A Weighted Mini-Bucket Bound for Solving Influence Diagrams

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

Abstract

Influence diagrams provide a modeling and inference framework for sequential decision problems, representing the probabilistic knowledge by a Bayesian network and the preferences of an agent by utility functions over the random variables and decision variables. The time and space complexity of computing the maximum expected utility (MEU) and its maximizing policy are exponential in the induced width of the underlying graphical model, which is often prohibitively large. In this paper, we develop a weighted mini-bucket approach for bounding the MEU. These bounds can be used as a standalone approximation that can be improved as a function of a controlling i-bound parameter, or as a heuristic function to guide subsequent search. We evaluate the scheme empirically against the current state of the art, illustrating its potential.

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 the corresponding optimal policy for each decision, subject to the history of observations and the previous decisions.

Computing the MEU is recognized as one of the hardest tasks over graphical models, and hence recent work aims at developing anytime bounding schemes that tighten the bounds if more time and memory are available [Lee et al., 2018]. Often, the goal is to incorporate such bounds as heuristic functions to guide search algorithms. In this paper, we focus on computing an upper bound on the MEU for a single agent sequential decision making problem with no-forgetting assumptions.

There exist several approaches to develop such bounds:

Information relaxation methods relax the constraints on the variable ordering in the ID, re-ordering the chance variables and resulting in a model with smaller induced width whose value upper bounds the original ID [Nilsson and Hohle, 2001]. These "re-ordered" upper bounds can be used as heuristics, for example to guide a branch-and-bound search to solve the original ID [Yuan et al., 2010]. It is notable that, since these methods alter the ID itself, they are orthogonal to the approximate inference techniques described in the sequel and can potentially be combined with such approaches.

Task transform methods convert the MEU task into a marginal MAP query on a related graphical model, and then apply existing approximation techniques to the MMAP query [Liu and Ihler, 2012; Mauá, 2016]. This allows state-of-the-art methods to be applied, but is often impractical due to the size and complexity of the transformed graph.

Decomposition bounds for MEU generalize bounding techniques developed on other inference tasks to operate on IDs. For example, a version of mini-bucket [Dechter and Rish, 2003] was proposed for MEU in Dechter [2000]; mini-bucket relaxes the variable elimination procedure to ensure that its computational cost is at most exponential in a user-controlled parameter called the i-bound. Mini-bucket's similarity to variable elimination also makes it well-positioned to provide precomputed and thus efficient to evaluate bounds during search. Lee et al. [2018] extended the generalized dual decomposition bounds (GDD) for MMAP [Ping et al., 2015] to MEU tasks using the framework of *valuation algebra* [Shenoy and Shafer,

1990; Mauá et al., 2012; Moral, 2018], which defines operators such as combination and marginalization over pairs of probability and utility functions called *potentials* [Jensen et al., 1994]. This results in a class of decomposed bounds that can be optimized through iterative cost-shifting updates; however, unlike GDD for MMAP, these bounds are non-convex and often difficult or slow to optimize, particularly for large clique sizes.

One of the most effective current techniques for building anytime bounds and search heuristics in other inference tasks, such as MAP, summation, or MMAP, is *weighted mini-bucket* [Liu and Ihler, 2011; Ihler et al., 2012; Marinescu et al., 2014, 2018]. Weighted mini-bucket bounds are constructed via approximate variable elimination, but perform a single round of cost-shifting during construction ("moment matching") and/or a few iterations of optimization after construction. This results in decomposition bounds with benefits from both techniques: moderate to large clique sizes (controlled by the i-bound), with much of the gains from cost-shifting without significantly more computation.

In this work, we develop a weighted mini-bucket scheme for generating upper bounds on the MEU. Given a consistent variable ordering, the scheme generates bounds for each variable conditioned on past histories, observations and decisions relative to a given variable ordering. We develop a stage-wise cost-shifting procedure that attempts to minimize the bound locally during each approximate elimination step, based on optimizing a simplified upper bound on the unprocessed remainder of the model. We show empirically that our scheme can offer more effective bounds faster than existing state-of-the-art schemes. Thus, these bounds have great potential to be used as a heuristic function for search algorithms for solving IDs.

In Section 2, we review preliminaries and background (i.e., definition of influence diagrams, valuation algebra and decomposition bounds). Section 3 develops our weighted mini-bucket algorithm generating parameterized bound for IDs, empirical evaluation is given in Section 4 and Section 5 concludes.

2 BACKGROUND

2.1 INFLUENCE DIAGRAMS

An ID is a tuple $\mathcal{M} \coloneqq \langle \mathbf{C}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$ consisting 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 $\mathbf{P} = \{P_i | i \in \mathcal{I}_{\mathbf{P}}\}$, a set of real-valued additive utility functions $\mathbf{U} = \{U_i | i \in \mathcal{I}_{\mathbf{U}}\}$, and a constrained variable ordering $\mathcal{O}$ that will be elaborated below. We use $\mathcal{I}_{\mathbf{S}} = \{0, 1, \ldots, |S| - 1\}$ to denote

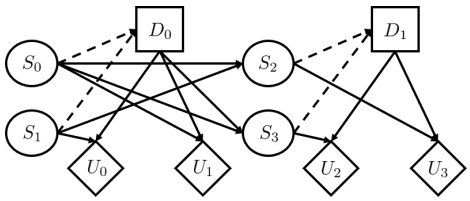

Figure 1: Factored 2 Stage Markov Decision Process (MDP) as an Influence Diagram

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, an ID can be associated with a DAG containing three types of nodes: chance nodes $\mathbf{C}$ drawn as circles, decision nodes $\mathbf{D}$ drawn as squares, and value nodes $\mathbf{U}$ drawn as diamonds representing utility functions defined over the chance and decision nodes in its scope. There are also three types of directed edges: edges directed into a chance node C_i from its parents $pa(C_i) \subseteq \mathbf{C} \cup \mathbf{D}$ representing the conditional probability function $P_i(C_i|pa(C_i))$, edges directed into a value node U_i denoting the utility function $U_i(\mathbf{X}_i)$ over its scope $\mathbf{X}_i \subseteq \mathbf{C} \cup \mathbf{D}$, and informational arcs (dashed arrows in Figure 1) directed from chance nodes to a decision node. The set of parent nodes I_i associated with a decision node D_i are assumed to be observed before making decision D_i. An ID implies the constrained partial variable ordering $\mathcal{O}$ which alternates between the observed chance variables and decision variables $\{\mathbf{I}_0 \prec D_0 \prec \ldots \prec \mathbf{I}_{|\mathbf{D}|-1} \prec D_{|\mathbf{D}|-1} \prec \mathbf{I}_{|\mathbf{D}|}\}$. Since the agent is no-forgetting, the partial ordering ensures the regularity property of an ID requiring a total ordering of all decision variables [Shachter, 1986], and dictates the available information at each decision D_i so that the agent makes decisions in a multi-staged manner based on the whole history available at each stage.

Solving an ID is computing the Maximum Expected Utility (MEU), $\mathbb{E}[\sum_i U_i | \mathbf{\Delta}]$ under an optimal strategy, $\mathbf{\Delta} = \{\Delta_i | \Delta_i : R(D_i) \mapsto D_i, \forall D_i \in \mathbf{D}\}$, where Δ_i is a policy for D_i and $R(D_i)$ is the requisite information to D_i [Nielsen and Jensen, 1999].

2.2 VALUATION ALGEBRA

The valuation algebra for IDs is an algebraic framework for computing the expected utility values, or values for short, using combination and marginalization operators applied to valuations, or "potentials" [Jensen et al., 1994; Mauá et al., 2012; Lee et al., 2018]. A valuation $\Psi(\mathbf{X})$ is a pair of probability and value functions $(P(\mathbf{X}), V(\mathbf{X}))$ over a set of variables $\mathbf{X}$ called its scope. We will sometime abuse notation, removing explicit reference to the scope of function, e.g., writing $P_1(\mathbf{X}_1)$ as P_1. We next define some operators over valuations.

