparameters for the number of gradient updates for the inner optimization to 10, and the number of updates M_2 for the weights and costs before updating utility constants to 20 and 50, which yielded a good convergence behavior.

4 EXPERIMENTS

We empirically compare the performance of our proposed algorithm JGDID with earlier approaches for bounding the MEU on four problem domains.

Benchmarks. The benchmarks are generated as follows. (1) Factored FH-MDP instances are generated from two stage factored MDP templates by controlling the number of state and action variables, the scope of functions, and the time horizon. (2) Factored FH-POMDP instances are generated similarly to FH-MDP instances, but we incorporate partial observability. (3) Random influence diagram (RAND) instances are generated by randomly generating influence diagram topology given the number of chance, decision, and value nodes. (4) BN instances are converted to ID from existing Bayesian networks used in the UAI-2006 probabilistic inference competitions by adding utility functions and randomly selecting decision variables. We generated 10 instances for each benchmark with increasing difficulty; Table 1 summarizes the instance statistics of the 4 benchmarks.

Algorithms. The first approach we compare against is the upper bounding algorithms for MMAP using the reduction from ID to MMAP. The reduction of Mauá [2016] generates standard MMAP instances, while Liu and Ihler [2012] generates MMAP instances with interleaved max and sum operators, which we call mixed MMAP. For the standard MMAP instances, we applied weighted mini-bucket with moment matching (WMBMM) [Liu and Ihler, 2011; Marinescu et al., 2014], and for the mixed MMAP instances, we applied GDD [Ping et al., 2015].

The second set of algorithms are applied directly to the In-

Table 2: Instance statistics of MMAP translation. Table shows the minimum, median, and maximum instance statistics of the standard MMAP reduction (MM) and the mixed MMAP reduction (MI).

Domain	Trans	n	k	w
FH-MDP	ID	25, 110, 170	2, 3 ,5	5, 25, 43
	MI	26, 111, 171	10, 27, 80	15, 86, 160
FH-POMDP	ID	15, 51, 96	2, 3, 2	10, 27, 47
	MM	28, 188, 5277	6, 16, 48	14, 141, 5192
	MI	16, 52, 97	6, 16 ,48	10, 28, 48
RAND	ID	22, 57, 91	2, 2, 2	6, 18, 41
	MM	29, 79, 142	2, 8, 21	8, 25, 58
	MI	23, 58, 92	2, 8, 21	6, 20, 42
BN	ID	54, 100, 115	2, 2, 2	18, 28, 45
	MM	69, 126, 202	3, 6, 12	20, 40, 92
	MI	55, 101, 116	3, 6, 12	19, 29, 46

fluence Diagram. One scheme is based on the mini-bucket idea which bounds the induced width by the i-bound, and then applies variable elimination using valuation algebra, yielding algorithm MBE-VA. The second is an information relaxation scheme, which relaxes the constrained variable ordering, denoted IR-SIS. [Nilsson and Hohle, 2001; Yuan et al., 2010]. The information relaxation is orthogonal to approximate elimination, and so both can be applied together; when we apply MBE-VA and JGDD together with the relaxed ordering of IR-SIS we call the hybrid algorithms MBE-VA+IR-SIS ad JGDID-IR-SIS.

In summary, we evaluated 6 algorithms. We have JGDID and MBE-VA applied directly to the input IDs assuming only constrained ordering. We have JGDID+IR-SIS and MBE-VA+IR-SIS applied to the IDs but allowing relaxed ordering by IR-SIS, and finally we have WMBMM and GDD that are applied to MMAP reductions from IDs.

Performance measure. We report the quality of upper bounds for individual instances, and the average quality of the upper bounds by the mean of the ratio between the best upper bound found by all configurations (6 algorithms with i-bounds 1 and 15) and the upper bound of each under comparison; the closer the value to 1.0, the better the quality.

4.1 COMPARING AGAINST MMAP TRANSLATIONS

Next, we compare our JGDID approach with WMBMM and GDD bounds based on MMAP translations.

Impact of MMAP translation. Table 2 summarizes the changes in the number of variables, the maximum domain size, and the induced width w due to the translation. When computing the induced width, we used the randomized min-fill algorithm. The reduction from ID to MMAP for the FH-MDP benchmark is not shown in the table because the translation was not feasible for most of the instances. Note that the standard MMAP reduction (MM) inflates all input statistics. The number of variables is exponential in the size of the largest information set, the maximum domain size is increased to the total number of utility functions, and the induced width also increased significantly higher than input IDs. The mixed MMAP translation (MI) increases the number of variables by 1 which has domain size equal to the total number of utility functions. The induced width is increased by 1 except for the FH-MDP domain.

