UNIVERSITY OF CALIFORNIA,
IRVINE

Decomposition Bounds for Influence Diagrams

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Junkyu Lee

Dissertation Committee:
Prof. Rina Dechter, Chair
Prof. Padhraic Smyth
Prof. Sameer Singh

2020

DEDICATION

To my wife, Younglim

To my daughters, Claire and Chloe

Contents

List of Figures

List of Tables

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

I am indebted to all those who have helped me finish this dissertation: my academic advisor, committee members, staff members of the ICS departments, colleagues, friends, and family.

I would like to express my very great appreciation to my academic advisor Professor Rina Decther for her endless support and advice during my M.S and Ph.D. program. Throughout her guidance, I was fortunate to have had a chance to learn the essence of automated reasoning and graphical models and study exciting topics advancing the frontier of AI research. Besides, I would like to thank her for the patience in times that allow me to spend a long time going back to basics. I am happy to be one of her students. I enjoyed the privilege of learning from her work directly, having discussions on the bigger pictures, and receiving critical comments. All those valuable times only improve the quality of our work.

I would like to offer my special thanks to my committee members, Professor Padharic Smyth and Sameer Singh, for allowing the defense in the middle of busy days. My special thanks are extended to the committe members for the Ph.D. candidacy exam, Professor Alexander Ihler, and John Turner.

I wish to acknowledge the help provided by Professor Alexander Ihler and Radu Marisuecu. The topic of my Ph.D. study divides into variational methods and search algorithms in graphical models. Professor Alexander Ihler provided me the necessary basics on variational methods throughout all the works. I am lucky to work with Radu Marinescu, a surprisingly productive and active researcher, and he gave me many lessons in search algorithms.

I wish to thank all the people in our research group. My work is heavily dependent on former students who graduated before I came into this group. Even if I didn't have a chance to meet all of them personally, I greatly appreciate their high-quality research outcomes that paved my way to study. In the earlier days of my Ph.D. program, it has been a joy to talk with Dr. Kalev Kask, Dr. Natasha Flerova, Dr. William Lam, Dr. Qi Lou, and other friends, Silu Yang, Zhengli Zhao, and specially Filjor Broka. In the latter days, we have new students Bobak Pezeshki, Yasaman Razeghi, and Sakshi Agarwal. My special thanks to Bobak for his positive moods and helps during writing my thesis and papers. It was also an enjoyable time to take a course together with Bobak and Yasaman.

Assistance provided by ICS Computing Support is greatly appreciated. The staff member of ICS Computing Support is one of the essential parts of our research activity. Thanks for the support to maintain our pedigree cluster and various instructions and troubleshooting that allowed me to use ICS open lab clusters to conduct numerical experiments on a larger scale. I cannot imagine that I could finish running all the experimental cases without Hans Wunsch and other staff members' support.

During the Ph.D. program, I had many opportunities to work as a teaching assistant for CS-171 introduction to AI with many professors, including Professor Richard Lathrop, Alexander Ihler, Eric Mjolsness, and Kalev Kask. I enjoyed all the discussions and course projects

together with students. My special thanks to Professor Richard Lathrop for nominating me as a TA several times and giving me valuable lessons.

Lastly, everything was possible only because of the endless support and love from my family. I thank my wife, Younglim, for everything. It is also the best part of my life to meet my daughters, Claire and Chloe, during the Ph.D. program. Special thanks to my wife's parents for caring for my wife and daughters. My parents and sister always support what I am doing, and this long time study was only possible because of their help.

I am grateful for the assistance I received for my graduate studies partially from the following funding sources: NSF grants (IIS-1526842, IIS-1254071, IIS-1065618), the U.S. Air Force (FA9453-16-C0508, FA8750-14-C-0011), and DARPA (PPAML Program, W911NF-18-C-0015). I also appreciate being a recipient of the Otto W. Shaler Scholarship for Spring 2019 and travel funds from Bren School of ICS and the association of graduate students.

CURRICULUM VITAE

Junkyu Lee

EDUCATION

Doctor of Philosophy in Computer Science **2021**
University of California, Irvine *Irvine, California*

Master of Science in Computer Science **2014**
University of California, Irvine *Irvine, California*

Master of Science in Engineering **2006**
Korea Advanced Institute of Science and Technology *Daejeon, Republic of Korea*

Bachelor of Science in Physics **2004**
Korea Advanced Institute of Science and Technology *Daejeon, Republic of Korea*

RESEARCH EXPERIENCE

Graduate Research Assistant **2014 – 2020**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Teaching Assistant **2014 – 2020**
University of California, Irvine *Irvine, California*

PROFESSIONAL EXPERIENCE

Research Intern **June 2019 – September 2019**
Big Data Experience Lab, Adobe Research *San Jose, California*

Senior Researcher **2006 – 2012**
Agency for Defense Development *Daejeon, Republic of Korea*

PUBLICATIONS

Junkyu Lee, Radu Marinescu, and Rina Dechter.
"Submodel Decomposition Bounds for Influence Diagrams" to appear in Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2021.

Radu Marinescu, Junkyu Lee, and Rina Dechter.
"A New Bounding Scheme for Influence Diagrams" to appear in Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2021.

Junkyu Lee, Radu Marinescu, and Rina Dechter.
"Heuristic AND/OR Search for Solving Influence Diagrams (Extended Abstract)" in Proceedings of the Symposium on Combinatorial Search (SoCS), 2020.

Junkyu Lee.
"Submodel Decomposition for Solving Limited Memory Influence Diagrams (Student Abstract)" in Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2020.

Junkyu Lee, Radu Marinescu, and Rina Dechter.
"A Weighted Mini-Bucket Bound for Solving Influence Diagram" in Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), 2019.

Radu Marinescu, Junkyu Lee, Rina Dechter, and Alexander Ihler.
"AND/OR Search for Marginal MAP" Journal of Artificial Intelligence Research (JAIR) volume 63, 2018.

Junkyu Lee, Alexander Ihler, and Rina Dechter.
"Case Study for Handling Hybrid Influence Diagrams in Probabilistic Programming Language Figaro with Discrete Algorithms" UCI ICS Technical Report, 2018.

Junkyu Lee, Alexander Ihler, and Rina Dechter.
"Join Graph Decomposition Bounds for Influence Diagrams" in Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), 2018.

Junkyu Lee.
"Probabilistic Planning with Influence Diagrams (Doctoral Consortium)" in Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2018.

Junkyu Lee, Alexander Ihler, and Rina Dechter.
"Generalized Dual Decomposition for Bounding Maximum" in AAAI Workshop on Planning and Inference, 2018.

Radu Marinescu, Junkyu Lee, Alexander Ihler, and Rina Dechter.
"Anytime Best+Depth-First Search for Bounding Marginal MAP" in Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2017.

Junkyu Lee, Radu Marinescu, and Rina Dechter.
"Applying Search Based Probabilistic Inference Algorithms to Probabilistic Conformant Planning: Preliminary Results" in Proceedings of International Symposium on Artificial Intelligence and Mathematics (ISAIM), 2016.

Junkyu Lee, Radu Marinescu, and Rina Dechter.
"From Exact to Anytime Solutions for Marginal MAP" in Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2016.

Junkyu Lee, Radu Marinescu, and Rina Dechter.
"Applying Marginal MAP Search to Probabilistic Conformant Planning: Initial Results" in the International Workshop on Statistical Realtional AI (STARAI) 2014.

Junkyu Lee, William Lam, and Rina Dechter.
"Benchmark on DAOOPT and GUROBI with the PASCAL2 Inference Challenge Problems" in NIPS Workshop on Discrete and Combinatorial Problems in Machine Learning (DISCML) 2013.

Junkyu Lee.
"Collaborative Maintenance Simulation System Using Virtual Mockup" Journal of Korea Multimedia Society volume 15, issue 2, 2012.

Jaehwan Kim, Junkyu Lee, and Joonhyuk Kang.
"A Novel Spectrum Sharing Scheme Using Relay Station with Intelligent Reception" International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, 2006.

Kwanghoon Kim, Junkyu Lee, Yusung Lee, Joonhyuk Kang, and Hyuncheol Park
"High Speed Transmission for a Coaxial Cable Based In-home Network" IEEE Transactions on Consumer Electronics volume 52, issue 2, 2006.

ABSTRACT OF THE DISSERTATION

Decomposition Bounds for Influence Diagrams

By

Junkyu Lee

Doctor of Philosophy in Computer Science

University of California, Irvine, 2020

Prof. Rina Dechter, Chair

Graphical models provide a unified framework for modeling and reasoning about complex tasks. Example tasks include probabilistic inference and sequential decision making under uncertainty. Influence diagrams (IDs) extend Bayesian networks with decision variables and utility functions to model the interaction between an agent and a system aiming to capture the agent's preferences. The standard task is to compute the maximum expected utility (MEU) over the influence diagram and a corresponding sequence of policies. However, finding the MEU is intractable, and it is one of the most challenging tasks in graphical models. The goal of this dissertation is the development of decomposition bounds for influence diagrams. Computing upper bounds on the MEU is desirable also because upper bounds can facilitate anytime-solutions by acting as heuristics to guide search or sampling-based methods. Most earlier approaches for upper bounding the MEU relied on polynomial reductions to other standard tasks in graphical models. However, such reductions are ineffective in practice since they highly inflate the size of the problems that needs to be solved. In this dissertation, we present two direct bounding methods for IDs. The first builds on the algebraic framework for influence diagrams, called valuation algebra. Using this framework, we extend two earlier decomposition bounds to the MEU task. The second method provides upper bounds by ex-

ploiting an exponentiated utility form of the MEU, which can be bounded by a conventional task (i.e., marginal MAP) over a standard probabilistic graphical model. This enables the use of two variational decomposition bounds for the well known marginal MAP task, yielding what we call exponentiated utility bounds for the MEU. For both cases (using valuation algebra and using exponentiated utilities), we present two kinds of message passing algorithms derived from bounding schemes on the marginal MAP task. One is the generalized dual decomposition algorithm, which propagates messages over a join-graph decomposition of an ID. The other is a weighted mini-bucket algorithm that propagates messages over the mini-bucket tree decomposition. We evaluated all the algorithms empirically and compared them against earlier approaches over four synthetic benchmark domains. The empirical evaluation results show that our direct bounding schemes generate upper bounds that are orders of magnitude tighter than previous methods. In addition, the exponentiated schemes turn out to be far more effective overall. We also evaluated our algorithms on the well-known system administration domain and observed that one of the exponentiated algorithms exhibited a remarkable performance generating upper bounds that are within a factor of two from the exact MEU.

Chapter 1

Introduction

Graphical models provide a unified framework for modeling complex systems and reasoning inference tasks such as probabilistic inference and decision making under uncertainty [Pearl, 1988, Lauritzen, 1996, Darwiche, 2009, Koller and Friedman, 2009, Dechter, 2013]. Probabilistic graphical models such as Bayesian networks (BNs) capture the local structure in probability models, and they offer the basis of analysis and design of graph-based algorithms. Influence diagrams (IDs) [Howard and Matheson, 1981] which extend Bayesian networks with decision variables and utility functions, are discrete graphical models of a single agent's sequential decision making under uncertainty. The standard inference task in IDs is computing the maximum expected utility (MEU) and finding an optimal set of decision rules that achieves this MEU. It is considered one of the most challenging tasks in graphical models.

Since its inception in the early 1980s, IDs have been a popular choice for modeling decision problems [Owens et al., 1997, Nielsen and Jensen, 2003, Sommestad et al., 2009, Carriger and Barron, 2011, Koenig and Matarić, 2017, Everitt et al., 2019]. IDs facilitate decision analysis paired with Bayesian analysis as shown in medical decision problems [Owens et al., 1997] and sensitivity analysis of the decisions subject to the changes in the model parameters [Nielsen and Jensen, 2003]. Due to its intuitive appeal we can see many applications of IDs in a variety of domains [Sommestad et al., 2009, Carriger and Barron, 2011, Koenig and

Matarić, 2017]. This popularity is partly because rational behavior can be formulated as an inference task in IDs [Russell and Norvig, 2010]. We also see a more recent application of IDs in modeling artificial general intelligence safety framework [Everitt et al., 2019] since IDs allow flexible structures and capture richer local structures compared with other frameworks such as Markov Decision Process.

Recent advances in graphical model inference algorithms mostly focus on developing new approximate inference schemes, often relying on reformulating inference tasks as optimization [Wainwright and Jordan, 2008], or improving the scalability and efficiency of algorithms to solve more practical and for larger scale problems by using search [Pearl, 1984, Hansen and Zhou, 2007]. However, in contrast to the popularity and success in advancing other probabilistic inference tasks, the MEU task has been less addressed. Often it is considered as a problem that can be solved through translation to other traditional well researched tasks, partly due to its close relationship to the marginal MAP task [Mauá, 2016].

Our primary focus in this thesis is on computing upper bounds on the MEU in IDs. The upper bound is desirable not only because it offers an approximate solution to the intractability of the MEU task, but also because we can use it as a building block in other algorithms such as heuristic search or in sampling-based schemes. As noted, earlier approaches for generating upper bounds on the MEU rely on its reduction to marginal MAP (MMAP). This however introduces a set of auxiliary variables and relations inflating the resulting problem size, as we will illustrate. In this thesis, we present two direct approaches for bounding MEU:

- Generalize the existing algebraic framework called valuation algebra for IDs with the powered-summation operation and extend variational decomposition bounds to this framework,
- Introduce a new exponentiated utility bound for IDs, and then bounding it using recently developed decomposition bounds.

In both approaches, namely that of extending decomposition bounds to valuation algebra and those decomposition bounds defined over the exponentiated utility bounds we present two types of message passing algorithms. The first are iterative algorithms that propagate messages over a join-graph decomposition [Mateescu et al., 2010] for IDs, and the second sends messages over a mini-bucket tree decomposition [Dechter and Rish, 2003] for IDs. We evaluate all the algorithms empirically and compared them against earlier approaches. The empirical evaluation results show that our direct bounding schemes generate upper bounds that are orders of magnitude tighter than earlier schemes, especially against translation based schemes.

Chapter 2 provides a background of earlier methods and notation conventions. Chapters 3 and 4 provide the contributions of this thesis. In the following subsections, we will detail the contributions in those two chapters.

1.1 Direct Decomposition Bounds for IDs

Chapter 3 focuses on developing direct decomposition bounds for the MEU task. One early work that yields direct bounds on the MEU is the mini-bucket elimination scheme [Dechter, 2000, Dechter and Rish, 2003]. Later, a translation method introduced by Liu and Ihler [2012] converts the MEU task into mixed inference task with the cost of introducing a selector variable having a large domain size. Mauá [2016] introduced a different translation showing the equivalence between the MEU task and MMAP task. In practice, the benefit of both reductions is outweighed by the cost caused by the inflation of the problem size as is clearly supported by our empirical evaluation. This observation motivated our development of direct decomposition bounds for IDs.

1.1.1 Bounding Schemes using Valuation Algebra for IDs

One difficulty in handling the MEU task is the presence of two kinds of combination operators. Namely, we have a multiplication operator applied to probability functions and a summation operator applied to utility or value functions. The valuation algebra processes pairs of probability and value functions using a single combination operator, thus simplifying the representation, inviting traditional schemes such as exact variable elimination, and decomposition bounds schemes. Specifically, using the valuation algebra [Jensen et al., 1994], we present bounding algorithms that apply generalized dual decomposition (GDD) [Liu and Ihler, 2012, Liu, 2014, Ping et al., 2015] and weighted mini-bucket elimination (WMBE) [Dechter, 1999, Dechter and Rish, 2003, Liu and Ihler, 2011, Ihler et al., 2012, Liu, 2014, Marinescu et al., 2014] schemes, developed for MMAP, to the MEU task.

Contributions

- We define the powered-summation operation for the valuation algebra for IDs.
- We generalize the decomposition bounds in probabilistic graphical models to the valuation algebra for IDs.
- We formulate an optimization problem over join-graph decomposition of IDs, yielding the algorithm JGD-ID applying the GDD scheme.
- We formulate an optimization problem over the mini-bucket tree decomposition of IDs, yielding algorithm WMBE-ID applying both GDD and the WMBE schemes.

1.1.2 Bounding Schemes using Exponentiated Utility Functions

While the valuation algebra simplifies representation and facilitates variable elimination procedures it is known to result in higher induced width due to using its pairs representation

over a constrained variable ordering. In addition, the optimization formulation over the valuation algebra yields a non-convex optimization problem that is much harder to solve.

Therefore, we moved back to developing bounding algorithms without using valuation algebra, by introducing the exponentiated utility bounds. Here also we present two algorithms that are built on top of the GDD scheme [Ping et al., 2015] and the weighted mini-bucket with moment matching (WMBMM) scheme [Marinescu et al., 2014].

Contributions

- We define the exponentiated ID, which exponentiates the sum of utility functions, yielding multiplicative exponentiated utility functions.
- Using the exponentiated ID we introduce a new bound to the MEU expressed as mixed inference.
- We combine the resulting exponentiated utility bound with decomposition bounds in a mixed inference task, allowing direct reuse of existing bounding algorithms
- We implement two bounding algorithms: JGD-EXP using GDD and WMBMM-EXP using WMBMM.

1.2 Empirical Evaluation of Decomposition Bounds for IDs

We dedicate Chapter 4 to the empirical evaluation of the proposed direct bounding algorithms, JGD-ID, WMBE-ID, JGD-EXP, and WMBMM-EXP. We also evaluate two reduction based approaches. The first approach uses the translation method by Liu and Ihler [2012], which yield mixed inference tasks to which we applied the GDD algorithm [Ping et al., 2015]. The second approach uses the translation by Mauá [2016], to which we applied WMBMM for MMAP task [Marinescu et al., 2014].

1.2.1 Benchmark Results from Synthetic Domains

We generated four synthetic benchmark domains including fully observable factored Markov Decision Process (MDP) problems, partially observable MDP problems, IDs with random graphs, and IDs modified from existing BNs. Each domain has instances with varying ranges of difficulties from easy to hard.

Contributions

- We generated new benchmark sets for evaluating algorithms over IDs.
- We illustrated the negative impact of reduction-based methods and in particular showed the extent to which such methods inflate problem sizes.
- We experimented with varying time and memory resource bounds and characterize the algorithms performance at different combinations of the resource limits.

1.2.2 A Case Study on System Administration Planning Domains

The system administration domains [Guestrin et al., 2003] are popular problem domains in probabilistic planning, and are used in the international planning competitions [Sanner et al., 2011]. We selected our best-performing algorithm, WMBMM-EXP and evaluated its power to yield upper bounds on a large scale, factored MDP/POMDP domains. Since the exact MEU is unknown for most of the problem instances, we evaluated using state-of-the-art online planners to obtain lower bounds of the MEU.

Contributions

- We converted into influence diagram framework system administration problems that are expressed in planning domain languages [Sanner, 2010].

- Our results show that the gap between the our upper bounds using our best direct decomposition schem and the lower bounds obtained from an online planner is at most 47 percent (within a factor of two).

Chapter 2

Background

This chapter reviews the background and basic notations. Section 2.1 introduces basic concepts in graphical models. Section 2.2 reviews decomposition methods in graphical models. Section 2.3 reviews variational decomposition bounds for generating upper bounds in probabilistic inference tasks. Section 2.4 reviews the basic concepts in influence diagrams (IDs) and earlier works for solving IDs.

2.1 Probabilistic Graphical Models

Graphs are the main language in graphical models. Graphs represent a model comprised of variables and relations, serve as a tool for analyzing the computational complexity and abstract schemes for solving inference tasks, and implementing data structures and algorithms.

We denote a set by the bold-face capital letters (e.g., $\mathbf{X}$, $\mathbf{D}$, $\mathbf{F}$), a set of indices of a set $\mathbf{S}$ by $\mathcal{I}_{\mathbf{S}}$, the number of elements of a set $\mathbf{S}$ by $|\mathbf{S}|$, a member of a set by the upper case letters often followed by its index in a set (e.g., $X_i \in \mathbf{X}$), a domain of a variable X_i by $\mathrm{dom}(X_i)$, an assignment of a value x_i to a variable X_i by $X_i = x_i$, an assignment of values $\mathbf{x}$ to a set of variables $\mathbf{X}$ by $\mathbf{X} = \mathbf{x}$ or $(X_1 = x_1, \ldots, X_N = x_N)$, and a scope of a function F_i by $\mathrm{sc}(F_i)$. Basic notations for graphs are given as follows. In a directed graph $\mathcal{G} := \langle \mathcal{V}, \mathcal{E} \rangle$ over a set of

nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$, we denote a set of parent nodes of Y by $\mathrm{pa}(Y)$, a set of child nodes of Y by $\mathrm{ch}(Y)$, a set of ancestor nodes of Y by $\mathrm{an}(Y)$, a set of descendant nodes of Y by $\mathrm{de}(Y)$, a family of a node of Y by $\mathrm{fa}(Y) := \mathrm{pa}(Y) \cup \{Y\}$, and a set of neighbor nodes of Y by $\mathrm{nhd}(Y) := \mathrm{pa}(Y) \cup \mathrm{ch}(Y)$. In an undirected graph, a set of neighbors of Y is the set of nodes adjacent to y. The above graph notations can be extended to a subset of nodes $\mathbf{Y}$ by taking the unions of the individual sets, e.g., $\mathrm{pa}(\mathbf{Y}) = \cup_{i \in \mathcal{I}_{\mathbf{Y}}} \mathrm{pa}(Y_i)$. While describing graph concepts in graphical models, we will interchange the notations for the variables and nodes when it is clear from the context.

Definition 2.1 (**Graphical Model $\mathcal{M}$**). *A* probabilistic graphical model $\mathcal{M}$ *is a tuple* $\langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$*, where*

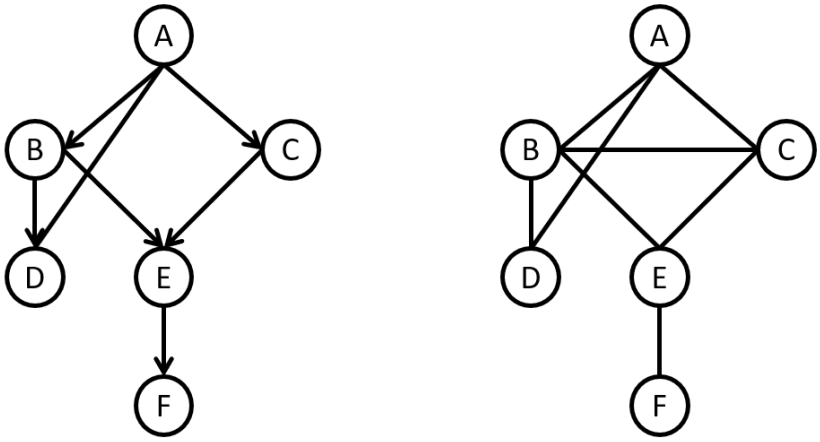
1. $\mathbf{X}$ *is a finite set of variables* $\{X_1, X_2, \ldots, X_n\}$*,*
2. $\mathbf{D}$ *is a set of finite domains of variables* $\{D_1, D_2, \ldots, D_n\}$ *s.t.* $\forall i \in \mathcal{I}_{\mathbf{X}}$ $D_i = dom(X_i)$
3. $\mathbf{F}$ *is a finite set of non-negative functions* $\{F_1, F_2, \ldots, F_m\}$ *s.t.* $\forall i \in \mathcal{I}_{\mathbf{F}}$ $sc(F_i) \subset \mathbf{X}$*.*

2.1.1 Bayesian Networks and Markov Networks

Bayesian networks and Markov networks are two popular probabilistic graphical models that each represents the local structure either by a directed acylic graph (DAG) or an undirected graph (UG).

Definition 2.2 (**Bayesian Networks $\mathcal{M}_{\mathrm{BN}}$**). *A Bayesian network* $\mathcal{M}_{BN} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ *is a graphical model with a set of conditional probability functions that factorizes the joint probability distribution* $P(\mathbf{X})$ *over DAG* $\mathcal{G}_{BN} := \langle \mathcal{V}, \mathcal{E} \rangle$ *by*

$$P(\mathbf{X}) = \prod_{X_i \in \mathbf{X}} P\big(X_i | pa(X_i)\big). \tag{2.1}$$



(a) Bayesian Network **(b)** Markov Network

Figure 2.1: Example of Bayesian Network and Markov Network

The $\mathcal{G}_{BN}$ represents a set of variables $\mathbf{X}$ by nodes $\mathcal{V}$, and the scope of each $P\big(X_i|pa(X_i)\big)$ by directed edges from $pa(X_i)$ to X_i.

Definition 2.3 (**Markov Networks $\mathcal{M}_{\text{MN}}$**). *A Markov network $\mathcal{M}_{MN} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ is a graphical model with a set of non-negative real valued functions that factorizes the join probability model $F(\mathbf{X})$ over an UG $\mathcal{G}_{MN} := \langle \mathcal{V}, \mathcal{E} \rangle$ by*

$$P(\mathbf{X}) = \frac{1}{Z} \prod_{F_i \in \mathbf{F}} F_i, \qquad\qquad Z = \sum_{\mathbf{X}} \prod_{F_i \in \mathbf{F}} F_i, \tag{2.2}$$

where Z is the normalization constant or partition function. The $\mathcal{G}_{MN}$ represents a set of variables $\mathbf{X}$ by nodes $\mathcal{V}$, and two nodes X_i and X_j are connected by an edge $(X_i, X_j) \in \mathcal{E}$ if they appear together in the scope of some function $F_i \in \mathbf{F}$.

Example 2.1. *Figure 2.1 illustrates an example of Bayesian network and Markov network. In the Bayesian network shown in Figure 2.1a, the nodes $\{A, B, C, D, E, F\}$ represents discrete random variables and the directed edges defines conditional probability functions that defines joint probability distribution*

$$P(A, B, C, D, E, F) = P(A)P(B|A)P(C|A)P(D|A, B)P(E|B, C)P(F|E). \tag{2.3}$$

In the Markov network shown in Figure 2.1b, the nodes $\{A, B, C, D, E, F\}$ *represents discrete random variables and the undirected edges define the scope of non-negative real-valued functions. Assuming all functions are defined over two variables the joint probability model can be written by*

$$F(A, B, C, D, E, F) = F_1(A, B) \cdot F_2(A, C) \cdot F_3(A, D) \cdot F_4(B, C) \cdot F_5(B, D) \cdot F_6(B, E) \cdot F_7(C, E) \cdot F_8(E, F). \quad (2.4)$$

Unlike Bayesian networks, the joint probability model in Markov network is not a probability distribution since it is not normalized. Therefore, we need to normalize $F(A, B, C, D, E, F)$ *by*

$$Z = \sum_{A,B,C,D,E} F(A, B, C, D, E, F)$$

to define a joint probability distribution

$$P(A, B, C, D, E, F) = \frac{1}{Z} F(A, B, C, D, E, F).$$

2.1.2 Inference Tasks in Probabilistic Graphical Models

For defining inference tasks, it is convenient to introduce the valuation algebra [Shenoy, 1997, Kohlas and Shenoy, 2000].

Definition 2.4 (**Valuation**). *Given a set of variables* $\mathbf{X}' \subset \mathbf{X}$*, a valuation* $\Psi_{\mathbf{X}'}$ *is the set of the values from all the possible assignments to the variables* $\mathbf{X}'$*. For the set of all variables* $\mathbf{X}$*,* $\Psi_{\mathbf{X}}$ *is the set of all valuations* $\Psi_{\mathbf{X}'}$ *over all possible subsets* $\mathbf{X}' \subset \mathbf{X}$*, and we denote the set of all valuations over any* $\mathbf{X}' \subset \mathbf{X}$ *by*

$$\mathbf{\Psi} := \cup_{\mathbf{X}' \subset \mathbf{X}} \Psi_{\mathbf{X}'}.$$

We say that $\mathbf{X}'$ *is the domain of* $\Psi_{\mathbf{X}'}$*,*

$$dom(\Psi_{\mathbf{X}'}) = \mathbf{X}'.$$

The set of all possible domains of the valuations is the power set of $\mathbf{X}$ *denoted by*

$$\mathbb{D} := 2^{\mathbf{X}}.$$

The combination operation $\otimes$ *is a binary operation that maps a pair of valuations to another valuation*

$$\otimes : \mathbf{\Psi} \times \mathbf{\Psi} \to \mathbf{\Psi},$$

and the projection operation $\Downarrow_{\mathbf{X}'}$ *projects the domain of a valuation by a mapping*

$$\Downarrow_{\mathbf{X}'}: \mathbf{\Psi} \times \mathbb{D} \to \mathbf{\Psi}, \quad s.t. \quad dom(\Downarrow_{\mathbf{X}'} \Psi) = \mathbf{X}'.$$

Definition 2.5 (**Valuation Algebra**). *A system of valuations* $\Upsilon := \langle \mathbf{\Psi}, \mathbb{D}, \otimes, \Downarrow \rangle$ *over the set of all valuations* $\mathbf{\Psi} := \{\Psi_{\mathbf{X}'_i} | \mathbf{X}'_i \in \mathbb{D}\}$ *with the combination operation* $\otimes$ *and the projection operation* $\Downarrow$ *defines a valuation algebra if it satisfies the following axioms [Kohlas and Shenoy, 2000].*

1. *Semigroup:* $\mathbf{\Psi}$ *is associative and commutative under the combination operation* $\otimes$*.*

2. *Domain of combination: For* $\Psi_{\mathbf{X}'_i}, \Psi_{\mathbf{X}'_j} \in \mathbf{\Psi}$*,*

$$dom(\Psi_{\mathbf{X}'_i} \otimes \Psi_{\mathbf{X}'_j}) = dom(\Psi_{\mathbf{X}'_i}) \cup dom(\Psi_{\mathbf{X}'_j}). \tag{2.5}$$

3. *Marginalization: For* $\Psi_{\mathbf{X}'_i} \in \mathbf{\Psi}$ *and* $\mathbf{X}'_j \in \mathbb{D}$,

$$\Downarrow_{\mathbf{X}'_j} \Psi_{\mathbf{X}'_i} = \Downarrow_{\mathbf{X}'_i \cap \mathbf{X}'_j} \Psi_{\mathbf{X}'_i}, \qquad dom(\Downarrow_{\mathbf{X}'_j} \Psi_{\mathbf{X}'_i}) = \mathbf{X}'_i \cap \mathbf{X}'_j, \qquad \Downarrow_{\mathbf{X}'_i} \Psi_{\mathbf{X}'_i} = \Psi_{\mathbf{X}'_i}. \tag{2.6}$$

4. *Transitivity of marginalization: For* $\Psi \in \mathbf{\Psi}$,

$$\Downarrow_{\mathbf{X}'_i} \left(\Downarrow_{\mathbf{X}'_j} \Psi \right) = \Downarrow_{\mathbf{X}'_i \cap \mathbf{X}'_j} \Psi. \tag{2.7}$$

5. *Distributivity of Marginalization over Combination: For* $\Psi_{\mathbf{X}'_i}, \Psi_{\mathbf{X}'_j} \in \mathbf{\Psi}$,

$$\Downarrow_{\mathbf{X}'_i} \left(\Psi_{\mathbf{X}'_i} \otimes \Psi_{\mathbf{X}'_j} \right) = \Psi_{\mathbf{X}'_i} \otimes \left(\Downarrow_{\mathbf{X}'_i} \Psi_{\mathbf{X}'_j} \right). \tag{2.8}$$

6. *Neutrality: For* $\mathbf{X}'_i, \mathbf{X}'_j \in \mathbb{D}$,

$$e_{\mathbf{X}'_i} \otimes e_{\mathbf{X}'_i} = e_{\mathbf{X}'_i \cup \mathbf{X}'_j}, \tag{2.9}$$

where $e_{\mathbf{X}'}$ *is a neutral element such that* $\Psi \otimes e_{\mathbf{X}'} = e_{\mathbf{X}'} \otimes \Psi = \Psi$ *for all valuations* $\Psi \in \Psi_{\mathbf{X}'}$.

Let $\mathbb{F}$ denote a set of all real valued factors defined over the set of discrete random variables $\mathbf{X}$ in $\mathcal{M}$. Then, the tuple $\langle \mathbb{F}, \mathbb{D}, +, \times \rangle$ is a semi-ring with addition $+$ and multiplication $\times$ operations that forms a valuation algebra [Shenoy, 1989, Kohlas and Shenoy, 2000, Kohlas and Wilson, 2008].

Proposition 2.1 (**Valuation Algebra over Discrete Factors**). *The tuple* $\langle \mathbb{F}, \mathbb{D}, \bigotimes, \Downarrow \rangle$ *over a probabilistic graphical model* $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ *is a valuation algebra with the following operations.*

1. *projection:* $\Downarrow_{\mathbf{Y}} F := \sum_{sc(F) \setminus \mathbf{Y}} F$,
2. *combination:* $F_i \bigotimes F_j := F_i \times F_j$.

For a probabilistic graphical model, the combination of all factors $F_i \in \mathbf{F}$ defines a joint probability model

$$F := \bigotimes_{F_i \in \mathbf{F}} F_i = \prod_{F_i \in \mathbf{F}} F_i. \tag{2.10}$$

We define inference tasks over the probabilistic graphical models by using the type of projection operators applied to the functions. The projection operator in Proposition 2.1 can be replaced by other operations than summation such as maximization, minimization, or the powered-summation that generalizes others.

Definition 2.6 (**Powered-summation**). *The powered-summation operation* $\sum_X^w$ *is defined by*

$$\sum_X^w F(\mathbf{X}) = \Big[\sum_X |F(\mathbf{X})|^{1/w} \Big]^w,$$

where the variable X *associated with a weight* $0 < w \leq 1$.

For a non-negative function, the powered summation reduces to the summation, maximization, and minimization when $w = 1$, $\lim_{w \to 0+} \sum_X^w F(\mathbf{X})$, and $\lim_{w \to 0-} \sum_X^w F(\mathbf{X})$, respectively.

In probabilistic inference, the summation task computes a marginal probability by marginalizing out a subset of variables, the normalization constant of an unnormalized distribution, or the probability of evidence by marginalizing out a subset of variables after assigning evidence to the observed variables.

Definition 2.7 (**Summation Task**). *Given a probabilistic graphical model* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ *defining a joint probability model* $F(\mathbf{X})$*, the summation task marginalizes out a subset of variables* $\mathbf{Y} \subset \mathbf{X}$ *from the scope of the joint distribution by*

$$\sum_{\mathbf{Y}} \prod_{F_i \in \mathbf{F}} F_i. \tag{2.11}$$

The optimization task seeks the extreme values of a function either by maximization or minimization depending on the context. In this thesis, we assume all optimization tasks maximize their respective objective function.

Definition 2.8 (**Maximization Task**). *Given a probabilistic graphical model* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ *defining a joint probability model* $\boldsymbol{f}(\mathbf{X})$*, the optimization task is defined by*

$$\max_{\mathbf{X}} \prod_{F_i \in \mathbf{F}} F_i, \tag{2.12}$$

and finds an optimal assignment

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{X}} F(\mathbf{X}). \tag{2.13}$$

The marginal MAP (MMAP) task generalizes the maximization task by allowing a subset of the variables to be marginalized.

Definition 2.9 (**Marginal MAP Task**). *Given a probabilistic graphical model* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ *defining a joint probability model* $F(\mathbf{X})$*, the marginal MAP task first marginalizes out the subset of variables by the summation, then performs the maximization task,*

$$\max_{\mathbf{X}_M} \sum_{\mathbf{X}_S} F(\mathbf{X}), \tag{2.14}$$

where $\mathbf{X}_M$ *is the maximization variables and* $\mathbf{X}_S = \mathbf{X} \setminus \mathbf{X}_M$ *is the summation variables. The optimal assignment* $\mathbf{x}_M^*$ *to the maximization variable is defined by*

$$\mathbf{x}_M^* = \operatorname*{argmax}_{\mathbf{X}_M} \sum_{\mathbf{X}_S} F(\mathbf{X}). \tag{2.15}$$

The mixed inference task also involves the summation operation and the maximization operation as the MMAP task, and we often use the MMAP task and the mixed inference task

interchangeably when it is clear from the context.

Definition 2.10 (**Mixed Inference Task**). *Given a probabilistic graphical model* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ *defining a joint probability model* $F(\mathbf{X})$*, the mixed inference task marginalizes out variables from the joint distribution by alternating* $\max$ *and* $\sum$ *operations subject to a partial elimination order* $\mathcal{O} := \{\mathbf{X}_{M_0}, \mathbf{X}_{S_0}, \mathbf{X}_{M_1}, \mathbf{X}_{S_1}, \ldots, \mathbf{X}_{M_N}, \mathbf{X}_{S_N}\}$*,*

$$\max_{\mathbf{X}_{M_0}} \sum_{\mathbf{X}_{S_0}} \cdots \max_{\mathbf{X}_{M_N}} \sum_{\mathbf{X}_{S_N}} F(\mathbf{X}), \tag{2.16}$$

where $\mathbf{X}_{M_k}$ *is a subset of variables eliminated by the maximization, and* $\mathbf{X}_{S_k}$ *is a subset of variables eliminated by the summation. The partial order* $\mathcal{O}$ *is the order of applying the respective sum and max projections, which do not commute. We define the function defining the optimal assignments for the set of variables* $\mathbf{X}_{M_k}$ *given each assignment to the previous variables* $\mathbf{x}^*_{M_0}, \ldots, \mathbf{x}^*_{S_{k-1}}$ *by*

$$\Delta^*(\mathbf{X}_{M_k} | \mathbf{X}_{M_0}, \mathbf{X}_{S_0}, \ldots, \mathbf{X}_{S_{k-1}}) = \operatorname*{argmax}_{\mathbf{X}_{M_k}} \sum_{\mathbf{X}_{S_k}} \cdots \sum_{\mathbf{X}_{M_N}} F\big(\mathbf{X}_{M_0} = \mathbf{x}^*_{M_0}, \ldots, \mathbf{X}_{S_{k-1}} = \mathbf{x}^*_{S_{k-1}}\big). \tag{2.17}$$

Theses can be called mixed best cost-to-go function.

2.1.3 Graph Separation and Independence Models

Given a graphical model $\mathcal{M}$, dependency relations between sets of variables can be assessed by the graph separation [Pearl, 1988].

Definition 2.11 (**Conditional Independence**). *Given a probability distribution* P*, we denote that a set of variables* $\mathbf{X}$ *and* $\mathbf{Y}$ *are conditionally independent given* $\mathbf{Z}$ *by* $(\mathbf{X} \perp \mathbf{Y} | \mathbf{Z})_{\mathcal{M}}$*.*

Definition 2.12 (**Separation in a Graph**). *By denoting* $(\mathbf{X} \perp \mathbf{Y} | \mathbf{Z})_{\mathcal{G}}$*, a set of nodes* $\mathbf{Z}$ *separates* $\mathbf{X}$ *and* $\mathbf{Y}$ *in* $\mathcal{G}$ *in a directed or undirected graph* $\mathcal{G}$ *if every path between nodes from* $\mathbf{X}$ *to* $\mathbf{Y}$ *is disconnected in the graph or blocked by nodes in* $\mathbf{Z}$*, as defined shortly.*

In Bayesian networks, separation is defined by the d-separation criterion over the DAG. It was shown that the d-separation captures conditional independence between variables in the presence of evidence in the given Bayesian network [Pearl, 1988].

Definition 2.13 (d-**Separation**). *Given a DAG $\mathcal{G}$, a path p is d-separated by a set of nodes $\mathbf{Z}$ if and only if*

1. *p contains a chain $i \to m \to j$ or a fork $i \leftarrow m \to j$ such that $m \in \mathbf{Z}$, or*
2. *p contains a collider $i \to m \leftarrow j$ such that $m \notin \mathbf{Z}$ and $de(m) \cap \mathbf{Z} = \emptyset$.*

A set $\mathbf{Z}$ d-separates $\mathbf{X}$ for $\mathbf{Y}$ if and only if $\mathbf{Z}$ d-separates every path from a node in $\mathbf{X}$ to a node in $\mathbf{Y}$. In undirected Markov networks $\mathcal{G}_{\text{MN}}$, a set $\mathbf{Z}$ separates $\mathbf{X}$ for $\mathbf{Y}$ if and only if $\mathbf{Z}$ disconnects every path from a node in $\mathbf{X}$ to a node in $\mathbf{Y}$.

The connection between a probability model $\mathcal{M}$ and a graph $\mathcal{G}$ is formalized by the notion of dependency and independency maps.

Definition 2.14 (**I-map**). *We say a graph $\mathcal{G}$ is an I-map (Independence map) of a probability model $\mathcal{M}$ if any graph separation $(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z})_{\mathcal{G}}$ implies conditional independence $(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z})_{\mathcal{M}}$. If removing an edge in $\mathcal{G}$ renders $\mathcal{G}$ not being an I-map, the $\mathcal{G}$ is minimal.*

Definition 2.15 (**D-map**). *We say a graph $\mathcal{G}$ is a D-map (Dependency map) of a probability model $\mathcal{M}$ if a conditional independence $(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z})_{\mathcal{M}}$ implies graph separation $(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z})_{\mathcal{G}}$.*

Definition 2.16 (**P-map**). *We say a graph $\mathcal{G}$ is a P-map (Perfect map) of a probability model $\mathcal{M}$ if it is both I-map and D-map.*

Remarkable results in graphical models make connections between two concepts, the one on the conditional independence assertions in a probability model and the other on the separation in a primal graph [Pearl, 1988].

Definition 2.17 (**Primal Graph $\mathcal{G}_{\mathrm{P}}$**). *A primal graph of a graphical model $\mathcal{M}_G$ is an undirected graph of nodes representing the variables. An edge connects two nodes if they appear in the scope of a function.*

Markov networks represent its structure by primal graphs, whereas the DAG of a Bayesian network can be transformed into a primal graph by moralizing the DAG.

Definition 2.18 (**Moral Graph $\mathcal{G}_{\mathrm{M}}$**). *A moral graph of a directed graph is an undirected graph obtained by processing the directed graph by*

1. *connecting all the parents of a node, and*
2. *remove the direction of all arcs.*

Then, the following theorems state that a primal graph $\mathcal{G}_{\mathrm{P}}$ of a probabilistic graphical model $\mathcal{M}$ indeed captures some conditional independence assertions in P.

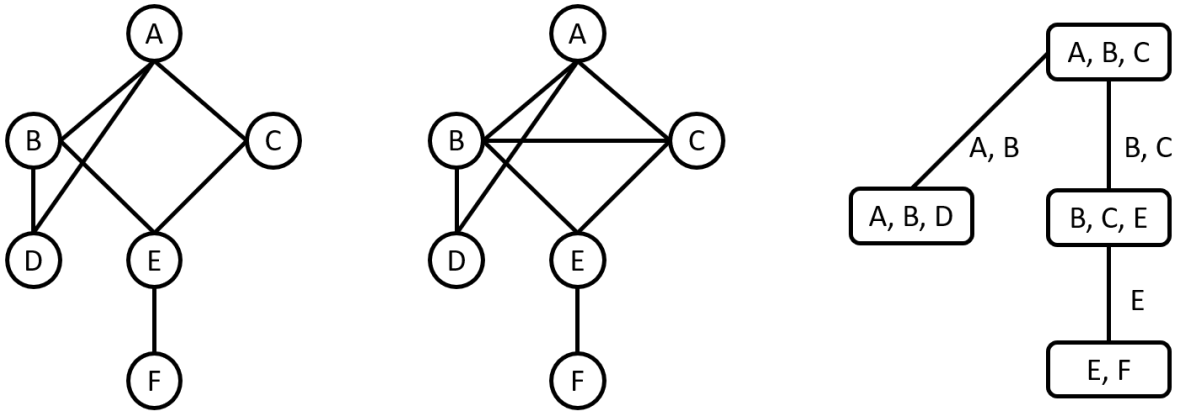
Theorem 2.1 (**Soundness of the graph separation**). *Let $\mathbf{p}(\mathbf{X})$ be a joint distribution of a probabilistic graphical model $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ either a Bayesian network or a Markov network. Then, the primal graph $\mathcal{G}_P$ of $\mathcal{M}$ is an I-map of $\mathcal{M}$.*

Theorem 2.2 (**Hammersley-Clifford** [Hammersley and Clifford, 1970]). *If $\mathbf{p}$ is a positive distribution, and $\mathcal{G}$ is an I-map of $\mathbf{p}$, then $\mathbf{p}$ can be factorizes over the cliques of $\mathcal{G}$.*

Now, we define a decomposable model that leads to the join-tree decomposition and exact algorithms for the inference tasks [Lauritzen and Spiegelhalter, 1988, Pearl, 1988].

Definition 2.19 (**Chordal Graph**). *A graph $\mathcal{G}$ is chordal if every cycle of length four or more has at least one chord.*

Definition 2.20 (**Decomposable Model**). *A graphical model $\mathcal{M}_G$ is decomposable if it has a graph $\mathcal{G}$ that is a minimal I-map and chordal. In this case, we say the probability model is decomposable relative to the chordal graph $\mathcal{G}$.*



(a) Primal Graph **(b)** Chordal Graph **(c)** Join Tree

Figure 2.2: Illustration of Graphs Related to Join-Tree.

Definition 2.21 (**Join-tree**). *For a chordal graph $\mathcal{G}$, a join-tree $\mathcal{T}_{JT} := \langle \mathcal{C}, \mathcal{S} \rangle$ is a tree with the cliques of $\mathcal{G}$ as its nodes $\mathcal{C}$ such that any two cliques $C_i, C_j \in \mathcal{C}$ containing a variable X are connected by a path of clique nodes that contain the variable X, which is called* running intersection property.

Theorem 2.3. *If a probability model P is decomposable relative to a chordal graph$\mathcal{G}$ and $\mathcal{T}_{JT} := \langle \mathcal{C}, \mathcal{S} \rangle$ is a join-tree of $\mathcal{G}$, then the joint distribution can be written as a product of the distributions of the cliques of $\mathcal{G}$ divided by a product of the distributions of their intersections.*

$$P(\mathbf{X}) = \prod_{C_i \in \mathcal{C}} \frac{P(\mathbf{X}_{C_i})}{P(\mathbf{X}_{C_i} \cap \mathbf{X}_{C_{j(i)}})}, \tag{2.18}$$

where C_i is a clique of nodes associated with the variables $\mathbf{X}_{C_i}$, and $C_{j(i)}$ is the parent of C_i in a tree traversal order [Lauritzen and Spiegelhalter, 1988, Pearl, 1988].

Example 2.2. *Figure 2.2 illustrates graphs related to a join-tree shown in Definition 2.21. Figure 2.2a shows a primal graph over the random variables $\{A, B, C, D, E, F\}$, and Figure 2.2b triangulates the primal graph by adding an edge (B, C). Figure 2.2c shows a join-tree of the chordal graph in Figure 2.2b.*

2.2 Decomposition Methods in Graphical Models

Exact inference algorithms in graphical models solve inference tasks by decomposing the input graphical model into a cluster tree, and propagating messages between clusters. The tree decomposition leads to approximate graph decomposition schemes by introducing relaxation to the decomposed tree structure, and the graph decomposition schemes provide a framework for developing approximate inference algorithms.

2.2.1 Join-tree Decomposition

The joint distribution of a probabilistic graphical model can be factorized over a join-tree due to Theorem 2.3. If a primal graph $\mathcal{G}_{\mathrm{P}}$ is not a chordal graph, we introduce additional edges to make it chordal. A join-tree generated by a chordal graph of a $\mathcal{G}_{\mathrm{P}}$ provides a decomposable probability model that can be parameterized over a tree, which allows application of dynamic programming for inference tasks [Dechter, 1999]. We next review the definitions related to join-tree decomposition [Dechter, 2013].

Definition 2.22 (**Join-Tree Decomposition**). *We define a join tree decomposition of a probabilistic graphical model* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ *with a primal graph* $\mathcal{G}_P$ *by a tuple* $\mathcal{T}_{JT} := \langle T(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$, *where* $T(\mathcal{C}, \mathcal{S})$ *is a tree,* χ *and* ψ *are labeling functions which associate each node* $C \in \mathcal{C}$ *to a set of variables by* $\chi(C)$ *and a set of functions by* $\psi(C)$. *For any two nodes* C_i *and* C_j, $\psi(C_i) \cap \psi(C_j) = \emptyset$. *For the variables assigned to a node* C, $\chi(C)$ *covers all the scope of functions* $\psi(C)$, *i.e.,* $\cup_{i \in \mathcal{I}_{\psi(C)}} sc(F_i) \subseteq \chi(C)$. *Lastly, the join tree* T *satisfies the running intersection property.*

Definition 2.23 (**Separator**). *For a join tree decomposition* $\mathcal{T}_{JT} := \langle T(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$, *the separator of cluster nodes* C_i *and* C_j *in a join tree* T *is* $sep(C_i, C_j) := \chi(C_i) \cap \chi(C_j)$, *or* S_{C_i, C_j} *for short. We also define eliminator of* C_i *with respect to* C_j *by* $elim(C_i, C_j) = \chi(C_i) \setminus \chi(C_j)$.

Definition 2.24 (**Width of Tree Decomposition**). *The tree width of a tree decomposition*

$\mathcal{T}_{JT} := \langle T(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ *is defined by the maximum cluster size,*

$$\max_{C \in \mathcal{C}} |\chi(C)| - 1.$$

The separator width of a tree decomposition is the maximum separator size.

Definition 2.25 (**Tree-width w^* of $\mathcal{G}$**). *For an undireced graph $\mathcal{G}$, we define induced width relative to the order $\mathcal{O}$ as the maximum clique size - 1 of its triangulated graph and denote it by $w(\mathcal{O})$. The tree width w^* of $\mathcal{G}$ is the smallest induced width along all orderings $\mathcal{O}$.*

We will often interchange the terms such as nodes in a join-tree to clusters, and join-tree decomposition to cluster-tree. A tree decomposition can be generated from any variable order by triangulating the primal graph along that ordering. For an inference task with a constrained elimination ordering, the constraints restrict the variable ordering to be consistent, and we denote an induced width from constrained ordering by $w_c(\mathcal{O})$.

2.2.2 Tree Clustering Schemes for Inference Tasks

Tree clustering schemes [Kask et al., 2005, Dechter, 2013] offer procedures for generating a join-tree decomposition from a primal graph, and solving inference tasks by propagating messages over a cluster tree. The join-tree decomposition also provides useful invariants for the message-passing algorithms, which characterize the solution of the inference tasks.

Suppose we have an inference task with a partial constrained elimination ordering, and a graphical model $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ with a primal graph $\mathcal{G}_{\mathrm{P}}$. Bucket elimination algorithm [Dechter, 1999] processes variables by a total order $\mathcal{O}$ that is compatible with the constrained ordering specified by the inference task.

Definition 2.26 (**Bucket and its Scope**). *A bucket $\mathbf{B}_{X_i}$ has its key associated with each*

variable $X_i \in \mathbf{X}$ *and collects functions that have* X_i *in its scope,*

$$\mathbf{B}_{X_i} := \{F_i | F_i \in \mathbf{F},\ X_i \in sc(F_i)\}. \tag{2.19}$$

We define the scope of a bucket $\mathbf{B}_{X_i}$ *as the union of the scopes of all functions in its bucket,*

$$sc(\mathbf{B}_{X_i}) := \cup_{F_j \in \mathbf{B}_{X_i}} sc(F_j) \tag{2.20}$$

Starting from the first variable in $\mathcal{O} := \{X_1, \ldots, X_{|\mathbf{X}|}\}$, the bucket elimination algorithm [Dechter, 1999] creates a bucket $\mathbf{B}_{X_i}$ and generates a new function, called message, by eliminating variable X_i from the combined function at $\mathbf{B}_{X_i}$,

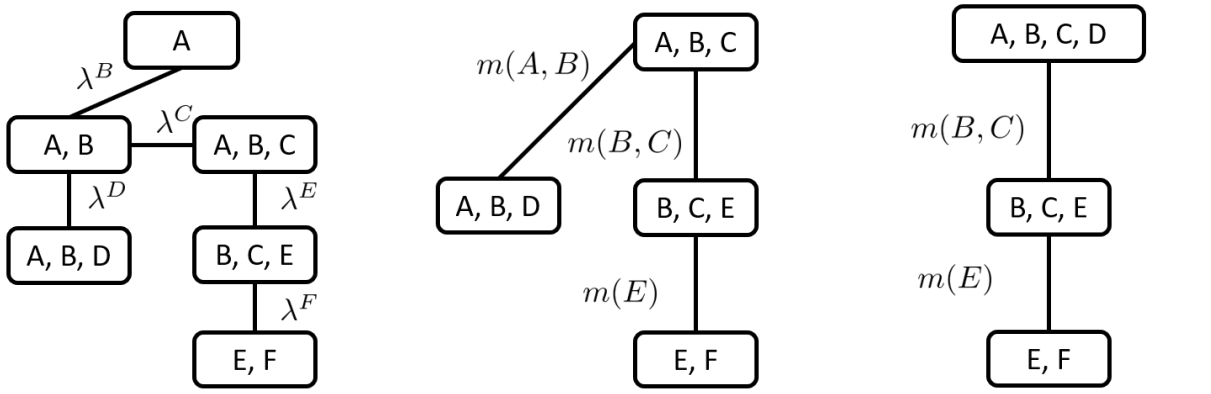
$$\lambda^{X_i} = \Big\Downarrow_{\mathrm{sc}(\mathbf{B}_{X_i}) \setminus \{X_i\}} \Big(\bigotimes_{F_j \in \mathbf{B}_{X_i}} F_j \Big). \tag{2.21}$$

Let's consider a tree of buckets connected by the edges associated with the messages. Then, the buckets $\mathbf{B}_{X_i}$ and $\mathbf{B}_{X_j}$ are connected by an edge if λ^{X_i} is placed at $\mathbf{B}_{X_j}$. After processing all variables, the combination of all the constant messages is the solution of the inference task, and the bucket-tree is a valid join-tree.

Theorem 2.4 (**Complexity of Bucket Tree Elimination Algorithm**). *The time complexity of the bucket tree elimination algorithm is* $O(n_f \cdot deg \cdot k^{(w^*+1)})$ *and the space complexity is* $O(n \cdot k^{w^*})$*, where* n *is the number of variables,* n_f *is the number of functions deg is the maximum degree of the bucket tree,* k *is the maximum domain size, and* w^* *is the tree width.*

Example 2.3. *Figure 2.3 illustrates an example of bucket tree. Suppose the set of functions in the graphical models is*

$$\mathbf{F} = \{F_1(A), F_2(A,B), F_3(A,C), F_4(A,B,D), F_5(B,C,E), F_6(E,F)\}$$



(a) Bucket Tree **(b)** Cluster Tree with w=2 **(c)** Cluster Tree with w=3

Figure 2.3: Example of Bucket-Tree and Cluster-Tree.

and the inference task is eliminating variables by $\max_A \sum_B \max_C \sum_D \max_E \sum_F \mathbf{F}(\mathbf{X})$ *subject to a total variable elimination order* $\mathcal{O} = \{F \prec E \prec D \prec C \prec B \prec A\}$. *The bucket tree elimination algorithm processes variables in the order appear in* $\mathcal{O}$. *In Figure 2.3a, The first bucket to process is* $\mathbf{B}_F = \{F_6(E, F)\}$, *and the message from* $\mathbf{B}_F$ *to* $\mathbf{B}_E$ *is* $\lambda^F = \sum_F F_6(E, F)$. *The next bucket to process is* $\mathbf{B}_E = \{\lambda^F(E), F_5(B, C, E)\}$, *and* $\lambda^E = \max_E[\lambda^F(E) \cdot F_5(B, C, E)]$. *Continuing the variable elimination process following the total order* $\mathcal{O}$ *to the end, we obtain the solution of the inference task and the bucket tree as shown in Figure 2.3a.*

Tree clustering for solving an inference task is not unique as shown in Figure 2.3b. Another cluster-tree can be obtained by merging cluster nodes as long as it satisfies Definition 2.22. Merging every cluster C_i to its adjacent cluster C_j when its scope $\chi(C_i)$ is a subset of $\chi(C_j)$, we can find a minimal tree decomposition.