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* $\Psi_2(\mathbf{X}_2){:=}(P_2(\mathbf{X}_2), V_2(\mathbf{X}_2))$*, the combination of the two valuations over* $\mathbf{X}_1 \cup \mathbf{X}_2$ *is defined by*

$$\Psi_1(\mathbf{X}_1) \otimes \Psi_2(\mathbf{X}_2) := (P_1P_2, P_1V_2 + P_2V_1).$$

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. ***(variable elimination for valuations)*** *Given a valuation* $\Psi(\mathbf{X}){:=}(P(\mathbf{X}), V(\mathbf{X}))$*, eliminating a variable* $Y \in \mathbf{X}$ *from a valuation by summation, maximization, or by powered-sum with weight* w*, is defined by*

$$\sum_Y \Psi(\mathbf{X}) := \Big(\sum_Y P(\mathbf{X}), \sum_Y V(\mathbf{X})\Big),$$
$$\max_Y \Psi(\mathbf{X}) := \Big(\max_Y P(\mathbf{X}), \max_Y V(\mathbf{X})\Big),$$
$$\sum_Y^{\mathbf{w}} \Psi(\mathbf{X}){:=}\Big(\sum_Y^w P(\mathbf{X}), \sum_Y^w V(\mathbf{X})\Big).$$

The powered-sum elimination operator $\sum_Y^w$ is defined by $\sum_Y^w f(\mathbf{X}) = [\sum_Y |f(\mathbf{X})|^{1/w}]^w$ for a variable Y associated with a weight $0 \leqslant w \leqslant 1$, and reduces to simple max and sum when $w{\to}0$ and $w{=}1$, respectively.

Definition 3. ***(comparison of two valuations)*** *Given two valuations* $\Psi_1 := (P_1, V_1)$ *and* $\Psi_2 : (P_2, V_2)$ *on the same scopes, we say that* $\Psi_1 \leqslant \Psi_2$ *iff* $P_1 \leqslant P_2$ *and* $V_1 \leqslant V_2$*.*

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

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

where $\mathcal{I}_\Psi$ is the set of indices of all valuations $\mathbf{\Psi}$, and $\mathbf{X}_\alpha$ denotes the scope of Ψ_α.

2.3 DECOMPOSITION BOUNDS

Graphical Model Decomposition The dependence relation between variables in a graphical model can be captured by a primal graph $\mathcal{G}_p = (V, E)$. In particular, the primal graph of an ID has chance variables and decision variables as nodes and an edge $e \in E$ connects two nodes if their corresponding variables appear in the scope of either a probabilistic or a utility function.

The bucket-tree decomposition [Dechter, 1999] decomposes a graphical model into a tree of buckets, one for each variable, that can be generated by triangulating the primal graph $\mathcal{G}_p$ along a total variable ordering. Each bucket has a subset of variables identified by the triangulated graph cliques and a subset of functions, satisfying some properties (e.g., the running intersection property). Subsequently, messages are passed between the buckets up and down the bucket-tree yielding exact computation [Dechter, 2013]. A bucket-tree is a type of tree-decomposition called join-tree. The mini-bucket scheme [Dechter and Rish, 2003] relaxes such exact tree-decomposition by splitting a bucket into mini-buckets whenever its size exceeds a bounding parameter termed the i-bound, yielding a mini-bucket tree. This splitting can be interpreted as variable duplication that forces additional independence. Join-graph decomposition [Mateescu et al., 2010] is a structure that facilitates another type of approximation via iterative message-passing. A join-graph can be generated by extending the mini-bucket tree into a graph by adding a chain of edges between mini-buckets of the same variable. The resulting structure is a graph connecting clusters called a join-graph. The edges of a join-graph can be labeled by a *separator*, which is a subset of variables shared by two adjacent clusters.

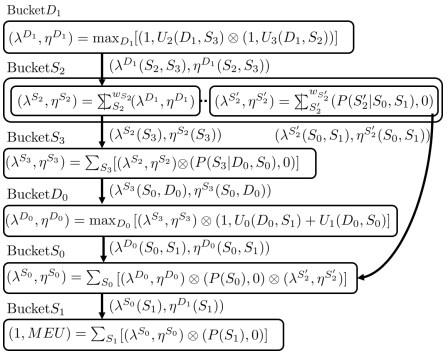

Figure 2: Weighted Mini-Bucket Elimination example. The details are elaborated in Example 1.

Weighted Mini-buckets and Cost-Shifting Schemes Weighted mini-bucket [Liu and Ihler, 2011] further tightens the mini-bucket relaxation by using Hölder's inequality,

$$\sum_X^{w_X} \prod_{\alpha=1}^q f_\alpha(\mathbf{X}_\alpha) \leqslant \prod_{\alpha=1}^q \sum_{X_\alpha}^{w_X^\alpha} f_\alpha(\mathbf{X}_\alpha), \quad (2)$$

where q is the number of mini-buckets of a given variable X, $f_\alpha(\mathbf{X}_\alpha)$ is the combined function at the α-th mini-bucket, w_X is the weight of the variable X (1 for the summation and 0 for the maximization), and w_X^α is a set of non-negative weights distributed to q mini-buckets such that $w_X = \sum_{\alpha=1}^q w_X^\alpha$.

For bounding MEU of IDs, Lee et al. [2018] extended the weighted mini-bucket scheme to valuation algebra using

Hölder's and Minkowski's inequality, showing that

$$\sum_{X}^{w_X} \otimes_{\alpha=1}^{q} \Psi_\alpha(\mathbf{X}_\alpha) \leqslant \bigotimes_{\alpha=1}^{q} \sum_{X_\alpha}^{w_X^\alpha} \Psi_\alpha(\mathbf{X}_\alpha), \qquad (3)$$

where $\Psi_\alpha(\mathbf{X}_\alpha)$ is a pair of probability and value functions.

Example 1. *Figure 2 illustrates the schematic trace of weighted mini-bucket elimination applied to the ID in Figure 1 with i-bound 2. First, a constrained elimination ordering consistent to the partial ordering specified by the informational arcs is $\mathcal{O} := \{D_1 \prec S_2 \prec S_3 \prec D_0 \prec S_0 \prec S_1\}$. To generate a mini-bucket tree, we process the variables following the order $\mathcal{O}$. For each variable, we collect all valuations that contain that variable and place them in a bucket. When the total number of variables in a bucket exceeds i+1, we partition the bucket into mini-buckets such that each mini-bucket contains $i+1$ or fewer variables. For example, bucket S_2 is partitioned into two mini-buckets in Figure 2. Then, we generate messages by eliminating the labelling variable of each mini-bucket from the combined valuation and send the result (a valuation message) to a mini-bucket in a lower layer. For example, the mini-bucket on the right of bucket S_2 sends its message to bucket S_0. Weights can be assigned to each bucket labelled by a summation variable. The weighted mini-bucket relaxation for bucket S_2 is $\sum_{S_2} \lambda^{D_1}(S_2, S_3) \otimes (P(S_2|S_0, S_1), 0) \leqslant [\sum_{S_2}^{w_{S_2}} \lambda^{D_1}(S_2, S_3)] \otimes [\sum_{S_2}^{w_{S_2'}} (P(S_2|S_0, S_1), 0)]$.*

Decomposition bounds can also be optimized by introducing auxiliary cost-shifting functions between decomposed clusters, thus reparameterizing the original functions allocated to each cluster [Sontag et al., 2011; Liu and Ihler, 2011; Ihler et al., 2012; Ping et al., 2015]. In particular, the generalized dual decomposition (GDD) bound [Ping et al., 2015] generates upper bounds of the MMAP task by jointly optimizing the weights associated with the variables at each cluster and the cost-shifting functions defined over the separators in a join graph decomposition under the fully decomposed setting.