Table 3: Average Quality of Upper Bounds.

Algorithm	FH-MDP	FH-POMDP	RAND	BN
JGDID+IR-SIS(i=1)	NA	0.88	0.87	0.99
JGDID+IR-SIS(i=15)	NA	0.76	0.85	0.64
JGDID(i=1)	0.88	0.38	0.86	0.89
JGDID(i=15)	0.49	0.38	0.85	0.64
MBE-VA+IR-SIS(i=1)	NA	0.03	0.01	0.00
MBE-VA+IR-SIS(i=15)	NA	0.54	0.46	0.15
MBE-VA(i=1)	0.00	0.00	0.00	0.00
MBE-VA(i=15)	0.40	0.29	0.46	0.17
GDD(i=1)	0.87	0.03	0.11	0.24
GDD(i=15)	0.22	0.11	0.15	0.05
WMBMM(i=1)	0.00	0.00	0.01	0.01
WMBMM(i=15)	0.01	0.23	0.35	0.24

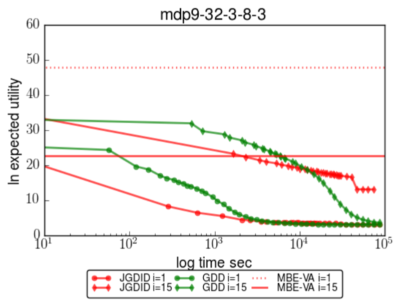
Upper bounds from individual instances. Figure 4 illustrates the quality of the obtained upper bounds for instances having the largest induced width in each benchmark. We ran JGDID and GDD algorithms up to 2000 iterations or until convergence. We can see from Figure 4 that JGDID dominates GDD and WMBMM on all instances except `pomdp8`. Comparing the upper bounds across i-bounds, JGDID and GDD do not show notable improvement on the speed of convergence with higher i-bounds due to the large overhead of a single iteration. We see that JGDID shows the step-wise improvement of upper bounds when it optimizes the utility constants.

Average quality of upper bounds. Table 3 shows the average quality of upper bounds. We see that the average quality of JGDID dominates both GDD and WMBMM in all benchmarks. Comparing JGDID and GDD, both generated upper bounds with similar quality on average in the FH-MDP benchmark. However, bounds from GDD are significantly worse than JGDID in other benchmarks. The upper bounds from WMBMM(i=1) is so large that the average ratios for all instances are close to 0.0.

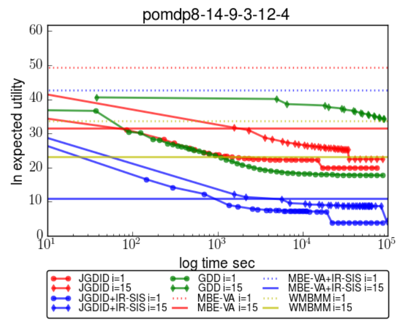
4.2 COMPARING AGANIST DIRECT ALGORITHMS

In this section we compare JGDID, MBE-VA, JGDID+IR-SIS, and MBE-VA+IR-SIS that are applied directly to IDs. They are all obtained by relaxing the input IDs either by decomposing graphical model or information relaxation.

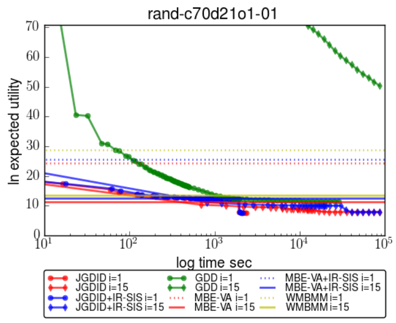
Impact of IR-SIS. The IR-SIS relaxation often produces



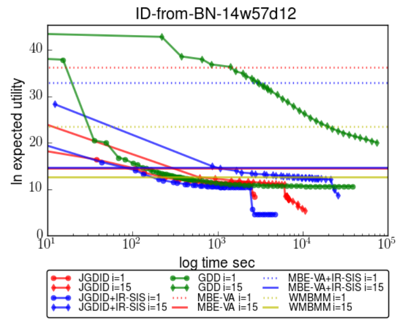
(a) n=99, f=120, w_c=43 w_{mi}=81



(b) n=94, f=140, w_c=47, w_{mm}=2057, w_{mi}=48, w_r=26



(c) n=91, f=91, w_c=41, w_{mm}=58, w_{mi}=42, w_r=42



(d) n=115, f=115, w_c=45, w_{mm}=92, w_{mi}=46, w_r=54

Figure 4: Upper Bounds of the MEU. Each plot shows the convergence of upper bounds over time. We draw horizontal lines for the bounds computed by non-iterative algorithms. The w_c, w_{mm}, w_{mi}, and w_r are the induced width of the input ID, standard MMAP reduction, mixed MMAP reduction, and relaxed ID with IR-SIS, respectively.