Definition 2.27 (**Minimal Tree Decomposition**). *We say a tree decomposition with a cluster tree* $\mathcal{T}_{CT} := \langle \mathcal{C}, \mathcal{S} \rangle$ *is minimal if* $sep(C_j, C_j) \not\subset \chi(C_i)$ *and* $sep(C_i, C_j) \not\subset \chi(C_j)$ *for* $(C_i, C_j) \in \mathcal{S}$.

Cluster trees shown in Figure 2.3b and 2.3c are minimal cluster trees because none of the

clusters subsumes to others. Cluster tree elimination algorithm [Kask et al., 2005, Dechter, 2013] eliminates more than one variable while computing a message between clusters in a cluster-tree $T(\mathcal{C}, \mathcal{S})$ by

$$m_{C_i \to C_j} := \Big\Downarrow_{\mathbf{X}_{S_{C_i,C_j}}} \Big(\bigotimes_{R \in \text{nhd}(C_i), R \neq C_j} m_{R \to C_i} \Big) \otimes \Big(\bigotimes_{i \in \psi(C_i)} F_i \Big). \tag{2.22}$$

Following a similar process illustrated in Example 2.3, we obtain cluster trees shown in Figure 2.3b and 2.3c. Until now, we implicitly assumed that the message propagation follows the order of underlying total order $\mathcal{O}$ that generated the tree decomposition. In general, the message propagation over a cluster tree could be processed in asynchronous manner until convergence as long as the elimination order along the message propagation remains valid in the inference task. Concretely, each cluster C_i sends a message shown in Eq. (2.22) to all adjacent cluster nodes C_j by pulling incoming messages from its neighbors except for the message from C_j.

Definition 2.28 (**Cluster Tree Calibration**). *We say that the cluster tree is calibrated when the messages are converged.*

Now, we review properties of the calibrated cluster-tree for the summation task.

Proposition 2.2. *Given a graphical model with a joint probability model $F(\mathbf{X}) = Z \cdot P(\mathbf{X})$, the calibrated cluster-tree reparameterizes the cluster tree $\mathcal{T}_{CT} := \langle \mathcal{C}, \mathcal{S} \rangle$ by*

$$F(\mathbf{X}) := \frac{\prod_{C_i \in \mathcal{C}} b(\mathbf{X}_{C_i})}{\prod_{(C_i,C_j) \in \mathcal{S}} b(\mathbf{X}_{S_{C_i,C_j}})} = \frac{\prod_{C_i \in \mathcal{C}} \left[F_{C_i}(\mathbf{X}_{C_i}) \cdot \prod_{C_j \in nhd(C_i)} M_{C_j,C_i}(\mathbf{X}_{S_{C_j,C_i}}) \right]}{\prod_{(C_i,C_j) \in \mathcal{S}} M_{C_i,C_j}(\mathbf{X}_{S_{C_i,C_j}}) \cdot M_{C_j,C_i}(\mathbf{X}_{S_{C_i,C_j}})}, \tag{2.23}$$

where $b(\mathbf{X}_{C_i})$ and $b(\mathbf{X}_{S_{C_i,C_j}})$ are unnormalized marginal probability functions or beliefs.

Therefore, the marginal probability $P(\mathbf{X}')$ of a subset of random variables $\mathbf{X}' \subset \mathbf{X}$ can be computed from the belief at calibrated cluster C if $\mathbf{X}' \subseteq \text{sc}(\mathbf{X}_C)$. In addition, the calibrated

cluster tree satisfies consistency relations over the separators,

$$b(\mathbf{X}_{S_{C_i,C_j}}) = \sum_{\text{elim}(C_i,C_j)} b(\mathbf{X}_{C_i}) = \sum_{\text{elim}(C_j,C_i)} b(\mathbf{X}_{C_j}). \tag{2.24}$$

We see that tree clustering schemes reparameterizes the joint probability model $F(\mathbf{X})$ to a decomposable model over over a cluster tree $\mathcal{T}_{\text{CT}}$, and the task of computing marginal probability can be performed locally to each cluster node once the tree is calibrated.

Finally, the complexity of cluster-tree elimination algorithms is also characterized by the graph-based parameter as follows.

Theorem 2.5 (**Complexity of Cluster-Tree Elimination Algorithm**). *The time complexity of the cluster tree elimination algorithm is* $O((n_f + n_c) \cdot deg \cdot k^{w^*+1})$ *and the space complexity is* $O(n_c \cdot k^{|sep|})$, *where* n_f *is the number of functions,* n_c *is the number of clusters, deg is the maximum degree of the cluster tree,* k *is the maximum domain size,* w^* *is the width of tree decomposition, and* $|sep|$ *is the separator width.*

2.2.3 Mini-bucket Tree and Join-graph Decomposition

The complexity of tree clustering schemes for inference tasks is intractable for practical problems, so designing approximate inference algorithms with a bounded complexity is a main research topic in graphical models. Our primary interest is in decomposition methods that generate upper bounds by graph decomposition schemes offering lower complexity structure with smaller clusters.

Approximate decomposition schemes such as mini-bucket elimination [Dechter and Rish, 2003] or join-graph decomposition [Mateescu et al., 2010] decompose a join-tree into possibly a loopy graph using the mini-bucket relaxations. Such mini-bucket based graph decomposition schemes can limit the maximum cluster size below a bounding parameter i-bound so

that the complexity of one iteration of message passing is bounded exponentially by the i-bound.

Consider a bucket $\mathbf{B}_X$ that collects functions having a variable X in its scope. Mini-bucket relaxation partitions the bucket to P mini-buckets $\{\mathbf{B}_X^1, \ldots, \mathbf{B}_X^P\}$ such that the i-bound limits the scope size of each mini-bucket by $|\mathrm{sc}(\mathbf{B}_X^p) \leq i + 1$. Then, the message λ^X from the bucket $\mathbf{B}_X$ can be bounded by the weighted mini-bucket relaxation [Dechter and Rish, 2003, Liu and Ihler, 2011].

Theorem 2.6 (Weighted Mini-bucket Relaxation)**.**

$$\lambda^X := \sum_X^{w_X} \Big[\prod_{i \in \mathbf{B}_X} F_i \Big] \leq \prod_{p=1}^{P} \Big[\sum_{X^p}^{w_{X^p}} \Big[\prod_{i \in \mathbf{B}_X^p} F_i \Big] \Big] \leq \prod_{p=1}^{P} \lambda^{X^p} \tag{2.25}$$

$$w_X = \sum_{p=1}^{P} w_{X^p}, \quad 0 < w_X, w_{X^p} \leq 1. \tag{2.26}$$

$$X = X^1 = X^2 = \cdots = X^P \tag{2.27}$$

If a variable X is eliminated by the maximization, we use $w_X = 0^+$, and all the weights $\{w_{X^1}, \ldots, w_{X^P}\}$ in Eq. (2.26) are arbitrary close to zero. Any combination of weights satisfying Eq. (2.26) generates a valid upper bound if X is eliminated by the summation, $w_X = 1$. Dechter and Rish [2003] presented the mini-bucket relaxation by the maximization and the summation operations, and Liu and Ihler [2011] generalized it to the weighted mini-buckets as shown in Eq. (2.25). Eq. (2.27) defines a set of equality constraints enforcing variables X at the bucket $\mathbf{B}_X$ and duplicated variables X^p at each mini-bucket $\mathbf{B}_{X^p}$ are the same. The idea of decomposing a coupled problem (bucket) into smaller problems (mini-buckets) appears frequently in the context of optimization, and we will revisit this mini-bucket relaxation from the optimization perspective later when we review variational decomposition bounds.

Algorithm 2.1 WMB(i): Weighted Mini-bucket Elimination

Require: Graphical model $\mathcal{M}_{\mathrm{G}} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, Total variable elimination ordering $\mathcal{O}$, i-bound
Ensure: Mini-bucket tree Decomposition $\mathcal{T}_{\mathrm{MB}} := \langle T(\mathcal{C}, \mathcal{E}), \chi, \psi \rangle$, Upper bound to a mixed inference task UB

1: UB $\leftarrow 1.0$ ▷ Initialize UB for the multiplicative functions
2: **for** each variable $X_i \in \mathcal{O}$ **do**
3: $\quad \mathbf{B}_{X_i} \leftarrow \{F_i | X_i \in \mathrm{sc}(F_i), F_i \in \mathbf{F}\}$ ▷ Collect functions to a bucket $\mathbf{B}_{X_i}$
4: $\quad \mathbf{F} \leftarrow \mathbf{F} \setminus \mathbf{B}_{X_i}$ ▷ Remove the functions assigned to bucket
5: $\quad$ Partition $\mathbf{B}_{X_i}$ to mini-buckets $\{\mathbf{B}_{X_i^1}, \dots \mathbf{B}_{X_i^P}\}$ such that $\max_p |\mathrm{sc}(\mathbf{B}_{X_i^p})| \leq i+1$
6: $\quad$ Assign positive weights $w_{X_i^p}$ to each variable X_i^p such that $w_X = \sum_{p=1}^{P} w_{X_i^p}$
7: $\quad$ **for** each mini-bucket $\mathbf{B}_{X_i^p} \in \{\mathbf{B}_{X_i^1}, \dots \mathbf{B}_{X_i^P}\}$ **do**
8: $\quad\quad$ Add a cluster node $\mathbf{B}_{X_i^p}$ to $\mathcal{T}_{\mathrm{MB}}$ ▷ Structure mini-bucket tree
9: $\quad\quad \psi(\mathbf{B}_{X_i^p}) \leftarrow \{F_i | F_i \in \mathbf{B}_{X_i^p}\}$ ▷ Update node labeling functions
10: $\quad\quad \chi(\mathbf{B}_{X_i^p}) \leftarrow \cup_{i \in \mathcal{I}_{\mathbf{B}_{X_i^p}}} \mathrm{sc}(F_i)$
11: $\quad\quad$ Connect two cluster nodes U, V in T if $\psi(V)$ contains the outgoing message from U
12: $\quad\quad \lambda^{X_i^p} \leftarrow \sum_{X_i^p}^{w_{X_i^p}} \left[\prod_{F_i \in \mathbf{B}_{X_i^p}} F_i \right]$ ▷ Compute weighted mini-bucket message
13: $\quad\quad$ **if** $\mathrm{sc}(\lambda^{X_i^p})$ is empty **then**
14: $\quad\quad\quad$ UB $\leftarrow$ UB $\cdot\, \lambda^{X_i^p}$ ▷ Multiply constant message to upper bound
15: $\quad\quad$ **else**
16: $\quad\quad\quad \mathbf{F} \leftarrow \mathbf{F} \cup \{\lambda^{X_i^p}\}$
return UB and message propagated mini-bucket tree $T(\mathcal{C}, \mathcal{E})$

By using the weighted mini-bucket relaxation, we can modify the bucket elimination algorithm to the weighted mini-bucket elimination in a straightforward manner. Algorithm 2.1 shows the weighted mini-bucket elimination algorithm for a mixed inference task with a combination operator $\otimes$ being the multiplication between functions. From line 3 to line 6, a bucket $\mathbf{B}_{X_i}$ is partitioned into P mini-buckets, and each mini-bucket cluster is added to a mini-bucket tree decomposition from line 8 to line 11. The weighted mini-bucket elimination algorithm computes the outgoing messages from the mini-buckets and accumulates constant messages to yield UB from line 12 to line 16, which we can skip computing the actual messages when structuring the mini-bucket tree decomposition. Since the size of all cluster scopes are bounded by i-bound, the space and time complexity for bounding the inference task by the mini-bucket elimination algorithm is exponential in the i-bound.

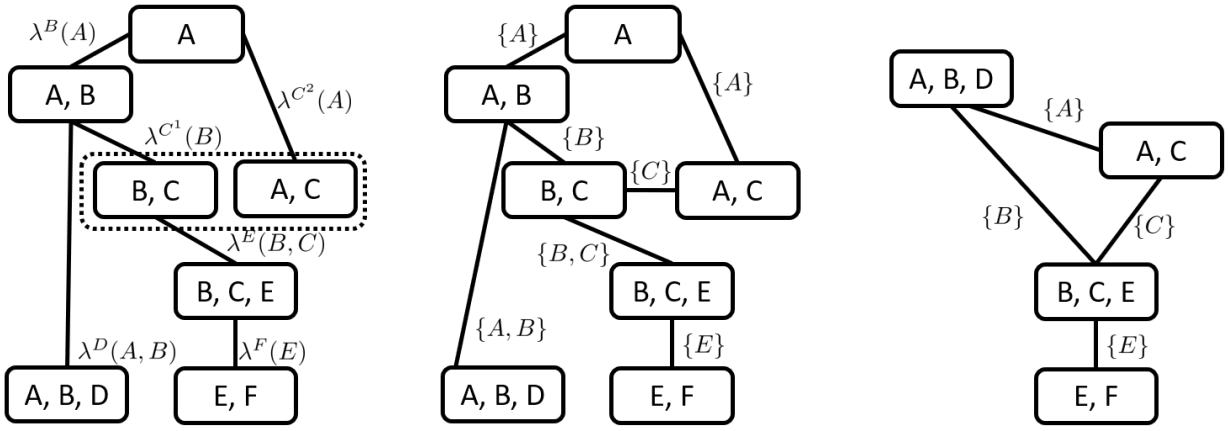
A join-graph decomposition [Mateescu et al., 2010] refines a join-tree into a join-graph with

smaller clusters.

Definition 2.29 (**Join-graph Decomposition** $\mathcal{G}_{\text{JG}}$). *A join-graph decomposition of a graphical model* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ *is a tuple* $\mathcal{G}_{JG} := \langle G, \chi, \psi \rangle$, *where* $G := \langle \mathcal{C}, \mathcal{S} \rangle$ *is a graph with nodes* $\mathcal{C}$ *and edges* $\mathcal{S}$, *and* χ *and* ψ *are labeling functions, where* χ *maps a node* $C \in \mathcal{C}$ *to a set of variables* $\chi(C) = \mathbf{X}_C \subset \mathbf{X}$, *and* ψ *allocates each function* $F_i \in \mathbf{F}$ *exclusively to a node* $C \in \mathcal{C}$ *such that* $sc(F_i) \subset \mathbf{X}_C$. *An edge* $(C_i, C_j) \in \mathcal{S}$ *is associated with a subset of variables shared between the two clusters* $\chi(C_i) \cap \chi(C_j)$, *called separator* $\mathcal{S}_{C_i,C_j}$. *The labeling function should ensure the running intersection property; for each variable* $X_i \in \mathbf{X}$, *the set* $\{C \in \mathcal{C} | X_i \in \psi(C)\}$ *induces a connected sub-graph.*

A valid join-graph can be systematically obtained from a mini-bucket tree by connecting mini-buckets $\{\mathbf{B}_{X_i^1}, \dots \mathbf{B}_{X_i^P}\}$ in a chain. The separators between mini-buckets of the same bucket $\mathbf{B}_X$ have a single variable $\{X\}$, and the scope of separators $\mathcal{S}_{C_i,C_j}$ is determined by the scope of the message sent from mini-bucket cluster C_i to C_j. The join-graph structured from a mini-bucket tree can further be simplified by merging adjacent clusters if the scope of a cluster subsumes the other. A nice property of a join-graph based on a mini-bucket tree is that the separators are *minimal* in the sense that removing any variable from a separator renders the join-graph decomposition invalid. The message propagation over a join-graph aims to provide an approximation that imrpoves over the belief propagation algorithm [Pearl, 1988]. However, it does not guarantee an upper bound to the optimal solution like the mini-bucket tree elimination. Yet, we can propagate messages in an iterative manner until convergence if it happens, or a time limit with space and time complexity for each iteration bounded exponentially by the i-bound.

Example 2.4. *Figure 2.4a shows an example of mini-bucket tree that approximate the bucket tree shown in Figure 2.3a. We can see that the bucket* $\mathbf{B}_C$ *is divided into two mini-buckets* $\{\mathbf{B}_{C^1}, \mathbf{B}_{C^2}\}$ *due to the i-bound 1. Figure 2.4b is a join-graph structured from the mini-bucket tree by adding a separator with single variable* $\{C\}$ *between clusters* $\mathbf{B}_{C^1}$ *and* $\mathbf{B}_{C^2}$



(a) Mini-bucket Tree (i=1) **(b)** Join-graph(i=1) **(c)** Simplified Join-graph

Figure 2.4: Example of Mini-bucket Tree and Join-graph Decomposition.

that introduces a cycle. Figure 2.4c is a simplified join-graph that merges clusters that can be subsumed to the adjacent ones.

The mini-bucket tree and join-graph decomposition provide a structural decomposition of an input graphical model for the approximate inference algorithms. In the following, we review variational inference framework that derives message passing algorithms sending messages over such graph decompositions. The message passing algorithms derived by variational inference use region graphs, which can be any graph reflecting the structure of the graphical model. The join-graph is a reasonable choice for the region graph since it allows a systematic construction procedure for generating higher-order region graphs that can improve the quality of approximation with increased i-bounds leading to anytime property.

2.3 Variational Decomposition Bounds

Variational inference is an optimization based approximate inference framework that greatly improves the quality of approximate solutions compared with naive message passing algorithms [Yedidia et al., 2001, Minka, 2001, Wainwright et al., 2003, Opper and Saad, 2001,

Wainwright and Jordan, 2008, Komodakis et al., 2010, Sontag et al., 2011]. The graph decomposition schemes such as the mini-bucket tree [Dechter and Rish, 2003] or the join-graph decomposition [Mateescu et al., 2010] provide a basis for approximate inference frameworks that can control the complexity of inference tasks by limiting the maximum cluster size [Dechter, 2013]. We can derive the upper bounds of the inference tasks in the form of message passing algorithms over the decomposed graph, and the variational inference framework tightens the gap between upper bounds and the optimal solution. The variational inference provides the primal and dual optimization view on the inference tasks [Wainwright and Jordan, 2008]. We say that the inference task is a primal problem when it is formulated in the natural parameter space. The dual problem can be obtained by taking conjugate dual or Lagrangian dual of the primal problem. Variational decomposition bounds develop upper bounds of inference tasks by combining two bounding frameworks defined in the primal and dual spaces. For example, decomposition schemes using the mini-bucket relaxation offer upper bounds in the primal problem. The conjugate dual formulation of the inference task also offers additional dimensions for tightening the upper bound. We now review the weighted mini-bucket elimination bounds [Liu and Ihler, 2011, 2012, Liu, 2014] and generalized dual decomposition bounds [Komodakis et al., 2010, Sontag et al., 2011, Ping et al., 2015] for the mixed inference task to provide background on the decomposition bounds for the MEU task in IDs in this thesis.

2.3.1 Exponential Family Representation

A discrete function $F_i \in \mathbf{F}$ in a discrete graphical model $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ has a tabular representation, and it can be represented by the exponential family form as follows.

Definition 2.30 (Discrete Functions in Exponential Family Form)**.** *A discrete function*

$F_i \in \mathbf{F}$ can be represented in exponential family form by

$$F_i(\mathbf{X}_{F_i}) = \exp\left[\sum_{\mathbf{x}_{F_i}} \theta_{\mathbf{x}_{F_i}} \mathbb{I}(\mathbf{X}_{F_i} = \mathbf{x}_{F_i})\right], \tag{2.28}$$

where $\mathbf{X}_{F_i}$ is the scope of the function F_i, and $\mathbf{x}_{F_i}$ is an assignment to $\mathbf{X}_{F_i}$. $\mathbb{I}(\mathbf{X}_{\boldsymbol{f}_i} = \mathbf{x}_{F_i})$ is the indicator function that returns 1 if $\mathbf{X}_{\boldsymbol{f}_i} = \mathbf{x}_{\boldsymbol{f}_i}$ and returns 0 otherwise, and $\theta_{\mathbf{x}_{F_i}} = \log F_i(\mathbf{X}_{F_i} = \mathbf{x}_{F_i})$.

The joint probability distribution $P(\mathbf{X}) = \frac{1}{Z}\prod_{F_i \in \mathbf{F}} F_i$ can be written as

$$P(\mathbf{X}) = \frac{1}{\sum_{\mathbf{X}} F(\mathbf{X})} \exp\left[\sum_{F_i \in \mathbf{F}} \sum_{\mathbf{x}_{F_i}} \theta_{\mathbf{x}_{F_i}} \mathbb{I}(\mathbf{X}_{F_i} = \mathbf{x}_{F_i})\right] = \exp\left[\sum_{F_i \in \mathbf{F}} \theta_i(\mathbf{X}_{F_i}) - \Phi(\boldsymbol{\theta})\right], \tag{2.29}$$

where $\theta_i(\mathbf{X}_{F_i}) = \log F_i(\mathbf{X}_{F_i})$, $\boldsymbol{\theta} = \{\theta_i | i \in \mathcal{I}_{\mathbf{F}}\}$, and $\Phi(\boldsymbol{\theta})$ is the log partition function,

$$\Phi(\boldsymbol{\theta}) := \log Z(\boldsymbol{\theta}) = \log \sum_{\mathbf{X}} \exp\left[\sum_{F_i \in \mathbf{F}} \theta_i(\mathbf{X}_{F_i})\right]. \tag{2.30}$$

While, we restrict our work to discrete random variables, many statements in this section hold for general exponential family distributions with minor modifications. In Eq. (2.29), the indicator functions $\mathbb{I}(\mathbf{X}_{F_i} = \mathbf{x}_{F_i})$ are the sufficient statistics. where each indicator function counts the occurrence of discrete events such as $\mathbf{X}_{F_i} = \mathbf{x}_{F_i}$ in the product space of the discrete random variables, and the coefficients $\theta_{\mathbf{x}_{F_i}}$ are called natural parameters.

The expectation of the sufficient statistics $\mathbb{I}(\mathbf{X}_{F_i} = \mathbf{x}_{F_i})$ over the joint distribution yields the first moment of the random variable that is the mean occurrence of $\mathbf{x}_{F_i}$,

$$\mu_{\mathbf{x}_{F_i}} := \mathbb{E}_{P(\mathbf{X})}\Big[\mathbb{I}(\mathbf{X}_{F_i} = \mathbf{x}_{F_i})\Big], \tag{2.31}$$

The first order moments can be generated by the gradients of the log partition function by

$$\frac{\partial \Phi(\boldsymbol{\theta})}{\partial \theta_{\mathbf{x}_{F_i}}} = \sum_{\mathbf{X}} \mathbb{I}(\mathbf{X}_{F_i} = \mathbf{x}_{F_i}) \frac{\exp\Big[\sum_{i \in \mathcal{I}_{\mathbf{F}}} \theta_i(\mathbf{X}_{F_i})\Big]}{Z(\boldsymbol{\theta})} = \mu_{\mathbf{x}_{F_i}}. \tag{2.32}$$

We denote a mean vector that concatenates all $\mu_{\mathbf{x}_{F_i}}$ in the $P(\mathbf{X})$ by $\boldsymbol{\mu}$, and the corresponding sufficient statistics vector by $\boldsymbol{\phi}(\mathbf{X})$ with its associated natural parameter vector by $\boldsymbol{\theta}$. Then,

$$\boldsymbol{\mu} = \mathbb{E}_{P(\mathbf{X})}\Big[\boldsymbol{\phi}(\mathbf{X})\Big], \quad P(\mathbf{X}) = \exp\Big[\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{X}) \rangle - \Psi(\boldsymbol{\theta})\Big], \tag{2.33}$$

where $\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{X}) \rangle$ denotes the inner product between $\boldsymbol{\theta}$ and $\boldsymbol{\phi}(\mathbf{X})$. The joint distribution parameterized by natural parameters $\boldsymbol{\theta} \in \Theta$ in Eq. (2.29) can also be represented by mean parameters $\boldsymbol{\mu} \in \mathbb{M}$. The collection of all the mean parameters $\mathbb{M}$ is called marginal polytope, and it is defined by

$$\mathbb{M} := \Big\{ \boldsymbol{\mu} \mid \mathbb{E}_{\mathbf{p}(\mathbf{X})}\Big[\boldsymbol{\phi}(\mathbf{X})\Big] \Big\}, \tag{2.34}$$

which can be obtained by the gradient mapping of the log partition function,

$$\nabla_{\boldsymbol{\theta}} \Psi(\boldsymbol{\theta}) = \boldsymbol{\mu}. \tag{2.35}$$

Uinsg the overcomplete representation of discrete graphical model, the set in Eq. (2.34) defines implicit constraints that enforce the consistency between all marginals. Note that the mapping in Eq. (2.35) is not one-to-one [Wainwright and Jordan, 2008]. The gradient mapping of the log partition function provides a mapping from the natural parameter space Θ to the mean parameter space $\mathbb{M}$. The reverse mapping is given by the entropy function. The entropy of a join probability distribution in Eq. (2.33) can be written by

$$\mathrm{H}(\boldsymbol{\mu}) := \sum_{\mathbf{X}} -P(\mathbf{X}) \cdot \log P(\mathbf{X}) = -\langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle - \Psi(\boldsymbol{\theta}). \tag{2.36}$$

The gradient mapping of the entropy in Eq. (2.36) with respect to $\boldsymbol{\mu}$ obeys

$$\nabla_{\boldsymbol{\mu}} \mathrm{H}(\boldsymbol{\mu}) = -\boldsymbol{\theta}. \tag{2.37}$$

We can see that the exponential family representation offers the dual representation of probability distributions by the natural parameters and the mean parameters. This is the consequence of the conjugate duality of the Legendre-Fenchel transformation [Rockafellar and Wets, 2009].

Definition 2.31 (**Legendre-Fenchel transformation**). *Let* $F : \mathbb{R}^n \to \overline{\mathbb{R}}$ *and* $F^* : \mathbb{R}^n \to \overline{\mathbb{R}}$ *be functions mapping from the* n*-dimensional real space to the extended real-line* $\mathbb{R} \cup \{-\infty, +\infty\}$*. We say* F^* *is conjugate to* F *iff.,*

$$F^*(\mathbf{X}^*) := \sup_{\mathbf{X}} \Big\{ \langle \mathbf{X}, \mathbf{X}^* \rangle - F(\mathbf{X}) \Big\}. \tag{2.38}$$

Now, we have the conjugate dual relationship between the log partition function $\Phi(\boldsymbol{\theta})$ and the entropy $\mathrm{H}(\boldsymbol{\mu})$ by applying the Legendre-Fenchel transformation to $\Phi(\boldsymbol{\theta})$,

$$\Phi(\boldsymbol{\theta}) = \sup_{\boldsymbol{\mu} \in M} \Big\{ \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle + \mathrm{H}(\boldsymbol{\mu}) \Big\}. \tag{2.39}$$

This conjugate duality is the basis of the variational inference framework, and it defines optimization problems over the primal natural parameter space Θ and the dual mean parameter space $\mathbb{M}$ with a certain optimization objective and a set of constraints.

2.3.2 Weighted Mini-bucket Bounds

Given a graphical model $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, consider the mixed inference task with a total variable elimination ordering $\mathcal{O} := \{X_1 \prec \ldots \prec X_N\}$, where $N = |\mathbf{X}|$, and each X_i is associated with the weight w_{X_i}, which is either 0 if X_i is a maximization variable or 1 if X_i is

a summation variable. Let a mini-bucket tree decomposition $\mathcal{T}_{\text{MB}}(\mathcal{B}, \mathcal{S})$ with some i-bound relaxes each bucket $\mathbf{B}_{X_i}$ in a bucket tree $\mathcal{T}_{\text{BT}}$ to a set of P_i mini-buckets $\{\mathbf{B}_{X_i^1}, \ldots, \mathbf{B}_{X_i^{P_i}}\}$. Then, the weighted mini-bucket relaxation scheme bounds the mixed inference task, and we can tighten upper bounds by enforcing the auxiliary constraints introduced by the relaxation. We can also view the relaxed graphical model as an augmented graphical model with additional duplicated random variables, one per mini-bucket by the mini-bucket relaxation.

Definition 2.32 (**Augmented Graphical Model**). *An augmented graphical model* $\overline{\mathcal{M}} := \langle \overline{\mathbf{X}}, \overline{\mathbf{D}}, \overline{\mathbf{F}} \rangle$ *over the mini-bucket tree* $\mathcal{T}_{MB}(\mathcal{B}, \mathcal{S})$ *is a graphical model obtained by introducing a set of auxiliary random variables* $\{\overline{X}_i^1, \ldots, \overline{X}_i^{P_i}\}$ *per each variable* X_i *by the mini-bucket relaxation. Namely, we introduce duplicated random variables* $\overline{\mathbf{X}} = \cup_{i=1}^{N} \{\overline{X}_i^p | p \in \{1, \ldots, P_i\}\}$ *with the same domain* $dom(\overline{X}_i^p) = dom(X_i) \;\; \forall \overline{X}_i^p \in \overline{\mathbf{X}}$*, and replace the original variables in the scope of a function* $sc(F_i)$ *with the duplicated random variables along a path in the mini-bucket tree starting from the mini-bucket* $\boldsymbol{B}_{\overline{X}_i^p}$ *that contains the function* F_i*.*

The symbols with an over-line denote the elements in the augmented model. For example, $\overline{N}$ denotes the total number of variables in the augmented model, $\overline{X}_i^p$ denotes a variable associated with X_i and the p-th mini-bucket $\mathbf{B}_{X_i^p}$, $\overline{F}_i$ denotes a function defined in the augmented model, $\overline{\boldsymbol{\theta}}_i$ denotes the natural parameters, $\overline{\boldsymbol{\mu}}$ denotes the the mean parameters, and so on.

The primal bound of the mixed inference task can be derived by applying the weighted mini-bucket relaxation as shown in Theorem 2.6.

$$\begin{aligned} \Phi_w(\boldsymbol{\theta}) &:= \log \sum_{X_N}^{w_{X_N}} \cdots \sum_{X_1}^{w_{X_1}} \exp\big(\boldsymbol{\theta}(\mathbf{X})\big) \\ \leq \; \Phi_{\overline{w}}(\overline{\boldsymbol{\theta}}) &:= \log \Big(\sum_{\overline{X}_N^{P_N}}^{w_{\overline{X}_N^{P_N}}} \cdots \sum_{\overline{X}_N^1}^{w_{\overline{X}_N^1}} \Big) \cdots \Big(\sum_{\overline{X}_1^{P_1}}^{w_{\overline{X}_1^{P_1}}} \cdots \sum_{\overline{X}_1^1}^{w_{\overline{X}_1^1}} \Big) \exp\big(\overline{\boldsymbol{\theta}}(\overline{\mathbf{X}})\big). \end{aligned} \tag{2.40}$$

The conjugate dual representation of the primal bound of the mixed inference can be obtained by using the dual representation of the mixed inference task [Liu, 2014].

Theorem 2.7 (**Conjugate Dual of the Mixed Inference Task**). *Given a graphical model* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ *and a mixed inference task with a total elimination order* $\mathcal{O}$ *consistent with the constrained ordering of the mixed inference task, and a set of the maximization variables* $\mathbf{X}_M$ *and the summation variable* $\mathbf{X}_S$*. We write the joint probability model* $p(\mathbf{X})$ *as a product of conditional probability functions,*

$$P\big(\mathbf{X}\big) = P\big(X_1|pa(X_1)\big) \cdot P\big(X_{N-1}|pa(X_{N-1})\big) \cdots P\big(X_N\big), \tag{2.41}$$

where each parent of X_i *is the induced parents in the triangulated graph induced by* $\mathcal{O}$*, and* $pa(X_i) \subset \{X_{i+1}, \ldots, X_N\}$*. The conjugate dual representation of the mixed inference task* $\Phi_w(\boldsymbol{\theta})$ *can be written as follows.*

$$\begin{aligned}\Phi_w(\boldsymbol{\theta}) := \log \sum_{X_N}^{w_{X_N}} \cdots \sum_{X_1}^{w_{X_1}} \exp(\boldsymbol{\theta}(\mathbf{X})) &= \sup_{\boldsymbol{\mu} \in \mathbb{M}} \Big\{ \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle + \sum_{i=1}^{N} w_{X_i} H(X_i|pa(X_i)) \Big\} \\ &= \sup_{\boldsymbol{\mu} \in \mathbb{M}} \Big\{ \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle + H(\boldsymbol{\mu}) - \sum_{X_i \in \mathbf{X}_M} H(X_i|pa(X_i)) \Big\}. \end{aligned} \tag{2.42}$$

The optimal mean parameters $\boldsymbol{\mu}^*$ of the dual optimization problem in Eq. (2.42) define the optimal pseudo probability distribution $\mathbf{q}^*(\mathbf{X})$,

$$\begin{aligned}\mathbf{q}^*(\mathbf{X}) &= \frac{\exp\big(\boldsymbol{\theta}(\mathbf{X})\big)^{1/w_{X_1}}}{\sum_{X_1} \exp\big(\boldsymbol{\theta}(\mathbf{X})\big)^{1/w_{X_1}}} \cdot \frac{\Big[\sum_{X_1}^{w_{X_1}} \exp\big(\boldsymbol{\theta}(\mathbf{X})\big)\Big]^{1/w_{X_2}}}{\sum_{X_2} \Big[\sum_{X_1}^{w_{X_1}} \exp\big(\boldsymbol{\theta}(\mathbf{X})\big)\Big]^{1/w_{X_2}}} \cdots \\ &\qquad \frac{\Big[\sum_{X_{N-1}}^{w_{X_{N-1}}} \cdots \sum_{X_1}^{w_{X_1}} \exp\big(\boldsymbol{\theta}(\mathbf{X})\big)\Big]^{1/w_{X_N}}}{\sum_{X_N} \Big[\sum_{X_{N-1}}^{w_{X_{N-1}}} \cdots \sum_{X_1}^{w_{X_1}} \exp\big(\boldsymbol{\theta}(\mathbf{X})\big)\Big]^{1/w_{X_N}}}, \\ &= \mathbf{q}^*\big(X_1|\mathrm{pa}(X_1)\big) \cdot \mathbf{q}^*\big(X_{N-1}|\mathrm{pa}(X_{N-1})\big) \cdots \mathbf{q}^*\big(X_N\big). \end{aligned} \tag{2.43}$$

The following weighted mini-bucket bounding scheme further tightens the variational bound in Eq. (2.42) by optimizing over the auxiliary constraints in the augmented graphical model.

Proposition 2.3 (**Weighted Mini-bucket Bounds**). *The dual optimization problem of the weighted mini-bucket bound for the mixed inference task* $\Phi_{\overline{w}}(\overline{\boldsymbol{\theta}})$ *can be written by*

$$\Phi_{\overline{w}}(\overline{\boldsymbol{\theta}}) = \sup_{\overline{\boldsymbol{\mu}} \in \overline{\mathbb{L}}} \Big\{ \langle \overline{\boldsymbol{\theta}}, \overline{\boldsymbol{\mu}} \rangle + \sum_{i=1}^{\overline{N}} \overline{w}_{\overline{X}_i} H(\overline{X}_i | pa(\overline{X}_i)) \Big\}, \tag{2.44}$$

$$s.t. \quad \begin{cases} \overline{X}_i^1 = \overline{X}_i^2 = \cdots = \overline{X}_i^{P_i} \quad \forall i \in \mathcal{I}_{\mathbf{X}} \\ \sum_{p=1}^{P_i} w_{\overline{X}_i^p} = 1 \;\; \forall i \in \mathcal{I}_{\mathbf{X}}, \;\; w_{\overline{X}_i^p} \geq 0, \end{cases} \tag{2.45}$$

where $\overline{\mathbb{L}}$ *denotes the marginal polytope of the augmented tree model, and the entropy decomposition assumes the probability model factorizes over the augmented distribution* $\boldsymbol{p}(\overline{\mathbf{X}})$ *by the mini-bucket tree decomposition. Eq.* (2.45) *shows the constraints on the duplicated random variables and the weights of the augmented graphical model introduced by the weighted mini-bucket relaxation.*

Fixing the weights and using Eq. (2.43), the optimal parameterization over the mini-bucket tree is

$$\mathbf{q}^*(\overline{\mathbf{X}}) = \mathbf{q}^*\big(\overline{X}_1 | \mathrm{pa}(\overline{X}_1)\big) \cdot \mathbf{q}^*\big(\overline{X}_{\overline{N}-1} | \mathrm{pa}(X_{\overline{N}-1})\big) \cdots \mathbf{q}^*\big(\overline{X}_{\overline{N}}\big). \tag{2.46}$$

Eq. (2.46) shows that Algorithm 2.1 computes the upper bounds of the original mixed inference task. The forward mini-bucket message sent from mini-bucket $\mathbf{B}_{\overline{X}_i^p}$ to $\mathbf{B}_{\overline{X}_j^q}$ is

$$\mathrm{m}(\mathbf{B}_{\overline{X}_i^p}, \mathbf{B}_{\overline{X}_j^q}) = \sum_{X_i^p}^{w_{\overline{X}_i^p}} \Big[\exp\big(\overline{\theta}_{\mathbf{B}_{\overline{X}_i^p}}\big) \cdot \prod_{(\mathbf{B}_{\overline{X}_r^p}, \mathbf{B}_{\overline{X}_i^p}) \in \mathcal{S}} \mathrm{m}(\mathbf{B}_{\overline{X}_r^s}, \mathbf{B}_{\overline{X}_i^p}) \Big], \tag{2.47}$$

where $X_r \prec_{\mathcal{O}} X_i \prec_{\mathcal{O}} X_j$, $\mathrm{m}(\mathbf{B}_{\overline{X}_r^s}, \mathbf{B}_{\overline{X}_i^p})$ is incoming messages to $\mathbf{B}_{\overline{X}_i^p}$. The backward mini-

bucket message from $\mathbf{B}_{\overline{X}_i^p}$ to $\mathbf{B}_{\overline{X}_j^q}$ is

$$\mathrm{m}(\mathbf{B}_{\overline{X}_i^p}, \mathbf{B}_{\overline{X}_j^q}) = \left[\sum_{\mathbf{X}_{\mathbf{B}_{\overline{X}_i^p}} \setminus \mathbf{X}_{\mathbf{B}_{\overline{X}_j^q}}} \frac{\left[\exp\left(\overline{\theta}_{\mathbf{B}_{\overline{X}_i^p}}\right) \cdot \prod_{\mathbf{B}_{\overline{X}_r^p} \in \mathrm{nhd}(\mathbf{B}_{\overline{X}_i^p})} \mathrm{m}(\mathbf{B}_{\overline{X}_r^s}, \mathbf{B}_{\overline{X}_i^p}) \right]^{1/w_{\overline{X}_i^p}}}{\mathrm{m}(\mathbf{B}_{\overline{X}_j^q}, \mathbf{B}_{\overline{X}_i^p})^{1/w_{\overline{X}_j^q}}} \right]^{w_{\overline{X}_j^q}}, \quad (2.48)$$

where $X_j \prec X_i$ and $\mathrm{m}(\mathbf{B}_{\overline{X}_i^p}, \mathbf{B}_{\overline{X}_i^p})$ is the forward message from $\mathbf{B}_{\overline{X}_j^q}$ to $\mathbf{B}_{\overline{X}_i^p}$, and $\mathrm{m}(\mathbf{B}_{\overline{X}_r^s}, \mathbf{B}_{\overline{X}_i^p})$ is the forward and the backward message sent to $\mathbf{B}_{\overline{X}_i^p}$.

We can tighten the weighted mini-bucket upper bound by enforcing the equality constraint by the Lagrange multipliers $\overline{\boldsymbol{\eta}}$ and optimizing the weights distributed to the mini-buckets $\overline{\mathbf{w}}$ subject to the constraint in Eq. (2.45).

The Lagrangian function $\mathcal{L}(\overline{\boldsymbol{\theta}}, \overline{\boldsymbol{\eta}})$ for optimizing $\overline{\boldsymbol{\eta}}$ can be written as

$$\mathcal{L}(\overline{\boldsymbol{\theta}}, \overline{\boldsymbol{\eta}}) := \langle \overline{\boldsymbol{\theta}}, \overline{\boldsymbol{\mu}} \rangle + \sum_{i=1}^{\overline{N}} \overline{w}_{\overline{X}_i} \mathrm{H}(\overline{X}_i | \mathrm{pa}(\overline{X}_i)) + \sum_{i=1}^{N} \left(\sum_{p=1}^{P_i} \langle \boldsymbol{\eta}_i^p, \boldsymbol{\mu}_{\overline{X}_i^p} \rangle - \psi_i \Big(\sum_{p=1}^{P_i} \boldsymbol{\eta}_i^p \Big) \right), \quad (2.49)$$

where $\boldsymbol{\eta}_i^p$ is the Lagrange multiplier for the constraint $\overline{X}_i^p - X_i = 0$, and ψ_i is the Lagrange multiplier for the constraint $\sum_{p=1}^{P_i} \boldsymbol{\eta}_i^p(X_i) = 0$, and $\boldsymbol{\mu}_{\overline{X}_i^p}$ is the mean parameters of the random variable $\overline{X}_i^p$. A fixed point update performing the moment matching on the mini-buckets $\{\mathbf{B}_{\overline{X}_i^1}, \ldots, \mathbf{B}_{\overline{X}_i^{P_i}}\}$ can be derived as

$$\overline{\boldsymbol{\theta}}_{\mathbf{B}_{\overline{X}_i^p}} = \overline{\boldsymbol{\theta}}_{\mathbf{B}_{\overline{X}_i^p}} + w_{\overline{X}_i^p} \Big(\log \mathbf{q}^*(\overline{X}_i^p) - \log \mathbf{q}(\overline{X}_i^p) \Big), \quad (2.50)$$

$$\mathbf{q}(\overline{X}_i^p) \propto \sum_{\mathbf{X}_{\mathbf{B}_{\overline{X}_i^p}} \setminus \{\overline{X}_i^p\}} \Big[\exp\left(\boldsymbol{\theta}\right) \cdot \prod_{\mathbf{B}_{\overline{X}_r^p} \in \mathrm{nhd}(\mathbf{B}_{\overline{X}_i^p})} \mathrm{m}(\mathbf{B}_{\overline{X}_r^s}, \mathbf{B}_{\overline{X}_i^p}) \Big]^{1/w_{\overline{X}_i^p}}, \quad (2.51)$$

$$\log \mathbf{q}^*(X_i) = \frac{1}{\sum_{p=1}^{P_i} w_{\overline{X}_i^p}} \Big(\sum_{p=1}^{P_i} w_{\overline{X}_i^p} \log \mathbf{q}(\overline{X}_i^p) \Big). \quad (2.52)$$

The Lagrangian function $\mathcal{L}(\overline{\boldsymbol{\theta}}, \overline{\mathbf{w}})$ for optimizing $\overline{\mathbf{w}}$ can be written by

$$\mathcal{L}(\overline{\boldsymbol{\theta}}, \overline{\mathbf{w}}) := \langle \overline{\boldsymbol{\theta}}, \overline{\boldsymbol{\mu}} \rangle + \sum_{i=1}^{\overline{N}} \overline{w}_{\overline{X}_i} \mathrm{H}(\overline{X}_i | \mathrm{pa}(\overline{X}_i)) + \sum_{i=1}^{N} \left(\sum_{p=1}^{P_i} \lambda_i^p w_{\overline{X}_i^p} - \psi_i \Big(\sum_{p=1}^{P_i} w_{\overline{X}_i^p} - 1 \Big) \right), \quad (2.53)$$

where λ_i^p is the Lagrange multiplier for the constraint $w_{\overline{X}_i^p} \geq 0$, and ψ_i is the Lagrange multiplier for the constraint $\sum_{p=1}^{P_i} w_{\overline{X}_i^p} = 1$. The entropy matching condition for the fixed point update of weights can be derived as

$$w_{\overline{X}_i^p} \left(\mathrm{H}(\overline{X}_i^p | \mathrm{pa}(\overline{X}_i^p) - \sum_{p=1}^{P_i} w_{\overline{X}_i^p} \mathrm{H}(\overline{X}_i^p | \mathrm{pa}(\overline{X}_i^p) \right). \quad (2.54)$$

The moment-matching by Eq. (2.50) and the entropy-matching by Eq. (2.54) can be performed in conjunction with the forward and backward message passing over the mini-bucket tree shown in Eq. (2.47) and Eq. (2.48). For further details on the message passing update, we refer to Liu [2014].

A simpler one iteration version of the message passing algorithm, called weighted mini-bucket with moment-matching (WMBMM) is shown to be useful in practice [Ihler et al., 2012, Marinescu et al., 2014]. The WMBMM only performs the moment-matching updates between mini-buckets with fixed uniform weights while performing the forward message passing. This only changes the weighted mini-bucket elimination algorithm shown in Algorithm 2.1 by adding one additional step, which matches the marginals between mini-buckets before generating messages.

2.3.3 Generalized Dual Decomposition Bounds

The bounding scheme using weighted mini-buckets interleaves optimization and message passing to compute pseudo marginals and to tighten the upper bounds. Due to cyclic dependency while updating the optimization parameters, the fixed-point updates in the weighted

mini-bucket bound may lead to unstable numerical behavior. We can push further the idea of duplicating one random variable between mini-buckets to duplicating all shared random variables between clusters in a join-graph. This approach leads to the generalized dual decomposition bounds for the mixed inference task [Ping et al., 2015], which is a generalization of the dual decomposition for the maximization task and the weighted mini-bucket for the summation task.

Given a graphical model $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ and its join-graph decomposition $\mathcal{G}_{\text{JG}}(\mathcal{C}, \mathcal{S}) := \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$, consider the mixed inference task with a total variable elimination order $\mathcal{O} := \{X_1, \ldots, X_N\}$ and a set of weights $\{w_{X_1}, \ldots, w_{X_N}\}$. The joint probability model can be written by

$$\mathbf{p}(\mathbf{X}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \exp\big(\boldsymbol{\theta}_C(\mathbf{X}_C)\big), \tag{2.55}$$

where $\mathbf{X}_C$ is the set of variables at C, θ_C is the natural parameters at C obatined by $\sum_{i \in \psi(C)} \theta_i$, and Z is the normalization constant.

The following primal bound of the mixed inference task can be obtained by fully decoupling the random variables between adjacent clusters,

$$\Phi_w(\boldsymbol{\theta}) := \log \sum_{X_N}^{w_{X_N}} \cdots \sum_{X_1}^{w_{X_1}} \exp\big(\sum_{C \in \mathcal{C}} \boldsymbol{\theta}_C(\mathbf{X}_C)\big) \leq \sum_{C \in \mathcal{C}} \left(\log \sum_{X_N^C}^{w_{X_N^C}} \cdots \sum_{X_1^C}^{w_{X_1^C}} \exp\big(\boldsymbol{\theta}_C(\mathbf{X}_C)\big) \right). \tag{2.56}$$

Since Eq. (2.56) duplicates all the variables between clusters, the probability distribution in the augmented model is simply the product of marginals at each cluster,

$$\mathbf{q}(\overline{\mathbf{X}}) = \prod_{C \in \mathcal{C}} \mathbf{q}(\overline{\mathbf{X}}_C) \tag{2.57}$$

where each marginal $\mathbf{q}(\overline{\mathbf{X}}_C)$ is at cluster C is factorized by the local conditional probability

functions,

$$\mathbf{q}(\overline{\mathbf{X}}_C) = \prod_{C\in\mathcal{C}} \mathbf{q}(\overline{X}_1^C|\overline{X}_2^C,\ldots,\overline{X}_N^C)\cdots\mathbf{q}(\overline{X}_{N-1}^C|\overline{X}_N^C)\cdot\mathbf{q}(\overline{X}_N^C).$$

The following generalized dual decomposition bounds tightens the primal bound in Eq. (2.56) by enforcing auxiliary constraints defined over the join-graph decomosition.

Proposition 2.4 (**Generalized Dual Decomposition Bounds**). *The dual optimization problem of the generalized dual decomposition bound for the mixed inference task $\Phi_w(\boldsymbol{\theta})$ can be written by*

$$\Phi_{\overline{w}}(\overline{\boldsymbol{\theta}}) = \sup_{\overline{\boldsymbol{\mu}}\in\overline{\mathbb{L}}} \Big\{ \langle\overline{\boldsymbol{\theta}},\overline{\boldsymbol{\mu}}\rangle + \sum_{C\in\mathcal{C}}\sum_{i=1}^{N} w_{\overline{X}_i^C} H\big(\overline{X}_i^C|pa(\overline{X}_i^C)\big)\Big\}, \tag{2.58}$$

$$s.t. \begin{cases} \mathbf{X}_{S_{C_i,C_j}} = \mathbf{X}^{C_i}_{S_{C_i,C_j}} = \mathbf{X}^{C_j}_{S_{C_i,C_j}} \quad \forall(C_i,C_j)\in\mathcal{S} \\ \sum_{C\in\mathcal{C}} w_{\overline{X}_i^C} = 1 \;\; \forall i\in\mathcal{I}_{\mathbf{X}}, \;\; w_{\overline{X}_i^C}\geq 0, \end{cases} \tag{2.59}$$

where $\overline{\mathbb{L}}$ denotes the marginal polytope of the augmented model that ensuring local consistency between marginals at adjacent clusters, and $\mathbf{X}^{C_i}_{S_{C_i,C_j}}$ and $\mathbf{X}^{C_j}_{S_{C_i,C_j}}$ in the equality constraints are the random variables over the separator between adjacent clusters C_i and C_j.

The optimal parameterization at each cluster C is

$$\begin{aligned}\mathbf{q}^*(\overline{\mathbf{X}}_C) &= \frac{\exp\big(\overline{\boldsymbol{\theta}}_C(\overline{\mathbf{X}}_C)\big)^{1/w_{\overline{X}_1^C}}}{\sum_{\overline{X}_1^C}\exp\big(\overline{\boldsymbol{\theta}}_C(\overline{\mathbf{X}}_C)\big)^{1/w_{\overline{X}_1^C}}}\cdot\frac{\Big[\sum_{\overline{X}_1^C}^{w_{\overline{X}_1^C}}\exp\big(\overline{\boldsymbol{\theta}}_C(\overline{\mathbf{X}}_C)\big)\Big]^{1/w_{\overline{X}_2^C}}}{\sum_{\overline{X}_2^C}\Big[\sum_{\overline{X}_1^C}^{w_{\overline{X}_1^C}}\exp\big(\overline{\boldsymbol{\theta}}_C(\overline{\mathbf{X}}_C)\big)\Big]^{1/w_{\overline{X}_2^C}}}\cdots \\ &\qquad \frac{\Big[\sum_{\overline{X}_{N-1}^C}^{w_{\overline{X}_{N-1}^C}}\cdots\sum_{\overline{X}_1^C}^{w_{\overline{X}_1^C}}\exp\big(\overline{\boldsymbol{\theta}}_C(\overline{\mathbf{X}}_C)\big)\Big]^{1/w_{\overline{X}_N^C}}}{\sum_{\overline{X}_N^C}\Big[\sum_{\overline{X}_{N-1}^C}^{w_{\overline{X}_{N-1}^C}}\cdots\sum_{\overline{X}_1^C}^{w_{\overline{X}_1^C}}\exp\big(\overline{\boldsymbol{\theta}}_C(\overline{\mathbf{X}}_C)\big)\Big]^{1/w_{\overline{X}_N^C}}}, \\ &= \mathbf{q}^*\big(\overline{X}_1^C|\mathrm{pa}(\overline{X}_1^C)\big)\cdot\mathbf{q}^*\big(\overline{X}_{N-1}^C|\mathrm{pa}(\overline{X}_{N-1}^C)\big)\cdots\mathbf{q}^*\big(\overline{X}_N^C\big),\end{aligned} \tag{2.60}$$

where $\overline{\mathbf{X}}^C = \{\overline{X}_1^C,\ldots\overline{X}_N^C\}$ and $q^*(\overline{\mathbf{X}}^C)$ is factorized similar to Eq. (2.43). The only difference is that the functions $\overline{\boldsymbol{\theta}}_C$ and variables $\overline{\mathbf{X}}_C$ are local in each cluster C. Fully eliminating the

variables at all clusters generates an upper bound of $\phi_w(\boldsymbol{\theta})$ that we can minimize by the moment matching and the entropy matching updates similar to the weighted mini-bucket bounds.

Introducing the cost-shifting functions over the separators, the primal bound can be rewritten by

$$\Phi_w(\boldsymbol{\theta}, \boldsymbol{\eta}) := \sum_{C \in \mathcal{C}} \left(\log \sum_{X_N^C}^{w_{X_N^C}} \cdots \sum_{X_1^C}^{w_{X_1^C}} \exp\left(\boldsymbol{\theta}_C(\mathbf{X}_C) - \sum_{(C,C') \in \mathcal{S}} \boldsymbol{\eta}_{(C,C')}(\mathbf{X}_{(C,C')}) \right) \right), \tag{2.61}$$

$$\text{s.t. } \boldsymbol{\eta}(\mathbf{X}_{S_{C,C'}}) + \boldsymbol{\eta}(C', C)(\mathbf{X}_{S_{C,C'}}) = 0, \tag{2.62}$$

where $\boldsymbol{\eta}(\mathbf{X}_{S_{C,C'}})$ is the Lagrange multiplier that moves cost from the cluster C to C'. The partial derivative of the upper bound with respect to the $\boldsymbol{\eta}(\mathbf{X}_{S_{C,C'}})$ can be evaluated as

$$\frac{\partial \Phi_w(\boldsymbol{\theta}, \boldsymbol{\eta})}{\partial \boldsymbol{\eta}(\mathbf{X}_{S_{C,C'}})} = - \sum_{\text{elim}(C,C')} \mathbf{q}^*\big(\overline{\mathbf{X}}_C\big) + \sum_{\text{elim}(C',C)} \mathbf{q}^*\big(\overline{\mathbf{X}}_{C'}\big). \tag{2.63}$$

We see that the partial derivative is the difference between the pseudo-marginals at the adjacent clusters C and C' that encourages moment-matching. The entropy matching update is the same as the condition shown in Eq. (2.54) after replacing the index running over the mini-buckets p with the index over the clusters C. By using the moment and entropy matching conditions, the upper bound $\Phi_w(\boldsymbol{\theta}, \boldsymbol{\eta}, \mathbf{w})$ can be tightened by gradient-based optimization such as the block coordinate descent algorithm that alternates updating the cost-shifting parameters and the weight parameters by gradient descent optimization algorithms.

2.4 Influence Diagrams

Influence Diagrams provide a modeling and inference framework for sequential decision making under uncertainty, representing the probabilistic knowledge by a Bayesian network and

the preference by utility functions over the chance variables and decision variables [Howard and Matheson, 1984, Nielsen and Jensen, 2009]. This section defines influence diagrams and reviews earlier graphical model algorithms.

2.4.1 Influence Diagrams and Perfect Recall Condition

Howard and Matheson [1981] introduced influence diagrams (IDs) as a compact graphical representation of decision trees for modeling and solving sequential decision making problems. We can view IDs as a discrete graphical model of the single-agent sequential decision making problem under uncertainty, which extend Bayesian networks with decision variables and utility functions. The standard task in IDs is to compute the maximum expected utility (MEU), and a set of optimal policy functions that jointly achieve the MEU subject to the constraints on the available information at each decision. The perfect recall influence diagrams assumes no limitation of memory in an agent, and it allows decisions can be made subject to the past history. The limited memory influence diagrams limits the memory in an agent, and the decisions can be made only subject to a partial history under the imperfect recall condition.