More recently, [Lee et al., 2018] extended this idea to a GDD reparameterized bound for the MEU task thus yielding a join-graph decomposition bound for IDs (JGDID). This scheme recursively applies the weighted mini-bucket scheme for IDs in Eq. (3) to the all variables at once, described as

$$\sum_{\mathbf{X}}^{\mathbf{w}} \otimes_{\alpha \in \mathcal{I}_\alpha} \Psi_\alpha(\mathbf{X}_\alpha) \leqslant \bigotimes_{\alpha \in \mathcal{I}_\alpha} \sum_{\mathbf{X}_\alpha}^{\mathbf{w}^\alpha} \Psi_\alpha(\mathbf{X}_\alpha), \qquad (4)$$

where $\mathcal{I}_\alpha$ is the index set of all nodes in a join graph, $\Psi_\alpha(\mathbf{X}_\alpha)$ is a valuation that combines all valuations in the α-th cluster, $\mathbf{w} = \{w_{X_1}, \ldots, w_{X_{|\mathbf{X}|}}\}$ is the set of all the weights that corresponds to all the variables $\mathbf{X}$, $\mathbf{X}_\alpha$ is the set of variables appear in the α-th cluster, and $\mathbf{w}^\alpha = \{w^\alpha_{X_1}, \ldots, w^\alpha_{X_{|\mathbf{X}|}}\}$ is the set of weights of the $\mathbf{X}_\alpha$ such that $w_{X_i} = \sum_{\alpha \in \mathcal{I}_\alpha} w^\alpha_{X_i}$ for all $X_i \in \mathbf{X}$. The auxiliary optimization parameters are the cost-shifting valuations, $\delta_{(\mathcal{C}_i, \mathcal{C}_j)}$ and $\delta_{(\mathcal{C}_j, \mathcal{C}_i)}$ between two clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ that are defined over the separator variables $\mathcal{S}_{(\mathcal{C}_i, \mathcal{C}_j)}$ such that both cancels to satisfy $\delta_{(\mathcal{C}_i, \mathcal{C}_j)} \otimes \delta_{(\mathcal{C}_j, \mathcal{C}_i)} = (1, 0)$, where $\mathbf{w}^{\mathcal{C}}$ are the weight parameters distributed to each cluster $\mathcal{C}$. Incorporating the optimization parameters into Eq. (4), we can rewrite the reparameterized bound for IDs as,

$$\sum_{\mathbf{X}}^{\mathbf{w}} \bigotimes_{\alpha \in \mathcal{I}_\alpha} \Psi_\alpha(\mathbf{X}_\alpha) \leqslant \bigotimes_{\alpha \in \mathcal{I}_\alpha} \sum_{\mathbf{X}_\alpha}^{\mathbf{w}^\alpha} [\Psi_\alpha(\mathbf{X}_\alpha) \otimes (\bigotimes_{(\alpha, \mathcal{C}_j) \in \mathcal{S}} \delta_{(\alpha, \mathcal{C}_j)})], \quad (5)$$

where each cluster's valuation Ψ_α is reparameterized by the shifts δ from the incident edges of the join graph. Minimizing the value component of the right hand-side of Eq. (5) relative to each $\mathbf{w}^\alpha$ for all $\alpha \in \mathcal{I}_\alpha$, and each $\delta_{(\mathcal{C}_i, \mathcal{C}_j)}$ and $\delta_{(\mathcal{C}_j, \mathcal{C}_i)}$ for all $(\mathcal{C}_i, \mathcal{C}_j) \in \mathcal{S}$, subject to constraints $w_X = \sum_\alpha w^\alpha_X$, $X \in \mathbf{X}$, and $\delta_{(\mathcal{C}_i, \mathcal{C}_j)} \otimes \delta_{(\mathcal{C}_j, \mathcal{C}_i)} = (1, 0)$, $(\mathcal{C}_i, \mathcal{C}_j) \in \mathcal{S}$, yields a local optimum with a tighter MEU bound.

3 A WEIGHTED MINI-BUCKET BOUND FOR IDS

The value component of the decomposition bounds for IDs such as Eq. (5) is not a convex function of the weights $\mathbf{w}$ and the cost-shifting functions δ because a bound on the global expected utility value is a weighted sum of value components, each multiplied by the product of the probability components of all other clusters. Note that the parameters $\mathbf{w}$ and δ appear in both probability and value components. This means that optimization difficulty can play a significant role in the quality of the final bounds. For example, we observed that the JGDID scheme [Lee et al., 2018] often shows *worse* upper bounds at higher i-bounds, in part because of the increased dimension of the objective function (which is exponential in the i-bound).

An alternative approach we explore here is to interleave the variable elimination and decomposition/optimization of the clusters "on the fly" in a more local manner. Ihler et al. [2012] presented a single pass approximate variable elimination algorithm that matches the max-marginal moments of the mini-buckets for MPE task, and Marinescu et al. [2014] showed a similar approach for MMAP. This way, the intermediate reparameterization step optimizes a partial decomposition scheme applied to a single cluster of the bucket-tree, yielding a lower dimensional optimization space. In other words, the optimization process reparameterizes the mini-bucket relaxation locally within a

single bucket. It then generates messages by the weighted power-sum for each mini-bucket independently, avoiding the difficult optimization steps altogether. In the following subsections, we develop a weighted mini-bucket elimination bounds for IDs (WMBE-ID) based on these ideas.

3.1 BOUND DERIVATION

Given an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{\Psi}, \mathcal{O} \rangle$, we could apply the weighted mini-bucket scheme in Eq. (3) to one variable at a time following the constrained elimination ordering $\mathcal{O} : \{X_{|\mathbf{X}|}, X_{|\mathbf{X}|-1}, \ldots, X_1\}$. The intermediate messages could be sent to mini-buckets at lower layers, as illustrated in Figure 2. To tighten the upper bound, we introduce cost-shifting functions between mini-buckets, yielding the following parameterized bound for each bucket of a variable X,

$$\sum_{X}^{w_X} \bigotimes_{\alpha=1}^{q} \Psi_\alpha(\mathbf{X}_\alpha) \leqslant \bigotimes_{\alpha=1}^{q} \sum_{X_\alpha}^{w_X^\alpha} [\Psi_\alpha(\mathbf{X}_\alpha) \otimes \frac{\delta_{(\alpha-1,\alpha)}(X)}{\delta_{(\alpha,\alpha+1)}(X)}], \quad (6)$$

where w_X is the weight of the variable X (1 or 0), q is the total number of mini-buckets, w_X^α is the weight of X at its α-th mini-bucket, and $\delta_{(\alpha,\alpha+1)}(X)$ is the cost-shifting valuation from the α-th mini-bucket to the $(\alpha+1)$-th mini-bucket. Using the example in Figure 2, the reparameterized upper bound at Bucket S_2 can be written as,

$$[\sum_{S_2}^{w_{S_2}} \frac{(\lambda^{D_1}, \eta^{D_1})}{\delta_{(S_2,S_2')}}] \otimes [\sum_{S_2'}^{w_{S_2'}} (P(S_2'|S_0,S_1),0) \otimes \delta_{(S_2,S_2')}].$$

However, the value component of Eq. (6) cannot be used as an objective function to be optimized; it is not a scalar quantity, but rather a function whose scope may be as large as the induced width. Therefore, we propose a surrogate optimization objective function based on the fully decomposed bound as follows.