IDs with a lower induced width by reordering the variables in the information sets. It is shown to be very effective for FH-POMDP instances because IR-SIS transforms POMDP instances to MDP. The minimum, the median and the maximum induced width of FH-POMDP instances decreased from 10 to 3, 27 to 14, and 47 to 35, respectively via this relaxation. In other benchmarks, the improvement was negligible and often IR-SIS increased induced width.

Upper bounds from individual instances. Figure 4 illustrates also upper bounds obtained by direct algorithms. Comparing JGDID with MBE-VA, we see in the figure that JGDID improved the quality of the upper bound significantly in all instances. IR-SIS often significantly improves the quality of the bounds when the upper bounds from decomposition methods is still weak. In case of `pomdp8`, we see that the JGDID+IR-SIS improved the upper bound significantly compared to JGDID.

Average quality of upper bounds. Table 3 also shows the average quality of the upper bounds for the 4 direct algorithms. We see that the average quality of JGDID dominates MBE-VA. The average quality of the upper bounds obtained from MBE-VA(i=1) are so weak that their value is closed to 0.0 in all benchmarks. Both JGDID and MBE-VA improve the average quality of the upper bounds when they are combined with IR-SIS, and JGDID+IR-SIS presents the best average quality overall.

5 CONCLUSIONS

We present a new algorithm, Join Graph Decomposition for solving Influence Diagrams, called JGDID, using the valuation algebra. Our scheme subsumes the decomposition bounds for marginal MAP, and also provide a bound for the MEU. Our experiments show the effectiveness of the translation free approach and the significant improvement in the quality of upper bounds compared with earlier state-of-the-art approaches. We also demonstrate that a join graph decomposition scheme can be combined with information relaxation scheme to yield superior bounds. The principle of decomposing a sequence of decision problems to a collection of weakly coupled subproblems can be further developed by incorporating advanced optimization frameworks, and the resulting, effective upper bounding schemes can be applied to probabilistic planning and stochastic programming.

ACKNOWLEDGEMENTS

This work was supported in part by NSF grants IIS-1526842 and IIS-1254071, the US Air Force (Contract FA9453-16-C-0508) and DARPA (Contract W911NF-18-C-0015).

References

Dechter, R. (1999). Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1):41–85.

Dechter, R. (2000). An anytime approximation for optimizing policies under uncertainty. In *AIPS-2000 Workshop on Decision Theoretic Planning*.

Dechter, R. and Rish, I. (2003). Mini-buckets: A general scheme for bounded inference. *Journal of the ACM (JACM)*, 50(2):107–153.

Howard, R. A. and Matheson, J. E. (2005). Influence diagrams. *Decision Analysis*, 2(3):127–143.

Jensen, F., Jensen, F. V., and Dittmer, S. L. (1994). From influence diagrams to junction trees. In *Proceedings of the 10th international conference on Uncertainty in artificial intelligence*, pages 367–373.

Kivinen, J. and Warmuth, M. K. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63.

Liu, Q. and Ihler, A. (2011). Bounding the partition function using Hölder's inequality. In *Proceedings of the 28th International Conference on Machine Learning*, ICML '11, pages 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*, pages 523–532.

Marinescu, R., Dechter, R., and Ihler, A. (2014). AND/OR search for marginal MAP. In *Uncertainty in Artificial Intelligence (UAI)*, pages 563–572, Quebec City, Canada.

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., de Campos, C. P., and Zaffalon, M. (2012). Solving limited memory influence diagrams. *Journal of Artificial Intelligence Research*, 44:97–140.

Nielsen, T. D. and Jensen, F. V. (1999). Welldefined decision scenarios. In *Proceedings of The 15th Conference on Uncertainty in Artificial Intelligence*, pages 502–511.

Nilsson, D. and Hohle, M. (2001). Computing bounds on expected utilties for optimal policies based on limited information. *Technical Report*, (94).

Ping, W., Liu, Q., and Ihler, A. T. (2015). Decomposition bounds for marginal MAP. In *Proceedings of Advances in Neural Information Processing Systems 28*, pages 3267–3275.

Sontag, D., Globerson, A., and Jaakkola, T. (2011). Introduction to dual decomposition for inference. *Optimization for Machine Learning*, 1(219-254):1.

Wright, S. and Nocedal, J. (1999). Numerical optimization. *Springer Science*, 35(67-68):7.

Yuan, C., Wu, X., and Hansen, E. A. (2010). Solving multistage influence diagrams using branch-and-bound search. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 691–700.