Definition 2.33 (**Influence Diagrams**). *An ID is a tuple* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$*, where*

1. $\mathbf{X} = \{X_1, \ldots, X_N\}$ *is a set of* N *discrete chance random variables, where the domain of* X_i *is denoted by* Ω_{X_i}
2. $\mathbf{D} = \{D_1, \ldots, D_M\}$ *is a set of* M *discrete decision variables, where the domain of* D_i *is denoted by* Ω_{D_i}
3. $\mathbf{P} = \{P_1, \ldots, P_N\}$ *is a set of conditional probability functions,*
4. $\mathbf{U} = \{U_1, \ldots, U_R\}$ *is a set of additive utility functions,*

5. $\mathcal{O} = \{\{pa(D_i) \prec \{D_i\} | D_i \in \mathbf{D}\}$ *is a set of precedence relations, where* $pa(D_i)$ *is the parents of* D_i *subject to the DAG* $\mathcal{G}$ *of an ID.*

A DAG $\mathcal{G} := \langle \mathcal{V}, \mathcal{E} \rangle$ *of an* $\mathcal{M}$ *is defined by three types of nodes,*

1. $\mathcal{V}_X$ *is the set of chance nodes associated with the chance variables* $\mathbf{X}$ *drawn as circles, and the parents of a node* $X_i \in \mathbf{X}$ *define the scope of the conditional probability function* $P\big(X_i | pa(X_i)\big)$,

2. $\mathcal{V}_D$ *is the set of decision nodes associated with the decision variables* $\mathbf{D}$ *drawn as squares. The incoming arcs to a decision node are called informational arcs and we enforce each* $pa(D_i)$ *to include all the observed chance variables and the previous decision variables remaining in the memory while making decision* D_i,

3. $\mathcal{V}_U$ *is the value nodes associated with the utility functions* $\mathbf{U}$ *drawn as diamonds, and the parents of a node* $U_i \in \mathbf{U}$ *defines the scope of the utility function* $sc(U_i) = pa(U_i)$.

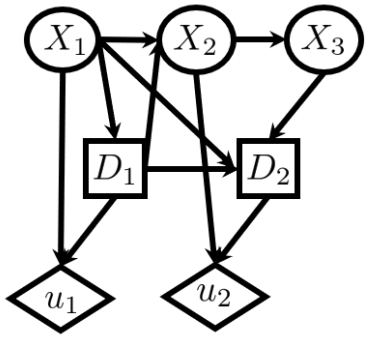
Note that the conventional IDs only require informational arcs to be specified from the immediately observed chance variables and ask for the additional regularity condition, which defines a direct path over all the decision variables. However, we must be explicit on the observed variables remaining in the memory for each decision in Definition 2.33. This way we don't need the regularity condition.

Definition 2.34 (**MEU Task**). *Given an ID* $\mathcal{M}$*, either a perfect recall influence diagram or a imperfect recall influence diagram, the MEU task computes*

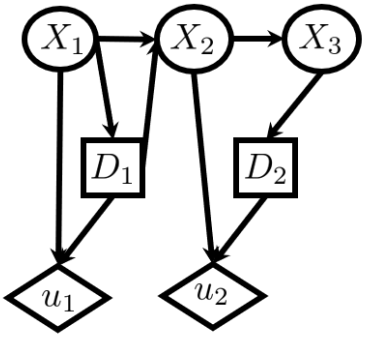
$$\max_{\boldsymbol{\Delta}} \mathbb{E}_{P(\mathbf{X},\mathbf{D})}\Big[\sum_{U_i \in \mathbf{U}} U_i\Big], \tag{2.64}$$

$$P(\mathbf{X}, \mathbf{D}) = \big(\prod_{P_i \in \mathbf{P}} P_i\big(X_i | pa(X_i)\big)\big) \times \big(\prod_{\Delta_i \in \boldsymbol{\Delta}} \Delta_i\big(D_i | pa(D_i)\big)\big), \tag{2.65}$$

$$\boldsymbol{\Delta} := \{\Delta(D_i | pa(D_i)) \;\mid\; \Delta(D_i | pa(D_i)) : \Omega_{pa(D_i)} \to \Omega_{D_i} \;\; \forall D_i \in \mathbf{D}\}, \tag{2.66}$$



(a) Perfect Recall ID



(b) Limited Memory ID

Figure 2.5: Example of perfect recall and limited memory influence diagram

where $\boldsymbol{\Delta}$ *in Eq.* (2.66) *is a set of policy functions that maps from the Cartesian product of the domains of the observed variables* $\Omega_{pa(D_i)} = \times_{X \in pa(D_i)} \Omega_X$ *to the domain of the decision variable* Ω_{D_i}*, and the expectation is taken over the joint distribution in Eq.* (2.65)*.*

The set of optimal policy functions $\boldsymbol{\Delta}^*$ *is obtained by*

$$\boldsymbol{\Delta}^* = \underset{\boldsymbol{\Delta}}{\operatorname{argmax}} \, \mathbb{E}_{P(\mathbf{X},\mathbf{D})} \Big[\sum_{U_i \in \mathbf{U}} U_i \Big]. \tag{2.67}$$

Example 2.5. *Figure 2.5a and 2.5b show an example of IDs of perfect recall and LIMIDs. The chance nodes* $\mathcal{V} = \{X_1, X_2, X_3\}$ *are drawn as circles, and the directed arcs to the chance nodes define the scope of probability functions* $\mathbf{P} = \{P(X_1), P(X_2|X_1, X_1), P(X_3|X_2)\}$*. The decision nodes* $\mathcal{V}_D := \{D_1, D_2\}$ *are drawn as squares, and the informational arcs define the scope of policy functions* $\boldsymbol{\Delta} = \{D_1|pa(D_1)), \Delta(D_2|pa(D_2))\}$*. The value nodes* $\mathcal{V}_U : \{U_1, U_2\}$ *are drawn as diamonds. The difference between Figure 2.5a and Figure 2.5b is in informational arcs. As we can see, Figure 2.5b has less informational arcs compared with Figure 2.5a that models the agent forgets* $\{D_1, X_1\}$ *when making decision* D_2*.*

2.4.2 Valuation Algebra over Jensen's Potentials

The valuation algebra is a system with combination and marginalization operations [Shenoy, 1989, 1997, Kohlas and Shenoy, 2000, Kohlas and Wilson, 2008] . Since there are multiplicative probability functions and additive utility functions in IDs, Jensen et al. [1994] proposed a valuation algebra over the pair of probability and utility or value functions for computing the MEU task in IDs of perfect recall.

Definition 2.35 (**Jensen's Potential**). *Given an ID* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$*, we define a pair of probability and value functions* $\Psi(\mathbf{X}_\Psi) = (P(\mathbf{X}_\Psi), V(\mathbf{X}_\Psi))$ *over a set of variables* $\mathbf{X}_\Psi \subset (\mathbf{X} \cup \mathbf{D})$ *as a potential, or valuation for IDs.*

The scope of a potential Ψ is the union of the scope of probability and value functions $\mathrm{sc}(\Psi) = \mathrm{sc}(P) \cup \mathrm{sc}(V)$. We next define operations over the Jensen's potentials for IDs.

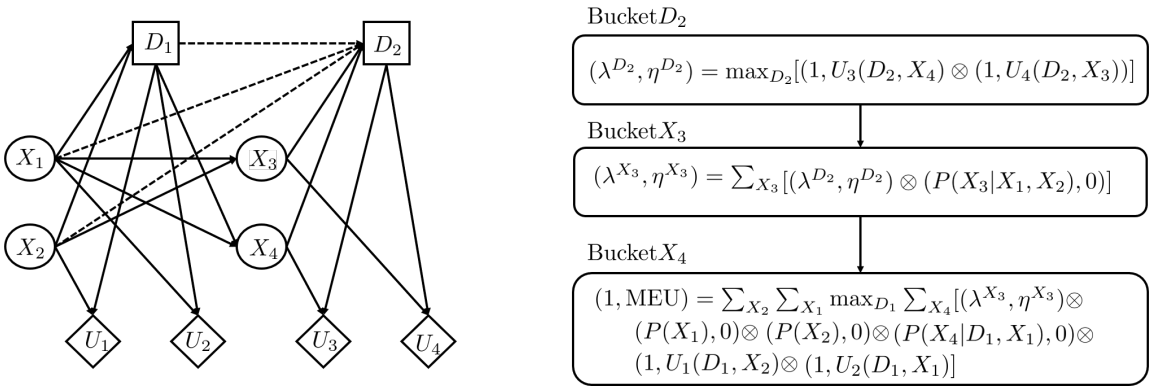
Definition 2.36 (**Combination**). *Given two potentials* $\Psi_1(\mathbf{X}_1) := (P_1(\mathbf{X}_1), V_1(\mathbf{X}_1))$ *and* $\Psi_2(\mathbf{X}_2) := (P_1(\mathbf{X}_2), V_1(\mathbf{X}_2))$*, the combination of the two valuations is defined by*

$$\Psi_1(\mathbf{X}_1) \otimes \Psi_2(\mathbf{X}_2) := (P_1(\mathbf{X}_1)P_2(\mathbf{X}_2), P_1(\mathbf{X}_1)V_2(\mathbf{X}_2) + P_2(\mathbf{X}_2)V_1(\mathbf{X}_1)), \tag{2.68}$$

where $\mathbf{X}_1$ *and* $\mathbf{X}_2$ *is the scope of the* Ψ_1 *and* Ψ_2*, respectively.*

Definition 2.37 (**Neutral Potential and Inverse**). *The neutral potential is* $\big(\mathbf{1}(\mathbf{X}), \mathbf{0}(\mathbf{X})\big)$*, where the* $\mathbf{1}$ *and* $\mathbf{0}$ *is the constant function that maps* $\Omega_\mathbf{X}$ *to to 1 and 0, respectively. We define the inverse of* $(P(\mathbf{X}), V(\mathbf{X}))$ *by* $\Big(\dfrac{1}{P(\mathbf{X})}, \dfrac{-V(\mathbf{X})}{P^2(\mathbf{X})}\Big)$.

Definition 2.38 (**Marginalization**). *Given* $\Psi(\mathbf{X}) := \big(P(\mathbf{X}), V(\mathbf{X})\big)$*, marginalizing out* $\mathbf{Y} \subset \mathbf{X}$ *by the summation, the maximization, and the powered-summation with weights* $\mathbf{w}$ *is*



(a) Perfect Recall ID **(b)** Constrained Joint-tree

Figure 2.6: Example of constrained join-tree decomposition of an ID of perfect recall.

defined by

$$\sum_{\mathbf{Y}}^{\mathbf{w}} \Psi(\mathbf{X}) := \big(\sum_{\mathbf{Y}} P(\mathbf{X}), \sum_{\mathbf{Y}}^{\mathbf{w}} V(\mathbf{X}) \big), \tag{2.69}$$

$$\max_{\mathbf{Y}}^{\mathbf{w}} \Psi(\mathbf{X}) := \big(\max_{\mathbf{Y}} P(\mathbf{X}), \max_{\mathbf{Y}}^{\mathbf{w}} V(\mathbf{X}) \big), \tag{2.70}$$

$$\sum_{\mathbf{Y}}^{\mathbf{w}} \Psi(\mathbf{X}) := \big(\sum_{\mathbf{Y}}^{\mathbf{w}} P(\mathbf{X}), \sum_{\mathbf{Y}}^{\mathbf{w}} V(\mathbf{X}) \big). \tag{2.71}$$

Definition 2.39 (Comparison). *Given two potentials* $\Psi_1(\mathbf{X}_1) := (P_1(\mathbf{X}_1), V_1(\mathbf{X}_1))$ *and* $\Psi_2(\mathbf{X}_2) := (P_1(\mathbf{X}_2), V_1(\mathbf{X}_2))$*, we define inequality between two potentials by*

$$\Psi_1(\mathbf{X}_1) \leq \Psi_2(\mathbf{X}_2) \iff P_1(\mathbf{X}_1) \leq P_2(\mathbf{X}_2) \ \ \& \ \ V_1(\mathbf{X}_1) \leq V_2(\mathbf{X}_2). \tag{2.72}$$

Now, we define an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$ using valuation algebra over the Jensen's potential, where $\mathbf{\Psi} = \{(P_i, 0) | P_i \in \mathbf{P}\} \cup \{(1, U_i) | U_i \in \mathbf{U}\}$. The MEU task can be rewritten as

$$\sum_{\text{pa}(D_1)} \max_{D_1} \cdots \sum_{\text{pa}(D_M)} \max_{D_M} \sum_{\mathbf{X} \setminus \text{pa}(D_M)} \bigotimes_{\alpha \in \mathcal{I}_{\mathbf{\Psi}}} \Psi_\alpha(\mathbf{X}_\alpha), \tag{2.73}$$

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

2.4.3 Constrained Join-tree Decomposition of IDs of Perfect Recall

By using the valuation algebra over the Jensen's potential, a tree decomposition of an ID of perfect recall can be defined as follows [Jensen et al., 1994].

Definition 2.40 (**Constrained Join-tree Decomposition**). *A constrained join-tree decomposition $\mathcal{T}_{CJT}$ of an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$ is a tuple $\mathcal{T}_{CJT} := \langle T(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$, where $T(\mathcal{C}, \mathcal{S})$ is a rooted tree with a root cluster C_{root}. χ and ψ is a labeling function mapping each cluster node $C \in \mathcal{C}$ to a set of variables by $\chi(C) \subset (\mathbf{X} \cup \mathbf{D})$ and to a set of potentials $\psi(C) \subset \mathbf{\Psi}$, respectively. The labeling functions satisfy additional conditions: (1) for any two nodes C_i and C_j, $\psi(C_i) \cap \psi(C_j) = \emptyset$, (2) for a potential $\psi \in \psi(C)$, $sc(\psi) \subset \chi(C)$, and (3) for any node C and a set of clusters $\mathbf{C}$ along the path from the node C to the root node C_{root}, $\cup_{C_i \in \mathbf{C}} \chi(C_i) \prec \chi(C)$ in the constrained order $\mathcal{O}$. Lastly, the join-tree T satisfies the running intersection property.*

The primal graph $\mathcal{G}_{\mathrm{P}}$ of the DAG $\mathcal{G}$ of an ID $\mathcal{M}$ can be obtained by moralizing the DAG and removing the value nodes for an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$, and a constrained join-tree $\mathcal{T}_{\mathrm{CJT}}$ can be obtained by triangulating the primal graph $\mathcal{G}_{\mathrm{P}}$ following any elimination ordering consistent with the precedence relations $\mathcal{O}$. Once we obtained a constrained join-tree, message passing over the tree performs the variable elimination for computing the MEU [Jensen et al., 1994, Mauá et al., 2012]. Algorithm 2.2 shows the message passing algorithm over the constrained join-tree decomposition $\mathcal{T}_{\mathrm{CJT}}$ for computing MEU. The message passing procedure starts from the leaf clusters in $\mathcal{T}_{\mathrm{CJT}}$, and it recursively collects, combines, and sends messages over the tree from leaf clusters to the root cluster. The MEU is the value component of the potential constant Ψ generated from the root cluster.

Example 2.6. *Figure 2.6a illustrates an ID of perfect recall and a schematic trace of the variable elimination algorithm using Jensen's potential. We use $\mathcal{O}_{elim} := \{D_2, X_3, X_4, D_1, X_1, X_2\}$ as an elimination order for generating the constrained join-tree. The first eliminated variable*

Algorithm 2.2 Message passing over constrained join-tree for computing MEU

Require: ID $\mathcal{M}_{\mathrm{G}} := \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$, Constrained join-tree decomposition $\mathcal{T}_{\mathrm{CJT}} := \langle T(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$
Ensure: $\mathcal{T}_{\mathrm{CJT}}$ augmented with messages sent out from clusters, MEU

1: **for** each cluster C from the leaf to the root **do**
2: Collect messages $(\lambda^{C_k \to C}, \eta^{C_k \to C})$ from incoming separators $\{(C_k \to C) | C_k \in \mathrm{de}(C)\}$
3: $\Psi^C(\chi(C)) \leftarrow (\bigotimes_{\Psi \in \psi(C)} \Psi) \otimes \big(\bigotimes_{C_k \in \mathrm{de}(C)} (\lambda^{C_k \to C}, \eta^{C_k \to C}) \big)$ ▷ Combine potentials at C
4: $(\lambda^{C \to \mathrm{pa}(C)}, \eta^{C \to \mathrm{pa}(C)}) \leftarrow \Downarrow_{\mathrm{sep}\left(C, \mathrm{pa}(C)\right)} \Psi^C(\chi(C))$ ▷ Compute out-going message
5: Send out-going message $(\lambda^{C \to \mathrm{pa}(C)}, \eta^{C \to \mathrm{pa}(C)})$ to separator $C \to \mathrm{pa}(C)$
6: $(\lambda^{\emptyset}, \eta^{\emptyset}) \leftarrow \Downarrow_{\emptyset} \Psi^{C_{\mathrm{root}}}$ ▷ Marginalize out all variables in the root cluster
7: **return** $\eta^{\emptyset}$ ▷ Return MEU

is D_2, so the variable elimination algorithm collects potentials whose scope includes D_2 in Bucket D_1. Then it generates the outgoing message $(\lambda^{D_1}, \eta^{D_1})$ and sends it to Bucket X_3. Bucket X_3 combines the allocated valuations and the incoming messages, and generates the outgoing message $(\lambda^{S_2}, \eta^{S_2})$. This variable elimination procedure continues until we obtain the MEU.

2.4.4 Earlier Bounding Schemes for IDs of Perfect Recall

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

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

Chapter 3

Direct Decomposition Bounds for Influence Diagrams

This chapter presents decomposition methods for bounding the MEU of influence diagrams having perfect recall, which we call IDs in the rest of the chapters. While earlier bounding schemes mostly used reductions to the MMAP or to the mixed inference task over a Bayesian Network [Mauá, 2016], we develop here a direct decomposition by extending the variational bounding schemes to the constrained join-graph decomposition of IDs to avoid the explosion in the model size due to reductions. Section 3.1 motivates developing direct decomposition bounds for IDs and overviews the chapter. Then, we present two types of bounding schemes. In Section 3.2, we present our first approach that extends the weighted mini-bucket relaxation to the valuation algebra and extends decomposition bounds in the MMAP task to the MEU task. In Section 3.3, we present our second approach also applied directly on the ID GM but without using Jensen's potential. In this later case, we present two bounding methods: (1) The first bounds the expected utility using Jensen's inequality [Jensen et al., 1906] applied to the exponential function ($e^{\mathbb{E}[X]} \leq \mathbb{E}[e^X]$) [Chandler, 1987], (2) the second uses variational decomposition bounds in probabilistic graphical models applied to IDs with exponentiated utility functions. We summarize and conclude in Section 3.4.

3.1 Introduction

Decomposition methods for bounding graphical model inference queries are composed of two techniques: (1) the parameterization of the inference query by introducing auxiliary parameters over the graphical model decomposition, and subsequently, (2) the numerical procedures that optimize the auxiliary parameters for tightening upper-bounds. For example, Generalized Dual Decomposition (GDD) bounds for the mixed inference task, shown in Proposition 2.4, minimizes the upper bounds by optimizing some parameters, called cost-shifting functions and weights that are defined over a join-graph. Various variational decomposition bounds are available in the literature and they vary by the decomposition schemes, methods of parameterization, and the optimization frameworks [Komodakis et al., 2010, Sontag et al., 2011, Liu and Ihler, 2011, 2012, Ping et al., 2015]. The common characteristic of such variational decomposition bounds is that they decompose the original graphical model to a relaxed lower complexity one, compute the global bounds for the relaxed model, and optimize the global bounds by some additional local computations. We propose two approaches for bounding the MEU in IDs. The first, presented in Section 3.2, is described on top of the valuation algebra representation [Shenoy, 1989, 1997, Kohlas and Shenoy, 2000]. Namely, we extend the variational decomposition bounds to the valuation algebra for IDs [Jensen et al., 1994, Mauá et al., 2012]. In this approach, we develop a bounding scheme that processes pairs of probability and value functions $\big(P(\mathbf{X}), V(\mathbf{X})\big)$. Given two potentials $\big(P_1(\mathbf{X}), V_1(\mathbf{X})\big)$ and $\big(P_2(\mathbf{X}), V_2(\mathbf{X})\big)$, the combination of the two is the componentwise product for the probability function and the weighted sum for the value function, $\big(P_1(\mathbf{X})P_2(\mathbf{X}), V_1(\mathbf{X})P_2(\mathbf{X}) + V_2(\mathbf{X})P_1(\mathbf{X})\big)$. Therefore, we present a method that decomposes the product of the probability functions and the weighted sum of the value functions using variational bounds. In particular, we extend two known variational decomposition schemes to IDs: (1) the GDD bound [Ping et al., 2015] extended to the constrained join-graph decomposition of IDs, which leads to a scheme we call JGD-ID, and (2) the weighted

mini-bucket (WMB) bound [Dechter and Rish, 2003, Liu and Ihler, 2011] over the mini-bucket tree decomposition, which lead to a scheme called WMBE-ID. Both bounding schemes formulate constrained optimization problems over the decomposed graph in order to tighten the upper bounds, which take the form of a message passing algorithm. The second approach, presented in Section 3.3, does not use the valuation algebra. We generate upper-bounds in two stages. First, we exponentiate the utility functions based on Jensen's inequality for the exponential function yielding $\mathbb{E}[X] \leq \log \mathbb{E}[e^X]$ [Chandler, 1987], and then apply variational decomposition bounds to the resulting IDs. Once all utility functions are exponentiated, we can combine functions by multiplication resulting in a regular probabilistic graphical model to which we can apply decomposition bounds for the mixed inference task directly. In particular, we present two bounding algorithms on the exponentiated representation: (1) a GDD bound yielding JGD-EXP scheme, and (2) a Weighted Mini-bucket with Moment Matching (WMBMM) bound [Liu and Ihler, 2012, Marinescu et al., 2014] called WMBMM-EXP scheme.

3.2 Bounding Schemes using Valuation Algebra for IDs

We will build on concepts and definitions provided in Section 2.4.

3.2.1 Powered-Summation and Decomposition Bounds

We recap the definition of the combination operation for Jensen's potentials [Jensen et al., 1994, Mauá et al., 2012] given in Definition 2.38 for ease of reading.

Definition 3.1 (**Powered-summation**). *Given* $\Psi(\mathbf{X}) := \big(P(\mathbf{X}), V(\mathbf{X})\big)$*, where* $P(\mathbf{X})$ *and* $V(\mathbf{X})$ *stand for a probability and value component of an ID* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$ *as defined in Definition 2.35, we define the powered-summation that marginalizes out a variable* $X \in \mathbf{X}$

from $\Psi(\mathbf{X})$ *by*

$$\sum_{X}^{w} \Psi(\mathbf{X}) := \big(\sum_{X}^{w} P(\mathbf{X}), \sum_{X}^{w} V(\mathbf{X}) \big), \tag{3.1}$$

where w *is a weight* $0 < w \leq 1$*, and the powered-summation reduces to the summation and maximization when* $w = 1$ *and* $w = 0^+$*, respectively.*

Note that the powered-summation in Eq. (3.1) is defined by

$$\sum_{X}^{w} \big(P(\mathbf{X}), V(\mathbf{X})\big) := \big(\big[\sum_{X} \big| P(\mathbf{X}) \big|^{1/w} \big]^{w}, \big[\sum_{X} \big| V(\mathbf{X}) \big|^{1/w} \big]^{w} \big), \tag{3.2}$$

where the value component $\big[\sum_X \big| V(\mathbf{X}) \big|^{1/w} \big]^w$ uses the absolute value $\big| V(\mathbf{X}) \big|$ raised to the $1/w$-th power. Namely, the powered-summation cannot distinguish between the positive and negative value of $V(\mathbf{X})$ for the weights $0 < w < 1$. This could lead to bad bounds. To mitigate this issue, we introduce an alternative definition for the powered-summation as follows.

Definition 3.2 (**Modified Powered-summation**). *Given* $\Psi(\mathbf{X}) := \big(P(\mathbf{X}), V(\mathbf{X})\big)$ *and for any constant* A*, we define the modified powered-summation by*

$$\sum_{\mathbf{X}}^{(\mathbf{w},A)} \big(P(\mathbf{X}), V(\mathbf{X})\big) := \Big(\sum_{\mathbf{X}}^{\mathbf{w}} P(\mathbf{X}), \sum_{\mathbf{X}}^{\mathbf{w}} h_{(P(\mathbf{X}),V(\mathbf{X}),A)}(\mathbf{X}) \Big) \otimes (1, -A), \tag{3.3}$$

where $\mathbf{w}$ *is the set of weights* $(0 < w \leq 1)$ *associated with the given set of variables* $\mathbf{X}$*. The function* $h_{(P(\mathbf{X}),V(\mathbf{X}),A)}(\mathbf{X})$ *transforms the value function* $V(\mathbf{X})$ *into a non-negative function as follows. For any constant* A*,*

$$h_{(P(\mathbf{X}),V(\mathbf{X}),A)}(\mathbf{X}) = \begin{cases} P(\mathbf{X})(\frac{V(\mathbf{X})}{P(\mathbf{X})} + A) & \text{if } \frac{V(\mathbf{X}=\mathbf{x})}{P(\mathbf{X}=\mathbf{x})} + A > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

In Definition 3.2, we see that the function $h_{(P(\mathbf{X}),V(\mathbf{X}),A)}(\mathbf{X})$ adds a real-valued constant A, and it clips negative utility values where the constant A ensures

$$\frac{V(\mathbf{X})}{P(\mathbf{X})} + A \geq 0,$$

so that the absolute value will not change the sign of value functions, namely,

$$h_{(P(\mathbf{X}),V(\mathbf{X}),A)}(\mathbf{X}) = \left|h_{(P(\mathbf{X}),V(\mathbf{X}),A)}(\mathbf{X})\right|.$$

In addition, when the weights $\mathbf{w}$ are close to 1 and if $\frac{V(\mathbf{X})}{P(\mathbf{X})} + A$ is a non-negative function, the result of eliminating all the variables $\mathbf{X}$ from $V(\mathbf{X})$ by the modified powered-summation becomes close to the result using the normal summation. Namely:

$$\sum_{\mathbf{X}}^{\mathbf{w}} h_{(P(\mathbf{X}),V(\mathbf{X}),A)} \approx \sum_{\mathbf{X}} V(\mathbf{X}) + A. \tag{3.5}$$

This is why in the modified powered-summation in Eq. (3.3), it adds and subtracts a utility constant A from the value function, which thus converges to the regular sum-marginalization when the weights $\mathbf{w}$ are close to 1. Namely,

$$\lim_{\mathbf{w}\to\mathbf{1}} \sum_{\mathbf{X}}^{(\mathbf{w},A)} \big(P(\mathbf{X}), V(\mathbf{X})\big) = \lim_{\mathbf{w}\to\mathbf{1}} \Big[\Big(\sum_{\mathbf{X}}^{\mathbf{w}} P(\mathbf{X}), \sum_{\mathbf{X}}^{\mathbf{w}} h_{(P(\mathbf{X}),V(\mathbf{X}),A)}\Big) \otimes (1, -A)\Big]$$
$$= \Big(\lim_{\mathbf{w}\to\mathbf{1}} \sum_{\mathbf{X}}^{\mathbf{w}} P(\mathbf{X}), \lim_{\mathbf{w}\to\mathbf{1}} \Big[\sum_{\mathbf{X}}^{\mathbf{w}} h_{(P(\mathbf{X}),V(\mathbf{X}),A)} - A \cdot \sum_{\mathbf{X}}^{\mathbf{w}} P(\mathbf{X})\Big]\Big) \approx \Big(1, \sum_{\mathbf{X}} V(\mathbf{X})\Big).$$

Equipped with a new powered-summation operation, we state the decomposition bounds for IDs. But first we need some more notations. We denote the sequence of the modified

powered-summation with the constant $A = 0$ by $\sum_{\mathcal{O}}^{(\mathbf{w},0)}$, where we define

$$\sum_{\mathcal{O}}^{(\mathbf{w},0)} \triangleq \sum_{\mathrm{pa}(\mathbf{D}_1)}^{\mathbf{w}^{\mathrm{pa}(\mathbf{D}_1)}} \sum_{\mathbf{D}_1}^{\mathbf{w}_{\mathbf{D}_1}} \cdots \sum_{\mathrm{pa}(\mathbf{D}_M)}^{\mathbf{w}^{\mathrm{pa}(\mathbf{D}_M)}} \sum_{D_M}^{\mathbf{w}_{\mathbf{D}_M}} \sum_{\mathbf{X}\setminus\mathrm{pa}(\mathbf{D}_M)}^{\mathbf{w}^{\mathrm{pa}(\mathbf{D}_M)}}, \tag{3.6}$$

following a constrained ordering $\mathcal{O}$. In Eq. (3.6), the weights $\mathbf{w}^{\mathrm{pa}(D_k)}$ are 1 for the summation variables, while weights $\mathbf{w}^{\mathbf{D}_k}$ are 0 for the maximization variables.

Theorem 3.1 (**Decomposition Bounds for IDs**). *Given an ID* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$*, where* $\mathbf{X}$ *is a set of chance variables,* $\mathbf{D}$ *is a set of decision variables,* $\mathbf{\Psi}$ *is a set of potentials representing probability and utility functions,* $\mathcal{O}$ *is the constrained ordering, and given a set of constants* A^α *for* $\alpha \in \mathcal{I}_{\mathbf{\Psi}}$ *and a set of weights* $\mathbf{w}$ *and* $\mathbf{w}^\alpha$ *such that* $w_i = \sum_{\alpha \in \mathcal{I}_\Psi} w_i^\alpha$*, where* w_i *is the weight of* $X_i \in \mathbf{X}$ *and* w_i^α *is the weight of* X_i *at* $\Psi_\alpha(\mathbf{X}_\alpha)$*, the MEU can be bounded by*

$$\sum_{\mathcal{O}}^{(\mathbf{w},\,0)} \bigotimes_{\alpha \in \mathcal{I}_{\mathbf{\Psi}}} \Psi_\alpha(\mathbf{X}_\alpha) \leq \bigotimes_{\alpha \in \mathcal{I}_{\mathbf{\Psi}}} \sum_{\mathcal{O}}^{(\mathbf{w}^\alpha,\, A^\alpha)} \Psi_\alpha(\mathbf{X}_\alpha), \tag{3.7}$$

where the left-hand side of Eq. (3.7) *is the MEU shown in Eq.* (2.73)*.*

Proof. The decomposition bound can be obtained by applying Minkowski's inequality and Hölder's inequality [Hardy et al., 1952] shown in Eq. (3.8) and Eq. (3.9), respectively.

$$\text{Minkowski's inequality: } \sum_X^w f(X) + g(X) \leq \sum_X^w f(x) + \sum_X^w g(X) \tag{3.8}$$

$$\text{Hölder's inequality: } \sum_X^w f(X) \cdot g(X) \leq \sum_X^{w_1} f(x) \sum_X^{w-w_1} g(X) \tag{3.9}$$

Denoting the probability component and the value component of a potential $\Psi(\mathbf{X})$ by $\Big[\Psi(\mathbf{X})\Big]_1$ and $\Big[\Psi(\mathbf{X})\Big]_2$, namely, $\Psi(\mathbf{X}) := \Big(\Big[\Psi(\mathbf{X})\Big]_1, \Big[\Psi(\mathbf{X})\Big]_2\Big)$, we rewrite bounding inequalities for each component. The probability component in the left-hand side of Eq. (3.7)

can be bounded by applying Hölder's inequality, namely:

$$\left[\sum_{\mathcal{O}}^{(\mathbf{w},\,0)} \bigotimes_{\alpha\in\mathcal{I}_{\boldsymbol{\Psi}}} \Psi_\alpha(\mathbf{X}_\alpha)\right]_1 = \sum_{\mathcal{O}}^{\mathbf{w}} \prod_{i\in\mathcal{I}_{\boldsymbol{\Psi}}} P_i(\mathbf{X}_i) \leq \prod_{i\in\mathcal{I}_{\boldsymbol{\Psi}}} \sum_{\mathcal{O}}^{\mathbf{w}^i} P_i(\mathbf{X}_i). \tag{3.10}$$

The value component can be bounded by the following steps. We rewrite the MEU by introducing constant utilities A_i as shown in Eq. (3.11), and collect utility functions and constant utilities separately as shown in Eq. (3.12). The sum of the constant utilities $\sum_{i\in\mathcal{I}_{\boldsymbol{\Psi}}} A^i$ in Eq. (3.12) subtracts a constant expected utility from the MEU, so it is independent to the maximization operations. Then, we bound the MEU by the function h_i defined in Eq. (3.4) as shown in Eq. (3.13). The non-constant term in Eq. (3.13) can be further bounded by applying Minkowski's inequality yielding Eq. (3.14) and then by Hölder's inequality yielding Eq. (3.15).

$$\left[\sum_{\mathcal{O}}^{(\mathbf{w},\,0)} \bigotimes_{\alpha\in\mathcal{I}_{\boldsymbol{\Psi}}} \Psi_\alpha(\mathbf{X}_\alpha)\right]_2 = \sum_{\mathcal{O}}^{\mathbf{w}} \sum_{i\in\mathcal{I}_{\boldsymbol{\Psi}}} \Big(V_i(\mathbf{X}_i) + P_i(\mathbf{X}_i)\cdot(A^i - A^i)\Big)\Big(\prod_{j\neq i} P_j(\mathbf{X}_j)\Big) \tag{3.11}$$

$$= \sum_{\mathcal{O}}^{\mathbf{w}} \sum_{i\in\mathcal{I}_{\boldsymbol{\Psi}}} \Big[P_i(\mathbf{X}_i)\cdot\Big(\frac{V_i(\mathbf{X}_i)}{P_i(\mathbf{X}_i)} + A^i\Big)\cdot\Big(\prod_{j\neq i} P_j(\mathbf{X}_j)\Big)\Big] - \Big(\prod_{j\in\mathcal{I}_{\boldsymbol{\Psi}}} P_j(\mathbf{X}_j)\Big)\cdot\sum_{i\in\mathcal{I}_{\boldsymbol{\Psi}}} A^i \tag{3.12}$$

$$\leq \sum_{\mathcal{O}}^{\mathbf{w}} \sum_{i\in\mathcal{I}_{\boldsymbol{\Psi}}} \Big[h_{(P_i(\mathbf{X}_i),V_i(\mathbf{X}_i),A^i)}(\mathbf{X}_i)\Big(\prod_{j\neq i} P_j(\mathbf{X}_j)\Big)\Big] - \sum_{i\in\mathcal{I}_{\boldsymbol{\Psi}}} A^i \tag{3.13}$$

$$\leq \sum_{i\in\mathcal{I}_{\boldsymbol{\Psi}}} \sum_{\mathcal{O}}^{\mathbf{w}} \Big[h_{(P_i(\mathbf{X}_i),V_i(\mathbf{X}_i),A^i)}(\mathbf{X}_i)\Big(\prod_{j\neq i} P_j(\mathbf{X}_j)\Big)\Big] - \sum_{i\in\mathcal{I}_{\boldsymbol{\Psi}}} A^i \tag{3.14}$$

$$\leq \sum_{i\in\mathcal{I}_{\boldsymbol{\Psi}}} \Big(\sum_{\mathcal{O}}^{\mathbf{w}^i} h_{(P_i(\mathbf{X}_i),V_i(\mathbf{X}_i),A^i)}(\mathbf{X}_i)\Big)\Big(\prod_{j\neq i} \sum_{\mathcal{O}}^{\mathbf{w}^j} P_j(\mathbf{X}_j)\Big) - \sum_{i\in\mathcal{I}_{\boldsymbol{\Psi}}} A^i. \tag{3.15}$$

Note that the weights $\mathbf{w}$ at the left hand side of Eq. (3.10) and in Eq. (3.15) are associated with variables $\mathbf{X}$ and $\mathbf{w}^i$ at the right hand side of Eq. (3.10) and in Eq. (3.15) are associated with individual variables $\mathbf{X}_i$. The final result can be obtained by representing the upper bound on the right hand side of the probability component in Eq. (3.10), $\prod_{i\in\mathcal{I}_{\boldsymbol{\Psi}}} \sum_{\mathcal{O}}^{\mathbf{w}^i} P_i(\mathbf{X}_i)$, and the upper bound on the value component given in Eq. (3.15),

$\sum_{i\in\mathcal{I}_{\mathbf{\Psi}}}\Big(\sum_{\mathcal{O}}^{\mathbf{w}^i} h_{(P_i(\mathbf{X}_i),V_i(\mathbf{X}_i),A^i)}(\mathbf{X}_i)\Big)\Big(\prod_{j\neq i}\sum_{\mathcal{O}}^{\mathbf{w}^j} P_j(\mathbf{X}_j)\Big) - \sum_{i\in\mathcal{I}_{\mathbf{\Psi}}} A^i$ as a potential. Namely as the pair:

$$\Bigg(\prod_{i\in\mathcal{I}_{\mathbf{\Psi}}}\sum_{\mathcal{O}}^{\mathbf{w}^i} P_i(\mathbf{X}_i), \sum_{i\in\mathcal{I}_{\mathbf{\Psi}}}\Big(\sum_{\mathcal{O}}^{\mathbf{w}^i} h_{(P_i,V_i,A^i)}(\mathbf{X}_i)\Big)\Big(\prod_{j\neq i}\sum_{\mathcal{O}}^{\mathbf{w}^j} P_j(\mathbf{X}_j)\Big) - \sum_{i\in\mathcal{I}_{\mathbf{\Psi}}} A^i.\Bigg) \tag{3.16}$$

We will now show in a sequence of expression equalities that this leads to the right hand side of expression Eq. (3.7).

$$\Bigg(\prod_{i\in\mathcal{I}_{\mathbf{\Psi}}}\sum_{\mathcal{O}}^{\mathbf{w}^i} P_i(\mathbf{X}_i), \sum_{i\in\mathcal{I}_{\mathbf{\Psi}}}\Big(\sum_{\mathcal{O}}^{\mathbf{w}^i} h_{(P_i,V_i,A^i)}(\mathbf{X}_i)\Big)\Big(\prod_{j\neq i}\sum_{\mathcal{O}}^{\mathbf{w}^j} P_j(\mathbf{X}_j)\Big)\Bigg) \bigotimes \Big(1, -\sum_{i\in\mathcal{I}_{\mathbf{\Psi}}} A^i\Big) \tag{3.17}$$

$$= \Bigg[\bigotimes_{i\in\mathcal{I}_{\Psi}}\Big(\sum_{\mathcal{O}}^{\mathbf{w}^i} P_i(\mathbf{X}_i), \sum_{\mathcal{O}}^{\mathbf{w}^i} h_{(P_i,V_i,A^i)}(\mathbf{X}_i)\cdot\prod_{j\neq i}\sum_{\mathcal{O}}^{\mathbf{w}^j} P_j(\mathbf{X}_j)\Big)\Bigg] \bigotimes \Bigg[\bigotimes_{i\in\mathcal{I}_{\Psi}}\Big(1, -A^i\Big)\Bigg] \tag{3.18}$$

$$= \bigotimes_{i\in\mathcal{I}_{\Psi}}\Bigg[\Big(\sum_{\mathcal{O}}^{\mathbf{w}^i} P_i(\mathbf{X}_i), \sum_{\mathcal{O}}^{\mathbf{w}^i} h_{(P_i,V_i,A^i)}(\mathbf{X}_i)\cdot\prod_{j\neq i}\sum_{\mathcal{O}}^{\mathbf{w}^j} P_j(\mathbf{X}_j)\Big) \bigotimes \Big(1, -A^i\Big)\Bigg] \tag{3.19}$$

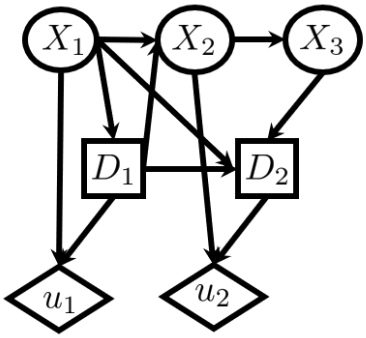
$$= \bigotimes_{i\in\mathcal{I}_{\Psi}}\sum_{\mathcal{O}}^{(\mathbf{w}^i,A^i)}(P_i(\mathbf{X}_i), V_i(\mathbf{X}_i)) = \bigotimes_{\alpha\in\mathcal{I}_{\Psi}}\sum_{\mathcal{O}}^{(\mathbf{w}^\alpha,A^\alpha)}\Psi_\alpha(\mathbf{X}_\alpha). \tag{3.20}$$

□

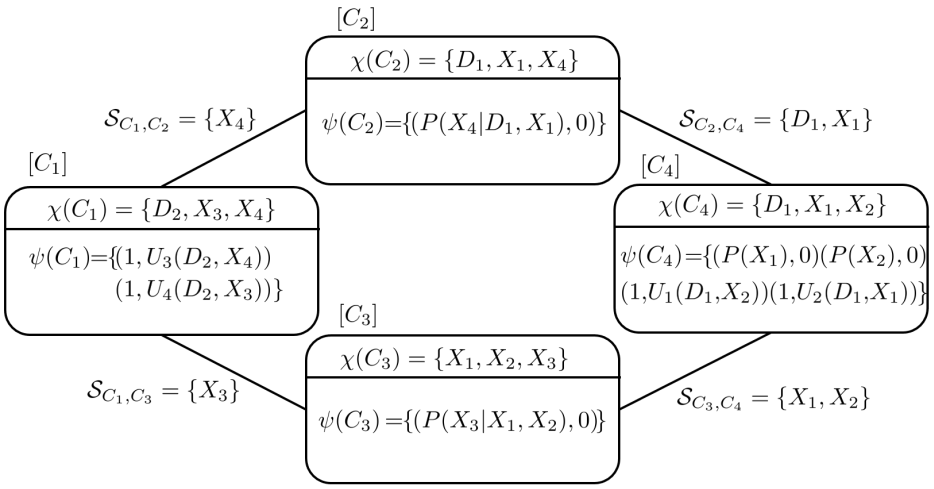
3.2.2 Join-graph Decomposition Bounds for IDs

Next, we will use Theorem 3.1 to extend GDD bounds [Ping et al., 2015] defined over a join-graph decomposition to IDs, yielding algorithm called JGD-ID described in Section 3.2.2. We subsequently also extend WMB bounds [Dechter, 2013, Liu, 2014] defined over mini-bucket tree decomposition, yielding a weighted mini-bucket elimination bounds for IDs, called WMBE-ID described in Section 3.2.3.

Given an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O}\rangle$, a join-graph decomposition $\mathcal{G}_{\text{JG}} := \langle G(\mathcal{C},\mathcal{S}), \chi, \psi\rangle$ can be structured from the mini-bucket tree of the constrained join-tree decomposition $\mathcal{T}_{\text{CJT}}$ as



(a) Perfect Recall ID



(b) Join-graph Decomposition

Figure 3.1: Join-graph Decomposition of an ID with the Valuation Algebra.

noted in Section 2.2.3. We next provide a running example illustrating our bounding scheme, JGD-ID. We start with the join-graph decomposition of IDs.

Example 3.1. *Figure 3.1a shows the graph of an ID. Figure 3.1b shows a possible join-graph decomposition* $\mathcal{G}_{JG} := \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$*. The primal graph* $\mathcal{G}_P$ *can be obtained by removing all informational arcs (which represent ordering constraints) before moralizing the parent nodes, and subsequently removing all value nodes (representing utility functions, see Section 2.4) after moralizing their parent nodes. From* $\mathcal{G}_p$ *and given a legal variable elimination ordering compatible with* $\mathcal{O}$*, the join-graph decomposition* $\mathcal{G}_{JG}$ *in Figure 3.1b was generated when*

limiting the maximum cluster size to 3. The labeling functions χ and ψ are displayed inside each node, and the separators $\mathcal{S}_{C_i,C_j}$ label the edges. Comparing with the constrained join-tree shown in Figure 2.6b, we see that we have an additional cluster node C_2 containing the probability function $P(X_4|D_1, X_1)$, which was extracted from Bucket X_4 in the constrained join-tree.

3.2.2.1 Derivation of Upper Bounds

Given a join-graph decomposition $\mathcal{G}_{\text{JG}} := \langle G(\mathcal{C}, \mathcal{S}), \chi, \Psi \rangle$ of an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$, we denote the combination of all the potentials at each cluster node $C \in \mathcal{C}$ by $\Psi_C(\mathbf{X}_C) = (P_C(\mathbf{X}_C), V_C(\mathbf{X}_C))$, where $\mathbf{X}_C = \chi(C)$ are the variables assigned at the cluster. We denote the potential $\Psi_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$ defined over the separator $(C_i, C_j) \in \mathcal{S}$ by

$$\Psi_{C_i,C_j}(\mathbf{X}_{C_i,C_j}) := \big(\lambda_{C_i,C_j}(\mathbf{X}_{C_i,C_j}),\ \ \eta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})\big).$$

The following Proposition 3.1 presents the parameterized decomposition bounds over a join-graph decomposition derived using Theorem 3.1.

Proposition 3.1. *Given an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$, $\mathbf{\Psi} := \{(P_i(\mathbf{X}_i), 0) | P_i \in \mathbf{P}\} \cup \{(1, V_i(\mathbf{X}_i) | V_i \in \mathbf{U}\}$ and a constant A and given a join-graph decomposition of $\mathcal{M}$, $\mathcal{G}_{JG} := \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$, then a parameterized decomposition bound can be obtained relative to $\mathcal{G}_{JG}$, which is denoted by $\Psi^U := \big(U_{prob}, U_{value}\big)$. Namely,*

$$MEU := \sum_{\mathcal{O}}^{(\mathbf{w},0)} \bigotimes_{\alpha \in \mathcal{I}_{\mathbf{\Psi}}} \Psi_\alpha(\mathbf{X}_\alpha) \leq \Psi^U, \tag{3.21}$$

where $\big(U_{prob}, U_{value}\big)$ *are defined by:*

$$U_{prob} = \prod_{C_i \in \mathcal{C}} \overset{\mathbf{w}^{C_i}}{\underset{\mathcal{O}}{\sum}} \tilde{P}_{C_i}(\mathbf{X}_{C_i}), \tag{3.22}$$

$$U_{value} = \sum_{C_i \in \mathcal{C}} \Big(\overset{\mathbf{w}^{C_i}}{\underset{\mathcal{O}}{\sum}} h_{(\tilde{P}_{C_i}(\mathbf{X}_{C_i}), \tilde{V}_{C_i}(\mathbf{X}_{C_i}), A_{C_i})}(\mathbf{X}_{C_i}) \Big) \cdot \Big(\prod_{C_j \neq C_i} \overset{\mathbf{w}^{C_j}}{\underset{\mathcal{O}}{\sum}} \tilde{P}_{C_j}(\mathbf{X}_{C_j}) \Big) - A^{C_i}, \tag{3.23}$$

and $\tilde{P}_C(\mathbf{X}_{C_i})$ *are defined by:*

$$\tilde{P}_C(\mathbf{X}_{C_i}) = P_C(\mathbf{X}_{C_i}) \prod_{(C_i, C_j) \in \mathcal{S}} \lambda_{C_i, C_j}(\mathbf{X}_{C_i, C_j}), \tag{3.24}$$

and $\tilde{V}_{C_i}(\mathbf{X}_{C_i})$ *are defined by:*

$$\tilde{V}_{C_i}(\mathbf{X}_{C_i}) = P_{C_i}(\mathbf{X}_{C_i}) \Big(\frac{V_{C_i}(\mathbf{X}_{C_i})}{P_{C_i}(\mathbf{X}_{C_i})} + \sum_{(C_i, C_j) \in \mathcal{S}} \frac{\eta_{C_i, C_j}(\mathbf{X}_{C_i, C_j})}{\lambda_{C_i, C_j}(\mathbf{X}_{C_i, C_j})} \Big), \tag{3.25}$$

and they obey the constraints

$$\Psi_{C_i, C_j}(\mathbf{X}_{C_i, C_j}) \triangleq \big(\lambda_{C_i, C_j}(\mathbf{X}_{C_i, C_j}), \eta_{C_i, C_j}(\mathbf{X}_{C_i, C_j}) \big), \tag{3.26}$$

$$\Psi_{C_i, C_j}(\mathbf{X}_{C_i, C_j}) \bigotimes \Psi_{C_j, C_1}(\mathbf{X}_{C_i, C_j}) = \big(\mathbf{1}(\mathbf{X}_{C_i, C_j}), \mathbf{0}(\mathbf{X}_{C_i, C_j}) \big) \quad \forall (C_i, C_j) \in \mathcal{S}. \tag{3.27}$$

In other wrods, $\tilde{P}_{C_i}(\mathbf{X}_{C_i})$ *and* $\tilde{V}_{C_i}(\mathbf{X}_{C_i})$ *are the reparameterized probability and value functions obtained by incorporating probability cost-shifting functions* $\lambda_{C_i, C_j}(\mathbf{X}_{C_i, C_j})$ *and utility cost-shifting functions* $\eta_{C_i, C_j}(\mathbf{X}_{C_i, C_j})$ *over the separators* $(C_i, C_j) \in \mathcal{S}$*, respectively.*

Proof. From Theorem 3.1 we can obtain the following inequality

$$\overset{(\mathbf{w}, 0)}{\underset{\mathcal{O}}{\sum}} \bigotimes_{\alpha \in \mathcal{I}_{\mathbf{\Psi}}} \Psi_\alpha(\mathbf{X}_\alpha) \leq \bigotimes_{C \in \mathcal{C}} \overset{(\mathbf{w}^C, A^C)}{\underset{\mathcal{O}}{\sum}} \Psi_C(\mathbf{X}_C) \tag{3.28}$$

by collecting potentials relative to a join-graph decomposition $\mathcal{G}_{\text{JG}} := \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$, and

introducing cost-shifting functions $\Psi_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$ over all separators $(C_i, C_j) \in \mathcal{S}$ such that $\Psi_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$ and $\Psi_{C_j,C_i}(\mathbf{X}_{C_j,C_i})$ cancel each other. Namely, $\Psi_{C_i,C_j}(\mathbf{X}_{C_i,C_j}) \otimes \Psi_{C_j,C_i}(\mathbf{X}_{C_j,C_i}) = (\mathbf{1}(\mathbf{X}_{C_j,C_i}), \mathbf{0}(\mathbf{X}_{C_j,C_i}))$. Re-arranging the potentials and introducing the above cost-shifting functions, we can obtain the following reparameterization over the join-graph:

$$\bigotimes_{\alpha \in \mathcal{I}_{\mathbf{\Psi}}} \Psi_\alpha(\mathbf{X}_\alpha) = \Big[\bigotimes_{C \in \mathcal{C}} \big(\bigotimes_{\alpha \in \mathcal{I}_{\psi(C)}} \Psi_\alpha(\mathbf{X}_\alpha) \big) \Big] \otimes \Big[\bigotimes_{(C_i,C_j) \in \mathcal{S}} \Psi_{C_i,C_j}(\mathbf{X}_{C_i,C_j}) \Big] \tag{3.29}$$

$$= \bigotimes_{C_i \in \mathcal{C}} \Big[\Psi_{C_i}(\mathbf{X}_{C_i}) \otimes \Big(\bigotimes_{C_i,C_j \in \mathcal{S}} \Psi_{C_i,C_j}(\mathbf{X}_{C_i,C_j}) \Big) \Big]. \tag{3.30}$$

Applying the inequality Eq. (3.7) of Theorem 3.1 to Eq. (3.30), we can obtain the desired result. Namely,

$$\sum_{\mathcal{O}}^{\mathbf{w},0} \bigotimes_{\alpha \in \mathcal{I}_{\mathbf{\Psi}}} \Psi_\alpha(\mathbf{X}_\alpha) \leq \bigotimes_{C_i \in \mathcal{C}} \sum_{\mathcal{O}}^{\mathbf{w}^{C_i}, A^{C_i}} \bigotimes_{C_i \in \mathcal{C}} \Big[\Psi_{C_i}(\mathbf{X}_{C_i}) \otimes \Big(\bigotimes_{C_i,C_j \in \mathcal{S}} \Psi_{C_i,C_j}(\mathbf{X}_{C_i,C_j}) \Big) \Big]. \tag{3.31}$$

To get the expression for U_{prob} and U_{value} as in Eq. (3.21), we should continue and unpack the expression in Eq. (3.31). □

Proposition 3.1 introduces four kinds of optimization parameters: (1) probability cost-shifting functions $\lambda_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$, (2) utility cost-shifting functions $\eta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$, (3) utility constants A_C introduced through the function $h_{P_C(\mathbf{X}_C),V_C(\mathbf{X}_C),A_C}$, and (4) weight parameters $\mathbf{w}^{C_i}$ that satisfies the following condition, $w_X = \sum_{C_i \in \mathcal{C}} w_X^{C_i}$ for all $X \in \chi(C_i)$.

Example 3.2. *Figure 3.2 illustrates the optimization parameters introduced in Proposition 3.1 relative to the join-graph decomposition $\mathcal{G}_{JG} := \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ in Figure 3.1b. The potentials at each node $C_i \in \mathcal{C}$ are displayed inside each node in Figure 3.1b and the utility constants A_{C_i} are attached next to each node. The cost-shifting potentials $\delta_{C_i,C_j} := (\lambda_{C_i,C_j}, \eta_{C_i,C_j})$ are shown next to the directed edges from C_i to C_j implying that the cost is moving from C_i to C_j, and $\delta_{C_i,C_j} = (\lambda_{C_i,C_j}, \eta_{C_i,C_j})$ and $\delta_{C_j,C_i} = (\lambda_{C_j,C_i}, \eta_{C_j,C_i})$ cancel each other. Namely,*

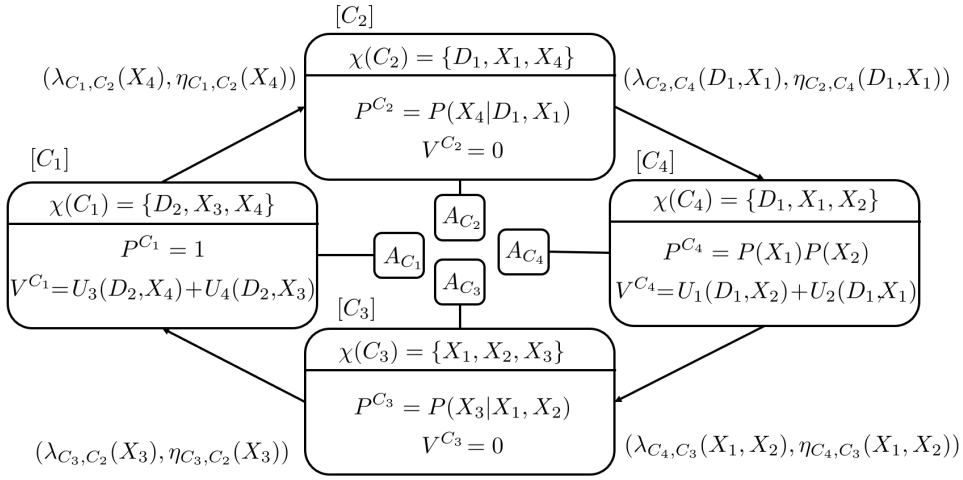


Figure 3.2: The Optimization Parameters for the Join-graph Decomposition Bounds for the ID in Figure 3.1b.

Algorithm 3.1 Block Coordinate Descent (BCD)

Require: An objective function $F(\mathbf{X})$ of a minimization problem , a set of optimization parameters $\mathbf{X}$, iteration limit M, inner-optimization routine `Inner-Optimization`

Ensure: a local optimal solution U^* and the assignments $\mathbf{X}^*$

Initialization Steps of BCD

1: Partition optimization parameters $\mathbf{X}$ to N block s.t. $\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2 \cup \cdots \cup \mathbf{X}_N$.