Theorem 1. ***(Weighted Mini-bucekt Elimination Bounds for IDs)*** *Given an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{\Psi}, \mathcal{O} \rangle$ and a constrained variable elimination ordering $\mathcal{O} := \{X_{|\mathbf{X}|}, X_{|\mathbf{X}|-1}, \ldots, X_1\}$, assume that the variables $\{X_{|\mathbf{X}|}, X_{|\mathbf{X}|-1}, \ldots X_{n+1}\}$ are already eliminated by the weighted mini-bucket elimination algorithm, and X_n is the current variable to eliminate. Let $\Psi^{X_i}(\mathbf{X}_{1:i})$ be the combination of all valuations allocated to bucket X_i of the bucket-tree, $Q_{X_i} := \{1, \ldots, q_{X_i}\}$ be the mini-bucket partitioning for bucket X_i, and $\Psi_\alpha^{X_i}(\mathbf{X}_\alpha^{X_i})$ be the combination of the valuations allocated at the α-th mini-bucket of bucket X_i. Then, we set up a surrogate upper bound, which we call weighted mini-bucket elimination bounds for IDs (WMBE-ID), on the MEU of the remaining subproblem and it is defined over those remaining variables $\mathbf{X}_{1:n} := \{X_1, \ldots, X_n\}$, as follows.*

$$\sum_{\mathbf{X}_{1:n-1}}^{\mathbf{w}_{1:n-1}} [\{\bigotimes_{i=1}^{n-1} \Psi^{X_i}(\mathbf{X}_{1:i})\} \otimes \{\sum_{X_n}^{w_{X_n}} \Psi^{X_n}(\mathbf{X}_{1:n})\}] \quad (7)$$

$$\leqslant \sum_{\mathbf{X}_{1:n-1}}^{\mathbf{w}_{1:n-1}} [\{\bigotimes_{i=1}^{n-1} \Psi^{X_i}(\mathbf{X}_{1:i})\} \otimes \{\bigotimes_{\alpha \in Q_{X_n}} \sum_{X_n}^{w_{X_n}^{X_n,\alpha}} \Psi_\alpha^{X_n}(\mathbf{X}_\alpha^{X_n})\}] \quad (8)$$

$$\leqslant \bigotimes_{i=1}^{n-1} [\bigotimes_{\alpha \in Q_{X_i}} \sum_{\mathbf{X}_{1:i}}^{\mathbf{w}_{1:i}^{X_i,\alpha}} \Psi_\alpha^{X_i}(\mathbf{X}_\alpha^{X_i})] \otimes [\bigotimes_{\alpha \in Q_{X_n}} \sum_{\mathbf{X}_{1:n}}^{\mathbf{w}_{1:n}^{X_n,\alpha}} \Psi_\alpha^{X_n}(\mathbf{X}_\alpha^{X_n})]. \quad (9)$$

The weights $\mathbf{w}_{1:n} := \{w_{X_1}, \ldots, w_{X_n}\}$ in Eq. (7) are the set of elimination weights for variables $\mathbf{X}_{1:n}$, either 1 (chance variable) or 0 (decision variable), and the weights $\mathbf{w}_{1:k}^{X_i,\alpha} := \{w_{X_1}^{X_i,\alpha}, \ldots, w_{X_k}^{X_i,\alpha}\}$ in Eq. (9) is a set of weights of the variables $\mathbf{X}_{1:k}$ in the α-th mini-bucket of bucket X_i such that $w_{X_j} = \sum_{l=1}^{k} \sum_{\alpha \in Q_{X_l}} w_{X_j}^{X_l,\alpha}$.

Proof. Concretely, Eq. (7) is the MEU of the subproblem after eliminating variables $\{X_{|X|}, \ldots, X_{n+1}\}$, and the inequality in Eq. (8) is obtained by applying weighted mini-bucket elimination of Eq. (3) to bucket X_n only. The final inequality, Eq. (9), is then given by applying the generalized dual decomposition bound for MEU in Eq. (4) to all mini-buckets $\{Q_{X_i} | X_i \in \mathbf{X_{1..n}}\}$ in the remaining subproblem, yielding a scalar valued function that bounds the MEU of the remaining subproblem. □

3.2 OPTIMIZING THE UPPER BOUND

Optimization Objectives and Parameters Given an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{\Psi}, \mathcal{O} \rangle$ and a constrained elimination ordering $\mathcal{O} := \{X_{|\mathbf{X}|}, X_{|\mathbf{X}|-1}, \ldots, X_1\}$, the weighted mini-bucket bounds as defined in Theorem 1 can be parameterized relative to a chain of mini-buckets Q_{X_n} when variable X_n is the next variable to be eliminated (after reparameterization) as follows.

$$[\bigotimes_{i=1}^{n-1} \bigotimes_{\alpha \in Q_{X_i}} \sum_{\mathbf{X}_{1:n-1}}^{\mathbf{w}_{1:n-1}^{X_i,\alpha}} \Psi_\alpha^{X_i}(\mathbf{X}_\alpha^{X_i})] \otimes$$

$$[\bigotimes_{\alpha \in Q_{X_n}} \sum_{\mathbf{X}_{1:n}}^{\mathbf{w}_{1:n}^{X_n,\alpha}} \Psi_\alpha^{X_n}(\mathbf{X}_\alpha^{X_n}) \frac{\delta_{(\alpha-1,\alpha)}^{X_n}(X_n)}{\delta_{(\alpha,\alpha+1)}^{X_n}(X_n)}] \quad (10)$$

The optimization parameters in Eq.(10) are the cost-shifting functions between adjacent mini-buckets and the weights over the mini-buckets Q_{X_n} namely,

$$\{\delta_{(\alpha,\alpha+1)}^{X_n}(X_n) | \forall \alpha, \alpha+1 \in Q_{X_n}\} \quad (11)$$

$$\{w_{X_n}^{X_n,\alpha} | \forall \alpha, \alpha+1 \in Q_{X_n}\}, \quad (12)$$

where $\delta_{(\alpha,\alpha+1)}^{X_n}(X_n)$ is a pair of probability and value components $\lambda_{(\alpha,\alpha+1)}^{X_n}(X_n)$ and $\eta_{(\alpha,\alpha+1)}^{X_n}(X_n)$. Dropping

Algorithm 1 Weighted Mini-Bucket Elimination Bounds for IDs (WMBE-ID)

Require: Influence diagram $\mathcal{M} = \langle \mathbf{X}, \mathbf{\Psi}, \mathcal{O} \rangle$, total constrained elimination order $\mathcal{O} := \{X_N, X_{N-1}, \ldots, X_1\}$, i-bound, iteration limit L,
Ensure: an upper bound of the MEU
1: Generate a schematic mini-bucket tree and allocate valuations to mini-buckets, $\{Q_{X_i} | X_i \in \mathbf{X}\}$.
2: Initialize weights $\{\mathbf{w}_{1:N}^{X_i,\alpha} | \forall X_i \in \mathbf{X}, \alpha \in Q_{X_i}\}$, and optionally tune the weights by a single JGDID update.
3: Compute fully decomposed bounds at all mini-buckets.
4: $Ub \leftarrow (1, 0)$
5: **for** $i \leftarrow N$ to 1 **do**
6: $iter = 0$
7: **while** $iter < L$ and bounds not converged **do**
8: Update cost functions $\{\delta_{(\alpha,\alpha+1)|\alpha \in Q_{X_i}}\}$ by SLSQP.
9: Update weights $\{w_{X_i}^{X_i,\alpha} | \alpha \in Q_{X_i}\}$ by EGD.
10: **end while**
11: **for** $\alpha \in Q_{X_i}$ **do**
12: $(\lambda_\alpha^{X_i}, \eta_\alpha^{X_i}) \leftarrow \sum_{X_i}^{w^{X_i,\alpha}} \tilde{\Psi}_\alpha^{X_i}(\mathbf{X}_\alpha^{X_i})$
13: **if** $(\lambda_\alpha^{X_i}, \eta_\alpha^{X_i})$ is constant **then**
14: $Ub \leftarrow Ub \otimes (\lambda_\alpha^{X_i}, \eta_\alpha^{X_i})$
15: **else**
16: Send message $(\lambda_\alpha^{X_i}, \eta_\alpha^{X_i})$ downward.
17: Combine the message and the valuation at the destination mini-bucket.
18: Merge the weights of the source and destination mini-buckets.
19: Recompute fully decomposed bound at the destination mini-bucket.
20: **end if**
21: **end for**
22: **end for**
23: Return value component of Ub

the scopes $\mathbf{X}_\alpha^{X_i}$, the objective function obtained (following some algebraic manipulation), derived as the value component of Eq. (10), can be written explicitly as

