The Workshops of the Thirty-Second
AAAI Conference on Artificial Intelligence

Generalized Dual Decomposition for Bounding Maximum Expected Utility of Influence Diagrams with Perfect Recall

Junkyu Lee, Alexander Ihler, Rina Dechter
Donald Bren School of Information and Computer Sciences
University of California, Irvine
Irvine, CA 92697, USA

Abstract

We introduce a generalized dual decomposition bound for computing the maximum expected utility of influence diagrams based on the dual decomposition method generalized to L^p space. The main goal is to devise an approximation scheme free from translations required by existing variational approaches while exploiting the local structure of sum of utility functions as well as the conditional independence of probability functions. In this work, the generalized dual decomposition method is applied to the algebraic framework called valuation algebra for influence diagrams which handles probability and expected utility as a pair. The proposed approach allows a sequential decision problem to be decomposed as a collection of sub-decision problems of bounded complexity and the upper bound of maximum expected utility to be computed by combining the local expected utilities. Thus, it has a flexible control of space and time complexity for computing the bound. In addition, the upper bounds can be further minimized by reparameterizing the utility functions. Since the global objective function for the minimization is nonconvex, we present a gradient based local search algorithm in which the outer loop controls the randomization of the initial configurations and the inner loop tightens the upper-bound based on block coordinate descent with gradients perturbed by a random noise. The experimental evaluation demonstrates highlights of the proposed approach on finite horizon MDP/POMDP instances.

Introduction

An Influence Diagram (*ID*) (Howard and Matheson 2005) is a graphical representation of a sequential decision problem for a single agent maximizing the total expected utility under uncertainty. In this paper, we assume that the agent is non-forgetting, i.e., the previous history is available when making a sequence of decisions. The Maximum Expected Utility (*MEU*) query asks for a strategy that maximizes the sum of the expected value of local utilities over the probability distribution conditioned on the strategy. Exact algorithms for solving *ID*s are based on either variable elimination or reduction of the diagram. The variable elimination algorithms include strong junction tree algorithm (Jensen, Jensen, and Dittmer 1994), bucket elimination algorithm (Dechter 2000b), and multi-operator cluster DAG architecture (Pralet, Schiex, and Verfaillie 2006). The reduction type of algorithms use node removal and arc reversal techniques to transform the input *ID* to an *ID* with a sinlge node representing *MEU* (Shachter 1986), (Tatman and Shachter 1990), and (Hansen, Shi, and Khaled 2016). Since the complexity of the *MEU* query is NP^{PP} complete (Mauá 2016), exact algorithms are intractable.

Previous works on the systematic search algorithms for solving *ID*s are depth-first AND/OR search algorithm exploiting the problem decomposition (Marinescu 2010), and depth-first branch and bound search with the heuristic evaluation function generated by relaxing the subset of hidden variables to be observed (Yuan, Wu, and Hansen 2010) and (Khaled, Hansen, and Yuan 2013). Although depth-first search only requires linear memory, the space complexity for finding the optimal strategy is exponential in the length of the past history due to the perfect recall assumption. Recently, stochastic constrain programming solvers are combined with AND/OR branch and bound search with the interval arithmetic heuristic (Babaki, Guns, and De Raedt 2017). Various approximation algorithms are proposed in the literature: mini-bucket elimination for *ID* (Dechter 2000a), sampling based methods (Ortiz and Kaelbling 2000) and (Garcia-Sanchez and Druzdzel 2004), and sum-product network learning approach (Melibari, Poupart, and Doshi 2016). On the other hand, local search algorithms improving a subset of policies are proposed for solving limited memory influence diagrams that relax the non-forgetting assumptions and avoid the exponential space complexity (Lauritzen and Nilsson 2001) and (Mauá and Cozman 2016). (Mauá 2016) presented a translation scheme from an *ID* to marginal MAP, hence any marginal MAP inference algorithm can be applied to solve *ID*s. The variational framework provides bounds of the *MEU* when the global utility function can be factorized as a product of local functions. (Liu and Ihler 2012) presented a variational form of the *MEU* query under multiplicative utility functions, and proposed message passing algorithms tightening the dual form of the *MEU*. (Cheng et al. 2013) applied variational belief propagation algorithm to MDP planning. (Ping, Liu, and Ihler 2015) generalized dual decomposition for MAP (Sontag, Globerson, and Jaakkola 2011) to marginal MAP which can also be applied to solve *ID*s with a translation.

Contributions: In this paper, we present a generalized dual decomposition bound for the *MEU* of perfect recall *ID*s free from translations to other queries. The proposed bounding scheme decomposes a sequential decision problem by the join-graph decomposition. Then, the generalized dual decomposition method is applied to the decomposed clusters of the join-graph providing the upper bound of the *MEU*. This bound can also be minimized by reparameterizing utility functions. We present and demonstrate a gradient based local search algorithm that tightens the proposed upper bound on several MDP/POMDP domains to highlight the potential of the proposed approach.

Background

Influence Diagrams

An influence diagram is a tuple, $\boldsymbol{\mathcal{M}} = \langle \mathbf{C}, \mathbf{D}, \mathbf{P}, \mathbf{U}, T \rangle$, where $\mathbf{C} = \{C_i : i \in \mathcal{N}_{\mathbf{C}}\}$ is a set of discrete chance variables with a set of domains $\{\Omega_{C_i} : i \in \mathcal{N}_{\mathbf{C}}\}$, and $D = \{D_i : i \in \mathcal{N}_{\mathbf{D}}\}$ is a set of discrete decision variables with a set of domains $\{\Omega_{D_i} : i \in \mathcal{N}_{\mathbf{D}}\}$. A chance variable is drawn as a circle and decision variable is drawn as a square. Index sets $\mathcal{N}_X = \{0, 1, \cdots, |X| - 1\}$ are collection of non-negative integers representing the index of each element in a set $\mathbf{X}$. $\mathbf{P} = \{p_i = Pr(C_i | pa(C_i)) : \forall i \in \mathcal{N}_c\}$ is a set of conditional probability functions indexed by the set $\mathcal{N}_c$, where each p_i is defined over a set of chance variables C_i and its parent variables $pa(C_i) \subset \mathbf{C} \cup \mathbf{D} - \{C_i\}$. The node for a parent variable connects its child node with a directed edge. $\mathbf{U} = \{u_i = u_i(\mathbf{X}_i) : \forall i \in \mathcal{N}_u\}$ is a set of discrete real-valued utility functions indexed by a set $\mathcal{N}_u$, where each u_i is defined on a subset of variables $\mathbf{X}_i \subset C \cup D$. A utility node is drawn as a diamond. The parent nodes of a utility node correspond to the variable in the scope of the utility function. For the ease of notation, we assume that all decision variables follow the predefined T stage temporal ordering, which is the sequence in the index set $\mathcal{N}_d$. *ID*s define a partition of chance variables, called *information sets* $\{\mathbf{I}_k : k \in \mathcal{N}_d\}$ associated with the decision variable D_k, where the chance variables in $\mathbf{I}_k$ are observed immediately before making a decision for D_k. The partial temporal ordering on the variables $\mathcal{O}$:$\{\mathbf{I}_0 \prec D_0 \prec \cdots \prec \mathbf{I}_{T-1} \prec D_{T-1} \prec \mathbf{I}_T\}$ can be read off from an *ID* since the informational arcs connect chance nodes in $\mathbf{I}_k$ to the decision variable D_k. Note that the chance variables in the last information set $\mathbf{I}_T$ are unobserved variables. Then, the *MEU* of a T stage *ID* can be written as,