2: Initialize all parameters $\mathbf{X}_1, \mathbf{X}_2, \ldots \mathbf{X}_N$ to $\tilde{\mathbf{X}}_1, \tilde{\mathbf{X}}_2, \ldots \tilde{\mathbf{X}}_N$

Outer-loop of BCD

3: $iter$=0, $U^* = \inf$

4: **while** $iter < M$ or U^* is not converged **do**

Inner-loop of BCD

5: **for** each block $\mathbf{X}_i \in \{\mathbf{X}_1, \mathbf{X}_2, \ldots \mathbf{X}_N\}$ **do**

6: $U^* \leftarrow \min$ (U^*, INNER-OPTIMIZATION($F(\tilde{\mathbf{X}}_1, \tilde{\mathbf{X}}_2, \ldots, \mathbf{X}_i, \tilde{\mathbf{X}}_{i+1}, \tilde{\mathbf{X}}_N)$, $\mathbf{X}_i$))

7: $\tilde{\mathbf{X}}_i \leftarrow \mathbf{X}_i^*$ ▷ update $\tilde{\mathbf{X}}_i$ by the local optimal solution $\mathbf{X}_i^*$.

8: $iter = iter + 1$

return U^*

$\lambda_{C_j,C_i} = \frac{1}{\lambda_{C_i,C_j}}$, $\eta_{C_j,C_i} = -\frac{\eta_{C_i,C_j}}{\lambda^2_{C_i,C_j}}$ *and* $\delta_{C_i,C_j} \otimes \delta_{C_j,C_i} = (\mathbf{1}, \mathbf{0})$

3.2.2.2 Optimizing the Parameterized Bounds

We will use the block coordinate descent (BCD) method [Boyd et al., 2004] to optimize the bound. The BCD method shown in Algorithm 3.1 is an iterative procedure that has an

Algorithm 3.2 Join-Graph GDD Bounds for IDs (JGD-ID)

Require: ID $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$, weights w_{X_i} associated with each variable $X_i \in \mathbf{X}$, i-bound, iteration limit M for the outer-loop of BCD.

Ensure: an upper bound of the MEU, U_{value}

Initialization Steps of BCD

1: Generate a join-graph decomposition $\mathcal{G}_{\text{JG}} = \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ with the i-bound

2: Execute a single pass cost-shifting by the messages generated by MBE algorithm.

3: Initialize the weights for each chance variable $X_i \in \mathbf{X}$ to uniform weights: $w^C_{X_i} = \frac{1}{|\{C|X_i \in \chi(C)\}|}$, and for each decision variable $D_i \in \mathbf{D}$, $w^C_{D_i} \approx 0$.

Outer-loop of BCD

4: $iter$=0, $U_{\text{value}} = \inf$

5: **while** $iter < M$ or U_{value} is not converged **do**

Inner-loop of BCD

6: **for** each variable $X_i \in \mathbf{X}$ **do**

7: $U_{\text{value}} \leftarrow \min(U_{\text{value}}, \text{UPDATE-WEIGHTS}(\mathcal{G}_{\text{JG}}, X_i))$ ▷ see Algorithm 3.3

8: **for** each edge $(C_i, C_j) \in \mathcal{S}$ **do**

9: $U_{\text{value}} \leftarrow \min(U_{\text{value}}, \text{UPDATE-COSTS}(\mathcal{G}_{\text{JG}}, (C_i, C_j)))$ ▷ see Algorithm 3.4

10: **for** each cluster $C_i \in \mathcal{C}$ **do**

11: $U_{\text{value}} \leftarrow \min(U_{\text{value}}, \text{UPDATE-UTIL-CONST}(\mathcal{G}_{\text{JG}}, C_i))$ ▷ see Algorithm 3.5

12: $iter = iter + 1$

return U_{value}

inner-loop and an outer-loop. In the inner-loop, BCD cycles through a subset of parameters, called a block, and each iteration minimizes a single block of parameters while keeping the rest fixed. For the inner-optimization routine, we can apply any off-the-shelf optimization routine. The common choices are gradient based first or second order optimization algorithms [Nocedal and Wright, 2006]. The outer-loop repeats the inner-loop until convergence or other termination criteria met. In earlier works on variational decomposition bounds, BCD was shown to perform well on maximization tasks [Sontag et al., 2011] and on mixed inference tasks [Ping et al., 2015]. For more details on the scheme see [Sontag et al., 2011, Ping et al., 2015].

Message Passing Algorithm (JGD-ID) We now present our JGD-ID algorithm for minimizing the upper bound U_{value} shown in Eq. (3.21) using the BCD method. Given an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$, and a join-graph decomposition $\mathcal{G} := \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$, we use the param-

eters over the join-graph as defined in Proposition 3.1. Namely, we have pairs of cost-shifting functions $(\lambda_{C_i,C_j}, \eta_{C_i,C_j})$ over separators $(C_i, C_j) \in \mathcal{S}$, weight parameters $\mathbf{w}_X = \{w_X^C | X \in \chi(C)\}$ for each variable $X \in \mathbf{X}$ over a sub-tree of the $G(\mathcal{C}, \mathcal{S})$, and utility constants A_C over clusters $C \in \mathcal{C}$. In the inner-loop of BCD method, each iteration optimizes a block of optimization parameters dictated by the join-graph decomposition $\mathcal{G}_{\text{JG}}$.

Algorithm 3.2 outlines the overall procedure of BCD updates in JGD-ID. Given an input ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$, the initialization step generates a join-graph decomposition $\mathcal{G}_{\text{JG}} = \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ with an input i-bound (line 1). For details on how to structure a join-graph decomposition, see [Mateescu et al., 2010]. Then, (line 2) we assign probability and utility functions to the clusters in $\mathcal{C}$ defining the labeling function ψ and χ, by executing a single pass cost-shifting over the join-graph using the messages generated by simple MBE [Dechter and Rish, 2003] applied to the valuation algebra [Jensen et al., 1994, Mauá et al., 2012]. We subsequently (line 3) initialize uniform weights $w_{X_i}^C$. The outer-loop of BCD begins by initializing the iteration counter and the minimum upper bound U_{value} to an arbitrary large number (line 4), and repeat the inner-loop until it reaches the iteration limit M or U_{value} converges (line 5). The inner-loop of JGD-ID minimizes over a subset of weight parameters $\mathbf{w}_X = \{w_X^C | X \in \chi(C)\}$ for each variable X by iterating over the clusters $\{C | X \in \chi(C)\}$ in a sub-tree that satisfies the running intersection property and calling `UPDATE-WEIGHTS`$(\mathcal{G}_{\text{JG}}, X_i)$ in Algorithm 3.3 (described ahead) (line 6-7), a pair of cost-shifting functions $(\lambda_{C_i,C_j}, \eta_{C_i,C_j})$ by visiting each edge $(C_i, C_j) \in \mathcal{S}$ and calling `UPDATE-COSTS`$(\mathcal{G}_{\text{JG}}, (C_i, C_j))$, in Algorithm 3.4 (described ahead) (line 8-9), and a utility constant A_C by visiting each cluster $C \in \mathcal{C}$ and calling `UPDATE-UTIL-CONST`$(\mathcal{G}_{\text{JG}}, X_i)$ in Algorithm 3.5 (described ahead) (line 10-11).

Next, we will present each optimization routine for updating the three types of parameters using gradient descent with line search [Nocedal and Wright, 2006], which is a popular choice in machine learning for numerical optimization due to its simplicity and efficiency [Murphy, 2012].

Gradient Descent for Minimizing U_{value} Given a minimization problem with an objective function $F(\mathbf{X})$ over a set of parameters $\mathbf{X}$, a gradient descent algorithm iteratively updates the parameters from the t-th step to the next step $t+1$ by using the gradient of $F(\mathbf{X})$ evaluated at the parameters $\mathbf{X}^t$ at the step t,

$$\mathbf{X}^{t+1} \leftarrow \mathbf{X}^t - s \cdot \nabla F(\mathbf{X}^t), \tag{3.32}$$

where s is a real-valued step size parameter determined by the line search algorithm that finds the s that satisfies $F(\mathbf{X}^t - s \cdot \nabla F(\mathbf{X}^t)) < \mathbf{X}^t)$. For more details on the gradient descent with line search, see [Nocedal and Wright, 2006].

We can see that computing the gradient of U_{value} is the core component for implementing the three optimization routines `UPDATE-WEIGHTS`$(\mathcal{G}_{\text{JG}}, X_i)$, `UPDATE-COSTS`$(\mathcal{G}_{\text{JG}}, (C_i, C_j))$, and `UPDATE-UTIL-CONST`$(\mathcal{G}_{\text{JG}}, X_i)$. Therefore, we will provide some details of the gradients of U_{value} with respect to the parameters w_X^C, $\lambda_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$, $\eta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$, and A_C. First, we define the concept of pseudo marginals [Wainwright and Jordan, 2008, Liu, 2014, Ping et al., 2015] that plays a central part in this approach, and other useful expressions for deriving the gradients.

Definition 3.3 (**Pseudo Marginals** [Liu, 2014])**.** *Given a non-negative real-valued function $Z_0(\mathbf{X}_{1:N})$ over a set of variables $\mathbf{X}_{1:N} = \{X_1, \ldots, X_N\}$, and the weights $\mathbf{w} = \{w_1, \ldots, w_N\}$ associated with $\mathbf{X}_{1:N}$, we define a partial powered-sum elimination of variables ranging from X_1 to X_i, $\mathbf{X}_{1:i} = \{X_1, \ldots, X_i\}$ recursively by*

$$Z_i(\mathbf{X}_{i+1:N}) \triangleq \sum_{x_1}^{w_1} \sum_{x_2}^{w_2} \cdots \sum_{x_i}^{w_i} Z_{i-1}(\mathbf{X}_{i:N}). \tag{3.33}$$

We define the pseudo marginal of $Z_0(\mathbf{X}_{1:N})$ denoted $\Lambda\big(Z_0(\mathbf{X}_{1:N})\big)$ by:

$$\Lambda\big(Z_0(\mathbf{X}_{1:N})\big) \triangleq \Big(\frac{Z_{N-1}(X_N)}{Z_N}\Big)^{1/w_N} \cdots \Big(\frac{Z_0(\mathbf{X}_{1:N})}{Z_1(\mathbf{X}_{2:N})}\Big)^{1/w_1}. \tag{3.34}$$

Note that $\Lambda(Z_0(\mathbf{X}_{1:N}))$ is a normalized probability distribution over variables $\mathbf{X}_{1:N}$, and each $\left(\frac{Z_{i-1}(\mathbf{X}_{i:N})}{Z_i(\mathbf{X}_{i+1:N})}\right)^{1/w_i}$ can be viewed as a conditional distribution over the variable X_i given $\mathbf{X}_{i+1:N} = \{X_{i+1}, \ldots, X_n\}$.

Next, let's define a selector function $F_i(C_j)$ that selects a probability or a value function from cluster C_j depending on the given index i as,

$$F_i(C_j) \triangleq \begin{cases} h_{(P_{C_j}, V_{C_j}, A_{C_j})}(\mathbf{X}_{C_j}), & \text{if } j = i \\ P_{C_j}(\mathbf{X}_{C_j}), & \text{otherwise.} \end{cases} \tag{3.35}$$

We will denote the product of the upper bounds on the probability functions from all clusters $C \in \mathcal{C}$ by ρ, which is defined by:

$$\rho \triangleq \sum_{\mathbf{X}_{C_1}}^{\mathbf{w}^{C_1}} P_{C_1}(\mathbf{X}_{C_1}) \sum_{\mathbf{X}_{C_2}}^{\mathbf{w}^{C_2}} P_{C_2}(\mathbf{X}_{C_2}) \cdots \sum_{\mathbf{X}_{C_{|\mathcal{C}|}}}^{\mathbf{w}^{C_{|\mathcal{C}|}}} P_{C_{|\mathcal{C}|}}(\mathbf{X}_{C_{|\mathcal{C}|}}). \tag{3.36}$$

The upper bound on the value function at cluster C can be simplified after substituting terms defined in Eq. (3.35) and (3.36). Namely,

$$\theta_{C_i} \triangleq \rho \cdot \frac{\sum_{\mathbf{X}_{C_i}}^{\mathbf{w}^{C_i}} h_{P_{C_i}, V_{C_i}, A_{C_i}}(\mathbf{X}_{C_i})}{\sum_{\mathbf{X}_{C_i}}^{\mathbf{w}^{C_i}} P_{C_i}(\mathbf{X}_{C_i})} = \prod_{C_j \in \mathcal{C}} \sum_{\mathbf{X}_{C_j}}^{\mathbf{w}^{C_j}} F_i(C_j). \tag{3.37}$$

Updating Weights The `UPDATE-WEIGHTS` routine in Algorithm 3.2 updates the weights $w_{X_i}^{C_j}$ for each variable X_i over clusters $\{C | X_i \in \chi(C)\}$. Following the implementation of GDD algorithm for the mixed inference task [Ping et al., 2015], we also used exponentiated gradient descent algorithm [Kivinen and Warmuth, 1997], which is a variant of gradient descent algorithm that exponentiates the objective function. Due to exponentiation, the

parameter update equation is given by

$$w_{X_k}^{C_i,t+1} \leftarrow \frac{w_{X_k}^{C_i,t} \exp\big[- s \cdot \big(\nabla_{w_{X_k}^{C_i}} U_{\text{value}}(w_{X_k}^{C_i,t})\big)\big]}{\sum_{C_j \in \mathcal{C}} w_{X_k}^{C_j,t} \exp\big[- s \cdot \big(\nabla_{w_{X_k}^{C_j}} U_{\text{value}}(w_{X_k}^{C_j,t})\big)\big]}, \tag{3.38}$$

where $w_{X_k}^{C_i,t}$ is the weight parameter for a variable X_k in cluster C_i at the t-th iteration of the gradient descent algorithm and s is the step size determined by line search [Nocedal and Wright, 2006]. Following standard methods for partial derivatives [Gillespie, 1955], we obtained the following closed-form expression for the partial derivatives of U_{value} with respect to $w_{X_k}^{C_i}$.

$$\begin{aligned}\frac{\partial U_{\text{value}}}{\partial w_{X_k}^{C_i}} =& w_{X_k}^{C_i} \cdot \Big[\big\{H_{\Lambda(P_{C_i}(\mathbf{X}_{C_i}))}\big(X_k|\mathbf{X}_{k+1:N}\big) \cdot \big(\sum_{C_l \in \{C|X_k \in \chi(C)\}} \theta_{C_l} - \theta_{C_i}\big) - \bar{H}_{\text{prob}}\big\} \\ &+ \big\{H_{\Lambda(F_i(C_i)(\mathbf{X}_{C_i}))}\big(X_k|\mathbf{X}_{k+1:N}\big) \cdot \theta_{C_i} - \bar{H}_{\text{value}}\big\}\Big],\end{aligned} \tag{3.39}$$

where,

$$H_{\Lambda(F(\mathbf{X}))}(X_k|\mathbf{X}_{k+1:N}) \triangleq -\sum_{\mathbf{X}} \Lambda(F(\mathbf{X})) \log \Lambda(X_k|\mathbf{X}_{k+1:N}), \tag{3.40}$$

$$\bar{H}_{\text{prob}} \triangleq \sum_{C_i} w_{X_k}^{C_i} H_{\Lambda(P_{C_i}(\mathbf{X}_{C_i}))}\big(X_k|\mathbf{X}_{k+1:N}\big) \cdot \big(\sum_{C_l \in \{C|X_k \in \chi(C)\}} \theta_{C_l} - \theta_{C_i}\big), \tag{3.41}$$

$$\bar{H}_{\text{value}} \triangleq \sum_{C_i} w_{X_k}^{C_i} H_{\Lambda(F_i C_i(\mathbf{X}_{C_i}))}\big(X_k|\mathbf{X}_{k+1:N}\big) \cdot \theta_{C_i}. \tag{3.42}$$

We see therefore from Eq. (3.39) that the partial derivative of U_{value} with respect to $w_{X_k}^{C_i}$ can be expressed as a function of conditional entropies $H_{\Lambda(P_{C_i}(\mathbf{X}_{C_i}))}$ and $H_{\Lambda(F_i(C_i)(\mathbf{X}_{C_i}))}$ of the pseudo marginals obtained from probability and value functions at cluster C_i, and the upper bound of the value functions at each cluster θ_{C_i} after rearranging terms. Algorithm 3.3 summarizes the gradient descent updates using the closed-form expression in Eq. (3.39).

Algorithm 3.3 UPDATE-WEIGHTS($\mathcal{G}_{\text{JG}}, X_i$) – JGD-ID

Require: a join-graph decomposition $\mathcal{G}_{\text{JG}}, X_i)$ of an ID $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$, weights $\mathbf{w}_{X_i} = \{w_{X_i}^{C_i} | X_i \in \chi(C_i)\}$ associated with variable $X_i \in \mathbf{X}$, iteration limit M for the gradient descent

Ensure: an upper bound of the MEU, U_{value},

1: t=0
2: **while** $t < M$ or U_{value} is not converged **do**
3: Compute gradients of U_{value} with respect to $\mathbf{w}_{X_i}$ using Eq. (3.39)
4: Update weights $\mathbf{w}_{X_i}^{t+1}$ from $\mathbf{w}_{X_i}^{t}$ using Eq. (3.38)
5: Compute new U_{value} with the updated weights $\mathbf{w}_{X_i}^{t}$
6: $t = t + 1$

Updating Cost-shifting Functions The `UPDATE-COSTS` routine in Algorithm 3.2 updates the cost-shifting functions $\big(\lambda_{C_i,C_j}(\mathbf{X}_{C_i,C_j}), \eta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})\big)$ over separators $(C_i, C_j) \in \mathcal{S}$ by a gradient descent algorithm using Eq. (3.32), where we obtain a closed-form gradient expression for the cost-shifting paramters by evaluating the partial derivatives of U_{value} in Eq. (3.23) with respect to $\big(\lambda_{C_i,C_j}(\mathbf{X}_{C_i,C_j}), \eta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})\big)$ as follows.

$$\frac{\partial U_{\text{value}}}{\partial \lambda_{C_i,C_j}(\mathbf{X}_{C_i,C_j})} = \sum_{C_k \in \mathcal{C}} \theta_{C_k} \cdot \Big[\sum_{\mathbf{X}_{C_j} \setminus \mathbf{X}_{C_i,C_j}} \Lambda\big(F_k(C_j)(\mathbf{X}_{C_j})\big) - \sum_{\mathbf{X}_{C_j} \setminus \mathbf{X}_{C_i,C_j}} \Lambda\big(F_k(C_i)(\mathbf{X}_{C_i})\big) \Big], \tag{3.43}$$

$$\begin{aligned} \frac{\partial U_{\text{value}}}{\partial \eta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})} =& \left[\theta_{C_j} \cdot \sum_{\mathbf{X}_{C_j} \setminus \mathbf{X}_{C_i,C_j}} \Lambda\big(F_j(C_j)(\mathbf{X}_{C_j})\big) \frac{P_{C_j}(\mathbf{X}_{C_j})}{F_j(C_j)(\mathbf{X}_{C_j})} \right] \\ & - \left[\theta_{C_i} \cdot \sum_{\mathbf{X}_{C_i} \setminus \mathbf{X}_{C_i,C_j}} \Lambda\big(F_i(C_i)(\mathbf{X}_{C_i})\big) \frac{P_{C_i}(\mathbf{X}_{C_i})}{F_j(C_i)(\mathbf{X}_{C_i})} \right]. \end{aligned} \tag{3.44}$$

Algorithm 3.4 summarizes the gradient descent update steps for the cost shifting parameters. In each iteration, we first set the cost-shifting functions to a neutral potential $(\mathbf{1}(\mathbf{X}_{C_i,C_j}), \mathbf{0}(\mathbf{X}_{C_i,C_j}))$, which has constant 1 and 0 for the probability function and the value function, respectively (line 3). Then, we compute gradients and find the cost-shifting parameters using Eq. (3.32) (line 4-6), and then we reparameterize the functions at cluster C_i and C_j (line 7-8). We repeat this until we reach the iteration limit M or until U_{value} converges.

Algorithm 3.4 UPDATE-COSTS($\mathcal{G}_{\text{JG}}, (C_i, C_j)$) – JGD-ID

Require: a join-graph decomposition $\mathcal{G}_{\text{JG}}, X_i)$ of an ID $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$, separator (C_i, C_j) iteration limit M for the gradient descent

Ensure: an upper bound of the MEU, U_{value},

1: t=0
2: **while** $t < M$ or U_{value} is not converged **do**
3: $\big(\lambda_{C_i,C_j}(\mathbf{X}_{C_i,C_j}), \eta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})\big) \leftarrow (\mathbf{1}(\mathbf{X}_{C_i,C_j}), \mathbf{0}(\mathbf{X}_{C_i,C_j}))$
4: Compute gradient of U_{value} with respect to the probability cost functions by Eq. (3.43)
5: Compute gradient of U_{value} with respect to the value cost functions by Eq. (3.44)
6: Find cost-shifting functions $\big(\lambda_{C_i,C_j}(\mathbf{X}_{C_i,C_j}), \eta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})\big)$ by Eq. (3.32).
7: $\Psi_{C_i}(\mathbf{X}_{C_i}) \leftarrow \Psi_{C_i}(\mathbf{X}_{C_i}) \otimes \frac{1}{\big(\lambda_{C_i,C_j}(\mathbf{X}_{C_i,C_j}), \eta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})\big)}$
8: $\Psi_{C_j}(\mathbf{X}_{C_j}) \leftarrow \Psi_{C_j}(\mathbf{X}_{C_j}) \otimes \big(\lambda_{C_i,C_j}(\mathbf{X}_{C_i,C_j}), \eta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})\big)$
9: Compute new U_{value} with the updated $\Psi_{C_i}(\mathbf{X}_{C_i})$ and $\Psi_{C_j}(\mathbf{X}_{C_j})$
10: $t = t + 1$

Updating Utility Constants The `UPDATE-UTIL-CONSTS` routine in Algorithm 3.2 updates the utility constant A_C at each cluster $C \in \mathcal{C}$ by a gradient descent algorithm. Similar to the previous update routines for the weights and cost-shifting functions, we can obtain a closed-from gradient expression by evaluating the partial derivatives of U_{value} with respect to A_C yielding,

$$\frac{\partial U_{\text{value}}}{\partial A_{C_i}} = \Big[\theta_C \cdot \sum_{\mathbf{X}_{C_i} \setminus \mathbf{X}_{C_i,C_j}} \Lambda(F_i(C_i)(\mathbf{X}_{C_i})) \frac{P_{C_i}(\mathbf{X}_{C_i})}{F_i(C_i)(\mathbf{X}_{C_i})}\Big] - 1. \tag{3.45}$$

Algorithm 3.5 summarizes the gradient descent update steps for updating the utility constant parameter at cluster $C_i \in \mathcal{C}$. Similar to the previous parameter updates, we iteratively update utility constant parameters until the iteration limit or convergence of U_{value}. For each internal iteration, we repeat computing the gradient of U_{value} with respect to A_{C_i} using Eq. (3.45), and find the updated parameter using Eq. (3.32).

Complexity of JGD-ID We will next discuss the complexity of our algorithm.

Theorem 3.2 (**Complexity of Algorithm JGD-ID**). *Given an ID* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$ *and a join-graph decomposition* $\mathcal{G}_{JG} := \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ *structured from a mini-bucket tree with*

Algorithm 3.5 UPDATE-UTIL-CONSTS($\mathcal{G}_{\text{JG}}, C_i$) – JGD-ID

Require: a join-graph decomposition $\mathcal{G}_{\text{JG}}$ of an ID $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$, cluster node C_i, iteration limit M for the gradient descent

Ensure: an upper bound of the MEU, U_{value},

1: t=0
2: **while** $t < M$ or U_{value} is not converged **do**
3: Compute ∇U_{value} with respect to the utility constant A_{C_i} using Eq. (3.45)
4: $A^t_{C_i} \leftarrow A^t_{C_i} - s \cdot \nabla U_{\text{value}}$ ▷ Update utility constant A_{C_i} by Eq. (3.32).
5: Compute new U_{value} with the updated A_{C_i}
6: $t = t + 1$

an i-bound, the time complexity of JGD-ID is $O\big(M_1 \cdot M_2 \cdot (|\mathbf{X} \cup \mathbf{D}| \cdot |\mathcal{C}| \cdot G_w \cdot k^{i+1} + \cdot |\mathcal{S}| \cdot G_p \cdot k^{i+1} + \cdot |\mathcal{S}| \cdot G_v \cdot k^{i+1} + |\mathcal{C}| \cdot G_u \cdot k^{i+1})\big)$ *and the space complexity is* $O\big(2 \cdot (|\mathcal{C}| \cdot k^{i+1} + |\mathcal{S}| \cdot k^{|sep|})\big)$, *where* M_1 *is the iteration limit of the outer-loop of the BCD method,* M_2 *is the iteration limit of the gradient descent algorithms in the inner-loop of the BCD method,* $|\mathbf{X} \cup \mathbf{D}|$ *is the number of variables,* k *is the maximum domain size,* $|\mathcal{C}|$ *and* $|\mathcal{S}|$ *are the number of clusters and separators in the join-graph decomposition* $\mathcal{G}_{JG}$, *and* $|sep|$ *is the separator width.* G_w, G_p, G_v, *and* G_u *are constants that bounds the number of factor operations for evaluating gradients.*

Proof. Algorithm JGD-ID is an iterative algorithm that terminates the outer-loop and the inner-loop either by the iteration limit or by the convergence. The complexity of a single iteration of the outer-loop can be characterized by the complexity of the gradient based inner-optimization routines in Algorithm 3.3, Algorithm 3.4, and Algorithm 3.5. Since a gradient descent algorithm is an iterative algorithm that repeats updating the optimization parameters by Eq. (3.32), processing gradient expressions dominate the overall complexity. The closed-form expressions for evaluating gradients involve a constant number of factor operations over the pseudo marginals at each cluster $C \in \mathcal{C}$ (see Eq. (3.39), Eq. (3.43), Eq. (3.44), and Eq. (3.45)). Since pseudo marginals are computed locally within each cluster in a join-graph decomposition, the time complexity for processing a single factor operation over pseudo marginals is bounded by $O(k^{i+1})$.

The time complexity of in Algorithm 3.3 is $O(G_w \cdot M_2 \cdot k^{i+1})$ since it repeats a single gradient update at most M_2 times. The number of calls to Algorithm 3.3 in a single outer-loop iteration is bounded by $O(|\mathbf{X} \cup \mathbf{D}| \cdot |\mathcal{C}|)$, so the overall time complexity for updating the weight parameters is $O(M_1 \cdot |\mathbf{X} \cup \mathbf{D}| \cdot |\mathcal{C}| \cdot G_w \cdot M_2 \cdot k^{i+1})$. The time complexity of Algorithm 3.4 is $O((G_p + G_v) \cdot M_2 \cdot k^{i+1})$, and the overall time complexity for updating the cost-shifting parameters is $O(M_1 \cdot |\mathcal{S}| \cdot (G_p + G_v) \cdot M_2 \cdot k^{i+1})$, where $|\mathcal{S}|$ bounds the number of calls to Algorithm 3.4. The time complexity of Algorithm 3.5 is $O(G_u \cdot M_2 \cdot k^{i+1})$, and overall time complexity is $O(M_1 \cdot |\mathcal{C}| \cdot (G_u) \cdot M_2 \cdot k^{i+1})$, where $|\mathcal{C}|$ bounds the number of calls to Algorithm 3.5.

For the space complexity, we store pseudo marginals and cost-shifting functions for the probability component and the value component at clusters and separators in $\mathcal{G}_{\mathrm{JG}}$, respectively. Therefore, the space complexity is $O\big(2 \cdot (|\mathcal{C}| \cdot k^{i+1} + |\mathcal{S}| \cdot k^{|\mathrm{sep}|})\big)$. □

Summary We have presented a message-passing algorithm JGD-ID for computing the upper bound of the MEU in IDs over a join-graph decomposition [Mateescu et al., 2010] based on GDD bound in the mixed inference task [Ping et al., 2015]. We first modified the powered-sum operations for the valuation algebra to handle the issue with negative utility values. Then, we derived a decomposition bound for IDs that uses the valuation algebra representation, which extends the previous decomposition bounds for the MMAP task. We subsequently formulated an optimization problem over a join-graph decomposition of IDs by introducing optimization parameters along with the join-graph decomposition. Algorithm JGD-ID implemented a block coordinate descent type algorithm using gradient descent algorithms as its inner optimization routines.

3.2.3 Weighted Mini-bucket Elimination Bounds for IDs

The optimization objective function U_{value} for JGD-ID algorithm derived in Proposition 3.1 is not in a convex form because the upper bound expression of the MEU U_{value} involves the multiplication of the local upper bounds on the probability and value functions,as given next

$$U_{\text{value}}=\sum_{C_i\in\mathcal{C}}\left[\Big(\sum_{\mathcal{O}}^{\mathbf{w}^{C_i}} h_{(\tilde{P}_C(\mathbf{X}_{C_i}),\tilde{V}_{C_i}(\mathbf{X}_{C_i}),A_{C_i})}(\mathbf{X}_{C_i})\Big)\cdot\Big\{\prod_{C_j\neq C_i}\sum_{\mathcal{O}}^{\mathbf{w}^{C_j}}\Big(P_C(\mathbf{X}_{C_i})\prod_{(C_i,C_j)\in\mathcal{S}}\lambda_{C_i,C_j}(\mathbf{X}_{C_i,C_j})\Big)\Big\}-A^{C_i}\right],$$

$$\text{where, }\begin{cases}\tilde{P}_C(\mathbf{X}_{C_i})=P_C(\mathbf{X}_{C_i})\prod_{(C_i,C_j)\in\mathcal{S}}\lambda_{C_i,C_j}(\mathbf{X}_{C_i,C_j}),\\ \tilde{V}_{C_i}(\mathbf{X}_{C_i})=P_{C_i}(\mathbf{X}_{C_i})\Big(\frac{V_{C_i}(\mathbf{X}_{C_i})}{P_{C_i}(\mathbf{X}_{C_i})}+\sum_{(C_i,C_j)\in\mathcal{S}}\frac{\eta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})}{\lambda_{C_i,C_j}(\mathbf{X}_{C_i,C_j})}\Big).\end{cases}\tag{3.46}$$

This non-convex objective function (see Boyd et al. [2004] for the basic properties of convex functions) may degrade the quality of the upper bounds as the number of optimization parameters grows. Indeed, in our empirical evaluation (see Chapter 4), JGD-ID often provides worse upper bounds even when using higher i-bounds, consuming more time and memory resources. The degradation is due to the expansion of the dimension of the optimization parameter space, which grows exponentially with the i-bounds. This motivated developing an alternative approach, where we interleave the variable elimination and the decomposition of the clusters on the fly while performing the variational inference more locally. This can be achieved by using mini-bucket tree decomposition of IDs, resulting in our WMBE-ID, another new weighted mini-bucket elimination bounds for IDs. The algorithm will be described in the following paragraphs.

Given an ID $\mathcal{M}:=\langle\mathbf{X},\mathbf{D},\mathbf{\Psi},\mathcal{O}\rangle$, a mini-bucket tree decomposition $\mathcal{T}_{\text{MBT}}$ can be structured from the constrained join-tree decomposition $\mathcal{T}_{CJT}$ by limiting the maximum cluster size to the i-bound. For the details of obtaining a mini-bucket tree, see [Dechter and Rish, 2003]. We will use a running example for illustrating algorithm WMBE-ID.

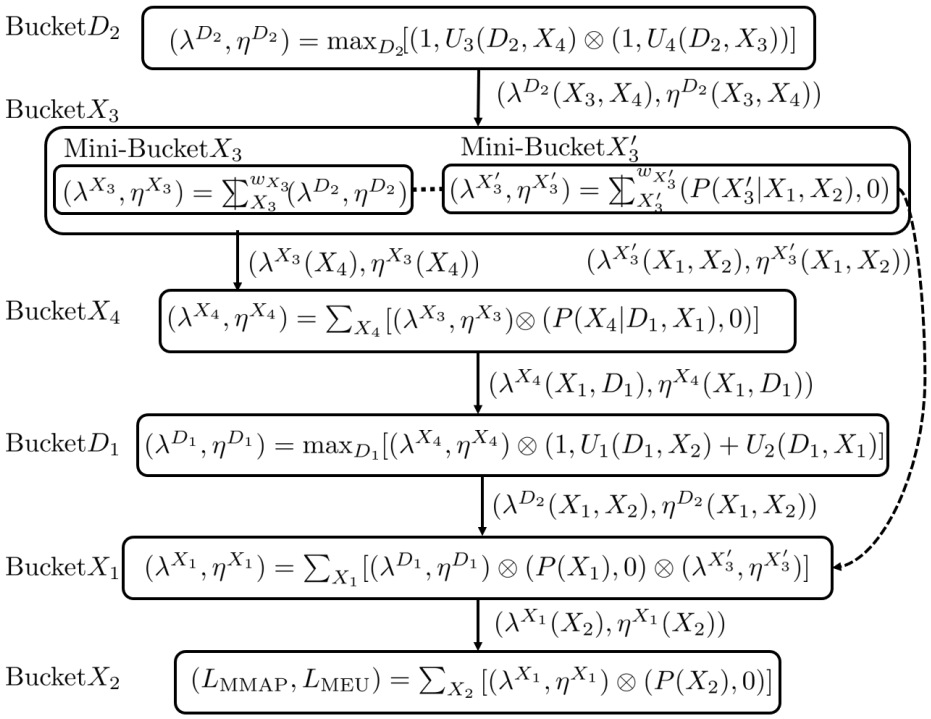


Figure 3.3: The Weighted Mini-bucket Decomposition of an ID with Valuation Algebra.

Example 3.3. *Figure 3.3 illustrates the schematic trace of weighted mini-bucket elimination applied to the ID in Figure 3.1a with i-bound 2. We will review the mini-bucket tree decomposition [Dechter and Rish, 2003] and introduce some necessary modification to the notation to accommodate Jensen's potentials [Jensen et al., 1994]. First, we use a constrained elimination ordering consistent with the precedence constraints $\mathcal{O}$ as dictated by the informational arcs, which is $\mathcal{O}_{elim} := \{D_2 \prec X_3 \prec X_4 \prec D_1 \prec X_1 \prec X_2\}$. To generate a mini-bucket tree, we process the variables along the order $\mathcal{O}_{elim}$. For each variable X_i, we collect all the functions having X_i in their scopes and place them in the bucket X_i. If 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 X_3 is partitioned into two mini-buckets in Figure 3.3. Subsequently, we generate messages by eliminating the bucket variable of each mini-bucket from its combined potential and send the message having a pair of probability and value functions to a mini-bucket in a lower layer along the tree. For ex-*

ample, the mini-bucket X_3' sends its message $(\lambda^{X_3'}(X_1, X_2), \eta^{X_3'}(X_1, X_2))$ to bucket X_1. We can also assign weights to each bucket labelled by a summation variable [Liu and Ihler, 2011] to facilitate use of powered-sum. We can obtain a weighted mini-bucket bound at bucket X_3 as follows

$$\sum_{X_3} \lambda^{D_2}(X_3, X_4) \otimes (P(X_3|X_1, X_2), 0) \leq \Big(\sum_{X_3}^{w_{X_3}} \lambda^{D_2}(X_3, X_4) \Big) \otimes \Big(\sum_{X_3}^{w_{X_3'}} (P(X_3|X_1, X_2), 0) \Big),$$

where the left-hand side is the exact message function computed at bucket X_3 in the bucket-tree elimination [Dechter, 2013], and the right-hand side is the combination of two mini-bucket message functions, from the mini-bucket X_3 and the mini-bucket X_3'.

3.2.3.1 Deriving the Upper-Bounds

We will now apply Theorem 3.1 to each layer of the mini-bucket tree associated with one variable, along the constrained elimination ordering $\mathcal{O}_{\text{elim}}$. As usual, the intermediate messages are sent to mini-buckets at lower layers, as illustrated in Figure 3.3. To tighten the upper bounds, we introduce cost-shifting functions between mini-buckets, yielding the following parameterized bound for each bucket of a variable X. Namely,

$$\sum_{X}^{w_X} \bigotimes_{\alpha=1}^{q} \Psi_\alpha(\mathbf{X}_\alpha) \leq \bigotimes_{\alpha=1}^{q} \sum_{X_\alpha}^{(w_X^\alpha, A)} \Big(\Psi_\alpha(\mathbf{X}_\alpha) \otimes \frac{\delta_{\alpha-1,\alpha}(X)}{\delta_{\alpha,\alpha+1}(X)} \Big), \tag{3.47}$$

where w_X is the weight of the variable X (1 or 0), q is the number of mini-buckets, w_X^α is the weight of X at its α-th mini-bucket, and $\delta_{\alpha,\alpha+1}(X) := (\lambda_{\alpha,\alpha+1}(X), \eta_{\alpha,\alpha+1}(X))$ is a cost-shifting potential from the α-th mini-bucket to the $(\alpha+1)$-th mini-bucket. In the example in Figure 3.3, the reparameterized upper bound at Bucket X_3 can be written as,

$$\Big(\sum_{X_3}^{(w_{X_3}, A)} \big(\lambda^{D_2}(X_3, X_4), \eta^{D_2}(X_3, X_4) \big) \otimes \frac{1}{\delta_{X_3, X_3'}(X_3)} \Big) \otimes \Big(\sum_{X_3'}^{(w_{X_3'}, A')} (P(X_3'|X_1, X_2), 0) \otimes \delta_{X_3, X_3'}(X_3) \Big).$$

However, we can immediately see some issues in Eq. (3.47) when we try to optimize the parameters such as the cost-shifting functions, the weight parameters, and the utility constant. First, the value component on the right-hand side of Eq. (3.47) is not a scalar quantity, but rather a function. Second, the powered-summation used in Theorem 3.1 is well-defined when we fully eliminate all variables, but here we only eliminate a single variable X. To be more specific, the partial elimination cannot use Definition 3.2 since the constants A will be multiplied by all the probability functions at other mini-buckets. As a result, it is not a simple shift of the overall expected utility value by the constant A.

To address all this, we propose a surrogate optimization objective function based on the fully decomposed bound in Theorem 3.1, which uses the powered-sum elimination operation in Definition 3.1 as follows.

Proposition 3.2. *Given an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$ and a constrained variable elimination ordering $\mathcal{O}_{elim}$ over all variables $\mathbf{X} \cup \mathbf{D}$, $\{X_N, X_{N-1}, \ldots, X_1\}$, assume that X_n is the current variable to be eliminated and the variables $\{X_N, X_{N-1}, \ldots, X_{n+1}\}$ were already processed by the weighted mini-bucket elimination algorithm [Dechter and Rish, 2003, Liu and Ihler, 2011]. For the remaining variables $\mathbf{X}_{1:n} = \{X_n, X_{n-1}, \ldots, X_1\}$, let $\Psi^{X_i}(\mathbf{X}_{1:i})$ be the combination of all potentials allocated to bucket X_i $(1 \leq i \leq n)$ of the bucket-tree, let $Q_{X_i} := \{1, \ldots, q_{X_i}\}$ be the mini-bucket partitioning of bucket X_i, and $\Psi_\alpha^{X_i}(\mathbf{X}_\alpha^{X_i})$ be the combination of the potentials at the α-th mini-bucket of the bucket X_i. Then, we define a surrogate upper bound on the MEU for the remaining subproblem as follows.*

$$(1, \mathit{MEU}) = \sum_{\mathbf{X}_{1:n-1}}^{\mathbf{w}_{1:n-1}} \Big[\Big(\bigotimes_{i=1}^{n-1} \Psi^{X_i}(\mathbf{X}_{1:i}) \Big) \otimes \Big(\sum_{X_n}^{w_{X_n}} \Psi^{X_n}(\mathbf{X}_{1:n}) \Big) \Big] \tag{3.48}$$

$$\leq \sum_{\mathbf{X}_{1:n-1}}^{\mathbf{w}_{1:n-1}} \Big[\Big(\bigotimes_{i=1}^{n-1} \Psi^{X_i}(\mathbf{X}_{1:i}) \Big) \otimes \Big(\bigotimes_{\alpha \in Q_{X_n}} \sum_{X_n}^{w_{X_n}^{X_n,\alpha}} \Psi_\alpha^{X_n}(\mathbf{X}_\alpha^{X_n}) \Big) \Big] \tag{3.49}$$

$$\leq \bigotimes_{i=1}^{n-1} \Big(\bigotimes_{\alpha \in Q_{X_i}} \sum_{\mathbf{X}_{1:i}}^{\mathbf{w}_{1:i}^{X_i,\alpha}} \Psi_\alpha^{X_i}(\mathbf{X}_\alpha^{X_i}) \Big) \otimes \Big(\bigotimes_{\alpha \in Q_{X_n}} \sum_{\mathbf{X}_{1:n}}^{\mathbf{w}_{1:n}^{X_n,\alpha}} \Psi_\alpha^{X_n}(\mathbf{X}_\alpha^{X_n}) \Big), \tag{3.50}$$

where the weights $\mathbf{w}_{1:n} := \{w_{X_1}, \ldots, w_{X_n}\}$ *in Eq.* (3.48) *are the set of original 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.* (3.50) *is a set of weights of all the variables* $\{X_1, X_2, \ldots, X_k\}$ *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. Eq. (3.48) is the MEU computed from the remaining subproblem after eliminating variables $\{X_N, X_{N-1}, \ldots, X_{n+1}\}$. The inequality in Eq. (3.49) is obtained by applying weighted mini-bucket relaxation to bucket X_n only. The final inequality, Eq. (3.50), is obtained by applying the JGD-ID upper-bounds to all the mini-buckets $\cup_{X_i \in \mathbf{X_{1..n}}} Q_{X_i}$ in the remaining subproblem, independently, yielding a scalar valued function that bounds the MEU. □

When variable X_n is the variable to be eliminated, we can introduce parameters relative for a chain of mini-buckets $Q_{X_n} = \{1, \ldots, q_{X_n}\}$ into the WMBE-ID bound according to Proposition 3.2, yielding the bound,

$$(1, \text{MEU}) = \sum_{\mathbf{X}_{1:n-1}}^{\mathbf{w}_{1:n-1}} \Big[\Big(\bigotimes_{i=1}^{n-1} \Psi^{X_i}(\mathbf{X}_{1:i}) \Big) \otimes \Big(\sum_{X_n}^{w_{X_n}} \bigotimes_{\alpha \in Q_{X_n}} \Psi_{\alpha}^{X_n}(\mathbf{X}_{\alpha}^{X_n}) \frac{\delta_{\alpha-1,\alpha}^{X_n}(X_n)}{\delta_{\alpha,\alpha+1}^{X_n}(X_n)} \Big) \Big] \tag{3.51}$$

$$\leq \sum_{\mathbf{X}_{1:n-1}}^{\mathbf{w}_{1:n-1}} \Big[\Big(\bigotimes_{i=1}^{n-1} \Psi^{X_i}(\mathbf{X}_{1:i}) \Big) \otimes \Big\{ \Big(\sum_{\mathbf{X}_{1:n}}^{\mathbf{w}_{1:n}^{X_n,1}} \Psi_1^{X_n}(\mathbf{X}_1^{X_n}) \frac{1}{\delta_{1,2}^{X_n}(X_n)} \Big) \otimes \Big(\sum_{\mathbf{X}_{1:n}}^{\mathbf{w}_{1:n}^{X_n,2}} \Psi_2^{X_n}(\mathbf{X}_2^{X_n}) \frac{\delta_{1,2}^{X_n}(X_n)}{\delta_{2,3}^{X_n}(X_n)} \Big)$$

$$\cdots \otimes \Big(\sum_{\mathbf{X}_{1:n}}^{\mathbf{w}_{1:n}^{X_n,q_{X_n}}} \Psi_{q_{X_n}}^{X_n}(\mathbf{X}_{q_{X_n}}^{X_n}) \delta_{q_{X_n}-1,q_{X_n}}^{X_n}(X_n) \Big) \Big\} \Big] \triangleq (U_{\text{prob}}, U_{\text{value}}) \tag{3.52}$$

The parameters to be optimized over in Eq.(3.52) denoted by $\delta_{\alpha-1,\alpha}(X_n)$ are the pair of cost-shifting functions between two adjacent mini-buckets, and the weights over the mini-buckets

in Q_{X_n} namely, we have the sets of parameters:

$$\text{Cost-shifting functions} : \{\delta^{X_n}_{\alpha,\alpha+1}(X_n) | \forall \alpha, \alpha+1 \in Q_{X_n}\} \tag{3.53}$$

$$\text{Weights} : \{w^{X_n,\alpha}_{X_n} | \forall \alpha, \alpha+1 \in Q_{X_n}\}, \tag{3.54}$$

where $\delta^{X_n}_{\alpha,\alpha+1}(X_n)$ is the pair $\big(\lambda^{X_n}_{\alpha,\alpha+1}(X_n), \eta^{X_n}_{\alpha,\alpha+1}(X_n)\big)$.

We can obtain a closed-form expression of the objective function U_{value} by some algebraic manipulation of Eq. (3.52). Dropping the arguments of the functions, we obtain:

$$U_{\text{value}} = \Gamma \cdot \left[\sum_{i=1}^{n-1} \Big(\sum_{\alpha \in Q_{X_i}} \frac{\sum_{\mathbf{X}_{1:n-1}}^{\mathbf{w}^{X_i,\alpha}_{1:n-1}} V^{X_i}_{\alpha}}{\sum_{\mathbf{X}_{1:n-1}}^{\mathbf{w}^{X_i,\alpha}_{1:n-1}} P^{X_i}_{\alpha}} \Big) + \sum_{\alpha \in Q_{X_n}} \frac{\sum_{\mathbf{X}_{1:n}}^{\mathbf{w}^{X_n,\alpha}_{1:n}} P^{X_n}_{\alpha} \frac{\lambda^{X_n}_{\alpha-1,\alpha}}{\lambda^{X_n}_{\alpha,\alpha+1}} \big[\frac{V^{X_n}_{\alpha}}{P^{X_n}_{\alpha}} - \frac{\eta^{X_n}_{\alpha,\alpha+1}}{\lambda^{X_n}_{\alpha,\alpha+1}} + \frac{\eta^{X_n}_{\alpha-1,\alpha}}{\lambda^{X_n}_{\alpha-1,\alpha}} \big]}{\sum_{\mathbf{X}_{1:n}}^{\mathbf{w}^{X_n,\alpha}_{1:n}} P^{X_n}_{\alpha} \frac{\lambda^{X_n}_{\alpha-1,\alpha}}{\lambda^{X_n}_{\alpha,\alpha+1}}} \right], \tag{3.55}$$

where Γ is just the product of probability components, namely,

$$\Gamma \triangleq \Big(\prod_{i=1}^{n-1} \prod_{\alpha \in Q_{X_i}} P^{X_i}_{\alpha} \Big) \cdot \Big(\prod_{\alpha \in Q_{X_n}} P^{X_n}_{\alpha} \frac{\lambda^{X_n}_{\alpha-1,\alpha}}{\lambda^{X_n}_{\alpha,\alpha+1}} \Big). \tag{3.56}$$

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

$$P^{X_n}_{\alpha}(\mathbf{X}_{\alpha}) \, \frac{\lambda^{X_n}_{\alpha-1,\alpha}(X_n)}{\lambda^{X_n}_{\alpha,\alpha+1}(X_n)}. \tag{3.57}$$

Similarly, the value component at the α-th mini-bucket of the bucket of X_n is reparameterized by subtracting the outgoing utility cost $\dfrac{\eta^{X_n}_{\alpha,\alpha+1}(X_n)}{\lambda^{X_n}_{\alpha,\alpha+1}(X_n)}$ from $\dfrac{V^{X_n}_{\alpha}(\mathbf{X}_{\alpha})}{P^{X_n}_{\alpha}(\mathbf{X}_{\alpha})}$ and adding to it the incoming utility cost $\dfrac{\eta^{X_n}_{\alpha-1,\alpha}(X_n)}{\lambda^{X_n}_{\alpha-1,\alpha}(X_n)}$. Then, multiplying by the new probability component in Eq. (3.57), we get

$$\Big(P^{X_n}_{\alpha}(\mathbf{X}_{\alpha}) \frac{\lambda^{X_n}_{\alpha-1,\alpha}(X_n)}{\lambda^{X_n}_{\alpha,\alpha+1}(X_n)} \Big) \cdot \Big(\frac{V^{X_n}_{\alpha}(\mathbf{X}_{\alpha})}{P^{X_n}_{\alpha}(\mathbf{X}_{\alpha})} - \frac{\eta^{X_n}_{\alpha,\alpha+1}(X_n)}{\lambda^{X_n}_{\alpha,\alpha+1}(X_n)} + \frac{\eta^{X_n}_{\alpha-1,\alpha}(X_n)}{\lambda^{X_n}_{\alpha-1,\alpha}(X_n)} \Big). \tag{3.58}$$

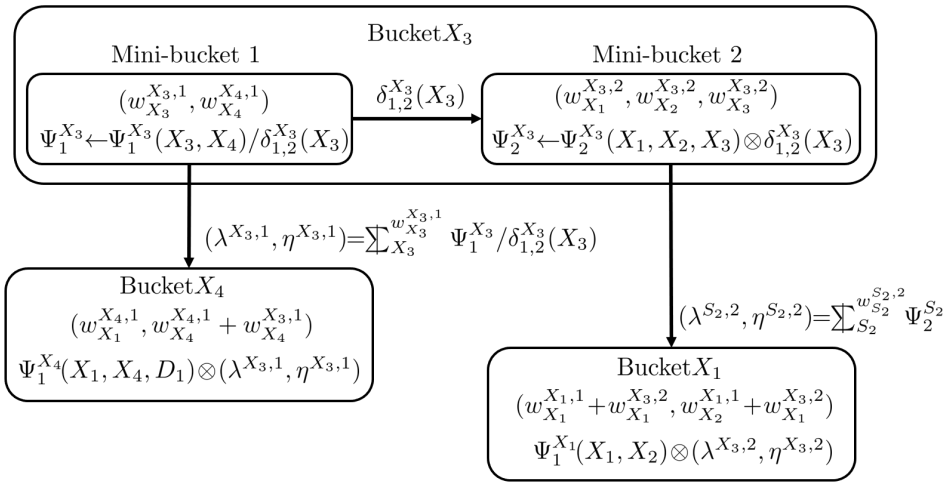


Figure 3.4: Optimization Parameters for WMBE-ID Bounds for IDs.

Example 3.4. *Figure 3.4 illustrates how we associate the cost-shifting parameters between mini-buckets of the bucket X_3 in Figure 3.3. Before entering the optimization loop, the mini-buckets of the bucket X_3 are pre-allocated the potentials and the messages received from the upper layers. Namely, $\Psi_1^{X_3}(X_3, X_4) = \big(\lambda^{D_2}(X_3, X_4), \eta^{D_2}(X_3, X_4)\big)$ is the message sent from bucket D_2, and $\Psi_2^{X_3} = (P(X_3|X_1, X_2), 1)$ is the potential allocated to bucket X_3 at the initialization step. We first show the cost-shifting potential $\delta_{1,2}^{X_3}(X_3)$ between* `Mini-bucket1` *and* `Mini-bucket2`*. We associated with the two mini-buckets the weight parameters denoted $w_{X_3}^{X_3,1}$ and $w_{X_3}^{X_3,2}$. These will be optimized under the constraint $1 = w_{X_3}^{X_3,1} + w_{X_3}^{X_3,2}$. Once we finish updating the cost-shifting valuation and the weight parameters by the optimization algorithms which we will introduce shortly, we compute the out-going messages from each mini-bucket. In* `Mini-bucket1`*, the message $\big(\lambda^{X_3,1}(X_4), \eta^{X_3,1}(X_4)\big)$ is computed by $\sum_{X_3}^{w_{X_3}^{X_3,1}} \Psi^{X_3}(X_3, X_4) \otimes \frac{1}{\delta_{1,2}^{X_3}(X_3)}$, and sent to bucket X_4. Bucket X_4 combines this incoming message with the preassigned potential $\Psi_1^{X_4}(X_1, X_4, D_1)$, and adds the incoming weight $w_{X_4}^{X_3,1}$ to $w_{X_4}^{X_4,1}$. In* `Mini-bucket2`*, the message $\big(\lambda^{X_3,2}(X_4)(X_1, X_2), \eta^{X_3,2}(X_4)(X_1, X_2)\big)$ is computed by $\sum_{X_3}^{w_{X_3}^{X_3,2}} \Psi_1^{X_3}(X_3, X_4) \otimes \delta_{1,2}^{X_3}(X_3)$, and sent to the bucket of X_1. Bucket X_1 combines this incoming message with its preassigned potential and update the weights. Since there are no more mini-buckets in the remaining subproblem, we compute the upper bound on the MEU*

by the variable elimination algorithm [Dechter, 2013].

In summary, Example 3.4 illustrates the procedure WMBE-ID, which interleaves a message passing stage by the weighted mini-bucket elimination algorithm [Dechter, 2013, Liu and Ihler, 2011] and a reparameterization stage that optimizes the parameters defined over a chain of mini-buckets. Next, we will present the message passing algorithm for optimizing the WMBE-ID bound, which is based on the BCD method [Boyd et al., 2004].

3.2.3.2 Optimizing the Paramterized Bounds

Message Passing Algorithm (WMBE-ID) Given an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$, and a mini-bucket tree decomposition $\mathcal{T}_{MBT}$ defining a set of mini-buckets $Q_{X_i} := \{1, \ldots, q_{X_i}\}$ for each of bucket X_i, where $X_i \in \mathbf{X} \cup \mathbf{D}$, Proposition 3.2 introduces three kinds of optimization parameters: (1) probability cost-shifting function $\{\lambda^{X_i}_{\alpha,\alpha+1}(X_i) | \alpha, \alpha+1 \in Q_{X_i}\}$, (2) value cost-shifting function $\{\eta^{X_i}_{\alpha,\alpha+1}(X_i) | \alpha, \alpha+1 \in Q_{X_i}\}$, and (3) weight parameters over the mini-buckets $\{w^{X_i,\alpha}_{X_i} | \alpha, \alpha+1 \in Q_{X_i}\}$. Reviewing the notation for the cost-shifting functions, $\lambda^{X_i}_{\alpha,\alpha+1}(X_i)$ and $\eta^{X_i}_{\alpha,\alpha+1}(X_i)$, the superscript X_i denotes the label of bucket X_i, and subscript $(\alpha, \alpha+1)$ denotes a separator between the α-th mini-bucket and the $(\alpha+1)$-th mini-bucket partitioning bucket X_i. For the weight parameter, $w^{X_i,\alpha}_{X_i}$, the superscript (X_i, α) denotes the α-th mini-bucket of bucket X_i. To define the inner-loop of the BCD method, we partition the optimization parameters into two types of blocks, one containing the weight parameters and the other containing the cost-shifting functions. Algorithm 3.6 presents the WMBE-ID algorithm. The algorithm minimizes the surrogate upper bound U_{value} in Eq. (3.55) by alternating the optimization routines for the costs and weights, respectively.