$$\Gamma \cdot \Bigg[\sum_{i=1}^{n-1} \sum_{\alpha \in Q_{X_i}} \frac{\sum_{\mathbf{X}_{1:n-1}}^{\mathbf{w}_{1:n-1}^{X_i,\alpha}} V_\alpha^{X_i}}{\sum_{\mathbf{X}_{1:n-1}}^{\mathbf{w}_{1:n-1}^{X_i,\alpha}} P_\alpha^{X_i}} + \sum_{\alpha \in Q_{X_n}} \frac{\sum_{\mathbf{X}_{1:n}}^{\mathbf{w}_{1:n}^{X_n,\alpha}} P_\alpha^{X_n} \frac{\lambda_{(\alpha-1,\alpha)}^{X_n}}{\lambda_{(\alpha,\alpha+1)}^{X_n}} \left[\frac{V_\alpha^{X_n}}{P_\alpha^{X_n}} - \frac{\eta_{(\alpha,\alpha+1)}^{X_n}}{\lambda_{(\alpha,\alpha+1)}^{X_n}} + \frac{\eta_{(\alpha-1,\alpha)}^{X_n}}{\lambda_{(\alpha-1,\alpha)}^{X_n}} \right]}{\sum_{\mathbf{X}_{1:n}}^{\mathbf{w}_{1:n}^{X_n,\alpha}} P_\alpha^{X_n} \frac{\lambda_{(\alpha-1,\alpha)}^{X_n}}{\lambda_{(\alpha,\alpha+1)}^{X_n}}} \Bigg], \quad (13)$$

where Γ is the probability component of Eq. (10),

$$\Gamma = \Big[\prod_{i=1}^{n-1} \prod_{\alpha \in Q_{X_i}} P_\alpha^{X_i} \Big] \cdot \Big[\prod_{\alpha \in Q_{X_n}} P_\alpha^{X_n} \frac{\lambda_{(\alpha-1,\alpha)}^{X_n}}{\lambda_{(\alpha,\alpha+1)}^{X_n}} \Big]. \quad (14)$$

The probability component at the α-th mini-bucket of the bucket of X_n is reparameterized by the probability cost-shifting functions $\lambda_{(\alpha-1,\alpha)}^{X_n}$ and $\lambda_{(\alpha,\alpha+1)}^{X_n}$ as,

$$P_\alpha^{X_n} \frac{\lambda_{(\alpha-1,\alpha)}^{X_n}}{\lambda_{(\alpha,\alpha+1)}^{X_n}}. \quad (15)$$

Similarly, the value component at the the α-th mini-bucket of the bucket of X_n is reparameterized by subtracting the outgoing utility cost and adding the incoming utility cost to $\frac{V_\alpha^{X_n}}{P_\alpha^{X_n}}$, and multiplying the new probability component in Eq. (15) yielding,

$$\Big[P_\alpha^{X_n} \frac{\lambda_{(\alpha-1,\alpha)}^{X_n}}{\lambda_{(\alpha,\alpha+1)}^{X_n}} \Big] \Big[\frac{V_\alpha^{X_n}}{P_\alpha^{X_n}} - \frac{\eta_{(\alpha,\alpha+1)}^{X_n}}{\lambda_{(\alpha,\alpha+1)}^{X_n}} + \frac{\eta_{(\alpha-1,\alpha)}^{X_n}}{\lambda_{(\alpha-1,\alpha)}^{X_n}} \Big]. \quad (16)$$

Constraints for Optimizing the Cost-shifting Functions Since the power-sum operator is defined over the absolute value of a function, the value components of all mini-buckets must remain non-negative after reparameterization. Let $\Psi_\alpha^{X_n}(\mathbf{X}_\alpha)$ be the valuation at the α-th mini-bucket of the bucket of X_n having the probability component $\lambda_\alpha^{X_n}(\mathbf{X}_\alpha)$ and the value component $\eta_\alpha^{X_n}(\mathbf{X}_\alpha)$, and $\delta_{(\alpha,\alpha+1)}(X_n) := (\lambda_{(\alpha,\alpha+1)}(X_n), \eta_{(\alpha,\alpha+1)}(X_n))$ be the cost-shifting valuation between the α-th and $(\alpha+1)$-th mini-buckets. Then, the non-negativity of the value components after the reparameterization is ensured by:

$$\frac{\eta_\alpha^{X_n}(\mathbf{X}_\alpha)}{\lambda_\alpha^{X_n}(\mathbf{X}_\alpha)} - \frac{\eta_{(\alpha,\alpha+1)}(X_n)}{\lambda_{(\alpha,\alpha+1)}(X_n)} + \frac{\eta_{(\alpha-1,\alpha)}(X_n)}{\lambda_{(\alpha-1,\alpha)}(X_n)} \geqslant 0. \quad (17)$$

In addition, the non-negativity of the probability components is ensured by:

$$\lambda_{(\alpha,\alpha+1)}(X_n) \geqslant 0 \quad (18)$$

for all mini-buckets $\alpha \in Q_{X_n}$. Equipped with the above optimization objective and constraints, any constrained optimization procedure can be applied to reparameterize the mini-buckets.

Optimization Routines In the empirical evaluation, we implemented optimization routines that optimize the probability cost-shifting functions, the value cost-shifting functions, and the weights. In order to optimize the probability and value cost-shifting functions, we integrated sequential least square programming (SLSQP) [Kraft, 1988] to optimize Eq. (13) subject to the constraints of Eq. (17) and Eq. (18). In order to optimize the weights, we implemented the exponentiated gradient descent (EGD) [Kivinen and Warmuth, 1997] as shown in [Lee et al., 2018].

Interleaving Elimination and Optimization Algorithm 1 outlines our overall algorithm, weighted mini-bucket elimination that is interleaved with reparameterization, which computes an upper bound on the MEU. Given an input ID $\mathcal{M}$ and a constrained elimination order $\mathcal{O}$, the schematic mini-bucket elimination algorithm [Dechter and Rish, 2003] generates a mini-bucket tree and valuations are allocated to mini-buckets (line 1). Then, all the weights are initialized uniformly over a join-graph obtained from the mini-bucket tree; an optional tuning step that runs a single JGDID iteration on the weights often showed a significant impact in our empirical evaluation (line 2). Initialization is completed by computing the fully decomposed bounds at each mini-bucket using

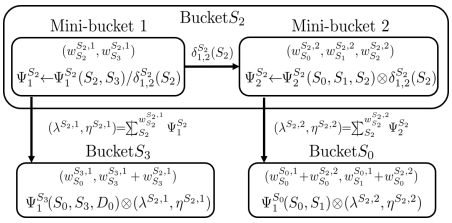

Figure 3: Optimizing the WMBE-ID Bounds for the example shown in Figure 2. The details are elaborated in Example 2.

the initial weights and valuations, $\sum_{\mathbf{X}_{1:N}}^{\mathbf{w}_{1:N}^{X_i,\alpha}} \Psi_\alpha^{X_i}(\mathbf{X}_\alpha^{X_i})$ for all $\{(X_i, \alpha) | X_i \in \mathbf{X},\ \alpha \in Q_{X_i}\}$ (line 3). In the main loop, variables are processed from first to last in the constrained elimination ordering. Given the current variable X_i, the cost-shifting functions and weights that parameterizes X_i's mini-buckets are updated to tighten the upper bound of Eq. (13) by the optimization routines (line 7 – 10). Subsequently, mini-bucket messages at the current layer are computed using the optimized weights and reparameterized valuations (line 12). These messages are sent downward along the mini-bucket tree, and the weights and valuations at the destination mini-bucket are incorporated into the fully decomposed bound (line 16 – 18).

Example 2. *Figure 3 illustrates how the mini-buckets of bucket S_2 in Figure 2 are optimized by the algorithm. Before entering the optimization loop, the mini-buckets of bucket S_2 contain the input valuations and the messages received from the upper layers. Namely, $\Psi_1^{S_2} = (\lambda^{D_1}, \eta^{D_1})$, and $\Psi_2^{S_2} = (P(S_2|S_0, S_1), 1)$. The cost-shifting $\delta_{1,2}^{S_2}(S_2)$ reparameterizes the valuations at each mini-bucket by shifting the cost $\delta_{1,2}^{S_2}(S_2)$ from mini-bucket1 to mini-bucket2. The weights $w_{S_2}^{S_2,1}$ and $w_{S_2}^{S_2,2}$ are optimized under the constraint $1 = w_{S_2}^{S_2,1} + w_{S_2}^{S_2,2}$. In mini-bucket1 , the weighted mini-bucket message $(\lambda^{S_2,1}, \eta^{s_2,1})$ is computed by $\sum_{S_2}^{w_{S_2}^{S_2,1}} \Psi_1^{S_2}$, and sent to the bucket of S_3 Once the optimization steps ran up to the maximum number of iterations or until convergence. The bucket of S_3 combines the incoming message and the preassigned valuation $\Psi_1^{S_3}$, merges the incoming weights $w_{S_3}^{S_2,1}$ to $w_{S_3}^{S_3,1}$, and recomputes the fully decomposed bound to incorporate the modification.*

4 EXPERIMENTS

We compare the performance of our proposed bounding scheme WMBE-ID with earlier approaches on 4 domains each containing 5 problem instances. The benchmark statistics are summarized in Table 1.