$$\sum_{\mathbf{I}_0} \max_{D_0} \cdots \sum_{\mathbf{I}_{T-1}} \max_{D_{T-1}} \sum_{\mathbf{I}_T} \Big(\prod_{p_i \in \mathbf{P}} p_i \Big) \Big(\sum_{u_i \in \mathbf{U}} u_i \Big). \qquad (1)$$

The set of policies is called a strategy $\boldsymbol{\Delta} = \{\Delta_k : k \in \mathcal{N}_d\}$ and each policy Δ_k is a probability distribution defined over the past history and the decision variables by a mapping $\Delta_k : \times_{i \leq k} ((\times_{C \in \mathbf{I}_i} \Omega_C) \times \Omega_{D_i}) \rightarrow [0, 1]$. The optimal strategy $\boldsymbol{\Delta}^*$ can be found by maximizing each decision variable D_k per all instantiations of variables in the past history.

Variable Elimination for Solving IDs

The valuation algebra (Shenoy and Shafer 1990) is a general algebraic framework for various reasoning architectures and it allows unified representation for the variable elimination algorithm for *ID*s. Here, we introduce essentials of valuation algebra for *ID*s following the notations in (Mauá, de Campos, and Zaffalon 2012). A ***valuation of IDs*** $\psi(\mathbf{X})$ with a scope $\mathbf{X} \subseteq C \cup D$ is a *pair* of functions (p, u), each defined over a set of variables $\mathbf{X}$, where p is the probability component and u is the expected utility component. From a valuation $\psi(\mathbf{X})$, $P(\psi(\mathbf{X}))$ denotes the probability component and $EU(\psi(\mathbf{X}))$ denotes the expected utility component. Given two valuations of an *ID* $\psi_1(\mathbf{X}_1) := (p_1, u_1)$ with scope $\mathbf{X}_1$ and $\psi_2(\mathbf{X}_2) := (p_2, u_2)$ with scope $\mathbf{X}_2$, the ***combination*** of two valuations is defined as $\psi_1(\mathbf{X}_1) \otimes \psi_2(\mathbf{X}_2) := (p_1 p_2, p_1 u_2 + p_2 u_1)$ with new scope $\mathbf{X}_1 \cup \mathbf{X}_2$. Now, we define the ***marginalization*** of a valuation. Let $\psi := (p, u)$ with scope $\mathbf{X}$. Then, the marginalization of a variable $Y \in \mathbf{X}$ from valuation ψ is $\Downarrow_Y \psi(\mathbf{X}) := (\downarrow_Y p, \downarrow_Y u)$ with new scope $X - \{Y\}$. The marginalization operator $\downarrow_Y$ acting on each component is $\sum_Y$ if $Y \in \mathbf{C}$ and $\max_Y$ if $Y \in bD$. Then, the *MEU* of a T stage *ID* with the temporal ordering $\mathcal{O}$ can be rewritten by valuation algebra as,

$$EU(\Downarrow_{\mathcal{O}} \{\otimes_{p_i \in \mathbf{P}} (p_i, 0)\} \otimes \{\otimes_{u_i \in \mathbf{U}} (1, u_i)\}), \qquad (2)$$

where each conditional probability function $p_i \in \mathbf{P}$ and utility function $u_i \in \mathbf{P}$ is converted to a *valuation* respectively as, $(p_i, 0)$ and $(1, u_i)$. In equation (2), all valuations are combined with a single combination operator and a variable is eliminated following the elimination ordering $\mathcal{O}$.

Join-Graph Decomposition of IDs

Given an *ID* $\boldsymbol{\mathcal{M}} = \langle \mathbf{C}, \mathbf{D}, \mathbf{P}, \mathbf{U}, T \rangle$, a join-graph decomposition is a triple $\boldsymbol{\mathcal{D}} = \langle \mathcal{G}_J, \chi, \psi \rangle$, where $\mathcal{G}_J = (V, E)$ is a graph and χ and ψ are labeling functions which associate with each node $v \in V$ two sets, $\chi(v) \subseteq \mathbf{C} \cup \mathbf{D}$ and $\psi(v) \subseteq \mathbf{P} \cup \mathbf{U}$ such that: (1) for each function $f_i \in \mathbf{P} \cup \mathbf{U}$, there exists only one vertex $v \in V$ such that $f_i \in \psi(v)$, and $scope(f_i) \subseteq \chi(v)$, (2) for each variable $x_i \in \mathbf{C} \cup \mathbf{D}$, the set $\{v \in V : x_i \in \chi(v)\}$ induces a connected subgraph of $\mathcal{G}_J$. Given two adjacent nodes u and v in join-graph $\mathcal{G}_J$, the separator of u and v is defined as $\mathcal{S}_{uv} = \theta(u, v)$, where the $\theta(u, v)$ is an edge-labeling function satisfying $\theta(u, v) \subseteq \chi(u) \cap \chi(v)$ and two nodes containing a variable x_i can be connected by a path whose every edge label includes x_i. Finally, an edge-labeled join-graph is minimal if no variable can be removed from any label while maintaining the connectedness property.

Join-Graph Structuring Bucket elimination (BE) is a join-tree decomposition algorithm and its space time complexity is exponential in the induced-width of a problem (Dechter 1999). BE processes variable elimination in the following steps. First, collect all functions in a bucket with the variable to be eliminated by an elimination order. Then, combine functions in a bucket, marginalize the combined function by the elimination variable associated with the bucket, and send a message to the next bucket to be processed. Mini-bucket elimination (Dechter and Rish 2003) is a bounded inference scheme that controls the space and time complexity of variable elimination by limiting the maximum number of variables in each bucket to be less than the bound-

ing parameter $i+1$, called i-$bound$. Given an influence diagram, a minimal join-graph can be structured by schematic mini-bucket elimination with an i-$bound$ (Dechter, Kask, and Mateescu 2002). In this paper, each node $\mathcal{C}$ in the join-graph $\mathcal{G}_J$ is associated with a mini-bucket, $\chi(\mathcal{C})$ is a set of variables in the associated mini-bucket, and $\psi(\mathcal{C})$ is a set of functions allocated to the mini-bucket.