Given an input ID $\mathcal{M}$, and an i-bound, the initialization step generates a mini-bucket tree decomposition (line 1) (see [Dechter and Rish, 2003]). Then, we assign functions to the mini-buckets and initialize weights (line 2-4) as usual. The main procedure (line 5-16) pro-

Algorithm 3.6 Weighted Mini-Bucket Elimination Bounds for IDs (WMBE-ID)

Require: ID $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$, total constrained elimination order $\mathcal{O}_{\text{elim}} := \{X_N, X_{N-1}, \ldots, X_1\}$, i-bound, iteration limit M,

Ensure: an upper bound of the MEU, U_{value}

Initialization Steps

1: Generate a mini-bucket tree decomposition $\mathcal{T}_{\text{MBT}} = \langle T(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ with i-bound

2: Allocate functions to mini-buckets $\{Q_{X_i} | X_i \in \mathbf{X}\}$, where $Q_{X_i} = \{1, \ldots, q_{X_i}\}$

3: Initialize weights of chance variables $X_i \in \mathbf{X}$ by uniform weights $w_{X_i}^{X_l,\alpha} = \frac{1}{|\{C | X_i \in \chi(C)\}|}$,

4: Initialize weights of decision variables $D_i \in \mathbf{D}$ by a constant close to zero, $w_{D_i}^{X_l,\alpha} \approx 0$

Interleave Variable Elimination and Reparameterization

5: **for** $i \leftarrow N$ to 1 **do**

6: **for** each mini-bucket $\alpha \in Q_{X_i}$ **do**

7: Combine potentials at the α-th mini-bucket, $\Psi_\alpha(\mathbf{X}_\alpha)$

8: **for** each $X_k \in \text{sc}(\Psi_\alpha)$ **do**

9: $w_{X_k}^{X_i,\alpha} \leftarrow w_{X_k}^{X_i,\alpha} + \sum_{l>i} w_{X_k}^{X_l,\alpha_l}$ ▷ sum up the weights of incoming messages $w_{X_k}^{X_l,\alpha_l}$ to the pre-allocated $w_{X_k}^{X_i,\alpha}$

Outer-loop of BCD

10: $iter = 0$, compute U_{value} by Eq. (3.55)

11: **while** $iter < M$ or U_{value} is not converged **do**

Inner-loop of BCD

12: **for** each edge $(\alpha, \alpha+1) \in \{(\alpha, \alpha+1) | \alpha, \alpha+1 \in Q_{X_i})\}$ **do**

13: $U_{\text{value}} \leftarrow min(U_{\text{value}}, \text{UPDATE-COSTS}(\mathcal{T}_{\text{MBT}}, (X_i, \alpha, \alpha+1)))$ ▷ Algorithm 3.7

14: $U_{\text{value}} \leftarrow min(U_{\text{value}}, \text{UPDATE-WEIGHTS}(\mathcal{T}_{\text{MBT}}, X_i))$

15: **for** each mini-bucket $\alpha \in Q_{X_i}$ **do**

16: $\Psi^{X_i} \leftarrow \sum_{X_i}^{w_{X_i}^{X_i,\alpha}} \Psi_\alpha(\mathbf{X}_\alpha)$ ▷ Compute message at the α-th mini-bucket

17: Send message Ψ^{X_i} downward to the destination mini-bucket

return U_{value}

cesses each variable X_i following the elimination order $\mathcal{O}_{\text{elim}} = \{X_N, X_{N-1}, \ldots, X_1\}$. Before executing BCD updates for reparameterizing mini-buckets Q_{X_i} of bucket X_i, we combine all the potentials assigned at the α-th mini-bucket to $\Psi_\alpha(\mathbf{X}_\alpha)$ (line 7). Recall that we initialized the weights $w_{X_k}^{X_l,\alpha}$ associated with variable X_k uniformly over all mini-buckets in the mini-bucket tree $T(\mathcal{C}, \mathcal{S})$, which distributes the weight not only horizontally over a chain of mini-buckets, but also vertically through downward message passing. The weight update procedure in line 8 sums up all the weights $w_{X_k}^{X_l,\alpha_l}$ sent from the α_l-th upstream mini-bucket extracted from bucket X_k by $w_{X_k}^{X_i,\alpha} + \sum_{l>i} w_{X_k}^{X_l,\alpha_l}$. This step ensures that the evaluation

Algorithm 3.7 UPDATE-COST($\mathcal{T}_{\text{MBT}}, (X_i, \alpha, \alpha+1)$) – WMBE-ID

Require: a mini-bucket tree decomposition $\mathcal{T}_{\text{MBT}}$ of an ID $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{\Psi}, \mathcal{O} \rangle$, mini-buckets α and $\alpha+1$ of bucket X_i

Ensure: an upper bound of the MEU, U_{value}

1: $F(\delta_{\alpha,\alpha+1}(X_i)) \leftarrow U_{\text{value}}(\delta_{\alpha,\alpha+1}(X_i))$ (Eq. (3.55)) ▷ define objective function
2: $C_{\text{value}} \leftarrow$ constraints for the value function Eq. (3.59)
3: $C_{\text{prob}} \leftarrow$ constraints for the probability function Eq. (3.60)
Solve constrained optimization problem by defining the objective function $F(\delta_{\alpha,\alpha+1}(X_i))$, optimization parameters $\delta_{\alpha,\alpha+1}(X_i)$, and constraints C_{value} and C_{prob} by using off-the-shelf solver SLSQP [Kraft, 1988]
4: $U_{\text{value}} \leftarrow \text{SLSQP}(F(\delta_{\alpha,\alpha+1}(X_i)), \delta_{\alpha,\alpha+1}(X_i), C_{\text{value}}, C_{\text{prob}})$
5: **return** U_{value}

of the optimization objective U_{value} by Eq. (3.55) on the remaining subproblems would be valid. The outer-loop of BCD begins by initializing the iteration counter and the minimum upper bound U_{value} by Eq. (3.55). The inner-loop of BCD (line 11-13) alternates updating the cost-shifting functions by `UPDATE-COSTS`($\mathcal{T}_{\text{MBT}}, (X_i, \alpha, \alpha+1)$) and the weight parameters by `UPDATE-WEIGHTS`($\mathcal{T}_{\text{MBT}}, X_i$) to be defined later. Since WMBE-ID uses the powered-summation operation following Definition 3.1, which uses the absolute value of the value function, the gradient based optimization methods may encounter difficulty when the value function contains negative values [Boyd et al., 2004]. Therefore, we developed a constrained optimization routine for updating the cost-shifting functions that addresses this difficulty discussed in the following paragraph. We can use Algorithm 3.3 for updating the weight parameters without any modification. After reaching the iteration limit or convergence, we process each mini-bucket by computing the messages and sending them to their destinations in the mini-bucket tree decomposition. Next, we will present a constrained optimization routine for updating cost-shifting functions.

Updating Cost-shifting Functions The WMBE-ID bound uses the powered-summation operation defined by Definition 3.1, which transforms a value function $V(\mathbf{X})$ to $|V(\mathbf{X})|$ as shown in Eq. (3.2). Therefore, we constrain value functions be a non-negative function after the reparameterization by enforcing an additional constraints. Let $\Psi_{\alpha}^{X_n}(\mathbf{X}_{\alpha}) =$

$(\lambda_\alpha^{X_n}(\mathbf{X}_\alpha), \eta_\alpha^{X_n}(\mathbf{X}_\alpha))$ be the potential at the α-th mini-bucket of bucket X_n, and let $\delta_{(\alpha,\alpha+1)}(X_n) = (\lambda_{(\alpha,\alpha+1)}(X_n), \eta_{(\alpha,\alpha+1)}(X_n))$ be the cost-shifting functions between the α-th and $(\alpha+1)$-th mini-buckets. Then, we can ensure the non-negativity of the value function following reparameterization by enforcing the following constraints.

$$\frac{\eta_\alpha^{X_n}(\mathbf{X}_\alpha)}{\lambda_\alpha^{X_n}(\mathbf{X}_\alpha)} - \frac{\eta_{(\alpha,\alpha+1)}(X_n)}{\lambda_{(\alpha,\alpha+1)}(X_n)} + \frac{\eta_{(\alpha-1,\alpha)}(X_n)}{\lambda_{(\alpha-1,\alpha)}(X_n)} \geq 0. \tag{3.59}$$

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

$$\lambda_{(\alpha,\alpha+1)}(X_n) \geq 0, \qquad \forall(\alpha, \alpha+1) \in Q_{X_n}. \tag{3.60}$$

Equipped with the above constraints, any constrained optimization procedure can be applied to reparameterize the mini-buckets when we posed the task as a constrained optimization task. In our implementation, we used an off-the-shelf sequential least square programming solver(SLSQP) solver [Kraft, 1988] to update the cost-shifting functions. Algorithm 3.7 outlines the procedure for updating the cost-shifting functions $\delta_{(\alpha,\alpha+1)}(X_n)$ for the mini-buckets $\alpha, \alpha+1$ of bucket X_i. First, we define the objective function by assigning the values of all the current optimization parameters except $\delta_{(\alpha,\alpha+1)}(X_n)$, thus making U_{value} in Eq. (3.55) a function over $\delta_{(\alpha,\alpha+1)}(X_n)$ (line 1). Next, we define constraints for both the probability and the value functions via Eq. (3.59) and (3.60) (line 2). Then, we call the off-the-shelf solver SLSQP by passing the optimization problem and return the updated upper bound U_{value}.

Complexity of WMBE-ID We will next discuss the complexity of our algorithm.

Theorem 3.3 (**Complexity of Algorithm WBME-ID**). *Given an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$ and a mini-bucket tree decomposition $\mathcal{T}_{MBT}$ with an i-bound, the time complexity of WMBE-ID is $O\big(|\mathbf{X} \cup \mathbf{D}| \cdot |\mathbf{P} \cup \mathbf{U}| \cdot \cdot (k^{i+1} + M_1 \cdot (M_2 \cdot G_w \cdot k^{i+1} + M_3 \cdot G_c \cdot k^{i+1} + M_3 \cdot f(k)))\big)$ and the space complexity is $O\big(2 \cdot |\mathbf{P} \cup \mathbf{U}| \cdot (k^{i+1} \cdot k)\big)$, where M_1 is the iteration limit of the*

outer-loop of the BCD method, M_2 is the iteration limit of the gradient descent algorithm for updating the weight parameters, M_3 is the iteration limit of SLSQP algorithm for updating the cost-shifting functions, $|\mathbf{P} \cup \mathbf{U}|$ is the number of functions, $|\mathbf{X} \cup \mathbf{D}|$ is the number of variables, k is the maximum domain size, and G_w is a constant that bounds the number of factor operations for evaluating gradients for the weight parameters, G_c is a constant that bounds the number of factor operations in SLSQP algorithm.

Proof. The time complexity of WMBE-ID can be captured by the inner-loop of the BCD method for updating the cost-shifting functions using Algorithm 3.7 and for updating the weight parameters using Algorithm 3.3. Algorithm 3.7 solves a constrained optimization problem defined over two mini-buckets and the time complexity of a single call of SLSQP can be bounded by $O((G_c \cdot k^{i+1} + f(k))$ since evaluating the objective function involves a constant number of factor operations over pseudo marginals at each mini-bucket. The polynomial $f(k)$ bounds the rest of the operations in SLSQP [Kraft, 1988] since the dimension of the optimization parameter space is $O(k)$. The overall time complexity for updating the cost shifting functions is $O\big(|\mathbf{X} \cup \mathbf{D}| \cdot |\mathbf{P} \cup \mathbf{U}| \cdot M_1 \cdot M_3 \cdot (G_c \cdot k^{i+1} + f(k))\big)$ since we repeat the SLSQP calls between two mini-buckets M_3 times in Algorithm 3.7, M_1 times over the outer-loop iteration of the BCD methods, $|\mathbf{X} \cup \mathbf{D}|$ times for all the layers of mini-buckets. The number of mini-buckets extracted from a bucket can be bounded by $|\mathbf{P} \cup \mathbf{U}|$, so we repeat the SLSQP calls at most $|\mathbf{P} \cup \mathbf{U}|$ times. The time complexity for updating the weight parameters for a single inner-loop BCD method can be bounded by $O\big(|\mathbf{P} \cup \mathbf{U}| \cdot M_2 \cdot G_w \cdot k^{i+1}\big)$. The overall time complexity is the sum of the time complexity for updating optimization parameters and for performing variable elimination $O\big(|\mathbf{X} \cup \mathbf{D}| \cdot |\mathbf{P} \cup \mathbf{U}| \cdot k^{i+1}\big)$ [Dechter, 2013], which results in $O\big(|\mathbf{X} \cup \mathbf{D}| \cdot |\mathbf{P} \cup \mathbf{U}| \cdot (k^{i+1} + M_1 \cdot (M_2 \cdot G_w \cdot k^{i+1} + M_3 \cdot G_c \cdot k^{i+1} + M_3 \cdot f(k)))\big)$. For the space complexity, we store pseudo marginals for the probability component and the value component at each mini-bucket and cost-shifting potentials between two mini-buckets, which costs $O\big(2 \cdot Q \cdot k^{i+1}\big)$ and $O(2 \cdot Q \cdot k)$ space, respectively. □

Summary We have presented a message-passing algorithm WMBE-ID for computing the upper bound of the MEU in IDs using the weighted mini-bucket elimination algorithm [Dechter, 2013, Liu, 2014]. WMBE-ID tightens the mini-bucket relaxation by optimizing the cost-shifting functions and weight parameters on the fly. To formulate an optimization problem over the mini-buckets extracted from a single bucket, we introduced a surrogate optimization objective function to minimize the upper bounds, which relies on the decomposition bounds for IDs introduced in Section 3.2.2.1. Under this formulation, the number of optimization parameters reduces considerably compared with the algorithm JGD-ID. However, WMBE-ID doesn't have a simple closed-form update equation, so we have defined a constrained numerical optimization routine to find the cost-shifting functions emerging from the powered-summation operation.

3.3 Bounding Scheme using Exponentiated Utility Functions

Now we move to the second approach for bounding the MEU, which relies on the idea of exponentiating the utility functions, which is a well recognized approach in the control theory [Kopp, 1962, Ekeland, 1974] and control as inference [Todorov, 2006, Kappen et al., 2012, Botvinick and Toussaint, 2012]. Most of the earlier works in control as inference focused on continuous domains and on generating lower bounds or on sampling based methods. In this section, we develop a bounding schemes for the MEU using the Jensen's inequality [Jensen et al., 1906], which we apply to the expected utility. Here, we do not use the valuation algebra.

3.3.1 Exponentiated IDs and Decomposition Bounds

We first define a notation for the IDs with exponentaited utility functions as follows.

Definition 3.4 (**Exponentiated ID**). *Given an ID* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$*, we define expo-*

nentiated ID by $\mathcal{M}^e := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}^e, \mathcal{O} \}$*, where* $\mathbf{U}^e = \{e^{U_i} | U_i \in \mathbf{U}\}$*. In* $\mathcal{M}^e$*, the combination operation for the exponentiated utility functions in* $\mathbf{U}^e$ *is the multiplication operation.*

Next, we state the exponentiated utility bounds for IDs, which bounds the MEU of $\mathcal{M}$ by the log-partition function (see Section 2.3.1) of $\mathcal{M}^e$ as follows.

Theorem 3.4 (**Exponentiated Utility Bounds for IDs**). *Given an ID* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$*, where* $\mathbf{X}$ *is a set of chance variables,* $\mathbf{D}$ *is a set of decision variables,* $\mathbf{P}$ *is a set of probability functions,* $\mathbf{U}$ *is a set of additive utility functions, and* $\mathcal{O}$ *is the constrained ordering, we can bound the MEU of* $\mathcal{M}$ *by the log-partition function of the exponentiated ID* $\mathcal{M}^e := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}^e, \mathcal{O} \rangle$*, where* $\mathbf{U}^e$ *is a set of multiplicative functions obtained by exponentiating the utility functions in* $\mathbf{U}$*. Namely,*

$$MEU = \sum_{\mathcal{O}}^{\mathbf{w}} \prod_{P_i \in \mathbf{P}} P_i(\mathbf{X}_{P_i}) \cdot \Big(\sum_{U_i \in \mathbf{U}} U_i(\mathbf{X}_{U_i}) \Big) \leq \log \sum_{\mathcal{O}}^{\mathbf{w}} \Big[\prod_{P_i \in \mathbf{P}} P_i(\mathbf{X}_{P_i}) \cdot \prod_{F_i \in \mathbf{U}^e} F_i(\mathbf{X}_{F_i}) \Big]. \quad (3.61)$$

Proof. The left-hand side of Eq. (3.61) is the MEU definition shown in Eq. (2.64) under the perfect recall condition. We then rewrite the MEU in terms of an expectation over the utility functions and we will subsequently introduce exponentiated utility functions in Eq. (3.63). From Jensen's inequality, we get Eq. (3.64) (see [Chandler, 1987]). We can rewrite the upper bound by switching the order of the max operation and log operation and rewrite the expectation using the functions in $\mathcal{M}^e$ in Eq. (3.65). The right-hand side of Eq. (3.61) is obtained by rewriting the upper bound using the powered-sum operation.

$$\text{MEU} = \sum_{\mathcal{O}}^{\mathbf{w}} \Big(\prod_{P_i \in \mathbf{P}} P_i(\mathbf{X}_{P_i}) \Big) \cdot \Big(\sum_{U_i \in \mathbf{U}} U_i(\mathbf{X}_{U_i}) \Big) \quad (3.62)$$

We next stop using the powered-sum notation and will explicilty use $\max_{\mathbf{\Delta}}$ notation, where

$\boldsymbol{\Delta}$ is the set of policy functions defined in Section 2.4.

$$= \max_{\boldsymbol{\Delta}} \mathbb{E}\Big[\sum_{U_i \in \mathbf{U}} U_i(\mathbf{X}_{U_i}) \Big] = \max_{\boldsymbol{\Delta}} \mathbb{E}\Big[\log e^{\sum_{U_i \in \mathbf{U}} U_i(\mathbf{X}_{U_i})} \Big] \tag{3.63}$$

From Jensen's inequality, we get:

$$\leq \max_{\boldsymbol{\Delta}} \log \mathbb{E}\Big[e^{\sum_{U_i \in \mathbf{U}} U_i(\mathbf{X}_{U_i})} \Big] \tag{3.64}$$

$$= \log \max_{\boldsymbol{\Delta}} \mathbb{E}\Big[e^{\sum_{U_i \in \mathbf{U}} U_i(\mathbf{X}_{U_i})} \Big] = \log \max_{\boldsymbol{\Delta}} \mathbb{E}\Big[\prod_{F_i \in \mathbf{U}^e} F_i(\mathbf{X}_{F_i}) \Big] \tag{3.65}$$

$$\tag{3.66}$$

Note that F_i denotes an element of a set of exponentiated utility functions $\mathbf{U}^e$. Namely, $F_i \in \mathbf{U}^e$ and $F_i(\mathbf{X}_{F_i}) := e^{U_i(\mathbf{X}_{U_i})}$. Therefore,

$$= \log \sum_{\mathcal{O}}^{\mathbf{W}} \Big[\prod_{P_i \in \mathbf{P}} P_i(\mathbf{X}_{P_i}) \cdot \prod_{F_i \in \mathbf{U}^e} F_i(\mathbf{X}_{F_i}) \Big] \tag{3.67}$$

□

Next, we will use two bounding schemes for the mixed inference task to compute the log-partition function of $\mathcal{M}^e$ yielding two bounding algorithms: (1) by extending GDD bounds [Ping et al., 2015] over a join-graph decomposition of $\mathcal{M}^e$, which we call JGD-EXP (Section 3.3.2), and (2) by extending WMBMM bounds [Ihler et al., 2012, Marinescu et al., 2014] over a mini-bucket tree decomposition of $\mathcal{M}^e$, which we call WMBMM-EXP (Section 3.3.3).

3.3.2 Join-graph Decomposition Bounds for Exponentiated IDs

Given the exponentiated ID $\mathcal{M}^e := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}^e, \mathcal{O} \rangle$ of $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$, a join-graph decomposition $\mathcal{G}_{\text{JG}} := \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ for $\mathcal{M}^e$ can be obtained in the usual way using the mini-bucket tree decomposition (see Example 3.1). Figure 3.5 and Example 3.5 show the

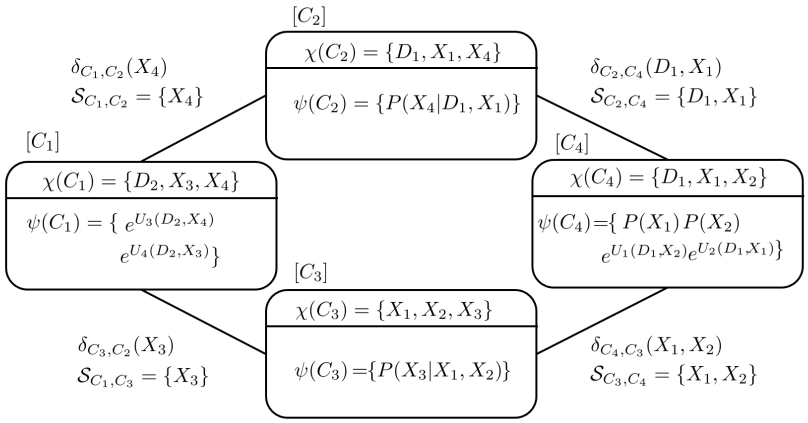


Figure 3.5: Join-graph Decomposition of the Exponentiated ID.

changes to the join-graph decomposition due to the exponentiated ID $\mathcal{M}^e$.

Example 3.5. *Figure 3.5 presents a join-graph decomposition $\mathcal{G}_{JG} := \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ of the ID in Figure 3.1a. Here we do not use the valuation algebra over the Jensen's potentials as was shown in Figure 3.1b. The join-graph in Figure 3.5 treats probability functions and exponentiated utility functions as unnormalized factors in a probabilistic graphical models.*

Theorem 3.4, shows that the MEU of an ID $\mathcal{M}$ can be bounded by the log-partition function of the exponentiated ID $\mathcal{M}^e$. Therefore, we can now apply any variational decomposition bounds in the mixed inference task for bounding the log-partition function to yield bounds to the MEU of $\mathcal{M}$. Namely, we bound the MEU by first applying Theorem 3.4 to obtain $\mathcal{M}^e$ from $\mathcal{M}$, and then by using variational decomposition bounds in the probabilistic graphical models to $\mathcal{M}^e$.

3.3.2.1 Derivation of Upper Bounds

Given a join-graph decomposition $\mathcal{G}_{\text{JG}} := \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ of $\mathcal{M}^e$, we denote a cost-shifting function over the separator $(C_i, C_j) \in \mathcal{S}$ by $\delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$. The upper bounds on the MEU can be parameterized over $\mathcal{G}_{\text{JG}}$ as described in the following proposition.

Proposition 3.3. *Given an ID* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$*, where* $\mathbf{U} = \{U_1, \ldots, U_l\}$ *and its exponentiated ID* $\mathcal{M}^e := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}^e, \mathcal{O} \rangle$*,* $\mathbf{U}^e = \{e^{U_1}, \ldots, e^{U_l}\}$*, a total elimination order* $\mathcal{O}_{elim} := \{X_1, X_2, \ldots, X_N\}$ *over the variables* $\mathbf{X} \cup \mathbf{D}$*, and given a join-graph decomposition* $\mathcal{G}_{JG} := \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ *of* $\mathcal{M}^e$*, the upper bound on the MEU can be parameterized over* $\mathcal{G}_{JG}$ *as follows.* U_{MEU} *denotes as usual an upper bound over the MEU. We claim that for a given set of weights.*

$$MEU \leq \sum_{C_i \in \mathcal{C}} \log \sum_{X_N^C}^{w_{X_N}^C} \cdots \sum_{X_1^C}^{w_{X_1}^C} \exp\Big(\log P_{C_i}(\mathbf{X}_{C_i}) + U_{C_i}(\mathbf{X}_{C_i}) + \sum_{(C_i, C_j) \in \mathcal{S}} \delta_{C_i, C_j}(\mathbf{X}_{C_i, C_j}) \Big), \quad (3.68)$$

and we denote the expression on the right hand side as U_{MEU} *where,*

$$P_{C_i}(\mathbf{X}_{C_i}) = \prod_{P_i \in \left(\psi(C_i) \cap \mathbf{P}\right)} P_i(\mathbf{X}_{P_i}), \quad (3.69)$$

$$U_{C_i}(\mathbf{X}_{C_i}) = \sum_{U_i \in \left(\psi(C_i) \cap \mathbf{U}\right)} U_i(\mathbf{X}_{U_i}), \quad (3.70)$$

$$\delta_{C_i, C_j}(\mathbf{X}_{C_i, C_j}) + \delta_{C_j, C_i}(\mathbf{X}_{C_i, C_j}) = \mathbf{0} \quad \forall (C_i, C_j) \in \mathcal{S}, \quad (3.71)$$

$$\sum_{C \in \mathcal{C}} w_{X_i}^C = w_{X_i}, \quad \forall X_i \in (\mathbf{X} \cup \mathbf{D}). \quad (3.72)$$

$P_{C_i}(\mathbf{X}_{C_i}) \in \mathbf{P}$ *and* $U_{C_i}(\mathbf{X}_{C_i}) \in \mathbf{U}$ *are the probability and utility functions assigned at the cluster node* C_i*,* $\delta_{C_i, C_j}(\mathbf{X}_{C_i, C_j})$ *and* $\delta_{C_j, C_i}(\mathbf{X}_{C_j, C_i})$ *are the cost-shifting function over the separator* (C_i, C_j) *that cancels each other, and* $w_{X_k}^C$ *is the weight for the variable* X_k *assigned at cluster* C *that satisfies* $w_{X_k} = \sum_{C \in \mathcal{C}} w_{X_k}^C$*.*

Proof. We start our derivation from Eq. (3.61) in Theorem 3.4 bounding the MEU of $\mathcal{M}$ by the log-partition function of $\mathcal{M}^e$ repeated in Eq. (3.73). To get Eq. (3.74), we rearrange the probability functions and exponentiated utility functions at each cluster C_i according to $\mathcal{G}_{\text{JG}}$, and introduce cost-shifting functions over the separators, which cancel each other. Namely,

$\delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})\delta_{C_j,C_i}(\mathbf{X}_{C_i,C_j}) = \mathbf{0}(\mathbf{X}_{C_i,C_j})$ Next, to derive Eq. (3.75), we rewrite functions in the log-space and replaces the exponentiated utility functions $F_i(\mathbf{X}_{F_i})$ with the original utility functions $U_i(\mathbf{X}_{U_i})$ Subsequently, we switch the order of the multiplication operation and the elimination operation using decomposition bounds on mini-buckets [Ping et al., 2015] yielding Eq. (3.76), and obtain the desired expression by changing log product to sum log and rewriting the powered-sum operation explicitly, yielding Eq. (3.77).

$$\text{MEU} = \sum_{\mathcal{O}}^{\mathbf{w}} \prod_{P_i \in \mathbf{P}} P_i(\mathbf{X}_{P_i}) \cdot \Big(\sum_{U_i \in \mathbf{U}} U_i(\mathbf{X}_{U_i}) \Big) \leq \log \sum_{\mathcal{O}}^{\mathbf{w}} \Big[\prod_{P_i \in \mathbf{P}} P_i(\mathbf{X}_{P_i}) \cdot \prod_{F_i \in \mathbf{U}^e} F_i(\mathbf{X}_{F_i}) \Big] \quad (3.73)$$

$$= \log \sum_{\mathcal{O}}^{\mathbf{w}} \prod_{C_i \in \mathcal{C}} \big(\prod_{P_i \in \left(\psi(C_i) \cap \mathbf{P}\right)} P_i(\mathbf{X}_{P_i}) \times \prod_{F_i \in \left(\psi(C_i) \cap \mathbf{U}^e\right)} F_i(\mathbf{X}_{F_i}) \big) \cdot \prod_{C_i, C_j \in \mathcal{S}} \exp(\delta_{C_i,C_j}) \quad (3.74)$$

$$= \log \sum_{\mathcal{O}}^{\mathbf{w}} \prod_{C_i \in \mathcal{C}} \exp \Big(\log P_{C_i}(\mathbf{X}_{C_i}) + U_{C_i}(\mathbf{X}_{C_i}) + \sum_{C_i, C_j \in \mathcal{S}} \delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j}) \Big) \quad (3.75)$$

$$\leq \log \prod_{C_i \in \mathcal{C}} \sum_{\mathcal{O}}^{\mathbf{w}^{C_i}} \exp \Big(\log P_{C_i}(\mathbf{X}_{C_i}) + U_{C_i}(\mathbf{X}_{C_i}) + \sum_{C_i, C_j \in \mathcal{S}} \delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j}) \Big) \quad (3.76)$$

$$= \sum_{C_i \in \mathcal{C}} \log \sum_{X_N^C}^{w_{X_N}^C} \cdots \sum_{X_1^C}^{w_{X_1}^C} \exp \Big(\log P_{C_i}(\mathbf{X}_{C_i}) + U_{C_i}(\mathbf{X}_{C_i}) + \sum_{(C_i, C_j) \in \mathcal{S}} \delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j}) \Big). \quad (3.77)$$

□

Proposition 3.3 introduces two kinds of optimization parameters: (1) cost-shifting functions $\delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$, and (2) weight parameters $\mathbf{w}^{C_i}$ that satisfies the following condition, $w_X = \sum_{C_i \in \mathcal{C}} w_X^{C_i}$ for all $X \in \chi(C_i)$. Compared with algorithm JGD-ID presented in Section 3.2.2, algorithm JGD-EXP has only two types of optimization parameters because JGD-EXP does not use the valuation algebra for IDs.

Algorithm 3.8 Join-graph GDD of Exponentiated IDs (JGD-EXP)

Require: Exponentiated ID $\mathcal{M}^e := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}^e, \mathcal{O} \rangle$ of $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$, weights w_{X_i} associated with each variable $X_i \in \mathbf{X}$, i-bound, iteration limit M for the outer-loop of BCD.

Ensure: an upper bound of the MEU, U_{MEU}

Initialization Steps

1: Generate a join-graph decomposition $\mathcal{G}_{\text{JG}} = \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ of $\mathcal{M}^e$ with given i-bound

2: Initialize weights for each chance variable $X_i \in \mathbf{X}$ by uniform weights, $w^C_{X_i} = \frac{1}{|\{C | X_i \in \chi(C)\}|}$, and for each decision variable $D_i \in \mathbf{D}$ by a constant close to zero, $w^C_{D_i} \approx 0$.

Outer-loop of BCD

3: $iter$=0, $U_{\text{MEU}} = \inf$

4: **while** $iter < M$ or U_{value} is not converged **do**

Inner-loop of BCD

5: **for** each variable $X_i \in \mathbf{X}$ **do**

6: $U_{\text{MEU}} \leftarrow \min(U_{\text{MEU}}, \text{UPDATE-WEIGHTS}(\mathcal{G}_{\text{JG}}, X_i))$ ▷ use Algorithm 3.3 with the gradient expression shown in Eq. (3.78)

7: **for** each edge $(C_i, C_j) \in \mathcal{S}$ **do**

8: $U_{\text{MEU}} \leftarrow \min(U_{\text{MEU}}, \text{UPDATE-COSTS}(\mathcal{G}_{\text{JG}}, (C_i, C_j)))$ ▷ see Algorithm 3.9

9: $iter = iter + 1$

return U_{MEU}

3.3.2.2 Optimizing the Paramterized Bounds

Next, we present algorithm JGD-EXP that solves a convex optimization problem for tightening the upper bound U_{MEU} defined in Eq. (3.68) by optimizing over the cost-shifting functions $\delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$ and the weight parameters $w^C_{X_k}$ parameterized over the join-graph $G(\mathcal{C}, \mathcal{S})$. As we will see, we will need to modify algorithm JGD-ID shown in Algorithm 3.2 in order to present JGD-EXP.

Message Passing Algorithm (JGD-EXP) Algorithm 3.8 outlines the overall procedure of BCD updates in JGD-EXP. In the initialization step of JGD-EXP, we generate a join-graph of $\mathcal{M}^e$ and initialize its weights as usual (line 1-2), The inner-loop of JGD-EXP alternates updating weights by `UPDATE-WEIGHTS`$(\mathcal{G}_{\text{JG},X_i})$ and updating cost-shifting functions by `UPDATE-COSTS`$(\mathcal{G}_{\text{JG},(C_i,C_j)})$ (line 5-8). During the updating if the weight parameters, we optimize a subset of weight parameters $\mathbf{w}_X = \{w^C_X | X \in \chi(C)\}$ for each variable X using

Algorithm 3.3, which was also used for updating the weight parameters in JGD-ID. Yet, we modify the gradient expression for the exponentiated utility bounds as will be shown in Eq. (3.78). For updating the cost-shifting functions, we use Algorithm 3.9 to be presented shortly which performs gradient descent updates for $\delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$ over the separators $\mathcal{S}$. Since both JGD-ID and JGD-EXP are based on GDD bounds [Ping et al., 2015], they are quite similar. However, JGD-EXP does not involve the complicated non-convex optimization problems for updating the parameters, and the derived gradient expressions are much simpler than JGD-ID, as we will show in the following Section. The main difference between these two bounding schemes is in how each handles the additive utility functions. In JGD-ID, we generalize the powered-sum elimination for IDs and derived decomposition bounds by Minkowski's inequality and Hölder's inequality. In JGD-EXP, we bound the MEU of $\mathcal{M}$ by first using the exponentiated utility bound, which gives $\mathcal{M}^e$, and then by applying the decomposition bounds to the log-partition function of $\mathcal{M}^e$.

Next, we will present the optimization routines for optimzing the weight parametrs and the cost-shifting functions.

Updating Weights The `UPDATE-WEIGHTS` routine in Algorithm 3.8 updates the weights $w_{X_i}^{C_j}$ for each variable X_i over clusters $\{C|X_i \in \chi(C)\}$ like in algorithm JGD-ID. The only modification required in Algorithm 3.3 to accommodate that is to provide the gradient expression due to the new objective function U_{MEU} defined in Eq. (3.68). Namely,

$$\frac{\partial U_{\text{MEU}}}{\partial w_{X_k}^{C}} = H_{\Lambda_C(\mathbf{X}_C)}(X_k|\mathbf{X}_{k+1:N}) = -\sum_{\mathbf{X}_C} \Lambda_C(\mathbf{X}_C) \log \Lambda_C(X_k|\mathbf{X}_{k+1:N}), \tag{3.78}$$

$$\text{where } \begin{cases} \Lambda_C(\mathbf{X}_C) = \left(\frac{Z_{N-1}(X_N)}{Z_N}\right)^{1/w_N} \cdots \left(\frac{Z_0(\mathbf{X}_{1:N})}{Z_1(\mathbf{X}_{2:N})}\right)^{1/w_1}, \\ Z_0(\mathbf{X}_{1:N}) = \prod_{P_i \in (\psi(C) \cap \mathbf{P})} P_i(\mathbf{X}_{P_i}) \exp\big(\sum_{U_i \in (\psi(C) \cap \mathbf{U})} U_i\big). \end{cases} \tag{3.79}$$

Algorithm 3.9 UPDATE-COSTS($\mathcal{G}_{\text{JG}}, (C_i, C_j)$)

Require: a join-graph decomposition $\mathcal{G}_{\text{JG}}, X_i)$ of an exponentiated ID $\mathcal{M}^e = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}^e, \mathcal{O} \rangle$ of $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$, separator (C_i, C_j) iteration limit M for the gradient descent

Ensure: an upper bound of the MEU, U_{MEU},

1: $\lambda_{C_i} \leftarrow \sum_{P_i \in (\mathbf{P} \cap \psi(C_i))} logP_i + \sum_{U_i \in (\mathbf{U} \cap \psi(C_i))} U_i$
2: $\lambda_{C_j} \leftarrow \sum_{P_i \in (\mathbf{P} \cap \psi(C_j))} logP_i + \sum_{U_i \in (\mathbf{U} \cap \psi(C_j))} U_i$
3: t=0
4: **while** $t < M$ or U_{MEU} is not converged **do**
5: $\quad \delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j}) \leftarrow \mathbf{0}(\mathbf{X}_{C_i,C_j})$
6: $\quad$ Compute gradient of U_{MEU} with respect to the cost-shifting function by Eq. (3.80)
7: $\quad$ Find cost-shifting functions $\delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$ by Eq. (3.32).
8: $\quad \lambda_{C_i}(\mathbf{X}_{C_i}) \leftarrow F_{C_i}(\mathbf{X}_{C_i}) - \delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$
9: $\quad \lambda_{C_j}(\mathbf{X}_{C_j}) \leftarrow F_{C_j}(\mathbf{X}_{C_j}) + \delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$
10: $\quad$ Compute new U_{MEU} using Eq. (3.68) with the updated $\lambda_{C_i}(\mathbf{X}_{C_i})$ and $\lambda_{C_j}(\mathbf{X}_{C_j})$
11: $\quad t = t + 1$

Eq. (3.78) is a conditional entropy of the pseudo marginal at cluster C defined by Eq. (3.79). For more details for deriving Eq. (3.78), see [Ping et al., 2015].

Updating Cost-shifting Functions We now derive the `UPDATE-COSTS` routine that is used in Algorithm 3.8 to update the cost-shifting functions $\delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$ over separators $(C_i, C_j) \in \mathcal{S}$. As usual, this is done via a gradient descent algorithm. We obtain a closed-form gradient expression for the cost-shifting paramters by evaluating the partial derivatives of U_{MEU} in Eq. (3.68) with respect to $\delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$ as follows.

$$\frac{\partial U_{\text{MEU}}}{\partial \delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})} = - \sum_{\mathbf{X}_{C_i} \setminus \mathbf{X}_{C_j}} \Lambda_{C_i}(\mathbf{X}_{C_i}) + \sum_{\mathbf{X}_{C_j} \setminus \mathbf{X}_{C_i}} \Lambda_{C_j}(\mathbf{X}_{C_j}). \tag{3.80}$$

For the derivation of the gradients, see [Ping et al., 2015].

Algorithm 3.9 presents the gradient descent algorithm for optimizing the cost-shifting functions. Given a separator (C_i, C_j), we combine the probability functions and the utility functions assigned at cluster C_i and C_j to $\lambda_{C_i}(\mathbf{X}_{C_i})$ and $\lambda_{C_j}(\mathbf{X}_{C_j})$, respectively (line 1-2). Note that this algorithm performs factor operations in the log-scale. In each iteration, we

set the cost-shifting function $\delta_{C_i,C_j}(\mathbf{X}_{C_i,C_j})$ to a neutral factor $\mathbf{0}(\mathbf{X}_{C_i,C_j})$. Then we compute gradient using Eq. (3.80) and find the cost-shifting function by Eq. (3.32) (line 6-7), and then we reparameterize $\lambda_{C_i}(\mathbf{X}_{C_i})$ and $\lambda_{C_j}(\mathbf{X}_{C_j})$ (line 8-9). We repeat this until we reach the iteration limit M or until convergence.

Complexity of JGD-EXP We can analyze the time and space complexity of Algorithm 3.8 by following the same reasoning for analyzing the complexity of algorithm JGD-ID (Theorem 3.2).

Theorem 3.5. *Given an ID* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$*, and its exponentiated ID* $\mathcal{M}^e := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}^e, \mathcal{O} \rangle$*, and given a join-graph decomposition* $\mathcal{G}_{JG} := \langle G(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ *of* $\mathcal{M}^e$ *with an* i*-bound, the time complexity of JGD-EXP is* $O\big(M_1 \cdot M_2 \cdot (|\mathbf{X} \cup \mathbf{D}| \cdot |\mathcal{C}| \cdot G_w \cdot k^{i+1} + \cdot |\mathcal{S}| \cdot G_c \cdot k^{i+1} +)\big)$ *and the space complexity is* $O\big((|\mathcal{C}| \cdot k^{i+1} + |\mathcal{S}| \cdot k^{|sep|})\big)$*, where* M_1 *is the iteration limit of the outer-loop of the BCD method,* M_2 *is the iteration limit of the gradient descent algorithms in the inner-loop of the BCD method,* $|\mathbf{X} \cup \mathbf{D}|$ *is the number of variables,* k *is the maximum domain size,* $|\mathcal{C}|$ *and* $|\mathcal{S}|$ *are the number of clusters and separators in the join-graph decomposition* $\mathcal{G}_{JG}$*, and* $|sep|$ *is the separator width.* G_w*, and* G_c *are constants that bounds the number of factor operations for evaluating gradients.*

Summary Algorithm JGD-EXP generates upper bounds by adapting the GDD algorithm over the join-graph decomposition after exponentiating the utility functions. This formulation avoids the non-convexity issue in the previous algorithms JGD-ID and WMBE-ID. However, the bounding scheme cannot recover the exact solution, even if the i bound exceeds the constrained induced width.

3.3.3 Weighted Mini-bucket Moment Matching Bounds for Exponentiated IDs

The WMBMM bound for the marginal MAP task [Marinescu et al., 2014] is a variant of the weighted mini-bucket bounds shown in Proposition 2.3, which performs a simple fixed-point moment matching message passing update only once between mini-buckets during the weighted mini-bucket elimination (See Section 2.3.2). This simple method was shown to often outperforms other more complex bounding schemes when using higher i-bounds since the iterative update on large number of parameters used by the more complex method is computationally intensive . In Section 3.2.3, we present algorithm WMBE-ID that interleaves the variable elimination and the reparameterization in a layer by layer manner over the mini-bucket tree similar to the WMBMM bound. However, it solves non-convex constrained optimization problems at every layer over a mini-bucket tree using surrogate objective functions derived from decomposition bounds before processing messages. We now investigate applying the WMBMM algorithm over the exponentiated IDs. Given $\mathcal{M}^e$, we can generate a weighted mini-bucket tree $\mathcal{T}_{\text{MBT}}$ in the usual manner (see Example 3.3) as we show next.

Example 3.6. *Figure 3.6 shows a weighted mini-bucket tree for $\mathcal{M}^e$, obtained by exponentiating the utility functions of the ID in Figure 2.6a. We see that the structure of the mini-bucket tree is the same as the tree shown in Figure 3.3. The only difference is that the $\mathcal{T}_{MBT}$ in Figure 3.6 uses probability and exponentiated utility functions directly rather than converting the functions into Jensen's potentials.*

Message Passing Algorithm (WMBMM-EXP) Given $\mathcal{M}^e$ and a mini-bucket tree decomposition $\mathcal{T}_{\text{MBT}}$, we can bound the MEU using the WMBMM bounds for the mixed inference task as implied by Theorem 3.4. Algorithm 3.10 present the WMBMM algorithm, applied to the exponentiated IDs. It is identical to the weighted mini-bucket elimination algorithm shown in Algorithm 2.1 except that the moment matching steps with the uniformly dis-

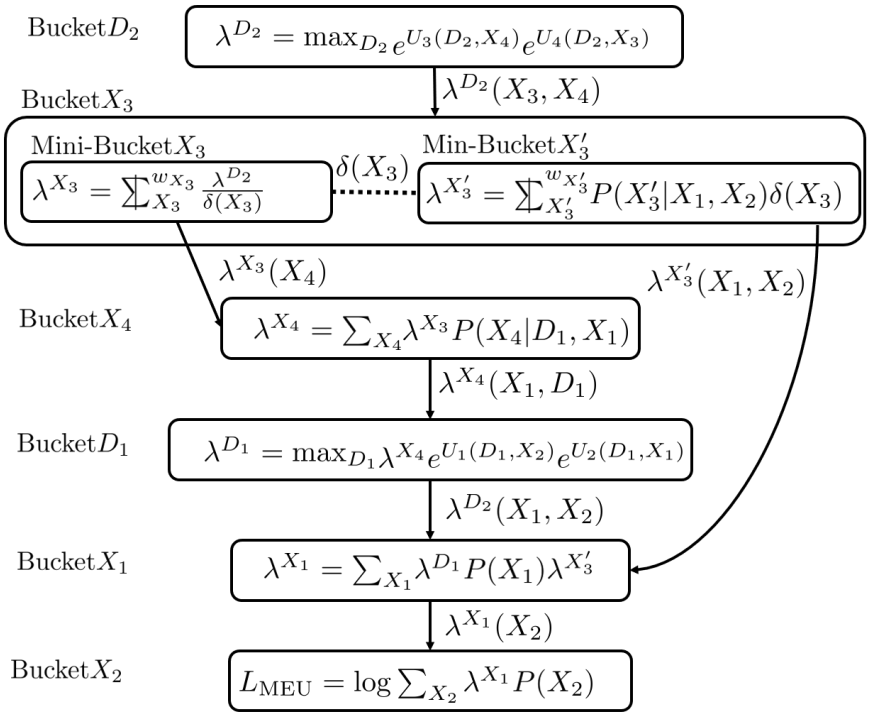


Figure 3.6: Weighted Mini-bucket Tree Decomposition of the Exponentiated ID.

tributed weights (line 8-16) is different. Specifically, when WMBMM-EXP partitions bucket $\mathbf{B}_{X_i}$ to P mini-buckets $\{\mathbf{B}_{X_i}^1, \ldots, \mathbf{B}_{X_i}^P\}$ (line 5), it combines functions at each mini-bucket $\mathbf{B}_{X_i}^p$ (line 7) and perform moment-matching by setting the weights uniformly $w_{X_i}^p = \frac{1}{P}$ if X_i is a chance variable (line 9). The cost-shifting function is computed by taking the pseudo marginals $\mu_p(X_i)$ from each mini-bucket (line 10, 14) and updating them by the geometrical mean (line 12, 16). After this moment-matching phase, we generate and send messages from each mini-bucket downstream as usual (line 17-21).

Complexity of WMBMM-EXP The complexity of algorithm WMBMM-EXP is the same as the complexity of the mini-bucket elimination algorithm [Dechter and Rish, 2003] since the only additional steps added to the naive mini-bucket elimination algorithm are moment-matching procedures (line 7-15) in Algorithm 2.1. We can see that the moment-matching steps only contain a finite number of operations that are bounded within each mini-bucket.

Algorithm 3.10 WMBMM for Exponentiated IDs (WMBMM-EXP)

Require: $\mathcal{M}^e = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathcal{O} \rangle$, total variable elimination ordering $\mathcal{O}$, i-bound
Ensure: Upper bound of the MEU, U_{MEU}

for each variable $X_i \in \mathcal{O}$ **do**
  Collect and Combine Functions
  $\mathbf{B}_{X_i} \leftarrow \{F_i(\mathbf{X}_{F_i}) | X_i \in \text{sc}(F_i), F_i \in \mathbf{F}\}$ ▷ Collect functions to a bucket $\mathbf{B}_{X_i}$
  $\mathbf{F} \leftarrow \mathbf{F} \setminus \mathbf{B}_{X_i}$ ▷ Remove the functions assigned to bucket
  Partition $\mathbf{B}_{X_i}$ to mini-buckets $\{\mathbf{B}_{X_i^1}, \dots \mathbf{B}_{X_i^P}\}$ such that $\max_p |\text{sc}(\mathbf{B}_{X_i^p})| \leq i + 1$
  for each mini-bucket $\mathbf{B}_{X_i^p} \in \{\mathbf{B}_{X_i^1}, \dots \mathbf{B}_{X_i^P}\}$ **do**
    $F_p^{X_i} \leftarrow \prod_{F_i \in \mathbf{B}_{X_i^p}} (F_i)$
    Moment Matching
    if $X_i \in \mathbf{X}$ **then** ▷ X_i is a chance variable
      $w_{X_i}^p = \frac{1}{P}$ ▷ assign uniform weights
      $\mu_p = \sum_{\text{sc}(\mathbf{B}_{X_i^p}) \setminus X_i} \left(F^{X_i^p}\right)^{\frac{1}{w_{X_i}^p}}$ ▷ compute pseudo marginal over X_i from $\mathbf{B}_{X_i^p}$
      $\mu = \prod_{p=1}^{P} \mu_p^{w_{X_i}^p}$ ▷ compute overall pseudo marginal
      $F_p^{X_i} \leftarrow F_p^{X_i} \cdot \left(\frac{\mu}{\mu_p}\right)^{w_{X_i}^p}$ ▷ moment matching update
    else
      $\mu_p = \max_{\text{sc}(\mathbf{B}_{X_i^p}) \setminus X_i} \left(F_p^{X_i}\right)$ ▷ compute pseudo marginal over decision X_i
      $\mu = \prod_{p=1}^{P} \mu_p^{\frac{1}{P}}$ ▷ compute overall pseudo marginal
      $F_p^{X_i} \leftarrow F_p^{X_i} \cdot \left(\frac{\mu}{\mu_p}\right)$ ▷ moment matching update
  Send Message Downwards
  for each mini-bucket $\mathbf{B}_{X_i^p} \in \{\mathbf{B}_{X_i^1}, \dots \mathbf{B}_{X_i^P}\}$ **do**
    if $X_i \in \mathbf{X}$ **then**
      $\lambda^{X_i^p} \leftarrow \sum_{X_i}^{w_{X_i}^p} F_p^{X_i}$ ▷ compute wmb message after moment matching
    else
      $\lambda^{X_i^p} \leftarrow \max_{X_i} F_p^{X_i}$ ▷ compute wmb message after moment matching
    $\mathbf{F} \leftarrow \mathbf{F} \cup \{\lambda^{X_i^p}\}$
$U_{\text{MEU}} \leftarrow \prod_{F_i \in \mathbf{F}} F_i$
return U_{MEU}

Theorem 3.6. *Given an ID* $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$*, and its exponentiated ID* $\mathcal{M}^e := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}^e, \mathcal{O} \rangle$*, and given a mini-bucket tree decomposition* $\mathcal{T}_{MBT} := \langle T(\mathcal{C}, \mathcal{S}), \chi, \psi \rangle$ *of* $\mathcal{M}^e$ *with an i-bound, the time and space complexity of WMBMM-EXP is* $O\big(|\mathbf{X} \cup \mathbf{D}| \cdot k^i\big)$*, where k is the maximum domain size of the variables.*

Summary of WMBMM-EXP We presented WMBMM-EXP algorithm for bounding the MEU extending WMBMM algorithm for the marginal MAP task into the MEU task. In contrast to other bounding schemes presented in this chapter, WMBMM-EXP is the only non-iterative single-pass algorithm. Note that WMBE-ID presented in Section 3.2.3 also parameterizes the weighted mini-bucket tree. However WMBE-ID iteratively optimizes the cost-shifting functions using constrained optimization, and the weight parameters by exponentiated gradient descent.

3.4 Conclusion

	Join-graph Decomposition	Mini-bucket Tree Decomposition
Decomposition Bounds for IDs	JGD-ID Section 3.2.2	WMBE-ID Section 3.2.3
Exponentiated Utility Bounds	JGD-EX Section 3.3.2	WMBMM-EXP Section 3.3.3

Table 3.1: Summary of Direct Decomposition Bounds for IDs

In this chapter, we developed new variational bounding schemes for the MEU task in IDs. Table 3.1 summarize the overall algorithms. The first approach that extending decomposition bounds for IDs generalizes the variational decomposition bounds to the valuation algebra over the Jensen's potential. Then, we extended generalized dual decomposition bounds and the weighted mini-bucket bounds to join-graph GDD bounds for IDs (JGD-ID), and weighted mini-bucket elimination bounds for IDs (WMBE-ID), respectively. This approach provides the bounds over the potentials, which is a pair of probability and the expected utility functions. Hence, both bounding schemes subsumes the upper bound for the MMAP task.

The second approach develops the exponentiated utility bounds, and we also extended the bounding schemes GDD and WMBMM to join-graph GDD bounds for exponentiated IDs

(JGD-EXP), and WMBMM bounds for exponentiated IDs (WMBMM-EXP). This approach bounds the MEU by first exponentiating the utility, and subsequently by bounding the log partition function of the exponentiated IDs with the variational bounding schemes in the mixed inference tasks.

Chapter 4

Empirical Evaluation of Decomposition Bounds for Influence Diagrams

This chapter presents empirical evaluation of our proposed algorithms JGD-ID, WMBE-ID, JGD-EXP, and WMBMM-EXP presented in Chapter 3 and compares the performance of algorithms for bounding the MEU task against earlier approaches. Section 4.1 presents the empirical evaluation methodology, which specifies the evaluated algorithms, the measure of performance, and benchmark domains. Section 4.2 presents the impact of reductions of the MEU task to the MMAP or the mixed inference task [Mauá, 2016, Liu, 2014]. Then, we present the empirical evaluation results from individual problem instances. Section 4.3 reports the performance of evaluated algorithms at varying time and memory bounds, and Section 4.4 shows the convergence behaviors of iterative algorithms from representative problem instances. In Section 4.5, we present the average quality of upper bounds in each domain and rank the algorithms in a competition setting. In Section 4.6, we present the result of a case study that evaluates the upper bounds from algorithm WMBMM-EXP in realistic and larger scale problem instances from probabilistic planning. We also compare the upper bounds from algorithm WMBMM-EXP and the lower bounds from the state-of-the-art online probabilistic planners. We conclude this chapter in Section 4.7

4.1 Methodology

In our experiments, we compared the upper bounds generated by the algorithms for the MEU task. All algorithms were implemented in Python3 language, and the experiment was conducted in `Openlab cluster` at University of California, Irvine. We next present the algorithms evaluated, the performance measure for comparing the algorithms performance in generating upper bounds and introduce the benchmark sets on which we evaluate the algorithms.

4.1.1 Evaluated Algorithms

Table 4.1 shows the list of algorithms to be evaluated. Our direct bounding schemes JGD-ID, WMBE-ID, JGD-EXP, and WMBMM-EXP were introduced in Chapter 3 as an extension to existing decomposition bounds, namely, WMB [Liu and Ihler, 2011, 2012], GDD [Ping et al., 2015], and WMBMM [Liu, 2014, Marinescu et al., 2014]. Our algorithms will be compared against translation based approaches that generate BNs from input IDs by introducing auxiliary variables and relations that allows bounding the MEU using mixed inference [Liu and Ihler, 2012] or MMAP tasks [Mauá, 2016]. We evaluated two translation based algorithms. GDD-MI translates the MEU task to the mixed inference task and applies a GDD bounding algorithm [Ping et al., 2015], and WMBMM-MMAP translates the MEU task to the MMAP task and applies the WMBMM bounding algorithm [Marinescu et al., 2014]. We also compare against MBE-ID, which is a non-variational bounding method that applies MBE bound [Dechter and Rish, 2003] to the potentials for IDs [Jensen et al., 1994]. As we explained in Chapter 3, direct decomposition bounds that are based on the GDD bounding method decompose input IDs to a join-graph decomposition using a bounding parameter i-bound, and they are all iterative algorithms which run until convergence or until iteration limits. Other algorithms such as WMBMM and MBE are non-iterative in nature that use

Algorithm	Decomposition Methods	Bounding Method	Iteration	Translation
JGD-ID	Join-graph	GDD over Potential	Yes	No
WMBE-ID	Mini-bucket Tree	WMB,GDD over Potential	Yes	No
JGD-EXP	Join-graph	GDD over $\mathcal{M}^e$	Yes	No
WMBMM-EXP	Mini-bucket Tree	WMBMM over $\mathcal{M}^e$	No	No
GDD-MI	Join-graph	GDD	Yes	Yes
WMBMM-MMAP	Mini-bucket Tree	WMBMM	No	Yes
MBE-ID	Mini-bucket Tree	MBE	No	No

Table 4.1: JGD-ID and WMBE-ID are the bounding schemes that are introduced in Section 3.2. JGD-EXP and WMBMM-EXP are the bounding scheme that are introduced in Section 3.3. GDD-MI and WMBM-MMAP are competing translation based approaches.

the mini-bucket tree decomposition with a bounding parameter i-bound.

4.1.2 Measure of Performance

Our target is to compare the quality of the upper bounds generated by algorithms listed in Table 4.1. To compare the quality of upper bounds from both iterative and non-iterative algorithms, we control the running time and report the upper bounds in an anytime manner. Namely, we compare the best upper bounds from the evaluated algorithms at sampled time points within the time interval between 10 seconds to 100,000 seconds under the varying i-bounds ranging from 1 to 20 in Section 4.3. In addition to reporting the individual upper bounds at sampled time points, we show the overall convergence behavior over the time interval between 10 seconds to 100,000 seconds. in Section 4.4. We next summarize the quality of upper bounds from each benchmark domain by averaging the ratio between the best known upper bound and the upper bound from each algorithm, $\frac{UB_{\text{best}}}{UB_{\text{alg}}}$. The closer the ratio 1.0, the better the quality. In the extreme, the ratio is 1.0 if the algorithm generated the best upper bound, and is 0.0 when an algorithm returned the infinity as an upper bound. We use the average quality of upper bounds to rank the evaluated algorithms under the combinations of varying time bounds and the i bounds in Section 4.5.