Benchmarks We generated 4 domains in the following way: (1) Factored Finite Horizon Markov Decision Process (FH-MDP) instances are generated from two stage factored MDP templates by varying the number of state and action variables, the scope size of functions, and the length of time steps between 3 and 10. (2) Factored Finite Horizon Partially Observable MDP (FH-POMDP) instances are generated similarly to MDP instances, but it incorporates observed variables. (3) Random influence diagrams (RAND) are generated from a random topology of influence diagram by varying the number of chance, decision, and value nodes. (4) BN influence diagram instances are IDs converted from the Bayesian networks `BN-0` and `BN-78` that are released in the UAI-2006 probabilistic inference challenge by converting random nodes to decision nodes and adding utility nodes.

Domain	n	f	k	s	w
FH-MDP	99, 145, 170	120, 170, 240	3, 3, 5	7, 9, 9	21, 39, 43
FH-POMDP	57, 92, 96	72, 128, 140	2, 2, 3	5, 6, 9	28, 43, 47
RAND	60, 77, 91	60, 77, 91	2, 2, 2	3, 3, 3	20, 27, 41
BN	54, 54, 100	54, 54, 100	2, 2, 2	6, 10, 10	19, 24, 28

Table 1: Benchmark statistics. We show the minimum, median, and maximum values for each of the problem parameters: n – the number of chance and decision variables, f – the number of probability and utility functions, k – the domain size, s – the scope size, and w – the induced width, respectively.

Algorithms We evaluated our algorithm WMBE-ID using 4 configurations: (1) uniform weights without cost-shifting updates (WMBE-U) (2) uniform weights with cost-shifting updates (WMBE-UC), (3) updating both weights and applying cost-shifting (WMBE-WC1) when initial weights assigned uniformly, and (4) updating both weights and cost-shifting (WMBE-WC2) when using the initial weight tuning step by a single JGDID iteration. For comparison, we consider the following earlier approaches: the mini-bucket elimination bound (MBE) [Dechter and Rish, 2003; Mauá et al., 2012], MBE combined with the re-ordering relaxation (MBE-Re) [Nilsson and Hohle, 2001], and the state-of-the-art join graph decomposition bounds for IDs (JGDID) [Lee et al., 2018]. WMBE-U, MBE, and MBE-Re are non-iterative algorithms that compute the upper bounds in a single pass. The others are iterative algorithms that run either until the maximum number of iterations or they convergence. We implemented all algorithms in Python using the NumPy [Oliphant, 2015] and SciPy [Jones et al., 2001] libraries.

4.1 COMPARING ON INDIVIDUAL INSTANCES

Table 2 shows the quality of the upper bounds obtained by all 7 algorithms using i-bounds 1, 5, 10, and 15, and iteration limits of 1, 5, 10, and 20, on selected instances from the 4 domains. We can see that the quality of the bounds from MBE and MBE-Re is orders of