Generalized Dual Decompoistion for MMAP

Here, we briefly review the generalized dual decomposition for marginal MAP inference in graphical models. Given a graphical model $\mathcal{G} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, where $\mathbf{X} = \{X_i : i \in \mathcal{N}_V\}$ is a set of random variables index by an index set $\mathcal{N}_V$, $\mathbf{D} = \{\Omega_{X_i} : X_i \in \mathbf{X}\}$ is a set of finite domains of variables, and $\mathbf{F} = \{F_\alpha(\mathbf{X}_\alpha) : \alpha \in \mathcal{N}_F)\}$ is a set of discrete non-negative real-valued functions indexed by the set $\mathcal{N}_F$, where F_α is defined over a subset of variables $\mathbf{X}_\alpha \in \mathbf{X}$, called its $scope(F_\alpha)$. $\mathcal{G}$ defines a factorized distribution $P(\mathbf{X}) = \exp[\sum_{\alpha \in \mathcal{N}_F} \theta_\alpha(X_\alpha) - \Phi(\theta)]$, where $\theta_\alpha = \log(F_\alpha)$ and $\Phi(\theta) = \log \sum_{\mathbf{X}} P(\mathbf{X})$. The powered-sum elimination operator is defined by $\sum_x^w f(x) = [\sum_x |f(x)|^{\frac{1}{w}}]^w$, which is the generalization of summation ($w = 1.0$) and maximization ($w \to 0+$). Then, the marginal MAP inference task can be written as computing the weighted log partition function,

$$\Phi(\theta) = \log \sum_{X_n}^{w_n} \cdots \sum_{X_1}^{w_1} \exp[\sum_{\alpha \in \mathcal{N}_F} \theta_\alpha(\mathbf{X}_\alpha)], \quad (3)$$

where the weights w_i are zero for maximization variables and one for summation variables.

The generalized dual decomposition bound for MMAP (Ping, Liu, and Ihler 2015) bounds the weighted log partition function by generalization of Hölder's inequality (Liu and Ihler 2011),

$$\log \sum_{\mathbf{X}}^{\mathbf{w}} \prod_{\alpha \in \mathcal{N}_F} \exp[\theta_\alpha(\mathbf{X}_\alpha)] \leqslant \log \prod_{\alpha \in \mathcal{N}_F} \sum_{\mathbf{X}_\alpha}^{\mathbf{w}_\alpha} \exp[\theta_\alpha(\mathbf{X}_\alpha)], \quad (4)$$

where the non-negative weights $w_i \in \mathbf{w}$ distributed to the factors θ_α are summed to the original weight $w_i = \sum_{\{\alpha : X_i \in \mathbf{X}_\alpha\}} w_i^\alpha$. This bound can be tightened by dual decomposition (Sontag, Globerson, and Jaakkola 2011). Introducing cost-shifting functions $\delta_{(\alpha,\beta)}(\mathbf{X}_\alpha \cap \mathbf{X}_\beta)$ between a pair of factors $\theta_\alpha(\mathbf{X}_\alpha)$ and $\theta_\beta(\mathbf{X}_\beta)$, the weighted log partition function can be reparameterized by

$$\Phi(\theta) \leqslant \log \prod_{\alpha \in \mathcal{N}_F} \sum_{\mathbf{X}_\alpha}^{\mathbf{w}_\alpha} \exp[\theta_\alpha(\mathbf{X}_\alpha) + \sum_{\beta \in \mathcal{N}_F} \delta_{(\alpha,\beta)}(\mathbf{X}_\alpha \cap \mathbf{X}_\beta)], \quad (5)$$

where the cost-shifting functions between two factors cancels each other $\delta_{\alpha,\beta} + \delta_{\beta,\alpha} = 0$. Since the upper-bound of $\Phi(\theta)$ in equation (5) is convex with respect to the cost-shifting functions and weights, efficient optimization algorithms are available for tightening the upper bound.

Generalized Dual Decomposition for MEU

In this section, we show the generalized dual decomposition upper bound for the *MEU* of *ID*s and a gradient based local search algorithm for optimizing the bound.

Derivation of the GDD Bound for MEU

Let $\mathcal{N}_C$ be the set of clusters of a join-graph decomposition $\mathcal{D} = \langle \mathcal{G}_J, \chi, \psi \rangle$ of an influence diagram $\mathcal{M} = \langle \mathbf{C}, \mathbf{D}, \mathbf{P}, \mathbf{U}, T \rangle$ with a partial variable ordering $\mathcal{O} : \{\mathbf{I}_0 \prec D_0 \prec \cdots \prec \mathbf{I}_{T-1} \prec D_{T-1} \prec \mathbf{I}_T\}$. From equation (2), the combination of all valuations can be factorized over the join-graph $\mathcal{G}_J$ by $\Psi(\mathbf{C}, \mathbf{D}) = \otimes_{\mathcal{C} \in \mathcal{N}_C} \Psi_\mathcal{C}(\mathbf{X}_\mathcal{C})$, where $\Psi_\mathcal{C}(\mathbf{X}_\mathcal{C})$ is a local combination of valuations at each cluster $\mathcal{C} \in \mathcal{N}_C$, i.e., $\Psi_\mathcal{C}(\mathbf{X}_\mathcal{C}) = \otimes_{f_i \in \psi(\mathcal{C})} \Psi_{f_i}(\mathbf{X}_{f_i})$ with $\Psi_{f_i} = (f_i, 0)$ if $f_i \in \mathbf{P}$ and $\Psi_{f_i} = (1, f_i)$ if $f_i \in \mathbf{U}$. Following the definition of marginalization of a valuation of *ID*s, we define the powered-sum elimination operator for valuations as $\sum_\mathcal{O}^\mathbf{w}(p, u) := (\sum_\mathcal{O}^\mathbf{w} p, \sum_\mathcal{O}^\mathbf{w} u)$, where $\sum_\mathcal{O}^\mathbf{w}$ eliminates all the variables in $\mathcal{O}$ with the set of weights $\mathbf{w} = \{w_i : \forall i \in \mathcal{O}, w_i \geqslant 0\}$ associated with each variable. Then, the generalized decomposition bound of *ID*s is shown in the following theorem 1.