4.1.3 Benchmarks

We used four types of benchmark sets, each has ten instances of varying difficulty characterized by the number of variables and the induced width of the DAG of input IDs.

- `FH-MDP`: The instances in this set are finite horizon factored Markov decision process (MDP) problems [Boutilier et al., 1999] generated by first creating a two-stage random Dynamic Bayesian Networks (DBN) [Dean and Kanazawa, 1989] and randomly selecting decision, chance variables from the nodes in the DBN and adding random utility functions. Then, we replicated the two-stage model to desired time steps from 3 to 10. The numbers shown next to the instance name encodes the number of chance variables c at each time step, the maximum domain size k, the number of utility functions u at each time step, and the number of time steps t replicated. For example, `mdp1-4-2-2-5` is generated by parameters c=4, k=2, u=2, and t=5.

	n	c	d	f	p	u	k	s	w
Min	25	20	3	30	20	10	2	4	5
Average	105.7	99.6	6.1	134.1	99.6	34.5	3.1	7.1	25.5
Max	170	160	10	240	160	80	5	9	43
mdp1-4-2-2-5	25	20	5	30	20	10	2	4	5
mdp2-8-3-4-5	45	40	5	60	40	20	3	5	41
mdp3-10-3-5-10	110	100	10	150	100	50	3	6	10
mdp4-10-3-5-10	110	100	10	150	100	50	3	6	13
mdp5-16-3-8-10	170	160	10	240	160	80	3	7	13
mdp6-20-5-5-5	105	100	5	125	100	25	5	9	21
mdp7-28-3-6-5	145	140	5	170	140	30	3	9	28
mdp8-28-3-6-4	116	112	4	136	112	24	3	9	40
mdp9-32-3-8-3	99	96	3	120	96	24	3	9	41
mdp10-32-3-8-4	132	128	4	160	128	32	3	7	43

Table 4.2: Problem Instance Statistics for `FH-MDP` domain. n is the number of variables, c is the number of chance variables, d is the number of decision variables, f is the number of functions p is the number of probability functions, u is the number of utility functions, k is the maximum domain size, s is the maximum scope size, and w is the constrained induced width from a constrained join tree computed by stochastic min-fill algorithm [Kask et al., 2011].

- `FH-POMDP`: The instances in this set are finite horizon factored partially observable MDP

(POMDP) problems [Boutilier et al., 1999] that are generated by the same method for FH-MDP except for the partial observability. To obtain finite horizon POMDP instances, We replicated a two-stage model to 3 to 5 horizons. For the observed variables, we randomly divided the chance variables into observed chance variables and unobserved chance variables in random. The numbers shown next to the instance name encodes the number of unobserved chance variables at each time step, the number of observed chance variables at each time step, the maximum domain size, the number of utility functions at each time step, and the number of time steps replicated. For example, pomdp1-4-2-2-2-3 is generated by replicating a two-stage model with 4 unobserved chance variables, 2 observed chance variables and 2 utility functions to 3 time steps.

	n	c	d	f	p	u	k	s	w
Min	15	12	3	18	12	6	2	3	10
Average	55.9	52.4	3.5	73.5	52.4	21.1	2.4	5.5	28
Max	96	92	5	140	92	48	3	9	46
pomdp1-4-2-2-2-3	21	18	3	24	18	6	2	4	11
pomdp2-2-2-2-2-3	15	12	3	18	12	6	2	3	10
pomdp3-4-4-2-2-3	27	24	3	30	24	6	2	4	16
pomdp4-4-4-2-2-5	45	40	5	50	40	10	2	4	27
pomdp5-6-4-3-5-3	33	30	3	45	30	15	3	5	17
pomdp6-12-6-2-6-3	57	54	3	72	54	18	2	9	28
pomdp7-20-10-2-10-3	93	90	3	120	90	30	2	9	44
pomdp8-14-9-3-12-4	96	92	4	140	92	48	3	6	46
pomdp9-14-8-3-10-4	92	88	4	128	88	40	3	6	43
pomdp10-12-7-3-8-4	80	76	4	108	76	32	3	5	38

Table 4.3: Problem Instance Statistics for FH-POMDP domain. For legend see Table 4.2.

- RAND: The instances in this set have a random graph structure that is generated by creating a random DAG given the number the number of chance, decision, and value nodes. The numbers shown next to the instance name encodes the parameters of random influence diagram generator. For example, rand-c20d2o1-01 is an influence diagram with 20 chance nodes, 2 decision nodes, and 1 immediate observed chance node. Unlike FH-MDP and FH-POMDP instances, RAND instances do not have a repeated structure over the time steps.

	n	c	d	f	p	u	k	s	w
Min	22	20	2	22	20	2	2	3	6
Average	56	47	9	56	47	9	2	3	17
Max	91	70	21	91	70	21	2	3	34
rand-c20d2o1-01	22	20	2	22	20	2	2	3	6
rand-c30d3o1-01	33	30	3	33	30	3	2	3	9
rand-c30d6o1-01	36	30	6	36	30	6	2	3	12
rand-c30d9o1-01	39	30	9	39	30	9	2	3	14
rand-c50d10o1-01	60	50	10	60	50	10	2	3	20
rand-c50d15o1-03	65	50	15	65	50	15	2	3	24
rand-c50d5o1-01	55	50	5	55	50	5	2	3	15
rand-c70d14o1-01	84	70	14	84	70	14	2	3	26
rand-c70d21o1-01	91	70	21	91	70	21	2	3	34
rand-c70d7o1-01	77	70	7	77	70	7	2	3	19

Table 4.4: Problem Instance Statistics for `RAND` domain. For legend see Table 4.2.

- `BN`: The instances in this set are modified from existing Bayesian networks by randomly selecting decision variables and adding random utility functions. While generating the benchmark instances, we took 3 BN instances, $BN-0$, $BN-14$, and $BN-78$, from the UAI-2006 probabilistic inference competitions. The decision nodes and value nodes were randomly selected from non-leaf nodes and leaf nodes in the BN, respectively.

	n	c	d	f	p	u	k	s	w
Min	54	48	3	54	48	3	2	6	12
Average	84	77	7	84	77	7	2	8	21
Max	115	109	12	115	109	12	2	10	42
BN-0-w28d6	100	94	6	100	94	6	2	6	23
BN-0-w29d6	100	94	6	100	94	6	2	6	17
BN-0-w32d11	100	89	11	100	89	11	2	6	24
BN-0-w33d11	100	89	11	100	89	11	2	6	27
BN-14w42d6	115	109	6	115	109	6	2	8	28
BN-14w57d12	115	103	12	115	103	12	2	8	42
BN-78-w18d3	54	51	3	54	51	3	2	10	12
BN-78-w19d3	54	51	3	54	51	3	2	10	14
BN-78-w23d6	54	48	6	54	48	6	2	10	13
BN-78-w24d6	54	48	6	54	48	6	2	10	18

Table 4.5: Problem Instance Statistics for `BN` domain. For legend see Table 4.2.

4.2 Impact of Translations

In this section, we present the statistics of problem instances obtained by translating IDs to BNs, and show the impact of translations. The translated instances are evaluated by bounding schemes for mixed inference and MMAP tasks.

4.2.1 Translation to the Mixed Inference Task

The reduction to the mixed inference task by Liu and Ihler [2011] converts additive utility functions in the MEU task into multiplicative functions by using the following equation Eq. (4.1).

Given an ID $\mathcal{M} := \langle \mathbf{X}, \mathbf{D}, \mathbf{P}, \mathbf{U}, \mathcal{O} \rangle$, the translation uses the following equality,

$$\sum_{U_i \in \mathbf{U}} U_i(\mathbf{X}_{U_i}) = \sum_{Z=1}^{|\mathbf{U}|} \prod_{U'_i \in \mathbf{U}} U'_i(\mathbf{X}_{U_i}, Z) F(Z) \tag{4.1}$$

$$\begin{cases} F(Z = z) = 1 \text{ for all z } \in \{1, 2, \ldots, |\mathbf{U}|\}, \\ U'_i(\mathbf{X}_{U_i}, Z = z) = U_i(\mathbf{X}_{U_i}) \text{ if } z = i, \\ U'_i(\mathbf{X}_{U_i}, Z = z) = 1 \text{ otherwise.} \end{cases} \tag{4.2}$$

We can see that the reduction to the mixed inference task introduces an auxiliary unobserved chance variable Z having domain size equals to the total number of utility functions, which inflates the maximum domain size to the total number of utility functions. The reduction also introduces directed arcs from the auxiliary node to all value nodes, modifying each additive utility function from U_i to multiplicative utility function U'_i as shown in Eq. (4.2), which extends the size of each utility function by a factor of the total number of utility functions. The constant function $F(Z)$ was introduced to define a valid directed graphical model. For the details of the translation see [Liu and Ihler, 2012].

	n	f	k	s	w	ρ_n	ρ_k	ρ_w
Min	26	30	10	4	15	1.04	5	3
Average	106	134	34	7	84	1.01	11.13	3.33
Max	171	240	80	9	160	1.01	16	3.72
mdp1-4-2-2-5	26	30	10	4	15	1.04	5	3
mdp2-8-3-4-5	46	60	20	5	25	1.02	6.67	0.61
mdp3-10-3-5-10	111	150	50	6	110	1.01	16.67	11
mdp4-10-3-5-10	111	150	50	6	100	1.01	16.67	7.69
mdp5-16-3-8-10	171	240	80	7	160	1.01	26.67	12.31
mdp6-20-5-5-5	106	125	25	9	70	1.01	5	3.33
mdp7-28-3-6-5	146	170	30	9	115	1.01	10	4.11
mdp8-28-3-6-4	117	136	24	9	88	1.01	8	2.2
mdp9-32-3-8-3	100	120	24	9	81	1.01	8	1.98
mdp10-32-3-8-4	133	160	32	7	84	1.01	10.67	1.95
pomdp1-4-2-2-2-3	22	24	6	4	12	1.05	3	1.09
pomdp2-2-2-2-2-3	16	18	6	3	10	1.07	3	1
pomdp3-4-4-2-2-3	28	30	6	4	16	1.04	3	1
pomdp4-4-4-2-2-5	46	50	10	4	27	1.02	5	1
pomdp5-6-4-3-5-3	34	45	15	6	17	1.03	5	1
pomdp6-12-6-2-6-3	58	72	18	9	29	1.02	9	1.04
pomdp7-20-10-2-10-3	94	120	30	9	46	1.01	15	1.05
pomdp8-14-9-3-12-4	97	140	48	6	48	1.01	16	1.04
pomdp9-14-8-3-10-4	93	128	40	6	44	1.01	13.33	1.02
pomdp10-12-7-3-8-4	81	108	32	5	38	1.01	10.67	1
rand-c20d2o1-01	23	22	2	4	6	1.05	1	1
rand-c30d3o1-01	34	33	3	4	10	1.03	1.5	1.11
rand-c30d6o1-01	37	36	6	4	14	1.03	3	1.17
rand-c30d9o1-01	40	39	9	4	18	1.03	4.5	1.29
rand-c50d10o1-01	61	60	10	4	24	1.02	5	1.2
rand-c50d15o1-03	66	65	15	4	33	1.02	7.5	1.38
rand-c50d5o1-01	56	55	5	4	17	1.02	2.5	1.13
rand-c70d14o1-01	85	84	14	4	33	1.01	7	1.27
rand-c70d21o1-01	92	91	21	4	42	1.01	10.5	1.24
rand-c70d7o1-01	78	77	7	4	22	1.01	3.5	1.16
BN-0-w28d6	101	100	6	7	29	1.01	3	1.26
BN-0-w29d6	101	100	6	6	30	1.01	3	1.76
BN-0-w32d11	101	100	11	7	33	1.01	5.5	1.38
BN-0-w33d11	101	100	11	7	34	1.01	5.5	1.26
BN-14w42d6	116	115	6	8	42	1.01	3	1.5
BN-14w57d12	116	115	12	9	46	1.01	6	1.1
BN-78-w18d3	55	54	3	10	19	1.02	1.5	1.58
BN-78-w19d3	55	54	3	11	20	1.02	1.5	1.43
BN-78-w23d6	55	54	6	10	23	1.02	3	1.77
BN-78-w24d6	55	54	6	10	25	1.02	3	1.39

Table 4.6: Problem Instance Statistics Translated to the Mixed Inference Task. n is the number of variables, f is the number of functions, k is the maximum domain size, s is the maximum scope size, and w is the constrained induced width. $\rho_n = \frac{n_{\text{trans}}}{n_{\text{ID}}}$, $\rho_k = \frac{k_{\text{trans}}}{k_{\text{ID}}}$, and $\rho_w = \frac{w_{\text{trans}}}{w_{\text{ID}}}$ are the ratio of the number of variables, the maximum domain size, and the constrained induced width between the translated instance denoted by the subscript trans and the original ID denoted by the subscript ID, respectively.

Table 4.6 shows the problem statistics obtained from the translated instances. In FH-MDP domain, we see that the maximum domain size k is increased on average by a factor of 11, and the constrained induced width w is also increased on average by a factor of 3.33. In FH-POMDP domain, we see that the maximum domain size k is increased on average by a factor of 8. In RAND domain, we see that the maximum domain size k is increased on average by a factor of 4.6. Since the number of variables and functions are relatively smaller than in the FH-MDP and FH-POMDP domains, the average ratio of the constrained induced width is close to 1.0. However, we can see the increase in the constrained induced width from 34 to 42 for rand-c70d21o1-01 instance. In BN domain, We see that the maximum domain size k is increased on average by a factor of 3.5.

4.2.2 Translation to MMAP

The reduction to MMAP task by Mauá [2016] modifies the input IDs in two stages. The first stage translates the additive utility functions to the multiplicative utility functions by the reduction to the mixed inference task. The second stage converts the alternating summation and maximization operations in the mixed inference task to the MMAP. For the detailed procedures, see Mauá [2016]. Although the reduction scheme inflates the size of the input problem in polynomial order [Mauá, 2016], many of the translated problem instances in the benchmark sets become intractable due to the increase in the number of variables and the induced width. Denoting a set of decision variables and a set of parent nodes of a decision variable by $\mathbf{D}$ and $\mathrm{pa}(D)$ for $D \in \mathbf{D}$, the reduction method introduces $O(|\mathbf{D}| \cdot k^{\mathrm{pa}(D)})$ decision variables and unobserved chance variables.

Table 4.7 presents the problem instance statistics obtained from the translated MMAP instances. In FH-POMDP domain, We see that the maximum domain size k is increased on average by a factor of 8.3. The constrained induced width w is now inflated by a factor of 32.31 on average. In RAND domain, we see that the maximum domain size k is increased on

	n	f	k	s	w	ρ_n	ρ_k	ρ_w
Min	28	31	6	4	14	1.62	3	1.27
Average	1385.7	1403.3	21.1	7.7	1334.4	16.31	8.3	32.31
Max	5277	5313	48	12	5192	57.36	16	120.74
pomdp1-4-2-2-2-3	34	37	6	4	14	1.62	3	1.27
pomdp2-2-2-2-2-3	28	31	6	4	14	1.87	3	1.4
pomdp3-4-4-2-2-3	76	79	6	6	52	2.81	3	3.25
pomdp4-4-4-2-2-5	126	131	10	6	84	2.8	5	3.11
pomdp5-6-4-3-5-3	82	94	15	6	52	2.48	5	3.06
pomdp6-12-6-2-6-3	250	265	18	9	198	4.39	9	7.07
pomdp7-20-10-2-10-3	3166	3193	30	12	3082	34.04	15	70.05
pomdp8-14-9-3-12-4	2145	2189	48	11	2057	22.34	16	44.72
pomdp9-14-8-3-10-4	5277	5313	40	10	5192	57.36	13.33	120.74
pomdp10-12-7-3-8-4	2673	2701	32	9	2599	33.41	10.67	68.39
rand-c20d2o1-01	29	29	2	4	8	1.32	1	1.33
rand-c30d3o1-01	42	42	3	4	11	1.27	1.5	1.22
rand-c30d6o1-01	53	53	6	4	19	1.47	3	1.58
rand-c30d9o1-01	60	60	9	4	28	1.54	4.5	2
rand-c50d10o1-01	87	87	10	4	29	1.45	5	1.45
rand-c50d15o1-03	104	104	15	4	49	1.6	7.5	2.04
rand-c50d5o1-01	72	72	5	4	18	1.31	2.5	1.2
rand-c70d14o1-01	121	121	14	4	45	1.44	7	1.73
rand-c70d21o1-01	142	142	21	4	58	1.56	10.5	1.71
rand-c70d7o1-01	98	98	7	4	23	1.27	3.5	1.21
Min	69	69	3	6	20	1.19	1.5	1.43
Average	122	122	7	8.5	43.1	1.45	3.5	2
Max	202	202	12	11	92	1.76	6	3.31
BN-0-w28d6	137	137	6	7	40	1.37	3	1.74
BN-0-w29d6	119	119	6	6	31	1.19	3	1.82
BN-0-w32d11	133	133	11	7	41	1.33	5.5	1.71
BN-0-w33d11	159	159	11	7	62	1.59	5.5	2.3
BN-14w42d6	150	150	6	8	46	1.3	3	1.64
BN-14w57d12	202	202	12	9	92	1.76	6	2.19
BN-78-w18d3	79	79	3	10	28	1.46	1.5	2.33
BN-78-w19d3	69	69	3	11	20	1.28	1.5	1.43
BN-78-w23d6	93	93	6	10	43	1.72	3	3.31
BN-78-w24d6	79	79	6	10	28	1.46	3	1.56

Table 4.7: Problem Instance Statistics Translated to MMAP Inference Task. in `FH-POMDP` domain. n is the number of variables, f is the number of functions, k is the maximum domain size, s is the maximum scope size, and w is the constrained induced width. $\rho_n = \frac{n_{\text{trans}}}{n_{\text{ID}}}$, $\rho_k = \frac{k_{\text{trans}}}{k_{\text{ID}}}$, and $\rho_w = \frac{w_{\text{trans}}}{w_{\text{ID}}}$ are the ratio of the number of variables, the maximum domain size, and the constrained induced width between the translated instance denoted by the subscript trans and the original ID denoted by the subscript ID, respectively.

average by a factor of 4.6. Since the number of parent nodes of a decision variable is relatively smaller than `FH-POMDP` domain, the average ratio of the number of variables and constrained induced widht are closed to 1.0, in both cases. In `BN` domain, We see that the maximum domain size k is increased on average by a factor of 3.5, and the constrained induced width is also increased on average by a factor of 2. Due to this inflation of the problem size, we failed to translate `FH-MDP` instances within 4 GB memory limit, except for the easiest instance `mdp1-4-2-2-5`, yielding statistics the number of variables 106, the number of functions 111, the maximum domain size 10, the maximum scope size 6, and the constrained induced width 84. Note that the constrained induced width of the `mdp1-4-2-2-5` instance is only 5 in the original ID, but it is inflated to 84 by the MMAP translation.

4.2.3 Summary of the Impact of Translation

From all the above statistics, we see that the translation based approach often converts the IDs to intractable problems. This can have harsh consequence on inference. For example, we can solve `mdp1-4-2-2-5` instance exactly by using variable elimination since its induced width is only 5 as shown in Table 4.2. However, the translation to MMAP task inflates the induced width to 84. In summary, the translation into the mixed inference task inflates the maximum domains size k to the total number of utility functions. On top of this inflation, the subsequent translation into MMAP introduces new decision variables and chance variables that are exponential in the size of the number of observed variables for each decision. All these illustrate clearly why developing direct decomposition bounds for the MEU task is desirable. In the next section, we will evaluate all our proposed algorithms that developed directly over the MEU representation and compare the results with the earlier schemes.

Iteration parameter	JGD-ID	WMBE-ID	JGD-EXP	GDD-MI
BCD outer-loop limit M	1000	5	1000	1000
BCD outer-loop convergence ϵ	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$
Inner-loop optimizer	gradient descent	SLSQP	gradient descent	gradient descent
Inner-loop iteration limits	20	100	10	20

Table 4.8: Hyper Parameters Used for Iterative Algorithms

4.3 Upper Bounds from Individual Instances

In this section, we report the upper bounds generated by all algorithms shown in Table 4.1 at individual instances over the time bounds 10, 100, 1000, 10,000, and 1,000,000 seconds, and i-bounds 1, 5, 10, 15, and 20. The total memory limit was 4 GB for all cases, and when an algorithm was terminated by the memory limit, we report the upper bound inf at all time bounds. Algorithms JGD-ID, JGD-EXP, and GDD-MI are iterative algorithms that improve the upper bounds when given more time until they reach convergence. Table 4.8 presents the hyper-parameters related to the iteration limits. For the outer-loop of BCD updates, M is the maximum iteration limit, and ϵ is the convergence parameter that terminates an algorithm when the improvement of the upper bound is below ϵ. The outer-loop iteration limit is set to 1,000. The iteration limit for the BCD outer-loop for algorithm WMBE-ID was set to 5 at each layer of a mini-bucket tree. For details on the BCD updates, refer back to Chapter 3. For inner-loop optimization, each algorithm uses different optimization routines. Algorithms JGD-ID, JGD-EXP, and GDD-MI use gradient descent algorithms with line search, where we set the iteration limit for the gradient descent update to 20 for JGD-ID and GDD-MI, and to 10 for JGD-EXP. Algorithm WMBE-ID uses SLSQP where we set the iteration limit parameter to 100. Note that algorithms WMBE-ID, WMBMM-EXP, WMBMM-MMAP, and MBE-ID generate an upper bound when they terminate.

4.3.1 FH-MDP Domain

We summarize the upper bounds for instances in the `FH-MDP` domain in Tables 4.9 to 4.13. We first observe that algorithm MBE-ID generated orders of magnitude worse upper bounds compared with the other algorithms. The translation based approach algorithm GDD-MI generated upper bounds that are worse than the direct decomposition bounds, except for the single case `mdp8-28-3-6-5` with i-bound 1.

Comparing two approaches in the direct decomposition bounds, namely, the bounding schemes using valuation algebra for IDs presented in Section 3.2 and the bounding schemes using exponentiated utility functions in Section 3.3, we observe that both generated similar upper bounds when problem instances are easy. As problem instances have a higher constrained induced width and a larger number of variables, algorithm JGD-EXP and WMBMM-EXP started to dominate other algorithms in shorter time bounds, yet algorithm WMBE-ID was able to provide the tightest upper bounds for all instances except for the `mdp5-16-3-8-10` instance when it used i-bound 15 or 20. From `mdp5-16-3-8-10` to `mdp10-32-3-8-4`, we see that algorithm WMBMM-EXP generated the best upper bounds when time bounds is less than 1,000 seconds, and algorithm JGD-EXP starts to generate tighter bounds than algorithm WMBMM-EXP given more time when i-bound is 1 or 5. Algorithm JGD-ID often provided high quality upper bounds at i-bound 1, but it generated very loose bounds when it was given higher i-bounds. Overall, we see that direct decomposition bounds dominated the earlier algorithms for generating upper bounds in the `FH-MDP` domain. In shorter time bounds, the non-iterative algorithm WMBMM-EXP generated high quality upper bounds quickly than other algorithms, and algorithm JGD-EXP often provided the tightest upper bounds with lower i-bounds, 1 or 5 at time bounds after 1,000 seconds. Comparing algorithms WMMM-EXP and WMBE-ID, WMBE-ID took much longer time than WMBMM-EXP, but it produced better upper bounds when it was given higher i-bound, 15 or 20.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
mdp1-4-2-2-5	1	10E+1	8.959	inf	7.66	**4.976**	99.263	3.04E+04	509.555
n:25 f:30		10E+2	8.959	inf	7.66	4.976	99.263	3.04E+04	509.555
k:2 s:4		10E+3	4.568	4.389	**4.176**	4.976	4.45	3.04E+04	509.555
w:5		10E+4	4.187	4.389	**4.176**	4.976	4.308	3.04E+04	509.555
		10E+6	4.187	4.389	**4.176**	4.976	4.308	3.04E+04	509.555
	5	10E+1	4.312	inf	inf	3.738	1681.511	1.20E+04	**3.695**
		10E+2	4.312	inf	inf	3.738	1681.511	1.20E+04	**3.695**
		10E+3	**3.695**	**3.695**	4.089	3.738	4.867	1.20E+04	**3.695**
		10E+4	**3.695**	**3.695**	4.07	3.738	4.217	1.20E+04	**3.695**
		10E+6	**3.695**	**3.695**	4.07	3.738	4.2	1.20E+04	**3.695**
	10	10E+1	4.312	inf	inf	3.738	inf	1758.53	**3.695**
		10E+2	4.312	inf	inf	3.738	inf	1758.53	**3.695**
		10E+3	**3.695**	**3.695**	4.07	3.738	5.001	1758.53	**3.695**
		10E+4	**3.695**	**3.695**	4.07	3.738	4.236	1758.53	**3.695**
		10E+6	**3.695**	**3.695**	4.07	3.738	4.186	1758.53	**3.695**
	15	10E+1	4.312	**3.695**	inf	3.738	inf	343.19	**3.695**
		10E+2	4.312	**3.695**	inf	3.738	inf	343.19	**3.695**
		10E+3	**3.695**	**3.695**	4.076	3.738	9.011	343.19	**3.695**
		10E+4	**3.695**	**3.695**	4.076	3.738	4.187	343.19	**3.695**
		10E+6	**3.695**	**3.695**	4.076	3.738	4.117	343.19	**3.695**
	20	10E+1	4.312	inf	inf	3.738	inf	206.404	**3.695**
		10E+2	4.312	inf	inf	3.738	inf	206.404	**3.695**
		10E+3	3.704	**3.695**	4.076	3.738	6.024	206.404	**3.695**
		10E+4	3.704	**3.695**	4.076	3.738	4.687	206.404	**3.695**
		10E+6	3.704	**3.695**	4.076	3.738	4.687	206.404	**3.695**
mdp2-8-3-4-5	1	10E+1	inf	inf	inf	**17.401**	1.51E+05	inf	1.02E+08
n:45 f:60		10E+2	inf	inf	inf	**17.401**	1.51E+05	inf	1.02E+08
k:3 s:5		10E+3	17.449	17.4	**15.797**	17.401	22.133	inf	1.02E+08
w:10		10E+4	15.451	17.4	**13.858**	17.401	16.51	inf	1.02E+08
		10E+6	15.139	17.4	**13.858**	17.401	16.301	inf	1.02E+08
	5	10E+1	inf	inf	inf	**16.634**	inf	inf	1.81E+05
		10E+2	inf	inf	inf	**16.634**	inf	inf	1.81E+05
		10E+3	1486.339	**13.946**	15.689	16.634	32.216	inf	1.81E+05
		10E+4	22.402	13.946	**13.326**	16.634	18.967	inf	1.81E+05
		10E+6	22.402	13.946	**13.326**	16.634	17.788	inf	1.81E+05
	10	10E+1	inf	inf	inf	12.197	inf	inf	**11.85**
		10E+2	inf	inf	inf	12.197	inf	inf	**11.85**
		10E+3	12.314	**11.85**	20.192	12.197	2072.068	inf	**11.85**
		10E+4	12.293	**11.85**	13.554	12.197	19.654	inf	**11.85**
		10E+6	12.293	**11.85**	13.554	12.197	16.512	inf	**11.85**
	15	10E+1	inf	inf	inf	12.197	inf	inf	**11.85**
		10E+2	inf	inf	inf	12.197	inf	inf	**11.85**
		10E+3	12.314	**11.85**	20.192	12.197	inf	inf	**11.85**
		10E+4	12.293	**11.85**	13.552	12.197	7.41E+06	inf	**11.85**
		10E+6	12.296	**11.85**	13.552	12.197	708.896	inf	**11.85**
	20	10E+1	inf	inf	inf	12.197	inf	inf	**11.85**
		10E+2	inf	inf	inf	12.197	inf	inf	**11.85**
		10E+3	12.296	**11.85**	19.592	12.197	inf	inf	**11.85**
		10E+4	12.296	**11.85**	13.552	12.197	inf	inf	**11.85**
		10E+6	12.296	**11.85**	13.552	12.197	inf	inf	**11.85**

Table 4.9: Upper Bounds from `mdp1-4-2-2-5` and `mdp2-8-3-4-5` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
mdp3-10-3-5-10	1	10E+1	inf	inf	inf	**53.828**	inf	inf	6.47E+23
n:110 f:150		10E+2	inf	inf	inf	**53.828**	inf	inf	6.47E+23
k:3 s:6		10E+3	4763.942	inf	69.633	**53.828**	1.40E+06	inf	6.47E+23
w:14		10E+4	91.116	48.137	**42.292**	53.828	76.975	inf	6.47E+23
		10E+6	49.434	48.137	**41.641**	53.828	49.257	inf	6.47E+23
	5	10E+1	inf	inf	inf	**48.686**	inf	inf	7.58E+19
		10E+2	inf	inf	inf	**48.686**	inf	inf	7.58E+19
		10E+3	7.22E+12	inf	68.995	**48.686**	2.38E+10	inf	7.58E+19
		10E+4	1.08E+07	44.828	**40.256**	48.686	105.823	inf	7.58E+19
		10E+6	505.832	44.828	**39.548**	48.686	50.349	inf	7.58E+19
	10	10E+1	inf	inf	inf	**39.077**	inf	inf	7.96E+08
		10E+2	inf	inf	inf	**39.077**	inf	inf	7.96E+08
		10E+3	1.36E+08	inf	66.699	**39.077**	2.83E+18	inf	7.96E+08
		10E+4	3.20E+05	**37.129**	37.197	39.077	5.97E+07	inf	7.96E+08
		10E+6	516.228	37.129	**36.995**	39.077	84.082	inf	7.96E+08
	15	10E+1	inf	inf	inf	31.114	inf	inf	**29.732**
		10E+2	inf	inf	inf	31.114	inf	inf	**29.732**
		10E+3	inf	inf	inf	31.114	inf	inf	**29.732**
		10E+4	inf	inf	68.688	31.114	inf	inf	**29.732**
		10E+6	30.022	**29.732**	39.545	31.114	2.54E+18	inf	**29.732**
	20	10E+1	inf	inf	inf	31.114	inf	inf	**29.732**
		10E+2	inf	inf	inf	31.114	inf	inf	**29.732**
		10E+3	inf	inf	inf	31.114	inf	inf	**29.732**
		10E+4	inf	inf	68.688	31.114	inf	inf	**29.732**
		10E+6	30.026	**29.732**	39.545	31.114	inf	inf	**29.732**
mdp4-10-3-5-10	1	10E+1	inf	inf	inf	**59.539**	inf	inf	1.55E+22
n:110 f:150		10E+2	inf	inf	inf	**59.539**	inf	inf	1.55E+22
k:3 s:6		10E+3	1.26E+05	inf	73.848	**59.539**	2.28E+06	inf	1.55E+22
w:14		10E+4	371.913	47.164	**45.715**	59.539	88.937	inf	1.55E+22
		10E+6	48.183	47.164	**44.889**	59.539	48.774	inf	1.55E+22
	5	10E+1	inf	inf	inf	**52.883**	inf	inf	1.64E+17
		10E+2	inf	inf	inf	**52.883**	inf	inf	1.64E+17
		10E+3	5.38E+12	inf	66.508	**52.883**	2.86E+13	inf	1.64E+17
		10E+4	4.37E+07	45.618	**43.467**	52.883	295.296	inf	1.64E+17
		10E+6	2.96E+05	45.618	**42.37**	52.883	68.431	inf	1.64E+17
	10	10E+1	inf	inf	inf	**41.475**	inf	inf	4.51E+09
		10E+2	inf	inf	inf	**41.475**	inf	inf	4.51E+09
		10E+3	2.36E+08	inf	64.352	**41.475**	4.11E+17	inf	4.51E+09
		10E+4	1.74E+06	39.042	**38.789**	41.475	1.07E+05	inf	4.51E+09
		10E+6	554.67	39.042	**38.454**	41.475	121.374	inf	4.51E+09
	15	10E+1	inf	inf	inf	32.752	inf	inf	**31.148**
		10E+2	inf	inf	inf	32.752	inf	inf	**31.148**
		10E+3	inf	inf	72.057	32.752	inf	inf	**31.148**
		10E+4	31.564	**31.148**	52.328	32.752	7.38E+19	inf	**31.148**
		10E+6	31.584	**31.148**	39.961	32.752	1.07E+14	inf	**31.148**
	20	10E+1	inf	inf	inf	32.752	inf	inf	**31.148**
		10E+2	inf	inf	inf	32.752	inf	inf	**31.148**
		10E+3	inf	inf	70.494	32.752	inf	inf	**31.148**
		10E+4	31.584	**31.148**	50.388	32.752	inf	inf	**31.148**
		10E+6	31.584	**31.148**	39.952	32.752	inf	inf	**31.148**

Table 4.10: Upper Bounds from `mdp3-10-3-5-10` and `mdp4-10-3-5-10` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
mdp5-16-3-8-10	1	10E+1	inf	inf	inf	**93.178**	inf	inf	1.40E+38
n:170 f:240		10E+2	inf	inf	inf	**93.178**	inf	inf	1.40E+38
k:3 s:7		10E+3	5.88E+07	inf	136.503	**93.178**	2.09E+17	inf	1.40E+38
w:22		10E+4	890.161	5.64E+11	100.564	**93.178**	1035.45	inf	1.40E+38
		10E+6	78.722	5.64E+11	**69.277**	93.178	90.01	inf	1.40E+38
	5	10E+1	inf	inf	inf	**85.719**	inf	inf	2.92E+31
		10E+2	inf	inf	inf	**85.719**	inf	inf	2.92E+31
		10E+3	2.38E+23	inf	127.47	**85.719**	8.81E+24	inf	2.92E+31
		10E+4	6.00E+14	**73.951**	82.668	85.719	8.89E+04	inf	2.92E+31
		10E+6	2.73E+08	73.951	**69.95**	85.719	117.693	inf	2.92E+31
	10	10E+1	inf	inf	inf	**79.597**	inf	inf	8.03E+24
		10E+2	inf	inf	inf	**79.597**	inf	inf	8.03E+24
		10E+3	4.70E+23	inf	147.733	**79.597**	4.98E+32	inf	8.03E+24
		10E+4	3.76E+17	**68.956**	123.112	79.597	8.22E+22	inf	8.03E+24
		10E+6	3.97E+13	68.956	**66.642**	79.597	737.515	inf	8.03E+24
	15	10E+1	inf	inf	inf	inf	inf	inf	inf
		10E+2	inf	inf	inf	inf	inf	inf	inf
		10E+3	inf	inf	inf	**65.463**	inf	inf	4.31E+13
		10E+4	inf	inf	inf	**65.463**	inf	inf	4.31E+13
		10E+6	79.724	**60.028**	inf	65.463	inf	inf	4.31E+13
	20	10E+1	inf	inf	inf	inf	inf	inf	inf
		10E+2	inf	inf	inf	inf	inf	inf	inf
		10E+3	inf	inf	inf	**49.817**	inf	inf	inf
		10E+4	inf	inf	inf	**49.817**	inf	inf	inf
		10E+6	inf	inf	inf	**49.817**	inf	inf	inf
mdp6-20-5-5-5	1	10E+1	inf	inf	inf	**39.74**	inf	inf	1.30E+22
n:105 f:125		10E+2	inf	inf	inf	**39.74**	inf	inf	1.30E+22
k:5 s:9		10E+3	5195.461	inf	57.82	**39.74**	2.25E+08	inf	1.30E+22
w:29		10E+4	40.691	inf	43.94	**39.74**	43.445	inf	1.30E+22
		10E+6	24.538	24.237	**23.859**	39.74	24.148	inf	1.30E+22
	5	10E+1	inf	inf	inf	**36.996**	inf	inf	1.18E+18
		10E+2	inf	inf	inf	**36.996**	inf	inf	1.18E+18
		10E+3	7.89E+09	inf	58.157	**36.996**	3.62E+10	inf	1.18E+18
		10E+4	8.32E+04	inf	40.832	**36.996**	66.786	inf	1.18E+18
		10E+6	29.062	**23.698**	23.892	36.996	24.249	inf	1.18E+18
	10	10E+1	inf	inf	inf	**31.842**	inf	inf	2.85E+14
		10E+2	inf	inf	inf	**31.842**	inf	inf	2.85E+14
		10E+3	2.58E+13	inf	58.334	**31.842**	8.36E+13	inf	2.85E+14
		10E+4	1.14E+09	**23.017**	43.35	31.842	3323.858	inf	2.85E+14
		10E+6	2.25E+06	**23.017**	23.266	31.842	27.458	inf	2.85E+14
	15	10E+1	inf	inf	inf	**29.394**	inf	inf	3.57E+10
		10E+2	inf	inf	inf	**29.394**	inf	inf	3.57E+10
		10E+3	inf	inf	64.178	**29.394**	inf	inf	3.57E+10
		10E+4	6.06E+08	**21.012**	59.757	29.394	3.42E+14	inf	3.57E+10
		10E+6	1.78E+07	**21.012**	25.22	29.394	8.08E+11	inf	3.57E+10
	20	10E+1	inf	inf	inf	inf	inf	inf	inf
		10E+2	inf	inf	inf	inf	inf	inf	inf
		10E+3	inf	inf	inf	**22.993**	inf	inf	1.39E+07
		10E+4	inf	inf	inf	**22.993**	inf	inf	1.39E+07
		10E+6	inf	**19.666**	inf	22.993	inf	inf	1.39E+07

Table 4.11: Upper Bounds from `mdp5-16-3-8-10` and `mdp6-20-5-5-5` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
mdp7-28-3-6-5	1	10E+1	inf	inf	inf	**51.671**	inf	inf	1.63E+32
n:145 f:170		10E+2	inf	inf	inf	**51.671**	inf	inf	1.63E+32
k:3 s:9		10E+3	3.72E+04	inf	81.035	**51.671**	2.03E+13	inf	1.63E+32
w:40		10E+4	60.965	inf	61.143	**51.671**	136.154	inf	1.63E+32
		10E+6	**29.414**	29.852	29.747	51.671	30.958	inf	1.63E+32
	5	10E+1	inf	inf	inf	**48.291**	inf	inf	2.66E+26
		10E+2	inf	inf	inf	**48.291**	inf	inf	2.66E+26
		10E+3	8.67E+13	inf	78.287	**48.291**	3.04E+16	inf	2.66E+26
		10E+4	3.77E+06	**29.003**	60.096	48.291	2272.862	inf	2.66E+26
		10E+6	36.814	**29.003**	30.343	48.291	32.238	inf	2.66E+26
	10	10E+1	inf	inf	inf	**41.855**	inf	inf	1.06E+22
		10E+2	inf	inf	inf	**41.855**	inf	inf	1.06E+22
		10E+3	2.25E+22	inf	83.847	**41.855**	9.23E+20	inf	1.06E+22
		10E+4	1.52E+15	**28.423**	65.732	41.855	1.23E+10	inf	1.06E+22
		10E+6	2.39E+09	**28.423**	32.051	41.855	40.234	inf	1.06E+22
	15	10E+1	inf	inf	inf	**41.843**	inf	inf	1.18E+19
		10E+2	inf	inf	inf	**41.843**	inf	inf	1.18E+19
		10E+3	inf	inf	inf	**41.843**	inf	inf	1.18E+19
		10E+4	6.31E+17	inf	85.044	**41.843**	1.53E+20	inf	1.18E+19
		10E+6	6.00E+12	**27.628**	44.361	41.843	2036.194	inf	1.18E+19
	20	10E+1	inf	inf	inf	inf	inf	inf	inf
		10E+2	inf	inf	inf	inf	inf	inf	inf
		10E+3	inf	inf	inf	**37.165**	inf	inf	4.46E+13
		10E+4	inf	inf	inf	**37.165**	inf	inf	4.46E+13
		10E+6	inf	**25.853**	inf	37.165	inf	inf	4.46E+13
mdp8-28-3-6-4	1	10E+1	inf	inf	inf	**38.793**	inf	inf	5.14E+24
n:116 f:136		10E+2	inf	inf	inf	**38.793**	inf	inf	5.14E+24
k:3 s:9		10E+3	3762.176	inf	61.281	**38.793**	1.11E+09	inf	5.14E+24
w:40		10E+4	**37.121**	inf	45.207	38.793	46.562	inf	5.14E+24
		10E+6	24.715	23.411	22.9	38.793	**22.715**	inf	5.14E+24
	5	10E+1	inf	inf	inf	**36.596**	inf	inf	1.27E+20
		10E+2	inf	inf	inf	**36.596**	inf	inf	1.27E+20
		10E+3	2.54E+08	inf	61.183	**36.596**	5.51E+11	inf	1.27E+20
		10E+4	1.07E+04	inf	46.127	**36.596**	88.122	inf	1.27E+20
		10E+6	29.626	**22.872**	23.444	36.596	23.192	inf	1.27E+20
	10	10E+1	inf	inf	inf	**32.162**	inf	inf	8.63E+16
		10E+2	inf	inf	inf	**32.162**	inf	inf	8.63E+16
		10E+3	4.72E+13	inf	60.016	**32.162**	1.55E+15	inf	8.63E+16
		10E+4	6.36E+08	inf	46.125	**32.162**	1.86E+05	inf	8.63E+16
		10E+6	1.66E+05	**22.408**	23.703	32.162	24.555	inf	8.63E+16
	15	10E+1	inf	inf	inf	**29.761**	inf	inf	7.52E+13
		10E+2	inf	inf	inf	**29.761**	inf	inf	7.52E+13
		10E+3	inf	inf	inf	**29.761**	2.66E+18	inf	7.52E+13
		10E+4	7.36E+11	inf	60.223	**29.761**	4.24E+13	inf	7.52E+13
		10E+6	3.50E+07	**21.131**	28.564	29.761	60.655	inf	7.52E+13
	20	10E+1	inf	inf	inf	inf	inf	inf	inf
		10E+2	inf	inf	inf	inf	inf	inf	inf
		10E+3	inf	inf	inf	**28.31**	inf	inf	1.21E+10
		10E+4	inf	inf	67.048	**28.31**	inf	inf	1.21E+10
		10E+6	inf	**20.243**	67.048	28.31	inf	inf	1.21E+10

Table 4.12: Upper Bounds from `mdp7-28-3-6-5` and `mdp8-28-3-6-4` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
mdp9-32-3-8-3	1	10E+1	inf	inf	inf	**29.285**	inf	inf	2.41E+21
n:99 f:120		10E+2	inf	inf	inf	**29.285**	inf	inf	2.41E+21
k:3 s:9		10E+3	648.758	inf	45.147	**29.285**	3.30E+04	inf	2.41E+21
w:43		10E+4	39.34	inf	28.888	29.285	**28.362**	inf	2.41E+21
		10E+6	23.087	22.731	**19.83**	29.285	21.984	inf	2.41E+21
	5	10E+1	inf	inf	inf	**28.513**	inf	inf	4.32E+14
		10E+2	inf	inf	inf	**28.513**	inf	inf	4.32E+14
		10E+3	3.23E+06	inf	42.747	**28.513**	5.40E+06	inf	4.32E+14
		10E+4	2394.589	**22.308**	30.015	28.513	34.67	inf	4.32E+14
		10E+6	27.36	22.308	**20.795**	28.513	22.275	inf	4.32E+14
	10	10E+1	inf	inf	inf	**27.292**	inf	inf	1.59E+12
		10E+2	inf	inf	inf	**27.292**	inf	inf	1.59E+12
		10E+3	7.75E+10	inf	48.277	**27.292**	1.59E+09	inf	1.59E+12
		10E+4	1.36E+07	inf	35.457	**27.292**	160.15	inf	1.59E+12
		10E+6	6.05E+04	**21.072**	21.772	27.292	22.587	inf	1.59E+12
	15	10E+1	inf	inf	inf	**24.966**	inf	inf	1.74E+10
		10E+2	inf	inf	inf	**24.966**	inf	inf	1.74E+10
		10E+3	inf	inf	52.37	**24.966**	1.16E+14	inf	1.74E+10
		10E+4	2.55E+08	inf	45.481	**24.966**	4.99E+08	inf	1.74E+10
		10E+6	1.60E+05	**20.352**	24.554	24.966	38.468	inf	1.74E+10
	20	10E+1	inf	inf	inf	inf	inf	inf	inf
		10E+2	inf	inf	inf	inf	inf	inf	inf
		10E+3	inf	inf	inf	**23.371**	inf	inf	4.15E+07
		10E+4	inf	inf	inf	**23.371**	inf	inf	4.15E+07
		10E+6	inf	**19.243**	inf	23.371	inf	inf	4.15E+07
mdp10-32-3-8-4	1	10E+1	inf	inf	inf	**44.327**	inf	inf	3.26E+25
n:132 f:160		10E+2	inf	inf	inf	**44.327**	inf	inf	3.26E+25
k:3 s:7		10E+3	1824.415	inf	59.087	**44.327**	2.60E+07	inf	3.26E+25
w:42		10E+4	47.76	inf	**38.992**	44.327	51.928	inf	3.26E+25
		10E+6	30.316	1.42E+12	**27.565**	44.327	30.361	inf	3.26E+25
	5	10E+1	inf	inf	inf	**41.004**	inf	inf	4.46E+19
		10E+2	inf	inf	inf	**41.004**	inf	inf	4.46E+19
		10E+3	2.12E+11	inf	61.484	**41.004**	4.64E+11	inf	4.46E+19
		10E+4	1.61E+05	inf	**37.024**	41.004	118.214	inf	4.46E+19
		10E+6	38.601	3.80E+12	**29.073**	41.004	31.747	inf	4.46E+19
	10	10E+1	inf	inf	inf	**38.788**	inf	inf	6.21E+16
		10E+2	inf	inf	inf	**38.788**	inf	inf	6.21E+16
		10E+3	1.07E+16	inf	71.321	**38.788**	7.55E+14	inf	6.21E+16
		10E+4	1.25E+10	5.99E+12	53.748	**38.788**	2.92E+04	inf	6.21E+16
		10E+6	7.01E+06	5.99E+12	**30.912**	38.788	36.787	inf	6.21E+16
	15	10E+1	inf	inf	inf	**37.556**	inf	inf	6.69E+12
		10E+2	inf	inf	inf	**37.556**	inf	inf	6.69E+12
		10E+3	inf	inf	inf	**37.556**	inf	inf	6.69E+12
		10E+4	1.62E+12	inf	72.865	**37.556**	1.71E+15	inf	6.69E+12
		10E+6	1.85E+09	**27.941**	37.073	37.556	464.69	inf	6.69E+12
	20	10E+1	inf	inf	inf	inf	inf	inf	inf
		10E+2	inf	inf	inf	inf	inf	inf	inf
		10E+3	inf	inf	inf	**35.536**	inf	inf	4.84E+10
		10E+4	inf	inf	inf	**35.536**	inf	inf	4.84E+10
		10E+6	inf	inf	inf	**35.536**	inf	inf	4.84E+10

Table 4.13: Upper Bounds from `mdp9-32-3-8-3` and `mdp10-32-3-8-4` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

4.3.2 FH-POMDP Domain

We next look at instances from the `FH-POMDP` domain from Table 4.14 to Table 4.18. We see immediately that algorithm MBE-ID generated loose upper bounds compared with other algorithms. The translation based approaches, algorithms GDD-MI and WMBMM-MMAP also generated orders of magnitude worse upper bounds than direct decomposition schemes, except for the easiest problem instances `pomdp1-4-2-2-2-3` and `pomdp2-2-2-2-2-3`.

Comparing the GDD based iterative algorithms, JGD-ID and JGD-EXP, we see that JGD-ID with i-bound 1 or 5 generated tighter upper bounds than JGD-EXP for easy problems (from `pomdp3-4-4-2-2-3` to `pomdp5-5-4-3-5-3`). As the induced width grows beyond 30 (from `pomdp6-12-6-2-6-3` to `pomdp10-12-7-3-8-4`), JGD-EXP generated orders of magnitude tighter upper bounds compared with JGD-ID. We also observe that both JGD-ID and JGD-EXP reached the 4 GB memory limit for i-bound 20 at hard instances having constrained induced width larger than 38. Comparing WMBE-ID and WMBMM-EXP, we observe that WMBMM-EXP dominated WMBE-ID over all time bounds and i-bounds at every instance except for the first two trivially easy instances.