Instance (n,f,k,s,w)	Algorithm	i=1 (time (sec), bound) iter=1	5	10	20	i=5 (time (sec), bound) iter=1	5	10	20
ID_from_BN_78 _w19d3 (54, 54, 2, 10, 19)	WMBE-U	(47, 418)	-	-	-	(52, 147.78)	-	-	-
	WMBE-UC	(605, 381)	(519, 381)	(694, 381)	(376, 381)	(531, 151.58)	(226, 151.58)	(364, 151.58)	(410, 151.58)
	WMBE-WC1	(164, 406.20)	(631, 404.56)	(879, 236.99)	**(746, 252.08)**	(323, 113.86)	(704, 108.07)	**(804, 87.72)**	(1034, 107.82)
	WMBE-WC2	(382, 181.10)	(416, 159.81)	**(994, 153.40)**	(531, 157.34)	(706, 82.49)	(413, 77.89)	**(944, 77.11)**	(1045, 78.56)
	JGDID	(0.51, 889)	(973, 28.34)	**(1281, 27.53)**	-	(0.7, 4362)	(915, 118)	(2731, 33.64)	**(6076, 32.59)**
	MBE	(1, 3.14E+5)	-	-	-	(1, 2957)	-	-	-
	MBE-Re	(1, 2.45E+5)	-	-	-	(1, 19059)	-	-	
	Algorithm	i=10 (time (sec), bound) iter=1	5	10	20	i=15 (time (sec), bound) iter=1	5	10	20
	WMBE-U	(214, 46.93)	-	-	-	(252, 27,79)	-	-	-
	WMBE-UC	(350, 44.94)	(282, 44.94)	(286, 44.94)	(170, 44.94)	(270, 27,79)	(399, 27.79)	(147, 27.79)	(228, 27.79)
	WMBE-WC1	(176, 46.98)	**(222, 43.32)**	(480, 43.39)	(368, 45.33)	(211, 25.76)	(208, 25.69)	(287, 25.69)	**(193, 25.69)**
	WMBE-WC2	(538, 42.62)	(928, 41.71)	(1459, 38.99)	**(1910, 38.97)**	(406, 25.43)	(740, 26.08)	(919, 24.84)	**(1118, 24.82)**
	JGDID	(1, 4561)	(3530, 90.23)	(7433, 48.15)	**(15235, 47.08)**	(1, 4513)	(7391, 45.21)	**(15720, 42.34)**	-
	MBE	(1, 113)	-	-	-	(1, 37.38)	-	-	-
	MBE-Re	(1, 329)	-	-	-	(1, 49.43)	-	-	-
Instance (n,f,k,s,w)	Algorithm	i=1 (time (sec), bound) iter=1	5	10	20	i=5 (time (sec), bound) iter=1	5	10	20
mdp9 -32_3_8_3 (54, 54, 2, 10, 19)	WMBE-U	(466, 2.86E+9)	-	-	-	(494, 1.25E+8)	-	-	-
	WMBE-UC	(4975, 1.01E+9)	(8707, 1.01E+9)	(6689, 1.01E+9)	(6294, 1.01E+9)	(2202, 7.49E+7)	(2497, 7.49E+7)	(2929, 7.49E+7)	(2616, 7.49E+7)
	WMBE-WC1	(4931, 182.18)	(16965, 777.23)	**(18317, 123.72)**	(19084, 167.08)	**(2282, 22.65)**	(14020, 183.96)	(12021, 389.51)	(16946, 110.97)
	WMBE-WC2	(3840, 30.64)	**(14553, 22.73)**	(12235, 22.88)	(16595, 23.86)	(6176, 22.56)	(8280, 22.31)	**(6546, 22.25)**	(10479, 22.33)
	JGDID	(2.8, 1.65E+11)	(7625, 48.22)	**(15340, 23.58)**	-	(3, 2.58E+13)	(6790, 7104)	(18589, 634)	**(30243, 27.57)**
	MBE	(3, 2.4E+21)	-	-	-	(3, 4.3E+14)	-	-	-
	MBE-Re	-	-	-	-	-	-	-	-
	Algorithm	i=10 (time (sec), bound) iter=1	5	10	20	i=15 (time (sec), bound) iter=1	5	10	20
	WMBE-U	(581, 1666593)	-	-	-	(790, 372057)	-	-	-
	WMBE-UC	(2968, 8.25E+5)	(2529, 8.25E+5)	(2586, 8.25E+5)	(2783, 8.25E+5)	(4158, 3.19E+5)	(5129, 3.33E+5)	(3586, 3.22E+5)	(4840, 3.21E+5)
	WMBE-WC1	(3819, 21.28)	(4778, 27.78)	(11776, 22.03)	**(7378, 21.38)**	(8298, 20.45)	(9299, 43.38)	(20944, 26.40)	**(21815, 19.77)**
	WMBE-WC2	(6548, 21.49)	(12698, 21.07)	(14260, 21.08)	**(6987, 21.05)**	(12809, 20.78)	(22651, 20.35)	**(30883, 20.22)**	(35086, 20.26)
	JGDID	(4, 4.01E+14)	(4282, 8.75E+8)	(9294, 4.40E+7)	**(21315, 4.62E+6)**	(7, 4.63E+14)	(9250, 2.42E+8)	(19791, 5.35E+7)	**(40074, 7.39E+5)**
	MBE	(2.6, 159E+12)	-	-	-	(2, 1.73E+10)	-	-	-
	MBE-Re	-	-	-	-	-	-	-	-
Instance (n,f,k,s,w)	Algorithm	i=1 (time (sec), bound) iter=1	5	10	20	i=5 (time (sec), bound) iter=1	5	10	20
pomdp8 -14_9_3_12_4 (96, 140, 2, 6, 47)	WMBE-U	(312, 9.3E+14)	-	-	-	(2, 3.4E+14)	-	-	-
	WMBE-UC	(899, 9.3E+14)	(1198, 9.3E+14)	(1540, 9.3E+14)	(574, 9.3E+14)	(980, 2.4E+13)	(1478, 2.4E+13)	(1229, 2.4E+13)	(894, 2.4E+13)
	WMBE-WC1	(502, 3.30E+13)	**(1659, 1.62E+13)**	(1923, 1.62E+13)	(3772, 1.62E+13)	(499, 1.67E+12)	(1534, 1.08E+12)	**(1378, 1.08E+12)**	(1892, 1.08E+12)
	WMBE-WC2	(708, 2.41E+12)	(2200, 2.32E+12)	(2631, 2.33E+12)	**(4283, 2.24E+12)**	(1531, 3.12E+11)	(2567, 3.14E+11)	**(3451, 2.96E+11)**	(2967, 2.97E+11)
	JGDID	(2.05, 1.8E+16)	(1920, 3.73E+10)	(4458, 8.65E+9)	**(7111, 6.12E+8)**	**(3, 2.75E+19)**	-	-	-
	MBE	(3, 1.94E+22)	-	-	-	(3, 2.57E+18)	-	-	-
	MBE-Re	(4, 2.9E+18)	-	-	-	(2, 3.4E+14)	-	-	-
	Algorithm	i=10 (time (sec), bound) iter=1	5	10	20	i=15 (time (sec), bound) iter=1	5	10	20
	WMBE-U	(755, 1.90E+10)	-	-	-	(1861, 9.49E+8)	-	-	-
	WMBE-UC	(773, 1.90E+10)	(1575, 1.90E+10)	(968, 1.90E+10)	(607, 1.90E+10)	(2022, 9.49E+8)	(3186, 9.49E+8)	(1606, 9.49E+8)	(2363, 9.49E+8)
	WMBE-WC1	**(510, 3.97E+9)**	(1409, 4.33E+9)	(1209, 4.33E+9)	(1867, 4.33E+9)	**(792, 4.42E+8)**	(2409, 1.19E+10)	(3588, 4.71E+8)	(9236, 4.70E+8)
	WMBE-WC2	(4582, 2.63E+9)	**(4099, 2.23E+9)**	(4394, 2.23E+9)	(4970, 2.23E+9)	(5923, 6.82E+7)	(12317, 6.72E+7)	(11869, 6.72E+7)	**(7892, 6.72E+7)**
	JGDID	(4, 3.2E+18)	(3224, 3.48E+12)	(7101, 5.55E+11)	**(14635, 4.42E+9)**	(12, 1.4E+18)	(12196, 4.39E+11)	(28338, 1.29E+11)	**(49188, 6.81E+9)**
	MBE	(2, 1.4E+15)	-	-	-	(3, 7.3E+12)	-	-	-
	MBE-Re	(1, 3.26E+8)	-	-	-	(1, 2.84E+5)	-	-	-
Instance (n,f,k,s,w)	Algorithm	i=1 (time (sec), bound) iter=1	5	10	20	i=5 (time (sec), bound) iter=1	5	10	20
rand -c70d7o1-01 (91, 91, 2, 3, 41)	WMBE-U	(64, 1028113)	-	-	-	(109, 9227)	-	-	-
	WMBE-UC	(371, 1.05E+6)	(232, 1.05+6)	(263, 1.05+6)	(266, 1.05+6)	(481, 7588)	(341, 8196)	(687, 8196)	(344, 7625)
	WMBE-WC1	**(76, 3.98E+5)**	(171, 4.93E+5)	(241, 4.90E+5)	(314, 4.89E+5)	**(417, 6081.49)**	(415, 7247.53)	(324, 6235.09)	(249, 8662.08)
	WMBE-WC2	(407, 6.04E+6)	**(856, 1.38E+5)**	(796, 1.35E+5)	(687, 1.60E+5)	(251, 4201.44)	**(633, 2567.81)**	(868, 2877.79)	(678, 3639.72)
	JGDID	(1, 2.15E+7)	(337, 61104)	(842, 758)	**(1453, 686)**	(1, 1.84E+7)	(1248, 11213)	(3195, 762)	**(4397, 736)**
	MBE	(1, 2.69E+9)	-	-	-	(1, 4.46E+5)	-	-	-
	MBE-Re	(1, 2.61E+9)	-	-	-	(2, 7.39E+5)	-	-	-
	Algorithm	i=10 (time (sec), bound) iter=1	5	10	20	i=15 (time (sec), bound) iter=1	5	10	20
	WMBE-U	(160, 1863)	-	-	-	(193, 1542)	-	-	-
	WMBE-UC	(793, 1835)	(1460, 1820)	(576, 1819)	(911, 1820)	(833, 1521)	(1560, 1518)	(1056, 1516)	(930, 1512)
	WMBE-WC1	**(315, 1803.52)**	(357, 2041.08)	(585, 1870.45)	(524, 1866.48)	(702, 2.08E+6)	(1005, 1956.82)	(1111, 1846.16)	**(1231, 1828.79)**
	WMBE-WC2	(1326, 1107.85)	(1083, 1105.79)	(953, 1140.48)	**(819, 1098.92)**	(2625, 1086.30)	**(2081, 1061.17)**	(2006, 1077.61)	(2583, 1077.98)
	JGDID	(1, 20049757)	(4228, 2167)	(8358, 1303)	**(15787, 795)**	(2, 2.37E+7)	(7837, 2006)	**(15134, 1357)**	-
	MBE	(1, 4937)	-	-	-	(1, 7027)	-	-	-
	MBE-Re	(1, 40695)	-	-	-	(1, 16139)	-	-	-

Table 2: The performance of the bounding schemes on individual instances. n is the number of variables, f is the number of functions, k is the maximum domain size, s is the maximum scope size, w is the constrained induced width. We show (time (sec), upper bound) for various i-bounds and number of iterations for the 7 algorithms. WMBE-U is the mini-bucket elimination with uniform weights, WMBE-UC preforms cost-shifting without optimizing the weight, WMBE-WC1/2 optimizes both weights and costs, JGDID is the fully decomposed bound over a join graph that optimizes both weights and costs, MBE is the mini-bucket elimination, and MBE-Re is the mini-bucket elimination with relaxed variable ordering. The best bounds from WMBE-WC1/2 and JGDID are highlighted by the boldface.