Theorem 1 (GDD Bound for *MEU*)**.** *Given an influence diagram $\mathcal{M}$ and its join-graph decomposition $\mathcal{D} = \langle \mathcal{G}_J, \chi, \psi \rangle$, the* MEU *can be bounded by the expected utility component of the combination of powered sum-marginalized valuations from each cluster $\mathcal{N}_C$,*

$$\sum_\mathcal{O}^\mathbf{w} (\otimes_{\mathcal{C} \in \mathcal{N}_C} \Psi_\mathcal{C}(\mathbf{X}_\mathcal{C}) \leqslant \otimes_{\mathcal{C} \in \mathcal{N}_C} \sum_\mathcal{O}^{\mathbf{w}^\mathcal{C}} \Psi_\mathcal{C}(\mathbf{X}_\mathcal{C}), \quad (6)$$

where $\mathbf{w}^\mathcal{C}$ is a set of non-negative weights distributed to the cluster $\mathcal{C}$ such that $w_i = \sum_{\mathcal{C} \in \mathcal{N}_C} w_i^\mathcal{C}$ and $0 \leqslant w_i^\mathcal{C} \leqslant 1$ for each variable $X_i \in \mathbf{C} \cup \mathbf{D}$.

Proof. Let $\Psi_{P_\mathcal{C}}(\mathbf{X}_\mathcal{C})$ and $\Psi_{EU_\mathcal{C}}(\mathbf{X}_\mathcal{C})$ denote a probability and expected utility component of valuation $\Psi_\mathcal{C}(\mathbf{X}_\mathcal{C})$ at cluster $\mathcal{C} \in \mathcal{N}_C$. The *MEU* can be bounded by applying absolute value inequality from (7) to (8), Minkowski's inequality from (8) to (9) and Hölder's inequality from (9) to (10) as shown in the following steps.

$$\begin{aligned}
& EU(\sum_\mathcal{O}^\mathbf{w} (\otimes_{\mathcal{C} \in \mathcal{N}_C} \Psi_\mathcal{C}(\mathbf{X}_\mathcal{C})) \\
& := \sum_\mathcal{O}^\mathbf{w} EU(\otimes_{\mathcal{C} \in \mathcal{N}_C} \Psi_\mathcal{C}(\mathbf{X}_\mathcal{C}) \\
& := \sum_\mathcal{O}^\mathbf{w} \sum_{i \in \mathcal{N}_C} \Psi_{EU_i}(\mathbf{X}_i) \prod_{j \in \mathcal{N}_C, j \neq i} \Psi_{P_j}(\mathbf{X}_j)) && (7) \\
& \leqslant \sum_\mathcal{O}^\mathbf{w} \sum_{i \in \mathcal{N}_C} |\Psi_{EU_i}(\mathbf{X}_i)| \prod_{j \in \mathcal{N}_C, j \neq i} \Psi_{P_j}(\mathbf{X}_j)) && (8) \\
& \leqslant \sum_{i \in \mathcal{N}_C} \sum_\mathcal{O}^\mathbf{w} |\Psi_{EU_i}(\mathbf{X}_i)| \prod_{j \in \mathcal{N}_C, j \neq i} \Psi_{P_j}(\mathbf{X}_j)) && (9) \\
& \leqslant \sum_{i \in \mathcal{N}_C} \sum_\mathcal{O}^{\mathbf{w}^i} |\Psi_{EU_i}(\mathbf{X}_i)| \prod_{j \in \mathcal{N}_C, j \neq i} \sum_\mathcal{O}^{\mathbf{w}^j} \Psi_{P_j}(\mathbf{X}_j)) && (10) \\
& = EU(\otimes_{\mathcal{C} \in \mathcal{N}_C} \sum_\mathcal{O}^{\mathbf{w}^\mathcal{C}} \Psi_\mathcal{C}(\mathbf{X}_\mathcal{C}))
\end{aligned}$$

□

Algorithm 1 Gradient Based Local Search GDD-ID(i)

Require: Influence diagram, $\mathcal{M} = \langle \mathbf{C}, \mathbf{D}, \mathbf{P}, \mathbf{U}, T \rangle$, elimination ordering $\mathcal{O}$, weights w_i associated with a variable $X_i \in \mathcal{O}$, i-bound, iteration limit M_{iter}, restarting limit M_{res}
Ensure: GDD upper bound L_{best} for *MEU*,
1: Find $\mathcal{D} = \langle \mathcal{G}_J(\mathcal{N}_C, \mathcal{N}_S), \chi, \psi \rangle$ with the input i-bound
2: Allocate Ψ_i to $\mathcal{G}_J$ by factor labeling function ψ of $\mathcal{D}$, where $\Psi_i \in \{(p_i, 0) \forall p_i \in \mathbf{P},\ (1, u_i) \forall u_i \in \mathbf{U}\}$
3: ASSIGN-UNIFORM-WEIGHT($\mathcal{G}_J$)
4: INITIALIZE-COST($\mathcal{G}_J$) ▷ Random or Minibucket cost shifting
5: $iter$=0, $L_{best} = \inf$, $L_{old} = \inf$.
6: **while** $iter < M_{iter}$ **do**
7: **for** $(i, j) \in \mathcal{N}_S$ **do** $L \leftarrow$ UPDATE-COST($\mathcal{G}_J$, (i, j))
8: **end for**
9: **for** $X_i \in \mathcal{O}$ **do** $L \leftarrow$ UPDATE-WEIGHT($\mathcal{G}_J$, X_i)
10: **end for**
11: **if** $L_{best} > L$ **then** $L_{best} \leftarrow L$
12: **else if** $abs(\frac{L_{old} - L}{L_{old}}) < \epsilon$ **then**
13: ASSIGN-UNIFORM-WEIGHT($\mathcal{G}_J$)
14: INITIALIZE-COST($\mathcal{G}_J$)
15: **end if**
16: $iter \leftarrow iter + 1$, $L_{old} \leftarrow L$
17: **end while**

Algorithm 2 Update-Cost by Randomized Gradient Descent

Require: Valuations at node i and j, Ψ_i, Ψ_j, initial step size η_0, final step size η_M iteration limit M_{iter},
1: **for** $(iter \leftarrow 0; iter < M_{iter}; iter{+}{+})$ **do**
2: Evaluate Gradient $\nabla L'$ by equation (14)
3: Interpolate gradient scaling factor $\eta \leftarrow f(iter, \eta_0, \eta_M)$
4: Add random noise to gradient $\nabla L' \leftarrow \nabla L' + \mathcal{U}$
5: $\delta_{U_{ij}} \leftarrow \delta_{U_{ij}} - \eta(\nabla L')$
6: $\Psi_{U_i} \leftarrow \Psi_{U_i} + \delta_{U_{ij}}, \Psi_{U_j} \leftarrow \Psi_{U_j} - \delta_{U_{ij}}$
7: **end for**

Note that, the upper bound in Theorem 1 bounds both joint probability and the total expected utility. The expected utility component of the bound can be obtained by combining local expected utilities of subproblems generated by the join-graph structuring process with i-*bound*, hence space and time complexity for computing the bound is exponential in i-*bound*. The complexity can be easily decreased by decreasing the i-*bound*, which also decrease the quality of the bound.