Overall, we see that direct decomposition schemes dominated the earlier algorithms. When problem instances have a smaller induced width less than 30, algorithm JGD-ID generated best upper bounds given i-bounds 1 or 5 and given time bounds greater than 1,000 seconds. However, JGD-EXP could generate tighter upper bounds than JGD-ID at harder problem instances. Algorithm WMBMM-EXP performed well when the time bound is less than 1,000 seconds because both JGD-ID and JGD-EXP couldn't generate the first upper bound within such a short time bound. When the i-bound increases up to 20, algorithm WMBMM-EXP generated the best upper bounds at problem instances having the constrained induced width larger than 30.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
pomdp1-4-2-2-2-3	1	10E+1	113.875	inf	7.842	**6.46**	90.747	370.259	5431.621
n:21 f:24		10E+2	113.875	inf	7.842	**6.46**	90.747	370.259	5431.621
k:2 s:4		10E+3	51.579	95.378	6.715	**6.46**	33.035	370.259	5431.621
w:10		10E+4	**4.813**	95.378	6.715	6.46	32.352	370.259	5431.621
		10E+6	**4.813**	95.378	6.715	6.46	32.352	370.259	5431.621
	5	10E+1	inf	inf	inf	**5.469**	67.981	78.489	95.493
		10E+2	inf	inf	inf	**5.469**	67.981	78.489	95.493
		10E+3	5.591	22.312	5.852	**5.469**	22.386	78.489	95.493
		10E+4	5.591	22.312	5.852	**5.469**	17.974	78.489	95.493
		10E+6	5.591	22.312	5.852	**5.469**	17.517	78.489	95.493
	10	10E+1	inf	inf	inf	4.099	93.129	11.701	**4.007**
		10E+2	inf	inf	inf	4.099	93.129	11.701	**4.007**
		10E+3	4.01	**4.007**	5.773	4.099	15.245	11.701	**4.007**
		10E+4	4.009	**4.007**	5.769	4.099	13.228	11.701	**4.007**
		10E+6	4.009	**4.007**	5.769	4.099	12.784	11.701	**4.007**
	15	10E+1	inf	inf	inf	4.099	inf	**4.005**	4.007
		10E+2	inf	inf	inf	4.099	inf	**4.005**	4.007
		10E+3	4.01	4.007	5.792	4.099	14.705	**4.005**	4.007
		10E+4	4.009	4.007	5.769	4.099	13.132	**4.005**	4.007
		10E+6	4.009	4.007	5.769	4.099	12.81	**4.005**	4.007
	20	10E+1	inf	inf	inf	4.099	inf	**4.005**	4.007
		10E+2	inf	inf	inf	4.099	inf	**4.005**	4.007
		10E+3	4.01	4.007	5.771	4.099	15.746	**4.005**	4.007
		10E+4	4.01	4.007	5.769	4.099	15.346	**4.005**	4.007
		10E+6	4.01	4.007	5.769	4.099	15.346	**4.005**	4.007
pomdp2-2-2-2-2-3	1	10E+1	61.441	inf	**6.483**	6.533	48.742	295.74	255.23
n:15 f:18		10E+2	61.441	inf	6.483	6.533	48.742	295.74	255.23
k:2 s:3		10E+3	**6.051**	51.379	6.398	6.533	27.338	295.74	255.23
w:10		10E+4	**5.771**	51.379	6.398	6.533	26.888	295.74	255.23
		10E+6	**5.771**	51.379	6.398	6.533	26.888	295.74	255.23
	5	10E+1	26.54	inf	5.591	**5.272**	21.846	74.229	11.142
		10E+2	26.54	inf	5.591	5.272	21.846	74.229	11.142
		10E+3	6.904	6.688	**5.022**	5.272	9.145	74.229	11.142
		10E+4	6.796	6.688	**5.022**	5.272	8.464	74.229	11.142
		10E+6	6.795	6.688	**5.022**	5.272	8.464	74.229	11.142
	10	10E+1	5.152	inf	5.675	4.421	24.017	10.393	**4.235**
		10E+2	5.152	inf	5.675	4.421	24.017	10.393	**4.235**
		10E+3	4.242	**4.235**	4.856	4.421	8.613	10.393	**4.235**
		10E+4	4.239	**4.235**	4.856	4.421	8.309	10.393	**4.235**
		10E+6	4.239	**4.235**	4.856	4.421	8.309	10.393	**4.235**
	15	10E+1	5.152	inf	5.675	4.421	24.017	**4.233**	4.235
		10E+2	5.152	inf	5.675	4.421	24.017	**4.233**	4.235
		10E+3	4.242	4.235	4.856	4.421	8.614	**4.233**	4.235
		10E+4	4.259	4.235	4.856	4.421	8.992	**4.233**	4.235
		10E+6	4.259	4.235	4.856	4.421	8.992	**4.233**	4.235
	20	10E+1	5.145	inf	5.675	4.421	28.162	**4.233**	4.235
		10E+2	5.145	inf	5.675	4.421	28.162	**4.233**	4.235
		10E+3	4.259	4.235	4.856	4.421	8.992	**4.233**	4.235
		10E+4	4.259	4.235	4.856	4.421	8.992	**4.233**	4.235
		10E+6	4.259	4.235	4.856	4.421	8.992	**4.233**	4.235

Table 4.14: Upper Bounds from `pomdp1-4-2-2-2-3` and `pomdp2-2-2-2-2-3` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
pomdp3-4-4-2-2-3	1	10E+1	inf	inf	9.587	**6.902**	367.034	931.408	2.73E+04
n:27 f:30		10E+2	inf	inf	9.587	**6.902**	367.034	931.408	2.73E+04
k:2 s:4		10E+3	96.288	inf	8.065	**6.902**	81.228	931.408	2.73E+04
w:16		10E+4	**4.442**	201.746	8.065	6.902	77.243	931.408	2.73E+04
		10E+6	**4.442**	201.746	8.065	6.902	77.243	931.408	2.73E+04
	5	10E+1	inf	inf	inf	**5.442**	587.364	164.775	130.195
		10E+2	inf	inf	inf	**5.442**	587.364	164.775	130.195
		10E+3	**4.539**	25.951	5.526	5.442	48.26	164.775	130.195
		10E+4	**4.539**	25.951	5.526	5.442	38.355	164.775	130.195
		10E+6	**4.539**	25.951	5.526	5.442	36.594	164.775	130.195
	10	10E+1	inf	inf	inf	**4.515**	inf	56.606	15.455
		10E+2	inf	inf	inf	**4.515**	inf	56.606	15.455
		10E+3	**4.514**	9.378	4.881	4.515	57.804	56.606	15.455
		10E+4	**4.483**	9.378	4.834	4.515	40.983	56.606	15.455
		10E+6	**4.471**	9.378	4.834	4.515	36.704	56.606	15.455
	15	10E+1	inf	inf	inf	**3.929**	inf	18.195	8.783
		10E+2	inf	inf	inf	**3.929**	inf	18.195	8.783
		10E+3	4.524	5.834	5.871	**3.929**	106.26	18.195	8.783
		10E+4	4.496	5.834	5.421	**3.929**	36.818	18.195	8.783
		10E+6	4.472	5.834	5.421	**3.929**	23.896	18.195	8.783
	20	10E+1	inf	inf	inf	3.538	inf	11.406	**3.51**
		10E+2	inf	inf	inf	3.538	inf	11.406	**3.51**
		10E+3	3.626	**3.51**	6.077	3.538	65.155	11.406	**3.51**
		10E+4	3.626	**3.51**	5.527	3.538	36.347	11.406	**3.51**
		10E+6	3.626	**3.51**	5.527	3.538	33.403	11.406	**3.51**
pomdp4-4-4-2-2-5	1	10E+1	inf	inf	inf	**13.413**	8.61E+05	9.56E+05	2.24E+08
n:45 f:50		10E+2	inf	inf	inf	**13.413**	8.61E+05	9.56E+05	2.24E+08
k:2 s:4		10E+3	6295.509	inf	**12.528**	13.413	2473.81	9.56E+05	2.24E+08
w:26		10E+4	**6.329**	3.15E+04	12.37	13.413	2126.838	9.56E+05	2.24E+08
		10E+6	**6.329**	3.15E+04	12.37	13.413	2114.334	9.56E+05	2.24E+08
	5	10E+1	inf	inf	inf	**10.6**	inf	6.15E+04	1.46E+05
		10E+2	inf	inf	inf	**10.6**	inf	6.15E+04	1.46E+05
		10E+3	2722.204	inf	11.018	**10.6**	1436.118	6.15E+04	1.46E+05
		10E+4	**7.959**	1899.055	10.443	10.6	844.58	6.15E+04	1.46E+05
		10E+6	**7.959**	1899.055	10.443	10.6	720.228	6.15E+04	1.46E+05
	10	10E+1	inf	inf	inf	**8.288**	inf	2566.97	358.656
		10E+2	inf	inf	inf	**8.288**	inf	2566.97	358.656
		10E+3	200.966	54.167	10.876	**8.288**	1401.642	2566.97	358.656
		10E+4	8.676	54.167	9.357	**8.288**	524.148	2566.97	358.656
		10E+6	8.644	54.167	9.357	**8.288**	404.372	2566.97	358.656
	15	10E+1	inf	inf	inf	**6.778**	inf	276.211	48.725
		10E+2	inf	inf	inf	**6.778**	inf	276.211	48.725
		10E+3	141.056	inf	10.502	**6.778**	3962.652	276.211	48.725
		10E+4	10.034	14.655	9.022	**6.778**	708.826	276.211	48.725
		10E+6	9.872	14.655	9.022	**6.778**	458.13	276.211	48.725
	20	10E+1	inf	inf	inf	**6.164**	inf	104.431	22.784
		10E+2	inf	inf	inf	**6.164**	inf	104.431	22.784
		10E+3	inf	inf	12.576	**6.164**	inf	104.431	22.784
		10E+4	8.238	inf	9.846	**6.164**	4.96E+04	104.431	22.784
		10E+6	8.187	9.963	9.065	**6.164**	6041.114	104.431	22.784

Table 4.15: Upper Bounds from `pomdp3-4-4-2-2-3` and `pomdp4-4-4-2-2-5` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
pomdp5-6-4-3-5-3	1	10E+1	inf	inf	inf	**15.261**	1.22E+05	2.02E+05	2.36E+07
n:33 f:45		10E+2	inf	inf	inf	**15.261**	1.22E+05	2.02E+05	2.36E+07
k:3 s:5		10E+3	1314.349	inf	15.82	**15.261**	734.544	2.02E+05	2.36E+07
w:17		10E+4	**14.244**	2773.993	15.328	15.261	565.204	2.02E+05	2.36E+07
		10E+6	**14.244**	2773.993	15.328	15.261	528.172	2.02E+05	2.36E+07
	5	10E+1	inf	inf	inf	**13.517**	inf	2.50E+04	1.36E+04
		10E+2	inf	inf	inf	**13.517**	inf	2.50E+04	1.36E+04
		10E+3	4449.24	inf	14.174	**13.517**	988.707	2.50E+04	1.36E+04
		10E+4	21.901	507.057	**13.389**	13.517	481.527	2.50E+04	1.36E+04
		10E+6	21.901	507.057	**13.389**	13.517	400.441	2.50E+04	1.36E+04
	10	10E+1	inf	inf	inf	**11.287**	inf	1578.03	741.242
		10E+2	inf	inf	inf	**11.287**	inf	1578.03	741.242
		10E+3	239.586	62.828	12.683	**11.287**	1914.186	1578.03	741.242
		10E+4	24.112	62.828	12.164	**11.287**	511.874	1578.03	741.242
		10E+6	23.28	62.828	12.164	**11.287**	405.597	1578.03	741.242
	15	10E+1	inf	inf	inf	**9.763**	inf	118.961	23.852
		10E+2	inf	inf	inf	**9.763**	inf	118.961	23.852
		10E+3	19.966	15.059	14.143	**9.763**	6795.026	118.961	23.852
		10E+4	18.99	15.059	12.344	**9.763**	878.623	118.961	23.852
		10E+6	20.901	15.059	12.344	**9.763**	279.246	118.961	23.852
	20	10E+1	inf	inf	inf	9.116	inf	inf	**8.664**
		10E+2	inf	inf	inf	9.116	inf	inf	**8.664**
		10E+3	8.675	inf	16.971	9.116	2.05E+04	57.139	**8.664**
		10E+4	8.668	**8.664**	13.857	9.116	3094.258	57.139	**8.664**
		10E+6	8.668	**8.664**	12.947	9.116	977.258	57.139	**8.664**
pomdp6-12-6-2-6-3	1	10E+1	inf	inf	inf	**26.516**	inf	1.97E+08	6.47E+14
n:57 f:72		10E+2	inf	inf	inf	**26.516**	inf	1.97E+08	6.47E+14
k:2 s:9		10E+3	3.05E+06	inf	32.021	**26.516**	2.35E+05	1.97E+08	6.47E+14
w:28		10E+4	3.85E+05	3.77E+07	**24.891**	26.516	5.08E+04	1.97E+08	6.47E+14
		10E+6	1097.621	3.77E+07	**23.388**	26.516	4.12E+04	1.97E+08	6.47E+14
	5	10E+1	inf	inf	inf	**25.077**	inf	5.23E+07	2.70E+11
		10E+2	inf	inf	inf	**25.077**	inf	5.23E+07	2.70E+11
		10E+3	2.64E+08	inf	30.783	**25.077**	1.69E+06	5.23E+07	2.70E+11
		10E+4	9.55E+05	8.83E+06	**24.238**	25.077	1.13E+05	5.23E+07	2.70E+11
		10E+6	7.60E+05	8.83E+06	**24.238**	25.077	6.62E+04	5.23E+07	2.70E+11
	10	10E+1	inf	inf	inf	**19.937**	inf	1.31E+06	1.10E+08
		10E+2	inf	inf	inf	**19.937**	inf	1.31E+06	1.10E+08
		10E+3	2.25E+08	inf	30.096	**19.937**	1.54E+07	1.31E+06	1.10E+08
		10E+4	2.28E+06	inf	22.217	**19.937**	1.19E+05	1.31E+06	1.10E+08
		10E+6	8145.456	1.45E+05	22.217	**19.937**	3.33E+04	1.31E+06	1.10E+08
	15	10E+1	inf	inf	inf	**17.23**	inf	4.44E+04	1.46E+06
		10E+2	inf	inf	inf	**17.23**	inf	4.44E+04	1.46E+06
		10E+3	inf	inf	35.031	**17.23**	inf	4.44E+04	1.46E+06
		10E+4	6.68E+06	5844.706	28.544	**17.23**	2.68E+07	4.44E+04	1.46E+06
		10E+6	547.373	5844.706	22.598	**17.23**	4.68E+05	4.44E+04	1.46E+06
	20	10E+1	inf	inf	inf	inf	inf	inf	**3666.166**
		10E+2	inf	inf	inf	inf	inf	inf	3666.166
		10E+3	inf	inf	inf	**13.857**	inf	2832.55	3666.166
		10E+4	inf	inf	27.063	**13.857**	inf	2832.55	3666.166
		10E+6	inf	226.2	19.375	**13.857**	inf	2832.55	3666.166

Table 4.16: Upper Bounds from `pomdp5-6-4-3-5-3` and `pomdp6-12-6-2-6-3` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
pomdp7-20-10-2-10-3	1	10E+1	inf	inf	inf	**45.636**	inf	inf	1.61E+24
n:93 f:120		10E+2	inf	inf	inf	**45.636**	inf	inf	1.61E+24
k:2 s:9		10E+3	3.26E+10	inf	57.456	**45.636**	3.56E+10	inf	1.61E+24
w:45		10E+4	8.17E+08	9.74E+11	47.028	**45.636**	2.34E+08	1.91E+13	1.61E+24
		10E+6	1.18E+08	9.74E+11	**42.09**	45.636	1.05E+08	1.91E+13	1.61E+24
	5	10E+1	inf	inf	inf	**43.971**	inf	inf	4.96E+19
		10E+2	inf	inf	inf	**43.971**	inf	inf	4.96E+19
		10E+3	7.29E+13	inf	59.793	**43.971**	5.81E+11	inf	4.96E+19
		10E+4	4.29E+10	1.56E+11	47.158	**43.971**	5.50E+08	3.12E+12	4.96E+19
		10E+6	2.45E+09	1.56E+11	**41.83**	43.971	1.52E+08	3.12E+12	4.96E+19
	10	10E+1	inf	inf	inf	**38.825**	inf	inf	7.34E+16
		10E+2	inf	inf	inf	**38.825**	inf	inf	7.34E+16
		10E+3	2.17E+15	inf	61.293	**38.825**	1.26E+14	inf	7.34E+16
		10E+4	1.58E+12	5.90E+09	50.401	**38.825**	1.49E+09	inf	7.34E+16
		10E+6	1.40E+10	5.90E+09	40.564	**38.825**	6.32E+07	inf	7.34E+16
	15	10E+1	inf	inf	inf	**36.723**	inf	inf	8.56E+12
		10E+2	inf	inf	inf	**36.723**	inf	inf	8.56E+12
		10E+3	inf	inf	65.436	**36.723**	inf	inf	8.56E+12
		10E+4	6.95E+12	inf	59.407	**36.723**	2.32E+14	4.68E+09	8.56E+12
		10E+6	1.19E+10	9.94E+09	44.867	**36.723**	3.27E+09	4.68E+09	8.56E+12
	20	10E+1	inf	inf	inf	inf	inf	inf	inf
		10E+2	inf	inf	inf	inf	inf	inf	inf
		10E+3	inf	inf	inf	**32.811**	inf	inf	2.13E+10
		10E+4	inf	inf	inf	**32.811**	inf	2.40E+08	2.13E+10
		10E+6	inf	8.47E+06	inf	**32.811**	inf	2.40E+08	2.13E+10
pomdp8-14-9-3-12-4	1	10E+1	inf	inf	inf	**56.01**	inf	inf	1.94E+22
n:96 f:140		10E+2	inf	inf	inf	**56.01**	inf	inf	1.94E+22
k:3 s:6		10E+3	2.31E+10	inf	68.31	**56.01**	1.34E+10	4.19E+14	1.94E+22
w:47		10E+4	4.70E+09	2.32E+12	57.581	**56.01**	8.45E+07	4.19E+14	1.94E+22
		10E+6	4.50E+08	2.32E+12	57.138	**56.01**	5.09E+07	4.19E+14	1.94E+22
	5	10E+1	inf	inf	inf	**53.079**	inf	inf	2.58E+18
		10E+2	inf	inf	inf	**53.079**	inf	inf	2.58E+18
		10E+3	4.55E+14	inf	65.339	**53.079**	1.25E+12	3.10E+13	2.58E+18
		10E+4	1.99E+11	3.14E+11	54.43	**53.079**	2.36E+08	3.10E+13	2.58E+18
		10E+6	4.06E+09	3.14E+11	**52.865**	53.079	3.97E+07	3.10E+13	2.58E+18
	10	10E+1	inf	inf	inf	**47.86**	inf	inf	1.45E+15
		10E+2	inf	inf	inf	**47.86**	inf	inf	1.45E+15
		10E+3	6.30E+13	inf	67.988	**47.86**	1.19E+15	3.60E+11	1.45E+15
		10E+4	1.32E+11	2.23E+09	54.017	**47.86**	1.42E+10	3.60E+11	1.45E+15
		10E+6	1.95E+09	2.23E+09	50.76	**47.86**	4.11E+07	3.60E+11	1.45E+15
	15	10E+1	inf	inf	inf	**43.878**	inf	inf	7.40E+12
		10E+2	inf	inf	inf	**43.878**	inf	inf	7.40E+12
		10E+3	inf	inf	76.321	**43.878**	inf	inf	7.40E+12
		10E+4	7.49E+12	inf	68.17	**43.878**	5.82E+16	inf	7.40E+12
		10E+6	5.83E+09	6.72E+07	52.082	**43.878**	7.43E+14	inf	7.40E+12
	20	10E+1	inf	inf	inf	inf	inf	inf	inf
		10E+2	inf	inf	inf	inf	inf	inf	inf
		10E+3	inf	inf	inf	**39.958**	inf	inf	1.44E+10
		10E+4	inf	inf	inf	**39.958**	inf	inf	1.44E+10
		10E+6	inf	3.62E+06	inf	**39.958**	inf	inf	1.44E+10

Table 4.17: Upper Bounds from `pomdp7-20-10-2-10-3` and `pomdp8-14-9-3-12-4` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
pomdp9-14-8-3-10-4	1	10E+1	inf	inf	inf	**49.939**	inf	inf	1.15E+28
n:92 f:128		10E+2	inf	inf	inf	**49.939**	inf	inf	1.15E+28
k:3 s:6		10E+3	1.32E+10	inf	59.792	**49.939**	3.51E+10	inf	1.15E+28
w:43		10E+4	7.96E+08	6.41E+11	**46.329**	49.939	1.78E+09	inf	1.15E+28
		10E+6	1.95E+08	6.41E+11	**45.994**	49.939	1.11E+09	inf	1.15E+28
	5	10E+1	inf	inf	inf	**46.207**	inf	inf	1.83E+18
		10E+2	inf	inf	inf	**46.207**	inf	inf	1.83E+18
		10E+3	1.14E+14	inf	61.176	**46.207**	1.91E+12	inf	1.83E+18
		10E+4	6.64E+10	1.90E+10	46.831	**46.207**	2.98E+09	inf	1.83E+18
		10E+6	2.54E+09	1.90E+10	**44.14**	46.207	9.86E+08	inf	1.83E+18
	10	10E+1	inf	inf	inf	**42.85**	inf	inf	1.41E+16
		10E+2	inf	inf	inf	**42.85**	inf	inf	1.41E+16
		10E+3	3.14E+15	inf	70.958	**42.85**	2.06E+16	inf	1.41E+16
		10E+4	8.16E+12	4.47E+08	56.431	**42.85**	1.36E+13	inf	1.41E+16
		10E+6	1.01E+11	4.47E+08	45.098	**42.85**	3.09E+09	inf	1.41E+16
	15	10E+1	inf	inf	inf	inf	inf	inf	inf
		10E+2	inf	inf	inf	inf	inf	inf	inf
		10E+3	inf	inf	inf	**39.545**	inf	inf	9.07E+11
		10E+4	inf	inf	73.755	**39.545**	inf	inf	9.07E+11
		10E+6	7.17E+10	9.54E+06	46.823	**39.545**	inf	inf	9.07E+11
	20	10E+1	inf	inf	inf	inf	inf	inf	inf
		10E+2	inf	inf	inf	inf	inf	inf	inf
		10E+3	inf	inf	inf	inf	inf	inf	inf
		10E+4	inf	inf	inf	**34.138**	inf	inf	inf
		10E+6	inf	inf	inf	**34.138**	inf	inf	inf
pomdp10-12-7-3-8-4	1	10E+1	inf	inf	inf	**38.242**	inf	inf	1.22E+18
n:80 f:108		10E+2	inf	inf	inf	**38.242**	inf	inf	1.22E+18
k:3 s:5		10E+3	2.75E+07	inf	38.535	**38.242**	1.33E+07	inf	1.22E+18
w:38		10E+4	2.17E+06	2.74E+09	**35.348**	38.242	3.83E+06	2.24E+11	1.22E+18
		10E+6	1.46E+06	2.74E+09	**35.348**	38.242	3.30E+06	2.24E+11	1.22E+18
	5	10E+1	inf	inf	inf	**33.62**	inf	inf	4.68E+12
		10E+2	inf	inf	inf	**33.62**	inf	inf	4.68E+12
		10E+3	1.24E+09	inf	38.703	**33.62**	5.31E+07	inf	4.68E+12
		10E+4	2.65E+06	2.98E+07	34.616	**33.62**	1.09E+06	8.46E+09	4.68E+12
		10E+6	2.56E+06	2.98E+07	34.616	**33.62**	5.48E+05	8.46E+09	4.68E+12
	10	10E+1	inf	inf	inf	**29.464**	inf	inf	1.69E+10
		10E+2	inf	inf	inf	**29.464**	inf	inf	1.69E+10
		10E+3	4.23E+09	inf	38.497	**29.464**	8.26E+11	inf	1.69E+10
		10E+4	9.69E+06	5.53E+05	31.889	**29.464**	5.05E+08	inf	1.69E+10
		10E+6	9.32E+06	5.53E+05	31.889	**29.464**	6.92E+06	inf	1.69E+10
	15	10E+1	inf	inf	inf	inf	inf	inf	**1.58E+07**
		10E+2	inf	inf	inf	inf	inf	inf	1.58E+07
		10E+3	inf	inf	inf	**26.152**	inf	inf	1.58E+07
		10E+4	1.69E+08	inf	48.273	**26.152**	inf	inf	1.58E+07
		10E+6	4.66E+05	3.74E+04	34.503	**26.152**	3.10E+11	inf	1.58E+07
	20	10E+1	inf	inf	inf	inf	inf	inf	inf
		10E+2	inf	inf	inf	inf	inf	inf	inf
		10E+3	inf	inf	inf	**23.726**	inf	inf	2.74E+05
		10E+4	inf	inf	inf	**23.726**	inf	inf	2.74E+05
		10E+6	inf	inf	inf	**23.726**	inf	inf	2.74E+05

Table 4.18: Upper Bounds from `pomdp9-14-8-3-10-4` and `pomdp10-12-7-3-8-4` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

4.3.3 RAND Domain

In this section, we summarize the upper bounds at instances in `RAND` domain from Table 4.19 to Table 4.23. Overall, we see that WMBMM-EXP is the best performing algorithm given i-bound larger than 10, or given time bounds shorter than 1,000 seconds. JGD-EXP often generated competitive upper bounds. For example, both JGD-EXP and WBMMM-EXP generated upper bounds that are close to each other at `rand-c50d10o1-01` instance and `rand-c50d15o1-03` over all i-bounds and time bounds longer than 1,000 seconds, as shown in Table 4.22. We observe a common trend that algorithm JGD-EXP generates tighter upper bounds than algorithm WMBMM-EXP when i-bound is 1 or 5 at time bounds longer than 1,000 seconds. On the other hand, algorithm WMBMM-EXP generates tighter upper bounds when i-bound is greater than 10, or given time bound is shorter than 1,000 seconds. In contrast with the previous benchmarks of `FH-MDP` and `FH-POMDP`, JGID-ID and WBME-ID were not competitive except for the instances having low induced-width, (w = 6, 10, 13), which are close to the input i-bounds. We see that the translation based approaches and MBE-ID were completely incompetitive, yielding orders of magnitude worse bounds.

Note that the first six instances presented in Table 4.19 to Table 4.21 are easy problem instances having the constrained induced width less than or equal to 20. In those easy instances, algorithms WMBE-ID and MBE-ID that are based on the mini-bucket tree decomposition generated the exact MEU if they were given i-bounds greater than the constrained induced width. The translation based approach WMBMM-MMAP generated the exact MEU if the constrained induced width of the translated BN is still less than i-bounds. However, JGD-ID and GDD-MI often failed to generate the exact MEU when they are given i-bounds larger than 10. JGD-EXP and WMBMM-EXP couldn't generate the exact MEU on easy problem instances because they apply two bounding inequalities, the exponenitated utility bounds and the decomposition bounds, in stages.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
rand-c20d2o1-01	1	10E+1	675.722	inf	172.68	**171.484**	446.857	1521.32	2546.296
n:22 f:22		10E+2	675.722	inf	172.68	171.484	446.857	1521.32	2546.296
k:2 s:3		10E+3	**166.846**	429.578	171.326	171.484	235.868	1521.32	2546.296
w:6		10E+4	**165.463**	429.578	171.326	171.484	234.028	1521.32	2546.296
		10E+6	**165.1**	429.578	171.326	171.484	234.028	1521.32	2546.296
	5	10E+1	135.356	inf	inf	170.632	869.981	289.132	**112.748**
		10E+2	135.356	inf	inf	170.632	869.981	289.132	**112.748**
		10E+3	124.57	112.765	170.728	170.632	143.159	289.132	**112.748**
		10E+4	124.548	112.765	170.728	170.632	129.196	289.132	**112.748**
		10E+6	124.531	112.765	170.728	170.632	129.196	289.132	**112.748**
	10	10E+1	137.169	inf	173.609	170.632	1041.224	**107.764**	112.665
		10E+2	137.169	inf	173.609	170.632	1041.224	**107.764**	112.665
		10E+3	116.724	112.665	170.724	170.632	138.607	**107.764**	112.665
		10E+4	116.722	112.665	170.724	170.632	128.137	**107.764**	112.665
		10E+6	116.721	112.665	170.724	170.632	128.137	**107.764**	112.665
	15	10E+1	137.169	inf	173.609	170.632	1041.224	**107.764**	112.665
		10E+2	137.169	inf	173.609	170.632	1041.224	**107.764**	112.665
		10E+3	116.723	112.665	170.724	170.632	138.312	**107.764**	112.665
		10E+4	116.722	112.665	170.724	170.632	147.213	**107.764**	112.665
		10E+6	120.217	112.665	170.724	170.632	147.213	**107.764**	112.665
	20	10E+1	137.122	inf	inf	170.632	2025.2	**107.764**	112.665
		10E+2	137.122	inf	inf	170.632	2025.2	**107.764**	112.665
		10E+3	120.217	112.665	170.726	170.632	147.213	**107.764**	112.665
		10E+4	120.217	112.665	170.726	170.632	147.213	**107.764**	112.665
		10E+6	120.217	112.665	170.726	170.632	147.213	**107.764**	112.665
rand-c30d3o1-01	1	10E+1	inf	inf	**276.487**	279.111	4.24E+04	1.02E+04	1.40E+05
n:33 f:33		10E+2	inf	inf	276.487	279.111	4.24E+04	1.02E+04	1.40E+05
k:2 s:3		10E+3	1929.77	2433.593	**274.262**	279.111	1242.861	1.02E+04	1.40E+05
w:10		10E+4	276.001	2433.593	**274.262**	279.111	1217.778	1.02E+04	1.40E+05
		10E+6	276.001	2433.593	**274.262**	279.111	1217.778	1.02E+04	1.40E+05
	5	10E+1	inf	inf	inf	**273.382**	9.51E+05	1359.92	3073.149
		10E+2	inf	inf	inf	**273.382**	9.51E+05	1359.92	3073.149
		10E+3	343.794	667.032	273.671	**273.382**	577.872	1359.92	3073.149
		10E+4	305.353	667.032	273.523	**273.382**	452.904	1359.92	3073.149
		10E+6	299.716	667.032	273.523	**273.382**	438.399	1359.92	3073.149
	10	10E+1	inf	inf	inf	272.711	inf	417.181	**222.005**
		10E+2	inf	inf	inf	272.711	inf	417.181	**222.005**
		10E+3	226.178	**222.005**	280.544	272.711	597.673	417.181	**222.005**
		10E+4	224.837	**222.005**	280.302	272.711	347.633	417.181	**222.005**
		10E+6	224.837	**222.005**	280.302	272.711	336.834	417.181	**222.005**
	15	10E+1	inf	inf	inf	272.711	inf	222.006	**222.005**
		10E+2	inf	inf	inf	272.711	inf	222.006	**222.005**
		10E+3	226.178	**222.005**	280.76	272.711	1116.989	222.006	**222.005**
		10E+4	224.869	**222.005**	280.564	272.711	349.126	222.006	**222.005**
		10E+6	224.869	**222.005**	280.564	272.711	441.774	222.006	**222.005**
	20	10E+1	inf	inf	inf	272.711	inf	222.006	**222.005**
		10E+2	inf	inf	inf	272.711	inf	222.006	**222.005**
		10E+3	224.925	**222.005**	280.599	272.711	2069.952	222.006	**222.005**
		10E+4	224.869	**222.005**	280.302	272.711	440.361	222.006	**222.005**
		10E+6	224.869	**222.005**	280.302	272.711	440.361	222.006	**222.005**

Table 4.19: Upper Bounds from `rand-c20d2o1-01` and `rand-c30d3o1-01` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
rand-c30d6o1-01	1	10E+1	inf	inf	inf	**536.596**	1.81E+06	1.90E+05	7.29E+05
n:36 f:36		10E+2	inf	inf	inf	536.596	1.81E+06	1.90E+05	7.29E+05
k:2 s:3		10E+3	3521.833	5165.121	**535.743**	536.596	2258.084	1.90E+05	7.29E+05
w:13		10E+4	552.87	5165.121	**535.743**	536.596	1773.476	1.90E+05	7.29E+05
		10E+6	552.415	5165.121	**535.743**	536.596	1773.476	1.90E+05	7.29E+05
	5	10E+1	inf	inf	inf	**535.437**	1.77E+13	1.76E+04	3850.815
		10E+2	inf	inf	inf	**535.437**	1.77E+13	1.76E+04	3850.815
		10E+3	648.628	773.618	536.524	**535.437**	3.80E+07	1.76E+04	3850.815
		10E+4	624.624	773.618	536.431	**535.437**	5.54E+06	1.76E+04	3850.815
		10E+6	622.095	773.618	536.431	**535.437**	3.18E+06	1.76E+04	3850.815
	10	10E+1	inf	inf	inf	**535.113**	inf	2163.31	805.127
		10E+2	inf	inf	inf	**535.113**	inf	2163.31	805.127
		10E+3	562.568	**433.766**	540.861	535.113	5.19E+08	2163.31	805.127
		10E+4	510.224	**433.766**	540.462	535.113	2.39E+07	2163.31	805.127
		10E+6	465.052	**433.766**	540.462	535.113	1.61E+07	2163.31	805.127
	15	10E+1	inf	inf	inf	534.979	inf	783.184	**377.614**
		10E+2	inf	inf	inf	534.979	inf	783.184	**377.614**
		10E+3	441.265	**377.614**	542.671	534.979	9.66E+09	783.184	**377.614**
		10E+4	418.715	**377.614**	541.969	534.979	2.11E+09	783.184	**377.614**
		10E+6	398.917	**377.614**	541.969	534.979	2.33E+09	783.184	**377.614**
	20	10E+1	inf	inf	inf	534.979	inf	**377.085**	377.614
		10E+2	inf	inf	inf	534.979	inf	**377.085**	377.614
		10E+3	412.851	377.614	542.713	534.979	1.43E+11	**377.085**	377.614
		10E+4	398.917	377.614	541.969	534.979	2.48E+09	**377.085**	377.614
		10E+6	398.917	377.614	541.969	534.979	2.33E+09	**377.085**	377.614
rand-c30d9o1-01	1	10E+1	inf	inf	inf	**742.014**	1.69E+08	5.50E+06	1.84E+06
n:39 f:39		10E+2	inf	inf	inf	**742.014**	1.69E+08	5.50E+06	1.84E+06
k:2 s:3		10E+3	7528.335	8798.054	742.287	**742.014**	1.56E+04	5.50E+06	1.84E+06
w:16		10E+4	766.687	8798.054	742.287	**742.014**	1.08E+04	5.50E+06	1.84E+06
		10E+6	766.687	8798.054	742.287	**742.014**	1.08E+04	5.50E+06	1.84E+06
	5	10E+1	inf	inf	inf	**739.365**	6.73E+17	2.05E+05	2.23E+04
		10E+2	inf	inf	inf	**739.365**	6.73E+17	2.05E+05	2.23E+04
		10E+3	806.441	2979.69	781.955	**739.365**	2.70E+10	2.05E+05	2.23E+04
		10E+4	793.615	2979.69	781.629	**739.365**	8.11E+06	2.05E+05	2.23E+04
		10E+6	793.615	2979.69	781.629	**739.365**	2.23E+06	2.05E+05	2.23E+04
	10	10E+1	inf	inf	inf	**732.835**	inf	8986.47	1327.517
		10E+2	inf	inf	inf	**732.835**	inf	8986.47	1327.517
		10E+3	1156.275	893.895	795.067	**732.835**	1.16E+11	8986.47	1327.517
		10E+4	830.039	893.895	794.406	**732.835**	1.25E+09	8986.47	1327.517
		10E+6	820.118	893.895	794.406	**732.835**	1.10E+09	8986.47	1327.517
	15	10E+1	inf	inf	inf	**730.436**	inf	2320.76	1409.596
		10E+2	inf	inf	inf	**730.436**	inf	2320.76	1409.596
		10E+3	960.234	790.751	794.757	**730.436**	1.60E+13	2320.76	1409.596
		10E+4	914.462	790.751	793.797	**730.436**	6.25E+09	2320.76	1409.596
		10E+6	763.819	790.751	793.797	**730.436**	6.22E+08	2320.76	1409.596
	20	10E+1	inf	inf	inf	729.753	inf	1189.79	**594.803**
		10E+2	inf	inf	inf	729.753	inf	1189.79	**594.803**
		10E+3	635.236	**594.803**	798.569	729.753	5.02E+13	1189.79	**594.803**
		10E+4	616.856	**594.803**	793.726	729.753	3.83E+12	1189.79	**594.803**
		10E+6	616.856	**594.803**	793.674	729.753	9.58E+11	1189.79	**594.803**

Table 4.20: Upper Bounds from `rand-c30d6o1-01` and `rand-c30d9o1-01` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
rand-c50d5o1-01	1	10E+1	inf	inf	inf	**433.91**	1.81E+12	7.90E+05	8.14E+07
n:55 f:55		10E+2	inf	inf	inf	433.91	1.81E+12	7.90E+05	8.14E+07
k:2 s:3		10E+3	9186.728	2.23E+04	**430.007**	433.91	5546.015	7.90E+05	8.14E+07
w:17		10E+4	450.84	2.23E+04	**429.997**	433.91	5220.345	7.90E+05	8.14E+07
		10E+6	450.84	2.23E+04	**429.997**	433.91	5215.389	7.90E+05	8.14E+07
	5	10E+1	inf	inf	inf	**430.406**	inf	2.55E+04	2.33E+04
		10E+2	inf	inf	inf	**430.406**	inf	2.55E+04	2.33E+04
		10E+3	1881.42	974.431	430.438	**430.406**	4.37E+06	2.55E+04	2.33E+04
		10E+4	532.138	974.431	**429.947**	430.406	7654.414	2.55E+04	2.33E+04
		10E+6	529.672	974.431	**429.947**	430.406	1941.346	2.55E+04	2.33E+04
	10	10E+1	inf	inf	inf	**429.001**	inf	1556.72	2340.09
		10E+2	inf	inf	inf	**429.001**	inf	1556.72	2340.09
		10E+3	686.227	inf	432.641	**429.001**	7.59E+06	1556.72	2340.09
		10E+4	467.486	**410.307**	431.306	429.001	1.23E+05	1556.72	2340.09
		10E+6	451.151	**410.307**	431.306	429.001	1.65E+04	1556.72	2340.09
	15	10E+1	inf	inf	inf	**428.851**	inf	692.522	645.601
		10E+2	inf	inf	inf	**428.851**	inf	692.522	645.601
		10E+3	517.937	**298.621**	434.683	428.851	6.85E+10	692.522	645.601
		10E+4	446.34	**298.621**	431.329	428.851	1.76E+08	692.522	645.601
		10E+6	416.142	**298.621**	431.329	428.851	2.67E+07	692.522	645.601
	20	10E+1	inf	inf	inf	428.375	inf	**272.002**	274.844
		10E+2	inf	inf	inf	428.375	inf	**272.002**	274.844
		10E+3	331.212	274.844	438.026	428.375	2.78E+11	**272.002**	274.844
		10E+4	289.712	274.844	432.042	428.375	5.21E+10	**272.002**	274.844
		10E+6	288.835	274.844	431.697	428.375	4.14E+09	**272.002**	274.844
rand-c70d7o1-01	1	10E+1	inf	inf	inf	**661.574**	2.51E+17	1.38E+07	2.69E+09
n:77 f:77		10E+2	inf	inf	inf	**661.574**	2.51E+17	1.38E+07	2.69E+09
k:2 s:3		10E+3	4.63E+04	3.70E+06	**660.847**	661.574	3.55E+04	1.38E+07	2.69E+09
w:20		10E+4	682.828	3.70E+06	**660.633**	661.574	2.42E+04	1.38E+07	2.69E+09
		10E+6	682.828	3.70E+06	**660.633**	661.574	2.39E+04	1.38E+07	2.69E+09
	5	10E+1	inf	inf	inf	**658.306**	inf	1.58E+05	4.46E+05
		10E+2	inf	inf	inf	**658.306**	inf	1.58E+05	4.46E+05
		10E+3	1.00E+04	2567.806	658.94	**658.306**	7.43E+09	1.58E+05	4.46E+05
		10E+4	725.743	2567.806	**657.36**	658.306	1.80E+08	1.58E+05	4.46E+05
		10E+6	722.234	2567.806	**657.36**	658.306	1.20E+07	1.58E+05	4.46E+05
	10	10E+1	inf	inf	inf	**657.373**	inf	1.09E+04	4937.762
		10E+2	inf	inf	inf	**657.373**	inf	1.09E+04	4937.762
		10E+3	2400.926	inf	662.015	**657.373**	7.61E+10	1.09E+04	4937.762
		10E+4	807.648	1105.79	658.246	**657.373**	1.10E+09	1.09E+04	4937.762
		10E+6	797.011	1105.79	658.246	**657.373**	5.51E+07	1.09E+04	4937.762
	15	10E+1	inf	inf	inf	**656.94**	inf	4980.38	7027.926
		10E+2	inf	inf	inf	**656.94**	inf	4980.38	7027.926
		10E+3	3580.138	inf	663.208	**656.94**	8.49E+14	4980.38	7027.926
		10E+4	1622.918	1078.383	657.773	**656.94**	3.72E+11	4980.38	7027.926
		10E+6	784.91	1078.383	657.773	**656.94**	4.40E+08	4980.38	7027.926
	20	10E+1	inf	inf	inf	655.954	inf	1290.27	**499.672**
		10E+2	inf	inf	inf	655.954	inf	1290.27	**499.672**
		10E+3	inf	inf	inf	655.954	inf	1290.27	**499.672**
		10E+4	inf	**499.672**	664.757	655.954	3.84E+15	1290.27	**499.672**
		10E+6	684.288	**499.672**	658.631	655.954	2.91E+13	1290.27	**499.672**

Table 4.21: Upper Bounds from `rand-c50d5o1-01` and `rand-c70d7o1-01` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
rand-c50d10o1-01	1	10E+1	inf	inf	inf	**869.844**	9.90E+21	6.06E+07	4.19E+08
n:60 f:60		10E+2	inf	inf	inf	**869.844**	9.90E+21	6.06E+07	4.19E+08
k:2 s:3		10E+3	4.27E+04	1.47E+05	871.617	**869.844**	7.59E+04	6.06E+07	4.19E+08
w:23		10E+4	1024.213	1.47E+05	871.608	**869.844**	4.03E+04	6.06E+07	4.19E+08
		10E+6	1024.213	1.47E+05	871.608	**869.844**	3.50E+04	6.06E+07	4.19E+08
	5	10E+1	inf	inf	inf	**866.954**	inf	1.45E+06	1.67E+05
		10E+2	inf	inf	inf	**866.954**	inf	1.45E+06	1.67E+05
		10E+3	1.26E+04	1.87E+04	**864.796**	866.954	1.23E+14	1.45E+06	1.67E+05
		10E+4	1162.645	1.87E+04	**863.936**	866.954	4.20E+10	1.45E+06	1.67E+05
		10E+6	1161.323	1.87E+04	**863.936**	866.954	1.49E+09	1.45E+06	1.67E+05
	10	10E+1	inf	inf	inf	**862.789**	inf	5.53E+04	4.09E+04
		10E+2	inf	inf	inf	**862.789**	inf	5.53E+04	4.09E+04
		10E+3	6771.519	2941.724	874.981	**862.789**	3.00E+15	5.53E+04	4.09E+04
		10E+4	1226.88	2941.724	870.313	**862.789**	1.00E+14	5.53E+04	4.09E+04
		10E+6	1205.696	2941.724	870.313	**862.789**	1.06E+13	5.53E+04	4.09E+04
	15	10E+1	inf	inf	inf	**862.416**	inf	8083.62	3995.708
		10E+2	inf	inf	inf	**862.416**	inf	8083.62	3995.708
		10E+3	5494.297	inf	875.264	**862.416**	1.19E+17	8083.62	3995.708
		10E+4	1313.536	1427.498	869.415	**862.416**	2.38E+12	8083.62	3995.708
		10E+6	1282.425	1427.498	869.415	**862.416**	8.63E+14	8083.62	3995.708
	20	10E+1	inf	inf	inf	**862.106**	inf	4638.09	7181.774
		10E+2	inf	inf	inf	**862.106**	inf	4638.09	7181.774
		10E+3	inf	inf	879.615	**862.106**	inf	4638.09	7181.774
		10E+4	inf	1000.242	875.892	**862.106**	9.52E+19	4638.09	7181.774
		10E+6	1701.032	1000.242	870.946	**862.106**	2.56E+18	4638.09	7181.774
rand-c50d15o1-03	1	10E+1	inf	inf	inf	**1247.159**	1.40E+31	2.60E+10	2.59E+08
n:65 f:65		10E+2	inf	inf	inf	**1247.159**	1.40E+31	2.60E+10	2.59E+08
k:2 s:3		10E+3	5.38E+04	3.51E+05	**1243.126**	1247.159	1.91E+05	2.60E+10	2.59E+08
w:27		10E+4	1314.326	3.51E+05	**1243.103**	1247.159	5.08E+04	2.60E+10	2.59E+08
		10E+6	1314.326	3.51E+05	**1243.103**	1247.159	4.81E+04	2.60E+10	2.59E+08
	5	10E+1	inf	inf	inf	**1242.412**	inf	1.65E+08	3.78E+05
		10E+2	inf	inf	inf	**1242.412**	inf	1.65E+08	3.78E+05
		10E+3	1.75E+04	9520.664	1254.971	**1242.412**	1.47E+12	1.65E+08	3.78E+05
		10E+4	1641.52	9520.664	1252.041	**1242.412**	2.27E+07	1.65E+08	3.78E+05
		10E+6	1638.078	9520.664	1252.041	**1242.412**	1.03E+06	1.65E+08	3.78E+05
	10	10E+1	inf	inf	inf	**1240.688**	inf	1.36E+06	1.03E+04
		10E+2	inf	inf	inf	**1240.688**	inf	1.36E+06	1.03E+04
		10E+3	4650.63	2636.03	1271.436	**1240.688**	3.02E+15	1.36E+06	1.03E+04
		10E+4	1749.403	2636.03	1265.567	**1240.688**	2.67E+10	1.36E+06	1.03E+04
		10E+6	1729.083	2636.03	1265.567	**1240.688**	3.77E+08	1.36E+06	1.03E+04
	15	10E+1	inf	inf	inf	**1240.61**	inf	9.92E+04	2443.418
		10E+2	inf	inf	inf	**1240.61**	inf	9.92E+04	2443.418
		10E+3	1979.445	inf	1275.254	**1240.61**	2.56E+24	9.92E+04	2443.418
		10E+4	1536.418	1388.087	1269.382	**1240.61**	9.93E+17	9.92E+04	2443.418
		10E+6	1494.398	1388.087	1269.382	**1240.61**	5.55E+14	9.92E+04	2443.418
	20	10E+1	inf	inf	inf	**1240.386**	inf	3.82E+04	1470.648
		10E+2	inf	inf	inf	**1240.386**	inf	3.82E+04	1470.648
		10E+3	inf	inf	1280.266	**1240.386**	inf	3.82E+04	1470.648
		10E+4	1545.091	**1156.419**	1274.473	1240.386	inf	3.82E+04	1470.648
		10E+6	1507.513	**1156.419**	1270.888	1240.386	inf	3.82E+04	1470.648

Table 4.22: Upper Bounds from `rand-c50d10o1-01` and `rand-c50d15o1-03` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
rand-c70d14o1-01	1	10E+1	inf	inf	inf	**1198.294**	1.14E+31	1.74E+10	4.71E+10
n:84 f:84		10E+2	inf	inf	inf	**1198.294**	1.14E+31	1.74E+10	4.71E+10
k:2 s:3		10E+3	1.12E+05	inf	1202.883	**1198.294**	2.28E+05	1.74E+10	4.71E+10
w:31		10E+4	1262.402	4.88E+05	1202.357	**1198.294**	1.40E+05	1.74E+10	4.71E+10
		10E+6	1262.402	4.88E+05	1202.357	**1198.294**	1.33E+05	1.74E+10	4.71E+10
	5	10E+1	inf	inf	inf	**1193.495**	inf	1.15E+08	6.39E+06
		10E+2	inf	inf	inf	**1193.495**	inf	1.15E+08	6.39E+06
		10E+3	7.33E+04	1.59E+04	1209.691	**1193.495**	3.08E+17	1.15E+08	6.39E+06
		10E+4	1627.243	1.59E+04	1207.103	**1193.495**	1.15E+15	1.15E+08	6.39E+06
		10E+6	1621.947	1.59E+04	1207.103	**1193.495**	2.10E+14	1.15E+08	6.39E+06
	10	10E+1	inf	inf	inf	**1192.697**	inf	9.59E+05	5.27E+04
		10E+2	inf	inf	inf	**1192.697**	inf	9.59E+05	5.27E+04
		10E+3	3.83E+04	inf	1210.847	**1192.697**	1.39E+21	9.59E+05	5.27E+04
		10E+4	1639.658	4306.811	1205.152	**1192.697**	9.10E+14	9.59E+05	5.27E+04
		10E+6	1636.008	4306.811	1205.152	**1192.697**	2.55E+12	9.59E+05	5.27E+04
	15	10E+1	inf	inf	inf	**1187.204**	inf	2.03E+05	8.17E+04
		10E+2	inf	inf	inf	**1187.204**	inf	2.03E+05	8.17E+04
		10E+3	4.98E+04	inf	1221.018	**1187.204**	1.49E+28	2.03E+05	8.17E+04
		10E+4	3903.284	1.76E+04	1209.388	**1187.204**	1.25E+20	2.03E+05	8.17E+04
		10E+6	1796.338	1.76E+04	1208.337	**1187.204**	3.07E+16	2.03E+05	8.17E+04
	20	10E+1	inf	inf	inf	inf	inf	inf	**1.24E+04**
		10E+2	inf	inf	inf	inf	inf	inf	1.24E+04
		10E+3	inf	inf	inf	**1191.487**	inf	2.23E+05	1.24E+04
		10E+4	inf	inf	1221.307	**1191.487**	inf	2.23E+05	1.24E+04
		10E+6	inf	2316.468	1207.994	**1191.487**	inf	2.23E+05	1.24E+04
rand-c70d21o1-01	1	10E+1	inf	inf	inf	**1755.734**	inf	3.20E+12	2.08E+10
n:91 f:91		10E+2	inf	inf	inf	**1755.734**	inf	3.20E+12	2.08E+10
k:2 s:3		10E+3	2.15E+05	1.34E+06	**1744.182**	1755.734	9.05E+05	3.20E+12	2.08E+10
w:41		10E+4	2081.899	1.34E+06	**1743.842**	1755.734	1.20E+05	3.20E+12	2.08E+10
		10E+6	2081.899	1.34E+06	**1743.842**	1755.734	1.08E+05	3.20E+12	2.08E+10
	5	10E+1	inf	inf	inf	**1745.283**	inf	7.74E+09	6.33E+06
		10E+2	inf	inf	inf	**1745.283**	inf	7.74E+09	6.33E+06
		10E+3	1.02E+05	4.01E+04	1782.238	**1745.283**	4.22E+25	7.74E+09	6.33E+06
		10E+4	2208.186	4.01E+04	1769.453	**1745.283**	1.06E+18	7.74E+09	6.33E+06
		10E+6	2208.186	4.01E+04	1769.453	**1745.283**	9.76E+15	7.74E+09	6.33E+06
	10	10E+1	inf	inf	inf	**1743.724**	inf	1.24E+07	7.77E+05
		10E+2	inf	inf	inf	**1743.724**	inf	1.24E+07	7.77E+05
		10E+3	5.12E+04	8270.178	1788.697	**1743.724**	7.11E+28	1.24E+07	7.77E+05
		10E+4	2560.783	8270.178	1783.18	**1743.724**	1.37E+19	1.24E+07	7.77E+05
		10E+6	2552.807	8270.178	1783.18	**1743.724**	7.73E+14	1.24E+07	7.77E+05
	15	10E+1	inf	inf	inf	**1734.844**	inf	7.87E+05	4.83E+04
		10E+2	inf	inf	inf	**1734.844**	inf	7.87E+05	4.83E+04
		10E+3	3.07E+04	inf	1801.88	**1734.844**	1.59E+41	7.87E+05	4.83E+04
		10E+4	6295.232	5264.345	1790.876	**1734.844**	3.16E+31	7.87E+05	4.83E+04
		10E+6	2717.454	5264.345	1790.621	**1734.844**	7.78E+21	7.87E+05	4.83E+04
	20	10E+1	inf	inf	inf	**1741.385**	inf	inf	1.89E+04
		10E+2	inf	inf	inf	**1741.385**	inf	inf	1.89E+04
		10E+3	inf	inf	1807.97	**1741.385**	inf	2.04E+05	1.89E+04
		10E+4	inf	inf	1802.177	**1741.385**	inf	2.04E+05	1.89E+04
		10E+6	5637.314	2712.705	1791.691	**1741.385**	inf	2.04E+05	1.89E+04

Table 4.23: Upper Bounds from `rand-c70d14o1-01` and `rand-c70d21o1-01` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

4.3.4 BN Domain

In this section, we summarize the upper bounds at instances in BN domain from Table 4.24 to Table 4.28. As in the previous benchmark domains, we can observe that the earlier approaches generated orders of magnitude worse upper bounds than directed decomposition bounds. Comparing the bounding schemes based on the valuation algebra and the bounding schemes based on the exponentiated utility bounds, we observe that both approaches generated comparable upper bounds, yet algorithms JGD-EXP and WMBMM-EXP generated overall tighter upper bounds at all problem instances. Problem instance BN-14-w42d6 shown in Table 4.28 has the largest constrained induced width 41, which is the second largest in BN domain. At BN-14-w42d6 instance, we observe that algorithm JGD-ID generated very loose upper bound (21,400 with i-bound 1) at a shorter time bound 1,000 seconds and it improved the upper bound (50.343 with i-bound 1) at the next time bound 10,000 seconds. On the other hand, JGD-EXP generated a relatively tighter upper bound (53.131 with i-bound 1) at 1,000 second time bound. We can observe similar trend that algorithm JGD-EXP generates higher quality upper bounds an order of magnitude faster than JGD-ID.

Overall, we see that direct decomposition schemes generated upper bounds that are orders of magnitude tighter than earlier algorithms. Comparing two iterative algorithms JGD-ID and JGD-EXP, both generate high quality upper bounds at all instances, yet the upper bounds from JGD-EXP is tighter than JGD-ID. In addition, JGD-EXP can generate upper bounds an order of mangnitude faster than JGD-ID. Algorithm WMBE-ID also generated relatively good upper bounds, but it was dominated by other direct bounding schemes in most of the cases. As in the previous benchmarks, algorithm JGD-EXP generated the best upper bounds when it was given i-bound 1 or 5, and given time bounds longer than 1,000 seconds. Algorithm WMBMM-EXP starts to dominate algorithm JGD-EXP when they are given shorter time bounds less than 1,000 seconds, or WMBMM-EXP was given higher i-bounds larger than 10.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
BN-78-w18d3	1	10E+1	inf	inf	inf	**25.885**	8052.847	383.224	3.05E+05
n:54 f:54		10E+2	inf	inf	inf	25.885	8052.847	383.224	3.05E+05
k:2 s:10		10E+3	85.273	161.048	**24.799**	25.885	48.477	383.224	3.05E+05
w:18		10E+4	29.429	161.048	**24.728**	25.885	45.056	383.224	3.05E+05
		10E+6	29.429	161.048	**24.728**	25.885	44.888	383.224	3.05E+05
	5	10E+1	inf	inf	inf	**25.358**	inf	217.126	2315.456
		10E+2	inf	inf	inf	**25.358**	inf	217.126	2315.456
		10E+3	129.854	109.384	**25.145**	25.358	49.883	217.126	2315.456
		10E+4	33.297	109.384	**24.523**	25.358	37.551	217.126	2315.456
		10E+6	33.287	109.384	**24.523**	25.358	36.176	217.126	2315.456
	10	10E+1	inf	inf	inf	**24.287**	inf	87.766	88.086
		10E+2	inf	inf	inf	**24.287**	inf	87.766	88.086
		10E+3	61.834	28.862	**24.205**	24.287	148.953	87.766	88.086
		10E+4	35.731	28.862	**23.864**	24.287	74.772	87.766	88.086
		10E+6	35.03	28.862	**23.864**	24.287	65.901	87.766	88.086
	15	10E+1	inf	inf	inf	23.375	inf	37.104	**22.328**
		10E+2	inf	inf	inf	23.375	inf	37.104	**22.328**
		10E+3	inf	**18.439**	26.849	23.375	1681.759	37.104	22.328
		10E+4	27.407	**18.439**	24.454	23.375	106.849	37.104	22.328
		10E+6	27.076	**18.439**	24.454	23.375	103.211	37.104	22.328
	20	10E+1	inf	inf	inf	22.798	inf	25.006	**15.953**
		10E+2	inf	inf	inf	22.798	inf	25.006	**15.953**
		10E+3	inf	inf	28.493	22.798	7067.541	25.006	**15.953**
		10E+4	22.418	**15.953**	25.316	22.798	752.169	25.006	**15.953**
		10E+6	22.241	**15.953**	24.838	22.798	123.33	25.006	**15.953**
BN-78-w23d6	1	10E+1	inf	inf	inf	**50.042**	2.23E+06	5449.14	3.92E+05
n:54 f:54		10E+2	inf	inf	inf	**50.042**	2.23E+06	5449.14	3.92E+05
k:2 s:10		10E+3	143.06	inf	50.133	**50.042**	90.357	5449.14	3.92E+05
w:22		10E+4	52.941	292.299	**49.88**	50.042	80.933	5449.14	3.92E+05
		10E+6	52.931	292.299	**49.88**	50.042	80.566	5449.14	3.92E+05
	5	10E+1	inf	inf	inf	**48.853**	inf	1789.0	3572.358
		10E+2	inf	inf	inf	**48.853**	inf	1789.0	3572.358
		10E+3	181.978	inf	**48.265**	48.853	510.227	1789.0	3572.358
		10E+4	69.909	127.377	**48.084**	48.853	147.663	1789.0	3572.358
		10E+6	68.512	127.377	**48.084**	48.853	110.716	1789.0	3572.358
	10	10E+1	inf	inf	inf	**47.266**	inf	336.531	438.383
		10E+2	inf	inf	inf	**47.266**	inf	336.531	438.383
		10E+3	189.313	79.834	48.882	**47.266**	1.27E+05	336.531	438.383
		10E+4	86.5	79.834	48.095	**47.266**	1.23E+04	336.531	438.383
		10E+6	85.31	79.834	48.095	**47.266**	3311.865	336.531	438.383
	15	10E+1	inf	inf	inf	**46.466**	inf	134.759	251.559
		10E+2	inf	inf	inf	**46.466**	inf	134.759	251.559
		10E+3	223.989	inf	53.871	**46.466**	1.94E+07	134.759	251.559
		10E+4	92.957	61.503	49.158	**46.466**	4.03E+05	134.759	251.559
		10E+6	96.344	61.503	48.8	**46.466**	1.36E+04	134.759	251.559
	20	10E+1	inf	inf	inf	**45.958**	inf	89.703	89.197
		10E+2	inf	inf	inf	**45.958**	inf	89.703	89.197
		10E+3	inf	inf	53.829	**45.958**	inf	89.703	89.197
		10E+4	inf	**44.487**	51.977	45.958	2.62E+07	89.703	89.197
		10E+6	73.049	**44.487**	48.49	45.958	3.68E+06	89.703	89.197

Table 4.24: Upper Bounds from `BN-78-w18d3` and `BN-78-w23d6` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
BN-78-w19d3	1	10E+1	inf	inf	inf	**26.099**	inf	656.807	3.14E+05
n:54 f:54		10E+2	inf	inf	inf	**26.099**	inf	656.807	3.14E+05
k:2 s:10		10E+3	74.825	159.81	26.48	**26.099**	59.696	656.807	3.14E+05
w:19		10E+4	26.963	159.81	**25.498**	26.099	54.236	656.807	3.14E+05
		10E+6	26.9	159.81	**25.498**	26.099	54.033	656.807	3.14E+05
	5	10E+1	inf	inf	inf	**25.566**	inf	237.886	2957.77
		10E+2	inf	inf	inf	**25.566**	inf	237.886	2957.77
		10E+3	178.157	77.889	27.165	**25.566**	121.056	237.886	2957.77
		10E+4	35.761	77.889	**25.386**	25.566	56.917	237.886	2957.77
		10E+6	35.656	77.889	**25.386**	25.566	50.888	237.886	2957.77
	10	10E+1	inf	inf	inf	**24.349**	inf	65.644	113.178
		10E+2	inf	inf	inf	**24.349**	inf	65.644	113.178
		10E+3	76.111	41.707	25.863	**24.349**	1042.103	65.644	113.178
		10E+4	42.769	41.707	24.878	**24.349**	388.272	65.644	113.178
		10E+6	41.78	41.707	24.878	**24.349**	341.115	65.644	113.178
	15	10E+1	inf	inf	inf	**23.801**	inf	27.633	37.386
		10E+2	inf	inf	inf	**23.801**	inf	27.633	37.386
		10E+3	inf	26.084	26.79	**23.801**	2.78E+04	27.633	37.386
		10E+4	36.846	26.084	24.926	**23.801**	961.642	27.633	37.386
		10E+6	41.335	26.084	24.926	**23.801**	270.253	27.633	37.386
	20	10E+1	inf	inf	inf	23.28	inf	17.719	**17.569**
		10E+2	inf	inf	inf	23.28	inf	17.719	**17.569**
		10E+3	inf	inf	29.446	23.28	inf	17.719	**17.569**
		10E+4	25.637	**17.569**	27.45	23.28	2.14E+04	17.719	**17.569**
		10E+6	23.78	**17.569**	25.355	23.28	1370.839	17.719	**17.569**
BN-0-w29d6	1	10E+1	inf	inf	inf	**53.195**	inf	6.28E+05	3.86E+11
n:100 f:100		10E+2	inf	inf	inf	**53.195**	inf	6.28E+05	3.86E+11
k:2 s:6		10E+3	1504.091	inf	**52.55**	53.195	1063.753	6.28E+05	3.86E+11
w:29		10E+4	53.531	8199.355	**52.085**	53.195	831.807	6.28E+05	3.86E+11
		10E+6	53.531	8199.355	**52.085**	53.195	821.859	6.28E+05	3.86E+11
	5	10E+1	inf	inf	inf	**50.305**	inf	2.08E+04	1.48E+06
		10E+2	inf	inf	inf	**50.305**	inf	2.08E+04	1.48E+06
		10E+3	1204.438	5393.312	52.005	**50.305**	1084.763	2.08E+04	1.48E+06
		10E+4	72.623	5393.312	**48.809**	50.305	327.087	2.08E+04	1.48E+06
		10E+6	72.602	5393.312	**48.809**	50.305	282.977	2.08E+04	1.48E+06
	10	10E+1	inf	inf	inf	**48.609**	inf	2713.49	3624.298
		10E+2	inf	inf	inf	**48.609**	inf	2713.49	3624.298
		10E+3	407.066	inf	55.745	**48.609**	9.28E+05	2713.49	3624.298
		10E+4	89.492	86.675	49.416	**48.609**	2322.398	2713.49	3624.298
		10E+6	87.79	86.675	49.416	**48.609**	347.933	2713.49	3624.298
	15	10E+1	inf	inf	inf	**47.596**	inf	410.212	1261.268
		10E+2	inf	inf	inf	**47.596**	inf	410.212	1261.268
		10E+3	437.082	inf	58.137	**47.596**	5.46E+08	410.212	1261.268
		10E+4	93.204	50.222	49.628	**47.596**	1.35E+05	410.212	1261.268
		10E+6	88.286	50.222	48.955	**47.596**	5.42E+04	410.212	1261.268
	20	10E+1	inf	inf	inf	**45.979**	inf	93.36	397.648
		10E+2	inf	inf	inf	**45.979**	inf	93.36	397.648
		10E+3	inf	inf	inf	**45.979**	inf	93.36	397.648
		10E+4	inf	inf	59.857	**45.979**	5.02E+09	93.36	397.648
		10E+6	103.309	54.302	49.738	**45.979**	3.00E+06	93.36	397.648