	WMBE-WC2												JGDID (i=10)	
Instance	i=10, iter=1		i=10, iter=5		i=15, iter=1		i=15, iter=5		i=20, iter=1		i=20, iter=5		i=10, max iter 100	
ID_from_BN_0_w28d6	13%	2.60	19%	2.74	61%	1.40	40%	2.13	193%	1.33	248%	1.37	410%	1.30
ID_from_BN_0_w29d6	10%	1.56	29%	1.52	26%	**0.96**	30%	**0.94**	132%	1.01	210%	1.02	255%	1.58
ID_from_BN_78_w19d3	40%	1.55	70%	1.51	31%	**0.92**	56%	**0.95**	244%	**0.64**	299%	**0.64**	1582%	1.70
ID_from_BN_78_w23d6	5%	1.53	5%	1.51	6%	1.24	8%	1.16	31%	**0.87**	38%	**0.84**	127%	1.79
ID_from_BN_78_w24d6	16%	2.12	21%	3.42	17%	1.15	31%	1.08	54%	**0.72**	90%	**0.70**	167%	1.83
mdp5-16_3_8_10	10%	**0.88**	13%	**0.87**	88%	**0.76**	91%	**0.76**	-	-	-	-	114%	4.68E+11
mdp6-20_5_5_5	13%	**0.94**	20%	**0.94**	28%	**0.85**	34%	**0.86**	771%	**0.80**	659%	**0.80**	339%	2.69E+04
mdp7-28_3_6_5	14%	1.14E+07	17%	**0.95**	20%	**0.93**	26%	**0.93**	-	-	333%	**0.87**	128%	1.56E+07
mdp8-28_3_6_4	21%	**0.97**	41%	**0.96**	34%	**0.91**	42%	**0.90**	287%	**0.87**	365%	**0.87**	228%	8.69E+03
mdp9-32_3_8_3	24%	**0.93**	46%	**0.91**	46%	**0.90**	81%	**0.88**	503%	**0.84**	520%	**0.83**	142%	2.65E+03
pomdp10-12_7_3_8_4	80%	**0.10**	93%	**0.07**	667%	17.11	1008%	**4.06E-03**	-	-	-	-	264%	**0.32**
pomdp6-12_6_2_6_3	26%	8.59	78%	4.25	43%	**0.17**	98%	**0.16**	607%	**0.01**	489%	**0.01**	109%	7.36
pomdp7-20_10_2_10_3	27%	1083.13	53%	20.72	51%	**0.89**	59%	34.93	827%	1.78	1247%	**0.03**	174%	72.80
pomdp8-14_9_3_12_4	42%	4.30	38%	3.64	55%	**0.11**	114%	**0.11**	1504%	**0.03**	2022%	**0.01**	199%	7.22
pomdp9-14_8_3_10_4	56%	1.27	87%	**0.87**	1455%	**0.02**	1775%	**0.02**	-	-	-	-	395%	17.00
rand-c50d15o1-03	56%	1.83	25%	1.98	51%	1.07	82%	1.04	379%	**0.87**	576%	**0.87**	1515%	1.35
rand-c50d5o1-01	42%	**0.91**	86%	**0.90**	44%	**0.69**	78%	**0.66**	61%	**0.61**	73%	**0.61**	1687%	1.02
rand-c70d14o1-01	79%	9.64	121%	3.31	172%	11.55	252%	13.53	2073%	1.65	2284%	1.78	1872%	1.23
rand-c70d21o1-01	56%	3.70	28%	3.72	85%	1.36	122%	1.24	604%	1.17	788%	1.22	885%	1.16
rand-c70d7o1-01	91%	1.61	75%	1.61	181%	1.58	143%	1.55	338%	**0.73**	422%	**0.73**	1475%	1.15
geometric mean	27%	4.96	37%	1.62	59%	**0.89**	81%	**0.71**	313%	**0.56**	384%	**0.41**	361%	66.03

Table 3: Comparing the ratio of time measured in seconds (left column) and quality of upper bounds (right column) against JGDID(i=1). WMBE-WC2 were provided with i-bound 10, 15 and 20, and the maximum number of iteration 1 and 5. JGDID were provided i-bound 1 and 10 with the maximum number iteration 100. All the quantities are normalized by the statistics of JGDID(i=1). WMBE-WC2 was terminated by time cut-off when a single layer optimization exceeds the 7200 second time limit. Normalized bounds tighter than JGDID(i=1) bound (less than 1.0) are highlighted by the boldface.

magnitude worse than the other algorithms. Algorithm JGDID generates tight bounds on many of the cases, especially at low i-bounds, but it consistently produced worse bounds with higher i-bounds and it takes more time; on the `mdp9-32-3-8-3` instance, the upper bound from JGDID(i=15) is 7.39E+5 (40074 sec), whereas it is only 23.58 (15340 sec) from JGDID(i=1).

In contrast, all the WMBE-ID algorithms consistently improved the quality of the bounds when using higher i-bounds. For example, on instance `ID-from-BN-78-w19d3`, WMBE-WC2 with 1 iteration computes upper bounds of 181.10, 82.49, 42.62 and 25.43 with i-bounds 1, 5, 10 and 15, respectively. We can also observe that WMBE-WC1 and WMBE-WC2 achieved the tightest bounds on all but `rand-c70d7o1-01` instances with i-bound 15. In the case of `ID-from-BN-78-w19d3`, WMBE-WC2 produced a better bound than the best of JGDID's bounds; 25.43 in 406 seconds by WMBE-WC2 with i-bound 15 and 1 iteration, and 27.53 in 1281 seconds by JGDID with i-bound 1. Similarly, WBME-WC1 generated better bounds for `mdp9-32-3-8-3` instance; 20.45 in 8298 seconds by WMBE-WC1 with i-bound 15 and 1 iteration, compared with 23.58 in 15340 seconds by JGDID with i-bound 1. Comparing the 3 variants of WMBE-ID algorithm, we see that optimizing both weights and cost-shifting functions greatly improved the quality of bounds but with additional time overhead.

4.2 COMPARING WMBE-ID VS. JGDID

Table 3 compares the quality of the upper bounds as well as the running time against JGDID(i=1) by normalizing both by the statistics of JGDID(i=1). Clearly, we see that JGDID with i-bound 10 shows degradation of the quality of the bounds on all but 1 instance. In contrast, WMBE-WC2 improved the upper bounds as the i-bounds increase. In particular, WMBE-WC2($iter = 5$) generated tighter bounds than JGDID(i=1); in 8 out of 20 instances, in 12 out of 20 instances, and in 13 out of 20 instances with i-bounds of 10, 15, and 20, respectively. The geometric mean over the normalized quantities summarizes the overall trend. It goes down from 1.62 to 0.71, and to 0.41 with longer running time, 37 %, 81%, and 384 % due to the increased i-bounds.

5 CONCLUSION

We presented a new weighted mini-bucket bounding scheme for influence diagrams, called WMBE-ID, which computes upper bounds of the MEU by interleaving variable elimination with optimizing partial decomposition within each variable's bucket. Compared with previous schemes, WMBE-ID yields tighter upper bounds faster, but may require more memory. In addition to being a better stand-alone bound, WMBE-ID bounds can be used as a static heuristic function for subsequent search and therefore facilitates an anytime algorithm for IDs in the spirit of earlier work for queries such as MAP and MMAP [Marinescu et al., 2018], a direction we plan to pursue.

Acknowledgements

We thank the reviewers for their helpful feedback. This work supported in part by NSF grants IIS-1526842 and IIS-1254071, the U.S. 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. (2013). Reasoning with probabilistic and deterministic graphical models: Exact algorithms. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 7(3):1–191.

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.

Ihler, A., Flerova, N., Dechter, R., and Otten, L. (2012). Join-graph based cost-shifting schemes. In *Uncertainty in Artificial Intelligence (UAI)*, pages 397–406, Corvallis, Oregon. AUAI Press.

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.

Jones, E., Oliphant, T., Peterson, P., et al. (2001). SciPy: Open source scientific tools for Python.

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

Kraft, D. (1988). A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*.

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 (UAI)*, pages 1053–1062.

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.

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

Moral, S. (2018). Divergence measures and approximate algorithms for valuation based systems. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Applications*, pages 591–602. Springer International Publishing.

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. *Dinar Research Report*.

Oliphant, T. E. (2015). *Guide to NumPy*. CreateSpace Independent Publishing Platform, USA, 2nd edition.

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.

Shachter, R. D. (1986). Evaluating influence diagrams. *Operations research*, 34(6):871–882.

Shenoy, P. P. and Shafer, G. (1990). Axioms for probability and belief-function propagations. In *Proceedings of The 4th Conference on Uncertainty in Artificial Intelligence*, pages 169–198.

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

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.