When combining the local expected utilities, the non-negative weights $\mathbf{w}$ are distributed to each cluster $\mathcal{C}$ by $\mathbf{w}^{\mathcal{C}}$. The upper bound of the total expected utility is the sum of the local expected utility $\sum_{\mathcal{O}}^{\mathbf{w}^{\mathcal{C}}} \Psi_{EU_{\mathcal{C}}}(\mathbf{X}_{\mathcal{C}})$ at each cluster $\mathcal{C}$ multiplied by marginalized probabilities of all other clusters $\prod_{i \in \mathcal{N}_C, i \neq \mathcal{C}} \sum_{\mathcal{O}}^{\mathbf{w}^i} \Psi_{P_i}(\mathbf{X}_i)$.

Cost Shifting Scheme for Tightening the Bound

Let $\mathcal{N}_S$ be the set of separators defined over the edges of the join-graph $\mathcal{G}_J(V, E)$ as $\{\mathcal{S}_{ij} : (i, j) \in E\}$, where each separator $\mathcal{S}_{ij}$ also defines the intersection of scopes at both clusters $\mathbf{X}_{ij} = \psi(i) \cap \psi(j)$. We can introduce arbitrary auxiliary valuations at the nodes adjacent to the separator $\mathcal{S}_{ij}$; $\delta_{ij}(\mathbf{X}_{ij})$ for cluster i and $\delta_{ji}(\mathbf{X}_{ij})$ for cluster j such that $\delta_{ij}(\mathbf{X}_{ij}) \otimes \delta_{ji}(\mathbf{X}_{ij}) = (1, 0)$. Then, the cost shifting scheme can be applied to the *GDD* bound for *MEU* shown in Theorem 1. The new bound incorporated with pairs of probability and expected utility functions $\delta_{ji}(\mathbf{X}_{ij})$ at each separator $\mathcal{S}_{ij}$ can be written as,

$$\sum_{\mathcal{O}}^{\mathbf{w}} (\otimes_{\mathcal{C} \in \mathcal{N}_C} \Psi_{\mathcal{C}}(\mathbf{X}_{\mathcal{C}}) \leqslant \otimes_{\mathcal{C} \in \mathcal{N}_C} \sum_{\mathcal{O}}^{\mathbf{w}^{\mathcal{C}}} \Psi'_{\mathcal{C}}(\mathbf{X}_{\mathcal{C}}), \tag{11}$$

where the cost shifted valuation at cluster $\mathcal{C}$ is combination of all the cost shifting valuations introduced at the cluster, $\Psi'_{\mathcal{C}}(\mathbf{X}_{\mathcal{C}}) = \Psi_{\mathcal{C}}(\mathbf{X}_{\mathcal{C}}) \otimes [\otimes_{\mathcal{S}_{\mathcal{C}j} \in \mathcal{N}_S} \delta_{\mathcal{C}j}(\mathbf{X}_{\mathcal{C}j})]$. Now, we set the reparameterized upper bound in equation (11) as an objective function to minimize,

$$L(\Delta, \omega) := \sum_{i \in \mathcal{N}_C} \sum_{\mathcal{O}}^{\mathbf{w}^i} \Psi'_{EU_i}(\mathbf{X}_i) \prod_{j \in \mathcal{N}_C, j \neq i} \sum_{\mathcal{O}}^{\mathbf{w}^j} \Psi'_{P_j}(\mathbf{X}_j). \tag{12}$$

The $L(\Delta, \omega)$ is a function of the set of all cost shifting functions $\Delta = \{\delta_{ij} : (i, j) \in E\}$ and the set of all weights $\omega = \{\mathbf{w}^{\mathcal{C}} : \mathcal{C} \in \mathcal{N}_C\}$.

Gradient based Optimization Algorithms

Algorithm 1 describes the outline of the gradient based local search for optimizing the bound $L(\Delta, \omega)$. First, it performs join-graph structuring with the input i-bound and decomposes the input influence diagram $\mathcal{M}$ to a graph of local clusters $\mathcal{N}_C$. Then, each function $f \in \mathbf{P} \cup \mathbf{U}$ is allocated to a cluster in $\mathcal{G}_J$ subject to the labeling function $\psi(f)$ of the join graph decomposition, and each weight $w_i^{\mathcal{C}}$ of the chance variable X_i at cluster $\mathcal{C}$ is assigned uniformly by $w_i^{\mathcal{C}} = \frac{1}{M}$, where the M is the number of clusters having X_i in its scope. Before calling the inner optimization procedures, we also initialize the bound by propagating mini-bucket messages or uniform random costs $\delta_{ij}(\mathbf{X}_{ij}), \forall \mathcal{S}_{ij} \in \mathcal{N}_S$ between cluster i and j as random initialization step of the local search. The inner optimization loop uses block coordinate descent updates that will be described in the following part. After each step of local optimization, weights and costs are initialized again when the improvement is less than ϵ to ensure the exploration of other regions of the state space. We set $\epsilon = 1e^{-4}$ in the experiments.

Block Coordinate Descent For the efficiency and simplicity, we present inner optimization procedures based on the block coordinate descent, which divides the whole set of optimization variables of $L(\Delta, \omega)$ into set of costs Δ and weights ω. Hence, each set is updated while the other is fixed. While optimizing $L(\Delta, \omega)$ with respect to Δ, the local optimization routine **Update-Cost**($\mathcal{G}_J, (i, j)$) is called per edge $(i, j) \in \mathcal{G}_J$. Similarly, **Update-Weight**($\mathcal{G}_J, X_i$) is called per variable $X_i \in \mathcal{O}$ to optimize the bound with respect to the weights $\mathbf{w}_i^{\mathcal{C}}$ associated with X_i.

Updating Costs For the local updates involving a cost shifting pair $(\delta_{P_{ij}}, \delta_{EU_{ij}})$, we will restrict the form of pairs to be $(1, \delta_{U_{ij}})$ with introducing a new notation for the normalized utility component $\Psi_U(\mathbf{X}) := \Psi_{EU}(\mathbf{X}) / \Psi_P(\mathbf{X})$. Note, $\delta_{EU_{ij}} = \delta_{U_{ij}}$ when $\delta_{P_{ij}} = 1$. This choice of fixing

$\delta_{P_{ij}} = 1$ renders the nonconvex and complicated global objective function $L(\Delta, \omega)$ in equation (12) to a simpler convex local objective function $L'(\delta_{U_{ij}})$ as follows.

$$L'(\delta_{U_{ij}}) = \rho_i \sum_{\mathcal{O}}^{\mathbf{w}^i} \Psi_{P_i} |[\Psi_{U_i} + \delta_{U_{ij}}]| + \rho_j \sum_{\mathcal{O}}^{\mathbf{w}^j} \Psi_{P_j} |[\Psi_{U_i} - \delta_{U_{ij}}]|, \tag{13}$$

where $\rho_i = \sum_{\mathcal{O}}^{\mathbf{w}^i} \Psi_{EU_i} / \sum_{\mathcal{O}}^{\mathbf{w}^i} \Psi_{P_i}$. The gradient of L' with respect to $\delta_{U_{ij}}$ can be evaluated by

$$\frac{\partial L'}{\partial \delta_{U_{ij}}} = \rho_i \sum_{\mathbf{X}_i \setminus \mathbf{X}_{ij}} \frac{\Lambda_i(\Psi_{EU_i})}{\Psi_{U_i}} - \rho_j \sum_{\mathbf{X}_j \setminus \mathbf{X}_{ij}} \frac{\Lambda_j(\Psi_{EU_j})}{\Psi_{U_j}}, \tag{14}$$

where $\Lambda_i(Z_0(\mathbf{X}))$ is called a pseudo belief of the cluster i (Liu 2014). When evaluating a pseudo belief, the $Z_0(\mathbf{X})$ can be either probability component Ψ_{P_i} or expected utility component Ψ_{EU_i}. In equation (14), pseudo beliefs are evaluated by expected utility components. Let X_k be the k-th variable of $\mathcal{O}$ associated with w_k, and $X_{i:j}$ denotes the sequence of variables from the i-th variable to j-th variable in $\mathcal{O}$. Then, $Z_i(\mathbf{X}_{i+1:|\mathcal{O}|})$ is defined recursively by a partial powered-sum up to $X_{1:i}$ by $Z_i(\mathbf{X}_{i+1:|\mathcal{O}|}) = \sum_{X_i}^{w_i} Z_{i-1}(\mathbf{X}_{i:|\mathcal{O}|})$, and $\Lambda_i(Z_0(\mathbf{X})) = \prod_{k=1..n} [\frac{Z_{k-1}}{Z_k}]^{\frac{1}{w_k}}$.

In principle, any non-linear optimization routine could be used to minimize the bound. From experimental evaluation, we observed that the computation of gradient and objective function becomes numerically unstable when weights $w_i^{\mathcal{C}}$ are small or expected utility values are close to zero. In addition, the number of parameters in the cost shifting functions $\delta_{U_{ij}}$ increases exponentially in *i-bound*, which challenges sophisticated second-order optimization routines. Therefore, we modified gradient descent to perturb the analytic gradient with a random noise vector with varying step sizes as described in Algorithm 2. The modification from the standard gradient descent is in line 4, adding a random noise $\mathcal{U}$ to the gradient $\nabla L'$. Note that, Algorithm 2 evaluates the gradient only once in each iteration and skips testing the step size of usual gradient descent, thus it willing to accept inferior solutions. The step size of the randomized gradient η is determined by a step interpolation function $f(iter, \eta_0, \eta_M)$ which decreases η by $1/iter$. In the experiment, we set tuning parameters of f as $\eta_0 = 0.1$, $\eta_M = 1e^{-4}$, and $M_{iter} = 20$, and sampled noise from $\mathcal{U}(-1, 1)$.

Updating Weights For updating the weights associated with a chance variable X_i, we used exponentiated gradient descent algorithm (Ping, Liu, and Ihler 2015). The exponentiated gradient descent algorithm transfroms optimization with a set of constraints, $\sum w_i^j = 1$ and $0 \leqslant w_i^j$, to unconstrained by the following parameterization $w_i^{\mathcal{C}} = \exp(v_i^{\mathcal{C}}) / \sum_{\alpha \in \mathcal{N}_C, X_i \in \chi(\alpha)} \exp(v_i^{\alpha})$.

The gradient of $L(\Delta, \omega)$ with respect to a single weight $w_i^{\mathcal{C}}$ at cluster $\mathcal{C}$ can be evaluated by

$$\rho_{\mathcal{C}} H_{EU_{\mathcal{C}}}(X_i | \mathbf{X}_{i+1:|\mathcal{O}|}) + \sum_{j \in \mathcal{N}_C, j \neq i} \rho_j H_{P_{\mathcal{C}}}(X_i | \mathbf{X}_{i+1:|\mathcal{O}|}). \tag{15}$$

The equation (15) involves the entropy terms of pseudo marginals of expected utility and probability components $\sum_{\mathbf{X}_{1:i-1}} \Lambda_{\mathcal{C}}(\Psi_{EU_{\mathcal{C}}})$ and $\sum_{\mathbf{X}_{1:i-1}} \Lambda_{\mathcal{C}}(\Psi_{P_{\mathcal{C}}})$. Following the standard exponentiated gradient descent, all the weights are updated by

$$w_i^{\mathcal{C}} \leftarrow w_i^{\mathcal{C}} \exp(\eta [\frac{\partial L}{\partial w_i^{\mathcal{C}}} - \sum_{j \in \mathcal{N}_C, X_i \in \chi(j)} w_i^j \frac{\partial L}{\partial w_j^{\mathcal{C}}}]) \tag{16}$$

and normalized to ensure $\sum_{j \in \mathcal{N}_C, X_i \in \chi(j)} w_i^j = 1$. In the experiments, the step size η in equation (16) was found by the line search with Armijo rule (Nocedal and Wright 2006).

	N_s, A, N_o	T, N_v, N_f, w	T, N_v, N_f, w
POMDP1	3,3,2	5,23,28,12	10,43,53,23
POMDP2	4,4,1	5,29,34,13	10,54,64,22
POMDP3	5,7,3	5,50,60,24	10,95,115,42
MDP1	6,8,-	10,76,106,10	20,146,206,10
MDP2	10,6,-	10,120,150,14	20,230,290,15
MDP3	16,8,-	10,186,256,25	20,356,496,26

Table 1: Random MDP/POMDP Problem Statistics.

Experimental Evaluation

We evaluated the proposed gradient based local search GDD-ID(i) on influence diagrams generated from random factored MDP/POMDP problems with a finite step size T. For reference, we also solved the same problem instances with SARSOP (Kurniawati, Hsu, and Lee 2008) with discounting factor 0.999 and compared the expected utility by simulating the infinite horizon policy for the length T, providing a lower bound of the optimal finite horizon *MEU*.

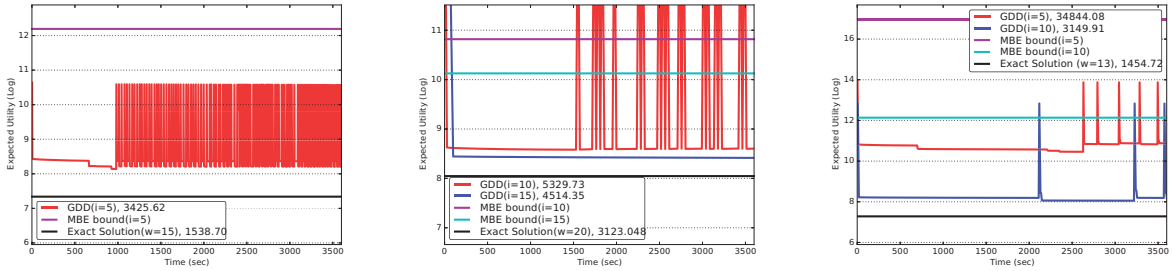
The experiment was designed to explore the quality of upper bounds by varying the i-bounds and the number of iterations on the random problem instances with controlled complexity. Table 1 summarizes the problem statistics of structure of factored MDP/POMDPs from which we generated 6 random instances. The first column characterizes the structure of MDP/POMDPs The N_s is the number of binary state variables, A is number of actions, and N_o is the number of binary observation variables. Influence diagrams were generated by unrolling MDP/POMDP for T time steps, N_v is the total number of variables N_f is the total number of functions, and w is the average constrained induced width. The problem statistics were chosen to reflect common benchmark instances. For the POMDP models, we matched (N_s, A, N_o) to produce models to reflect the sizes of common POMDP benchmark instances. The finite time steps were also chosen from common ranges shown in the probabilistic planning literature. The prototype of GDD-ID(i) is implemented in Python script language, we allowed 1 hour time limit, 1000 iteration limit, and provided *i-bound*s from 1, 5, 10, and 15.

Quality of GDD Bound for MEU

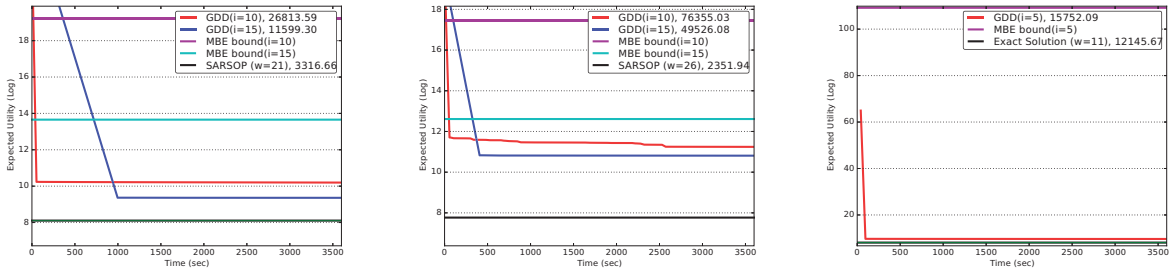
Table 2 reports the quality of the upper bound L_{gdd} for each problem instance. When problem instances can be solved exactly by variable elimination, the quality was measured by the ratio of the upper bound divided by the exact *MEU*, $Q^* = L_{gdd}/L^*$. If a problem instance cannot be solved

T		short (MDPs 10, POMDPs 5)						long (MDPs 20, POMDPs 10)					
Instance		0		1		2		0		1		2	
Quality		Q^*	Q_{mbe}	Q^*	Q_{mbe}	Q^*	Q_{mbe}	Q^*	Q_{mbe}	Q^*	Q_{mbe}	Q^*	Q_{mbe}
MDP	1	1.26	-42.71	1.31	-41.98	1.38	-37.8	1.3	-99.51	1.33	-83.55	1.45	-80.72
	2	1.47	-33.58	1.43	-22.51	1.62	-43.18	1.49	-68.22	1.44	-53	1.63	-91
	3	-	-103.37	-	-111.01	-	-106.15	-	-226.75	-	-210.8	-	-239.23
POMDP	1	2.23	-4.05	1.79	-1.50	2.19	-3.36	*4.06*	-4.16	1.45	-1.71	*1.88*	-0.76
	2	2.54	-1.84	2.17	-4.08	1.36	-1.08	*4.91*	-3.27	*1.42*	-4.50	*1352.56*	-4.30
	3	*3.28*	-2.29	*21.06*	-1.80	*17.28*	-1.99	*620.36*	-14.37	*1352.56*	-4.58	*5168.76*	-7.39

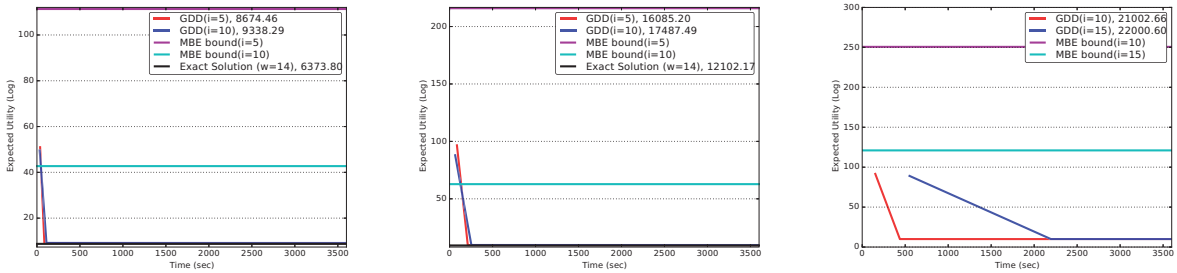
Table 2: Quality of GDD bound. Q^* is the ratio between the L_{gdd} and *MEU* (or lower bound of *MEU* simulated by SARSOP algorithm when exact solution is unknown.) Q_{mbe} is the improvement of the bound in **log scale**. Time steps are 10 and 20 for MDP instances, and 5 and 10 for POMDP instances.



(a) POMDP-1-0-5 T=5, w=15 (b) POMDP-1-1 T=10, w=20 (c) POMDP-2-1 T=5, w=13



(d) POMDP-2-2 T=10, w=21 (e) POMDP-2-2 T=10, w=26 (f) MDP-1-0 T=20, w=11



(g) MDP-2-0 T=10, w=14 (h) MDP-2-1 T=20, w=14 (i) MDP-3-1 T=10, w=15

Figure 1: Anytime convergence behavior of GDD-ID(i). Each subplot shows the trace of the GDD-ID(i) from the two largest i-*bound*s applicable to the problem instance. The upper horizontal lines are mini-bucket bounds found at the initialization step and the lower horizontal lines are the exact *MEU* or the lower bound for *MEU* simulated by SARSOP.

exactly, we used the sample average of the expected utility by simulating an infinite horizon policy by SARSOP (Kurniawati, Hsu, and Lee 2008) for 100 times. In the table, we show results from the maximum applicable i-*bound* tried. We also propagated mini-bucket elimination messages and uniform random costs at the initialization step.

$Q_{mbe} = \ln(L_{gdd}/L_{mbe})$ shows the improvement from the mini-bucket bound L_{mbe} to L_{gdd} in log scale. Compared with L_{mbe}, L_{gdd} presents significant improvements on all problem instances, and L_{gdd} is within a small constant factor from the optimal bound. The quality of the bound gets looser as the gap between the $i\text{-}bound$ and the induced width increases. In the case of instances generated from MDP-3 structure, GDD generated upper-bounds for the problems with $|S| = 2^{16}$ while variable elimination and SARSOP failed.

Convergence of GDD Bound for MEU

Figure 1 shows the anytime convergence behavior of L_{gdd} on several problem instances. We can observe that the higher $i\text{-}bound$ produces tighter upper bounds on all instances. However, the higher $i\text{-}bound$ generates larger scope sized functions and slows down the local updating routines. In the case of the instance in 1(b), GDD(i=15) iterated 51 times but GDD(i=10) iterated 381 times with 81 restarts. In all instances except 1(a) and 1(c), the gradient based local search almost converged to the local optimum after the first outerloop iteration. In 1(d), GDD(i=15) achieved $L_{gdd} = 11,674.47$ after the first iteration (took 996 seconds) while the L_{gdd} at the termination was only $11,599.30$. The spikes in 1(a), 1(b), and 1(c) show the restarting of the inner loop optimization. For the problem instance 1(c), GDD-ID(i=10) immediately improved after the first restarting at 2118 seconds. The bound before restarting was getting stuck at a local optima of around 3608 for almost 2000 seconds. However, restarting immediately produced a better $L_{gdd} = 3184$ only in two iterations. This instance illustrates the benefit of random restarting. Nevertheless, the rest of the 6 instances in Figure 1 spent the 1 hour time budget only for the first cycle due to the overhead of dealing with large scale gradient updates.

Conclusion and Future Work

In this paper, we proposed a new bounding scheme for the maximum expected utility of influence diagrams. First, we derived the bounding inequality that interchanges the combination and elimination operator for the valuation algebra for influence diagrams in Theorem 1. Then, we applied the generalized dual decomposition bound to the join-graph decomposition of influence diagram that decomposes a sequential decision problem into a collection of independent subproblems. The proposed scheme bounds the maximum expected utility by combining local expected utilities and tightens the bound by reparameterizing the normalized utility functions. We also demonstrated a gradient based local search algorithm by evaluating random MDP/POMDP instances.

From the experimental evaluation, we observed positive aspects of GDD-ID(i): (1) it converged to a local optimum in a small number of iterations with the noisy gradient update that does not require any function evaluations, (2) random restarting helped GDD-ID(i) to escape local optima, and (3) GDD-ID(i) can generate optimistic heuristics for problems with a large state space. However, there are several issues that need to be addressed in future. The tested GDD-ID(i) code implemented in Python language was not efficient enough to perform larger scale experiments that could empirically verify the positive sides of the algorithm.

The proposed scheme can be used as a new class of heuristic generators for the search based probabilistic planners. Since sequential decisions are localized to a set of independent clusters, it is interesting future work to extend the current GDD-ID(i) to guide online probabilistic planning in the form of dynamic heuristic. GDD-ID(i) produces not only the upper bound for the maximum expected utility but also generates a cost-optimized collection of sub-decision problems that can be immediately served as a sub-optimal strategy. This is an important aspect in probabilistic planning since the optimal strategy is exponential in the length of the history. GDD-ID(i) can generate a compressed strategy in a principled manner and the space complexity can be easily adjusted by the $i\text{-}bound$.

Acknowledgments

This work was supported in part by NSF grants IIS-1526842 and IIS-1254071, and by the US Air Force under Contract No. FA8750-14-C-0011 and FA9453-16-C-0508.

References

Babaki, B.; Guns, T.; and De Raedt, L. 2017. Stochastic constraint programming with and-or branch-and-bound. In *Proceeding of the 26th International Joint Conference on Artificial Intelligence*.

Cheng, Q.; Liu, Q.; Chen, F.; and Ihler, A. T. 2013. Variational planning for graph-based mdps. In *Advances in Neural Information Processing Systems 26*, 2976–2984.

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

Dechter, R.; Kask, K.; and Mateescu, R. 2002. Iterative join-graph propagation. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, 128–136.

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

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

Dechter, R. 2000b. A new perspective on algorithms for optimizing policies under uncertainty. In *Proceedings of the 5th Conference on Artificial Intelligence and Planning Systems*, 72–81.

Garcia-Sanchez, D., and Druzdzel, M. J. 2004. An efficient sampling algorithm for influence diagrams. In *Proceedings of the Second European Workshop on Probabilistic Graphical Models (PGM–04)*, 97–104.

Hansen, E. A.; Shi, J.; and Khaled, A. 2016. A pomdp approach to influence diagram evaluation. In *Proceeding of the 25th International Joint Conference on Artificial Intelligence*.

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*, 367–373.

Khaled, A.; Hansen, E. A.; and Yuan, C. 2013. Solving limited-memory influence diagrams using branch-and-bound search. In *Proceedings of The 29th Conference on Uncertainty in Artificial Intelligence*, 222–231.

Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Zurich, Switzerland.

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

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, 849–856.

Liu, Q., and Ihler, A. 2012. Belief propagation for structured decision making. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, 523–532.

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

Marinescu, R. 2010. *A New Approach to Influence Diagrams Evaluation*. Springer London. 107–120.

Mauá, D. D., and Cozman, F. G. 2016. Fast local search methods for solving limited memory influence diagrams. *Int. J. Approx. Reasoning* 68(C):230–245.

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

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.

Melibari, M. A.; Poupart, P.; and Doshi, P. 2016. Sum-product-max networks for tractable decision making. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 1846–1852.

Nocedal, J., and Wright, S. J. 2006. *Numerical Optimization, Second Edition*. Springer New York.

Ortiz, L. E., and Kaelbling, L. P. 2000. Sampling methods for action selection in influence diagrams. In *Proceedings of The 16th Conference on Uncertainty in Artificial Intelligence*, 378–385.

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

Pralet, C.; Schiex, T.; and Verfaillie, G. 2006. From influence diagrams to multi-operator cluster dags. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, 393–400.

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*, 169–198.

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

Tatman, J. A., and Shachter, R. D. 1990. Dynamic programming and influence diagrams. *IEEE transactions on systems, man, and cybernetics* 20(2):365–379.

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*, 691–700.