Table 4.25: Upper Bounds from `BN-78-w19d3` and `BN-0-w29d6` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
BN-78-w24d6	1	10E+1	inf	inf	inf	**51.57**	inf	7165.85	5.11E+06
n:54 f:54		10E+2	inf	inf	inf	**51.57**	inf	7165.85	5.11E+06
k:2 s:10		10E+3	184.082	inf	**51.251**	51.57	124.009	7165.85	5.11E+06
w:24		10E+4	54.145	2.70E+04	**50.183**	51.57	106.906	7165.85	5.11E+06
		10E+6	54.145	2.70E+04	**50.183**	51.57	106.275	7165.85	5.11E+06
	5	10E+1	inf	inf	inf	**50.688**	inf	2864.95	2.32E+04
		10E+2	inf	inf	inf	**50.688**	inf	2864.95	2.32E+04
		10E+3	391.873	inf	53.882	**50.688**	197.46	2864.95	2.32E+04
		10E+4	67.004	153.358	**49.995**	50.688	102.148	2864.95	2.32E+04
		10E+6	66.389	153.358	**49.995**	50.688	94.873	2864.95	2.32E+04
	10	10E+1	inf	inf	inf	**48.718**	inf	983.923	838.249
		10E+2	inf	inf	inf	**48.718**	inf	983.923	838.249
		10E+3	427.257	inf	54.362	**48.718**	1.28E+05	983.923	838.249
		10E+4	96.062	185.054	50.035	**48.718**	1272.374	983.923	838.249
		10E+6	95.592	185.054	50.035	**48.718**	640.569	983.923	838.249
	15	10E+1	inf	inf	inf	**47.763**	inf	272.525	219.028
		10E+2	inf	inf	inf	**47.763**	inf	272.525	219.028
		10E+3	inf	inf	54.609	**47.763**	1.64E+07	272.525	219.028
		10E+4	102.218	58.84	49.333	**47.763**	1.49E+05	272.525	219.028
		10E+6	98.031	58.84	49.202	**47.763**	2.10E+05	272.525	219.028
	20	10E+1	inf	inf	inf	45.378	inf	73.706	**44.35**
		10E+2	inf	inf	inf	45.378	inf	73.706	**44.35**
		10E+3	inf	inf	inf	45.378	inf	73.706	**44.35**
		10E+4	inf	inf	54.4	45.378	8.33E+06	73.706	**44.35**
		10E+6	92.981	**37.654**	49.843	45.378	7.48E+05	73.706	44.35
BN-0-w28d6	1	10E+1	inf	inf	inf	**49.841**	1.38E+09	7.99E+05	3.84E+10
n:100 f:100		10E+2	inf	inf	inf	**49.841**	1.38E+09	7.99E+05	3.84E+10
k:2 s:6		10E+3	1237.808	inf	**46.998**	49.841	1477.756	7.99E+05	3.84E+10
w:28		10E+4	53.546	2.10E+05	**46.788**	49.841	1219.724	7.99E+05	3.84E+10
		10E+6	53.546	2.10E+05	**46.788**	49.841	1208.315	7.99E+05	3.84E+10
	5	10E+1	inf	inf	inf	**47.135**	inf	2.93E+04	2.94E+05
		10E+2	inf	inf	inf	**47.135**	inf	2.93E+04	2.94E+05
		10E+3	752.778	508.212	47.238	**47.135**	7132.465	2.93E+04	2.94E+05
		10E+4	60.418	508.212	**45.817**	47.135	952.185	2.93E+04	2.94E+05
		10E+6	60.418	508.212	**45.817**	47.135	369.481	2.93E+04	2.94E+05
	10	10E+1	inf	inf	inf	**44.674**	inf	2207.28	2953.99
		10E+2	inf	inf	inf	**44.674**	inf	2207.28	2953.99
		10E+3	350.148	145.678	50.973	**44.674**	1.70E+05	2207.28	2953.99
		10E+4	74.306	145.678	45.899	**44.674**	5255.066	2207.28	2953.99
		10E+6	72.33	145.678	45.899	**44.674**	2036.703	2207.28	2953.99
	15	10E+1	inf	inf	inf	**44.118**	inf	1102.53	794.475
		10E+2	inf	inf	inf	**44.118**	inf	1102.53	794.475
		10E+3	217.447	inf	53.197	**44.118**	3.86E+07	1102.53	794.475
		10E+4	74.044	113.181	46.594	**44.118**	3.70E+04	1102.53	794.475
		10E+6	76.149	113.181	46.152	**44.118**	1.04E+04	1102.53	794.475
	20	10E+1	inf	inf	inf	**43.577**	inf	306.13	242.712
		10E+2	inf	inf	inf	**43.577**	inf	306.13	242.712
		10E+3	inf	inf	inf	**43.577**	inf	306.13	242.712
		10E+4	inf	inf	53.038	**43.577**	6.94E+08	306.13	242.712
		10E+6	76.948	72.787	46.255	**43.577**	1.17E+07	306.13	242.712

Table 4.26: Upper Bounds from `BN-78-w24d6` and `BN-0-w28d6` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
BN-0-w32d11	1	10E+1	inf	inf	inf	**89.287**	inf	1.83E+08	1.01E+12
n:100 f:100		10E+2	inf	inf	inf	**89.287**	inf	1.83E+08	1.01E+12
k:2 s:6		10E+3	4796.115	inf	**85.93**	89.287	4279.11	1.83E+08	1.01E+12
w:33		10E+4	97.977	3.22E+04	**85.234**	89.287	2952.051	1.83E+08	1.01E+12
		10E+6	97.977	3.22E+04	**85.234**	89.287	2869.502	1.83E+08	1.01E+12
	5	10E+1	inf	inf	inf	**86.421**	inf	2.37E+06	6.53E+06
		10E+2	inf	inf	inf	**86.421**	inf	2.37E+06	6.53E+06
		10E+3	9510.201	inf	87.183	**86.421**	1.38E+08	2.37E+06	6.53E+06
		10E+4	144.384	2889.173	**83.467**	86.421	2.72E+05	2.37E+06	6.53E+06
		10E+6	144.384	2889.173	**83.467**	86.421	4.15E+04	2.37E+06	6.53E+06
	10	10E+1	inf	inf	inf	**83.46**	inf	3.73E+04	4.88E+04
		10E+2	inf	inf	inf	**83.46**	inf	3.73E+04	4.88E+04
		10E+3	2724.513	449.163	94.645	**83.46**	4.48E+09	3.73E+04	4.88E+04
		10E+4	150.604	449.163	85.926	**83.46**	9.00E+06	3.73E+04	4.88E+04
		10E+6	149.646	449.163	85.926	**83.46**	1.17E+06	3.73E+04	4.88E+04
	15	10E+1	inf	inf	inf	**82.277**	inf	4833.2	1781.583
		10E+2	inf	inf	inf	**82.277**	inf	4833.2	1781.583
		10E+3	3657.637	inf	98.03	**82.277**	1.55E+13	4833.2	1781.583
		10E+4	178.251	175.244	88.596	**82.277**	2.78E+08	4833.2	1781.583
		10E+6	164.008	175.244	86.428	**82.277**	9.91E+07	4833.2	1781.583
	20	10E+1	inf	inf	inf	inf	inf	**2019.77**	2578.749
		10E+2	inf	inf	inf	inf	inf	2019.77	2578.749
		10E+3	inf	inf	inf	**80.796**	inf	2019.77	2578.749
		10E+4	inf	inf	99.281	**80.796**	3.03E+15	2019.77	2578.749
		10E+6	inf	234.636	87.043	**80.796**	1.29E+13	2019.77	2578.749
BN-0-w33d11	1	10E+1	inf	inf	inf	**92.152**	inf	4.72E+08	6.80E+11
n:100 f:100		10E+2	inf	inf	inf	**92.152**	inf	4.72E+08	6.80E+11
k:2 s:6		10E+3	1860.308	inf	**89.604**	92.152	2134.821	4.72E+08	6.80E+11
w:33		10E+4	98.71	7953.525	**88.914**	92.152	1487.654	4.72E+08	6.80E+11
		10E+6	98.71	7953.525	**88.914**	92.152	1466.044	4.72E+08	6.80E+11
	5	10E+1	inf	inf	inf	**89.245**	inf	2.89E+06	1.18E+06
		10E+2	inf	inf	inf	**89.245**	inf	2.89E+06	1.18E+06
		10E+3	2549.993	1248.837	91.365	**89.245**	1.17E+05	2.89E+06	1.18E+06
		10E+4	118.856	1248.837	**87.986**	89.245	3677.37	2.89E+06	1.18E+06
		10E+6	118.856	1248.837	**87.986**	89.245	1213.3	2.89E+06	1.18E+06
	10	10E+1	inf	inf	inf	**87.222**	inf	8.80E+04	1.93E+04
		10E+2	inf	inf	inf	**87.222**	inf	8.80E+04	1.93E+04
		10E+3	1224.124	278.203	91.106	**87.222**	4.94E+08	8.80E+04	1.93E+04
		10E+4	114.591	278.203	**87.057**	87.222	1.00E+05	8.80E+04	1.93E+04
		10E+6	114.591	278.203	**87.057**	87.222	1.95E+04	8.80E+04	1.93E+04
	15	10E+1	inf	inf	inf	**86.647**	inf	1.68E+04	4824.571
		10E+2	inf	inf	inf	**86.647**	inf	1.68E+04	4824.571
		10E+3	2512.24	inf	101.036	**86.647**	1.51E+15	1.68E+04	4824.571
		10E+4	153.255	219.76	91.637	**86.647**	2.42E+12	1.68E+04	4824.571
		10E+6	151.531	219.76	88.925	**86.647**	8.50E+04	1.68E+04	4824.571
	20	10E+1	inf	inf	inf	**86.359**	inf	5372.39	731.853
		10E+2	inf	inf	inf	**86.359**	inf	5372.39	731.853
		10E+3	inf	inf	inf	**86.359**	inf	5372.39	731.853
		10E+4	inf	inf	98.153	**86.359**	inf	5372.39	731.853
		10E+6	inf	173.278	88.108	**86.359**	inf	5372.39	731.853

Table 4.27: Upper Bounds from `BN-0-w32d11` and `BN-0-w33d11` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
BN-14w42d6	1	10E+1	inf	inf	inf	**52.827**	inf	4.13E+07	4.10E+14
n:115 f:115		10E+2	inf	inf	inf	**52.827**	inf	4.13E+07	4.10E+14
k:2 s:8		10E+3	2.14E+04	3.63E+06	53.131	**52.827**	1.97E+04	4.13E+07	4.10E+14
w:41		10E+4	50.343	3.63E+06	**48.692**	52.827	1.11E+04	4.13E+07	4.10E+14
		10E+6	50.343	3.63E+06	**48.692**	52.827	1.05E+04	4.13E+07	4.10E+14
	5	10E+1	inf	inf	inf	**50.342**	inf	8.89E+05	1.43E+09
		10E+2	inf	inf	inf	**50.342**	inf	8.89E+05	1.43E+09
		10E+3	1.23E+05	inf	51.108	**50.342**	2.91E+05	8.89E+05	1.43E+09
		10E+4	71.99	2.00E+04	**46.484**	50.342	5679.025	8.89E+05	1.43E+09
		10E+6	71.99	2.00E+04	**46.484**	50.342	2729.66	8.89E+05	1.43E+09
	10	10E+1	inf	inf	inf	**47.697**	inf	6.53E+04	3.49E+06
		10E+2	inf	inf	inf	**47.697**	inf	6.53E+04	3.49E+06
		10E+3	7.72E+04	inf	61.266	**47.697**	1.47E+08	6.53E+04	3.49E+06
		10E+4	79.66	1535.094	49.503	**47.697**	1.67E+05	6.53E+04	3.49E+06
		10E+6	79.66	1535.094	47.848	**47.697**	3.92E+04	6.53E+04	3.49E+06
	15	10E+1	inf	inf	inf	**45.27**	inf	5477.65	1.81E+05
		10E+2	inf	inf	inf	**45.27**	inf	5477.65	1.81E+05
		10E+3	inf	inf	66.573	**45.27**	7.05E+10	5477.65	1.81E+05
		10E+4	406.989	4.37E+05	56.533	**45.27**	2.58E+06	5477.65	1.81E+05
		10E+6	96.929	4.37E+05	49.784	**45.27**	2.78E+05	5477.65	1.81E+05
	20	10E+1	inf	inf	inf	inf	inf	**2261.68**	3480.875
		10E+2	inf	inf	inf	inf	inf	2261.68	3480.875
		10E+3	inf	inf	inf	**43.722**	inf	2261.68	3480.875
		10E+4	inf	inf	66.883	**43.722**	2.45E+13	2261.68	3480.875
		10E+6	inf	301.453	50.356	**43.722**	6.03E+10	2261.68	3480.875
BN-14w57d12	1	10E+1	inf	inf	inf	**101.915**	inf	1.58E+10	3.04E+16
n:115 f:115		10E+2	inf	inf	inf	**101.915**	inf	1.58E+10	3.04E+16
k:2 s:8		10E+3	8.04E+04	inf	103.701	**101.915**	8.39E+04	1.58E+10	3.04E+16
w:45		10E+4	1417.997	1.36E+08	**98.538**	101.915	4.18E+04	1.58E+10	3.04E+16
		10E+6	1417.997	1.36E+08	**98.538**	101.915	3.98E+04	1.58E+10	3.04E+16
	5	10E+1	inf	inf	inf	**98.305**	inf	1.95E+08	4.85E+10
		10E+2	inf	inf	inf	**98.305**	inf	1.95E+08	4.85E+10
		10E+3	2.01E+05	inf	106.332	**98.305**	6.72E+06	1.95E+08	4.85E+10
		10E+4	146.286	8.39E+04	**97.311**	98.305	8.64E+04	1.95E+08	4.85E+10
		10E+6	146.286	8.39E+04	**97.174**	98.305	1.69E+04	1.95E+08	4.85E+10
	10	10E+1	inf	inf	inf	**94.669**	inf	4.60E+06	2.65E+06
		10E+2	inf	inf	inf	**94.669**	inf	4.60E+06	2.65E+06
		10E+3	1.24E+05	inf	111.21	**94.669**	2.45E+11	4.60E+06	2.65E+06
		10E+4	170.58	5250.491	98.417	**94.669**	1.27E+07	4.60E+06	2.65E+06
		10E+6	170.58	5250.491	97.176	**94.669**	1.28E+06	4.60E+06	2.65E+06
	15	10E+1	inf	inf	inf	**93.452**	inf	2.93E+05	2.85E+05
		10E+2	inf	inf	inf	**93.452**	inf	2.93E+05	2.85E+05
		10E+3	2.11E+05	inf	115.103	**93.452**	1.04E+16	2.93E+05	2.85E+05
		10E+4	1510.737	1372.088	103.94	**93.452**	3.13E+11	2.93E+05	2.85E+05
		10E+6	192.74	1372.088	97.454	**93.452**	1.09E+09	2.93E+05	2.85E+05
	20	10E+1	inf	inf	inf	inf	inf	inf	inf
		10E+2	inf	inf	inf	inf	inf	inf	inf
		10E+3	inf	inf	inf	**92.062**	inf	6.45E+04	2.78E+04
		10E+4	inf	inf	116.813	**92.062**	1.92E+18	6.45E+04	2.78E+04
		10E+6	inf	772.404	99.368	**92.062**	1.77E+15	6.45E+04	2.78E+04

Table 4.28: Upper Bounds from `BN-14w42d6` and `BN-14w57d12` Instances. The table shows the best upper bound generated within the varying time bounds from 10 seconds to 100,000 seconds and i-bounds from 1 to 20. n is the number of variables, f is the number of functions, k is the maximum domains size s is the maximum scop e size, w is the constrained induced width, and i-bd is the i-bound.

4.3.5 Summary

Given all the results tabulated above a clear picture emerges.

- JGD-EXP and WMBMM-EXP are overall the best performing algorithm.
- JGD-EXP outperformed WMBMM-EXP when they use a low i-bounds and are given sufficient time (longer than 1,000 seconds in our experiments). Otherwise WMBMM-EXP generated tighter upper bounds. (Namely, when given less than 1,000 seconds, or for i-bounds larger than 10.)
- JGD-ID and WMBE-ID generated relatively good upper bounds when the problem instances were easy. Their strength is that if given an i-bound higher than the induced-width they generate the exact answer while JGD-EXP and WMBMM-EXP do not.
- The translation based approaches and MBE-ID didn't generate good upper bounds in most of the cases.

4.4 Convergence Behavior for Iterative Algorithms

In this section, we report the convergence behavior of the iterative algorithms, JGD-ID, WMBE-ID, JGD-EXP, and GDD-MI. Note that WMBE-ID and WMBM-EXP generate the upper bound only once when they terminate. Therefore, we report the upper bound from WMBE-ID and WMBMM-EXP when they terminate.

4.4.1 FH-MDP Domain

Figure 4.1 and 4.2 report upper bounds as a function of time at `mdp8-28-3-6-4` and `mdp9-32-3-8-3` instances. The plots on the left hand side show the upper bounds from all algorithms in log scale with varying i-bounds from 1 to 15. The plots on the right hand side focus on the expected utility close to the best upper bounds in linear scale. JGD-ID and GDD-MI showed slower speed of convergence especially when the i-bound is larger. The results from WMBE-ID are tighter than the bounds from JGD-EXP and WMBMM-EXP as was also shown in Table 4.12 and 4.13.

4.4.2 FH-POMDP Domain

Figure 4.3 and 4.4 report upper bounds at `pomdp5-6-4-3-5-3` and `pomdp10-12-7-3-8-4` instances. We first see that the quality of upper bounds from JGD-EXP and WMBMM-EXP are order of magnitude tighter than other algorithms, and WMBE-ID showed clear improvements in the upper bounds given higher i-bounds. Comparing the convergence behavior, we see that JGD-EXP and GDD-MI improved upper bounds smoothly over time, while JGD-ID showed step-wise improvements. At `pomdp5-6-4-3-5-3`, we can observe that the upper bounds from JGD-EXP is tighter than WBMMM-EXP after time bounds 1,000 seconds when they were given i-bounds 1 or 5. As we increase i-bound to 10 and 15, WBMMM-EXP produced tighter upper bound than JGD-EXP.

4.4.3 RAND Domain

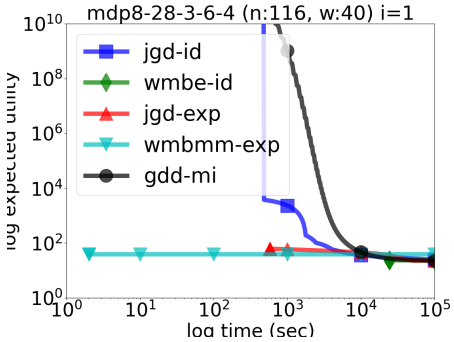
Figure 4.5 and 4.6 report upper bounds at `rand-c50d1501` and `rand-c70d21o1` instances. As in the previous domains, we observe similar convergence behavior at both instances. Namely, WBMMM-EXP terminated and generated the upper bound earlier than other algorithms, and JGD-EXP converged to tighter i-bound in shorter time bounds than other iterative algorithms JGD-ID and GDD-MI. Algorithm WMBE-ID improved the quality of upper bounds when it was given higher i-bounds.

4.4.4 BN Domain

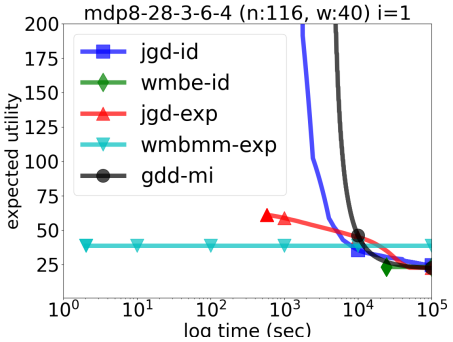
Figure 4.7 and 4.8 report upper bounds at `BN-14w57d12` and `BN-78-w24d6` instances. From both instances, we observe the transition of the best performing algorithms due to increased i-bounds. Namely, we see that the upper bounds from JGD-EXP is tighter than WBMMM-EXP when they were given i-bounds 1 or 5. However, WBMMM-EXP started to generate tighter upper bounds than JGD-EXP when it was given i-bound greater than 10.

4.4.5 Summary

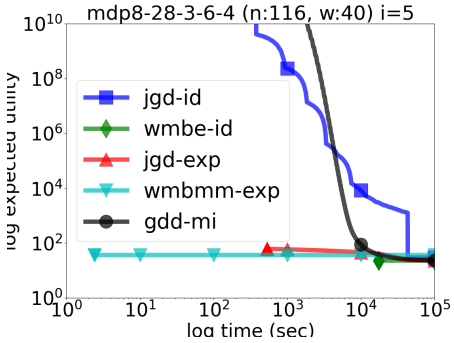
In this section, we presented the convergence behavior of our bounding algorithms. Among iterative algorithms, we see that algorithm JGD-EXP and WMBMM-EXP generated the tightest upper bounds that are order of magnitude tighter than other algorithms at shorter time bounds. Comparing JGD-ID and GDD-MI, We see that the JGD-ID showed step-wise improvement behavior until convergence, when GDD-MI showed smoother curves. We see that WMBE-ID greatly improved the upper bounds given higher i-bounds when other iterative algorithms often generated worse upper bounds or minor improvements. As we also observed in the tabular results earlier, WMBMM-EXP generated tighter upper bounds than JGD-EXP with higher i-bounds or when both have time bounds shorter than 1,000 seconds.



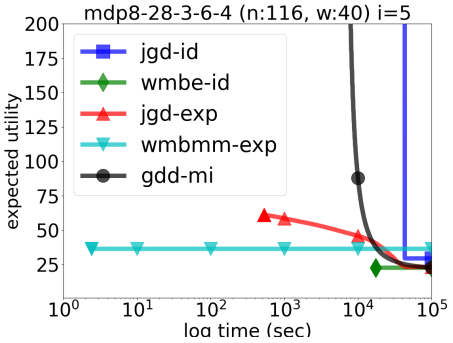
(a) mdp8-28-3-6-4 i=1



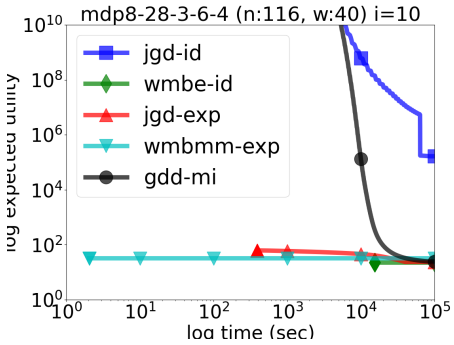
(b) mdp8-28-3-6-4 i=1



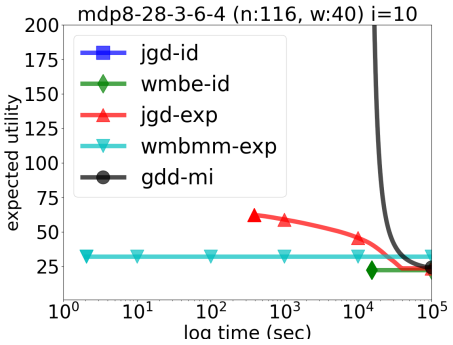
(c) mdp8-28-3-6-4 i=5



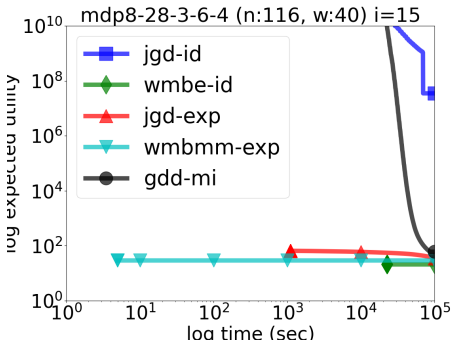
(d) mdp8-28-3-6-4 i=5



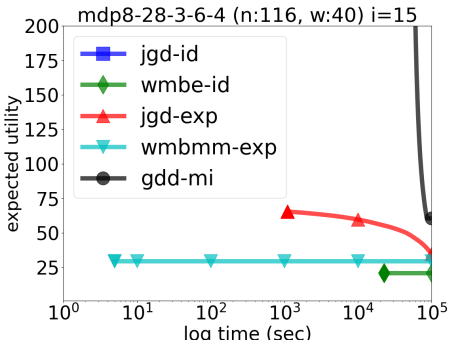
(e) mdp8-28-3-6-4 i=10



(f) mdp8-28-3-6-4 i=10

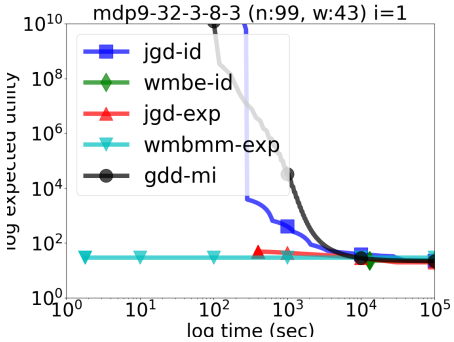


(g) mdp8-28-3-6-4 i=15

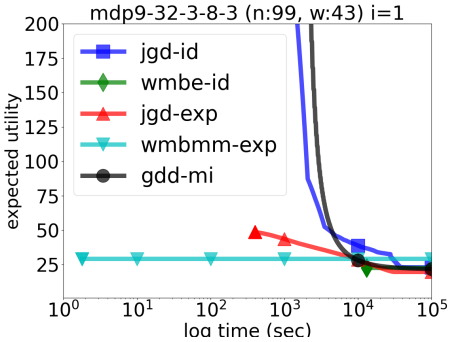


(h) mdp8-28-3-6-4 i=15

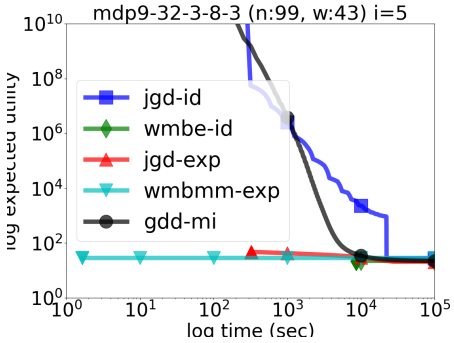
Figure 4.1: Convergence Behavior over Varying i-bounds at mdp8-28-3-6-4. The x-axis of is the time in log scale. Figures on the left hand side shows the expected utility in the log scale, and on the right hand side shows the expected utility in the linear scale in a focused region.



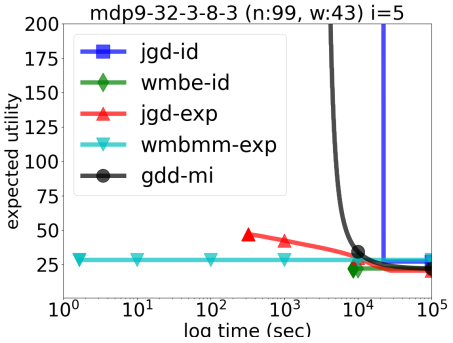
(a) mdp9-32-3-8-3 i=1



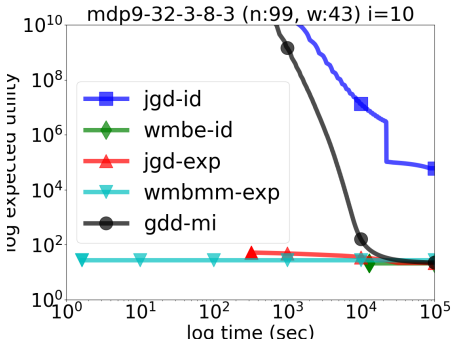
(b) mdp9-32-3-8-3 i=1



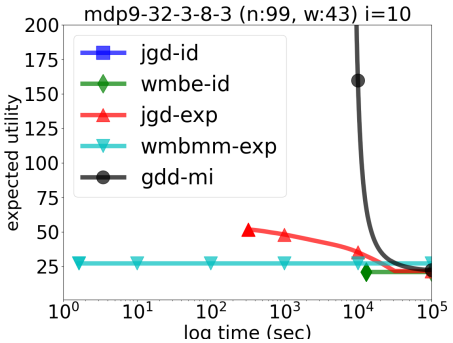
(c) mdp9-32-3-8-3 i=5



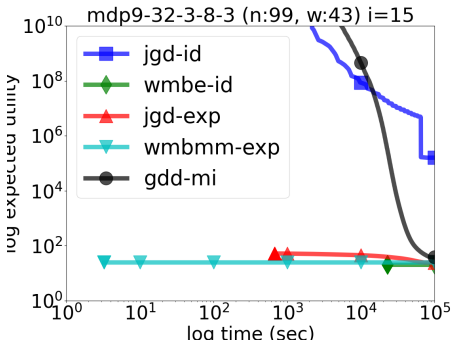
(d) mdp9-32-3-8-3 i=5



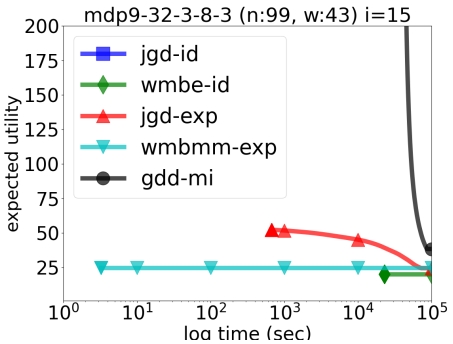
(e) mdp9-32-3-8-3 i=10



(f) mdp9-32-3-8-3 i=10

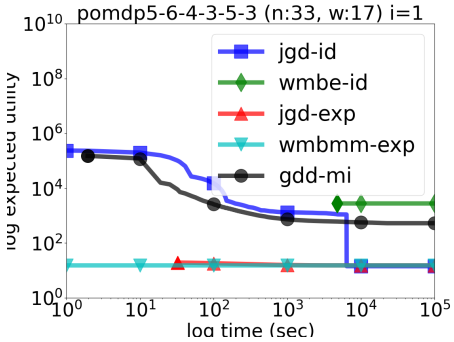


(g) mdp9-32-3-8-3 i=15

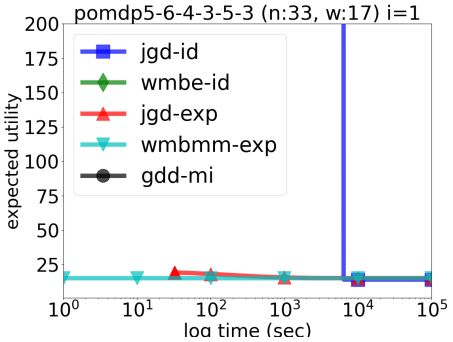


(h) mdp9-32-3-8-3 i=15

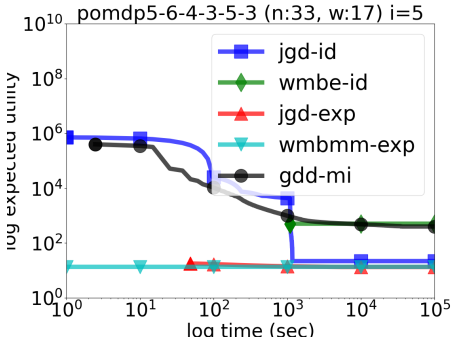
Figure 4.2: Convergence Behavior over Varying i-bounds at mdp9-32-3-8-3. The x-axis of is the time in log scale. Figures on the left hand side shows the expected utility in the log scale, and on the right hand side shows the expected utility in the linear scale in a focused region.



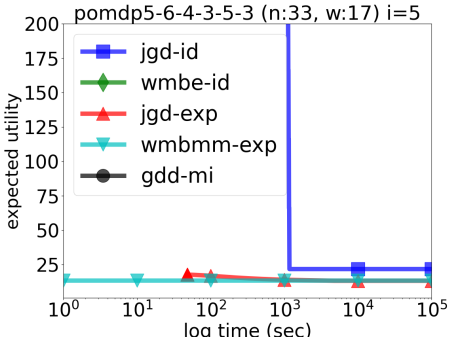
(a) pomdp5-6-4-3-5-3 i=1



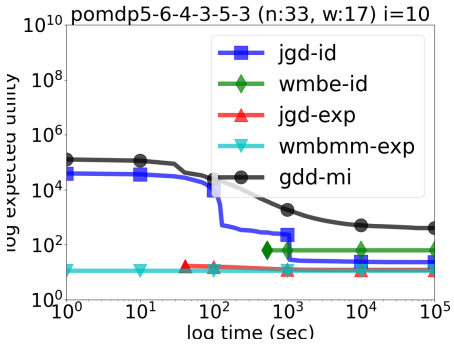
(b) pomdp5-6-4-3-5-3 i=1



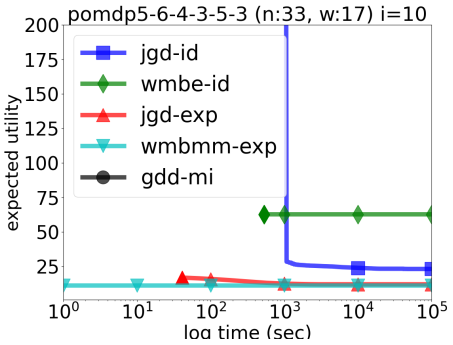
(c) pomdp5-6-4-3-5-3 i=5



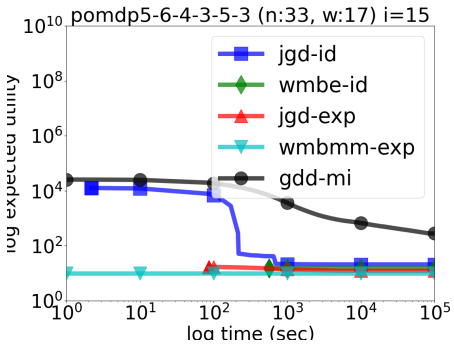
(d) pomdp5-6-4-3-5-3 i=5



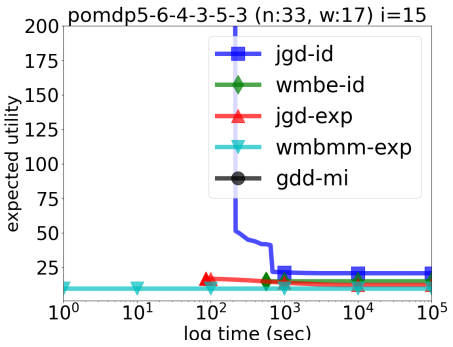
(e) pomdp5-6-4-3-5-3 i=10



(f) pomdp5-6-4-3-5-3 i=10

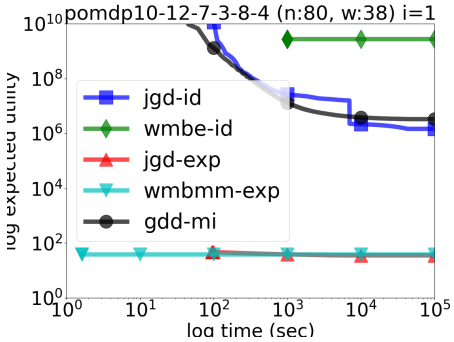


(g) pomdp5-6-4-3-5-3 i=15

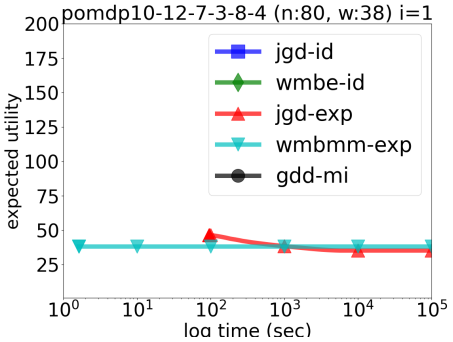


(h) pomdp5-6-4-3-5-3 i=15

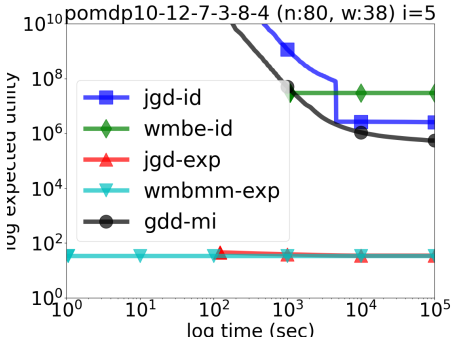
Figure 4.3: Convergence Behavior over Varying i-bounds at pomdp5-6-4-3-5-3. The x-axis of is the time in log scale. Figures on the left hand side shows the expected utility in the log scale, and on the right hand side shows the expected utility in the linear scale in a focused region.



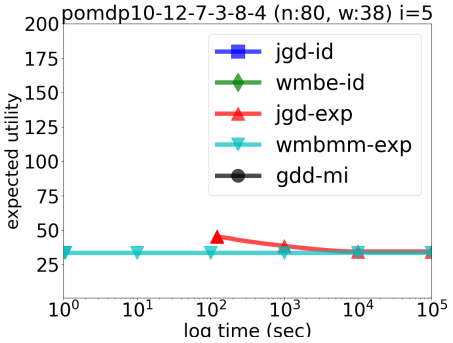
(a) pomdp10-12-7-3-8-4 i=1



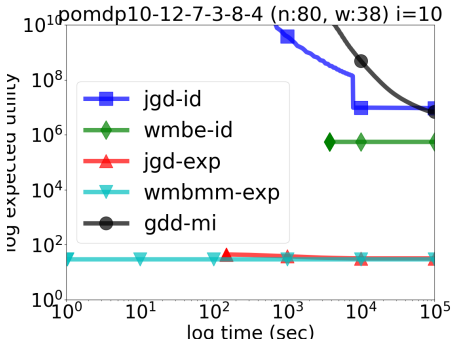
(b) pomdp10-12-7-3-8-4 i=1



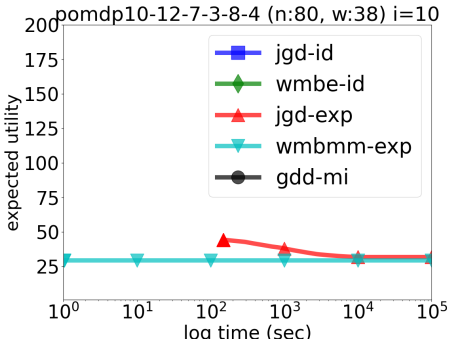
(c) pomdp10-12-7-3-8-4 i=5



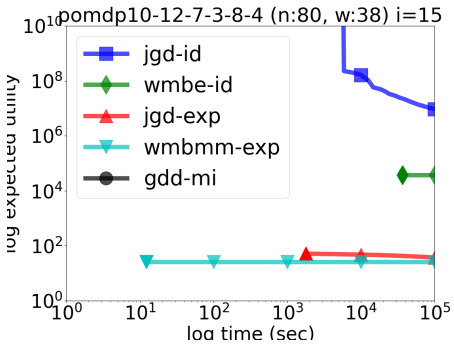
(d) pomdp10-12-7-3-8-4 i=5



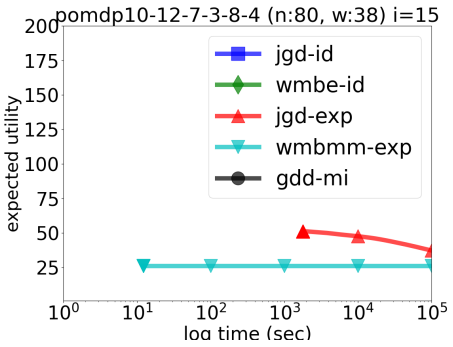
(e) pomdp10-12-7-3-8-4 i=10



(f) pomdp10-12-7-3-8-4 i=10

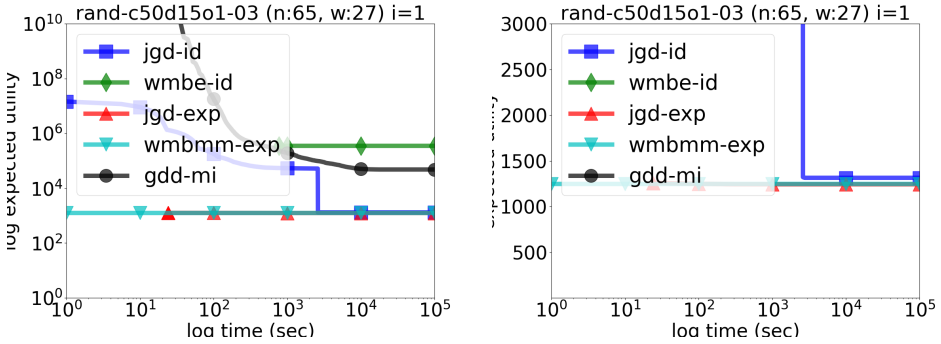


(g) pomdp10-12-7-3-8-4 i=15



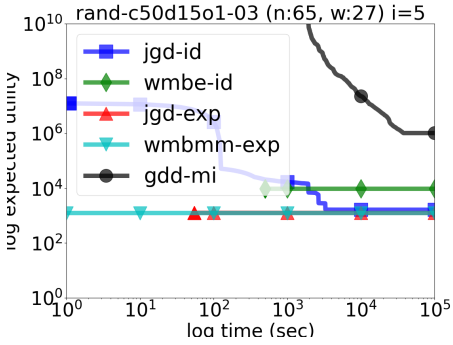
(h) pomdp10-12-7-3-8-4 i=15

Figure 4.4: Convergence Behavior over Varying i-bounds at pomdp10-12-7-3-8-4. The x-axis of is the time in log scale. Figures on the left hand side shows the expected utility in the log scale, and on the right hand side shows the expected utility in the linear scale in a focused region.

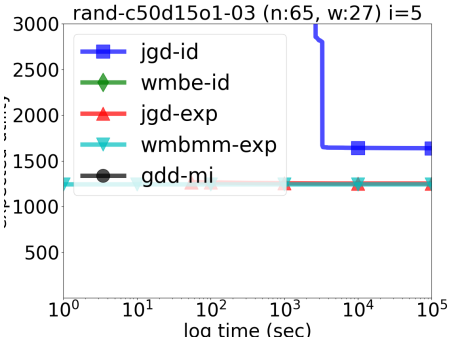


(a) rand-c50d15o1-03 i=1

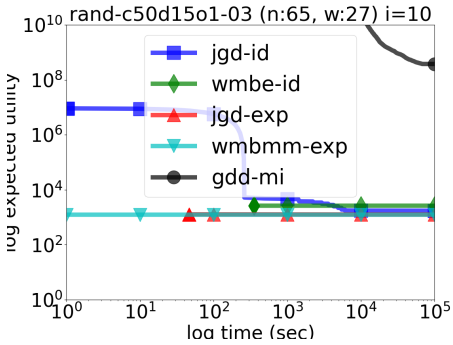
(b) rand-c50d15o1-03 i=1



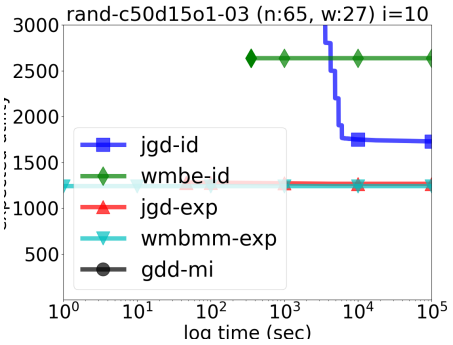
(c) rand-c50d15o1-03 i=5



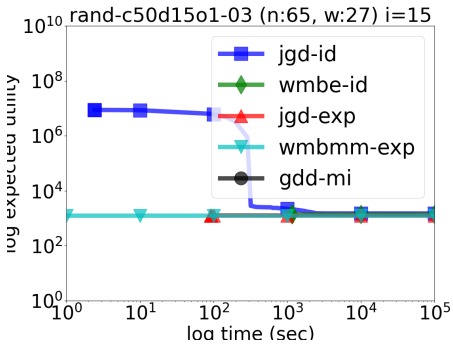
(d) rand-c50d15o1-03 i=5



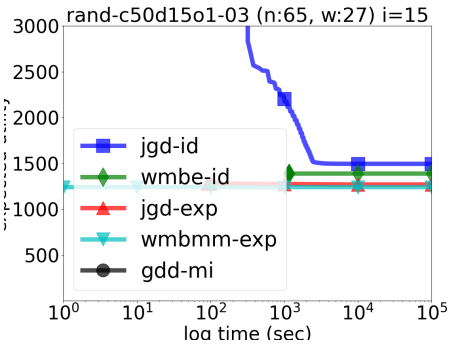
(e) rand-c50d15o1-03 i=10



(f) rand-c50d15o1-03 i=10

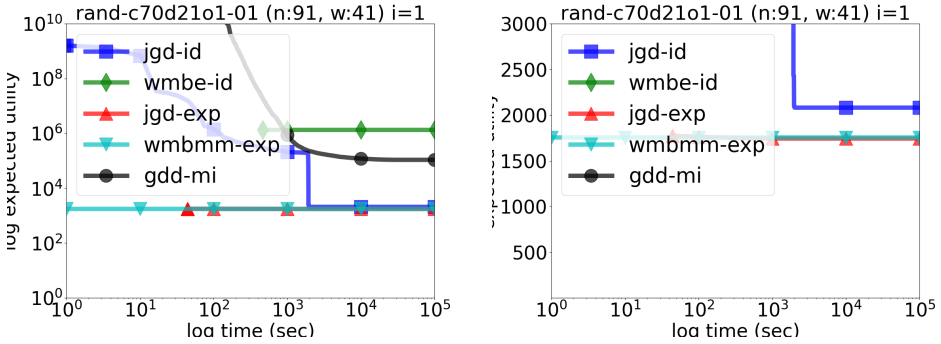


(g) rand-c50d15o1-03 i=15



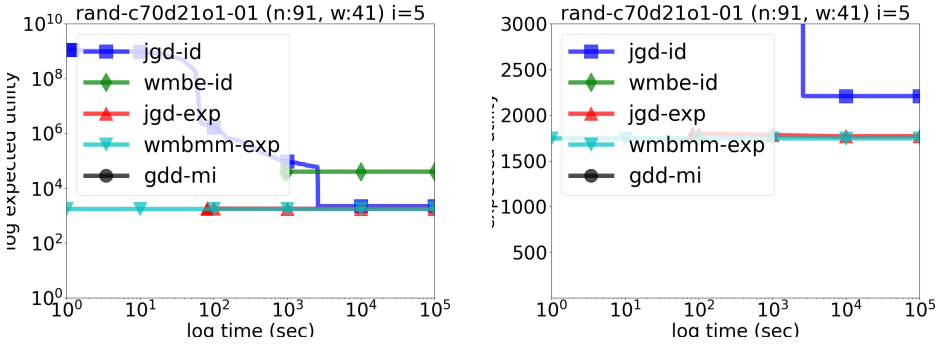
(h) rand-c50d15o1-03 i=15

Figure 4.5: Convergence Behavior over Varying i-bounds at rand-c50d15o1-03. The x-axis of is the time in log scale. Figures on the left hand side shows the expected utility in the log scale, and on the right hand side shows the expected utility in the linear scale in a focused region.



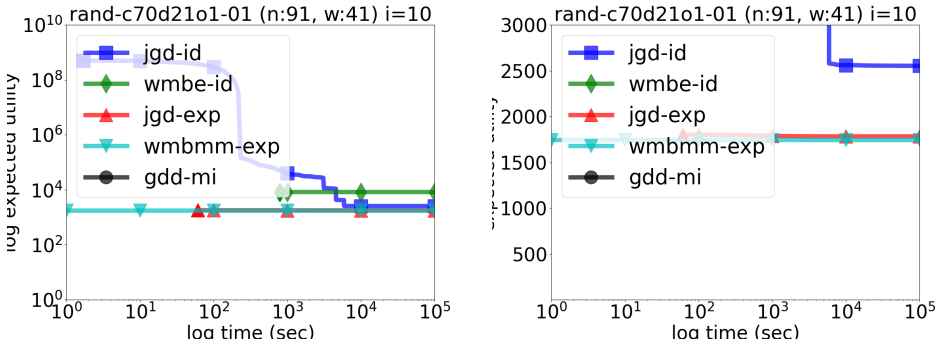
(a) rand-c70d21o1-01 i=1

(b) rand-c70d21o1-01 i=1



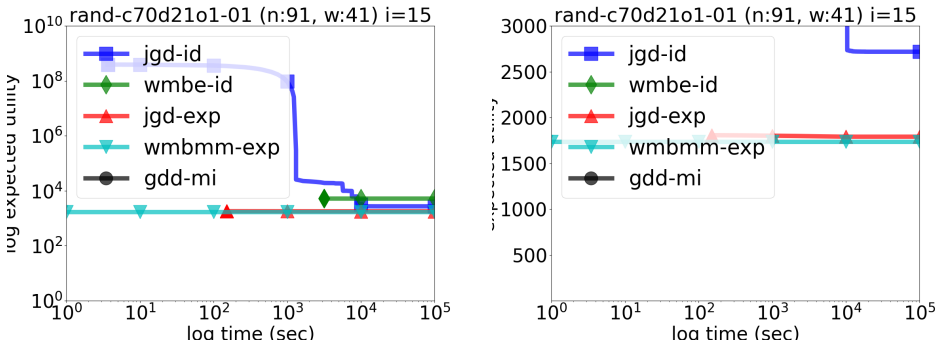
(c) rand-c70d21o1-01 i=5

(d) rand-c70d21o1-01 i=5



(e) rand-c70d21o1-01 i=10

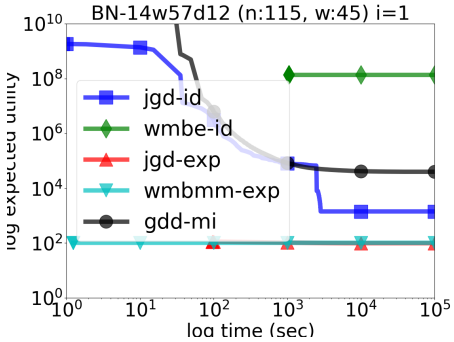
(f) rand-c70d21o1-01 i=10



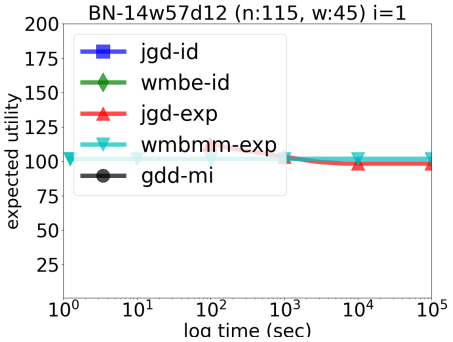
(g) rand-c70d21o1-01 i=15

(h) rand-c70d21o1-01 i=15

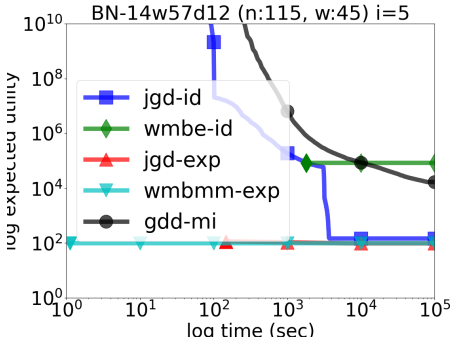
Figure 4.6: Convergence Behavior over Varying i-bounds at rand-c70d21o1-01. The x-axis of is the time in log scale. Figures on the left hand side shows the expected utility in the log scale, and on the right hand side shows the expected utility in the linear scale in a focused region.



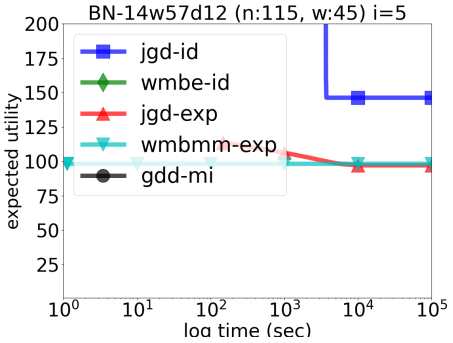
(a) BN-14w57d12 i=1



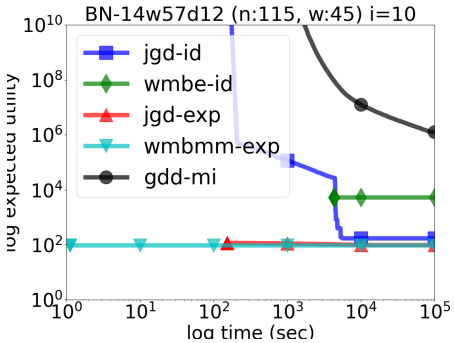
(b) BN-14w57d12 i=1



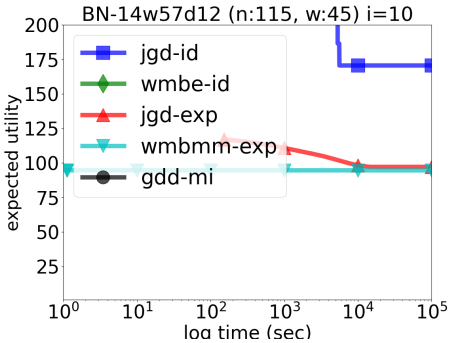
(c) BN-14w57d12 i=5



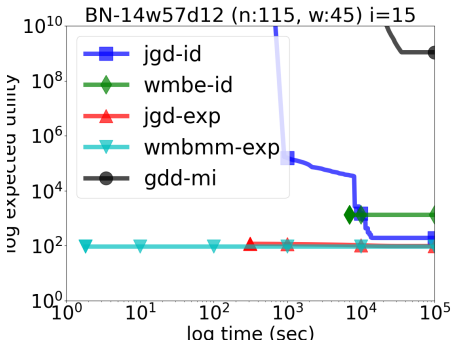
(d) BN-14w57d12 i=5



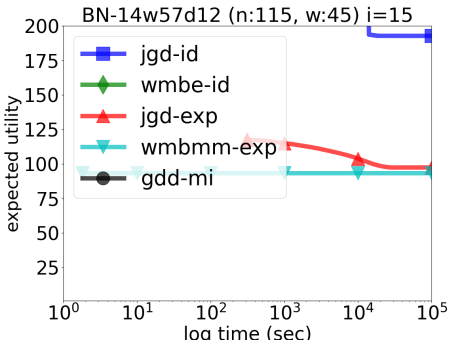
(e) BN-14w57d12 i=10



(f) BN-14w57d12 i=10

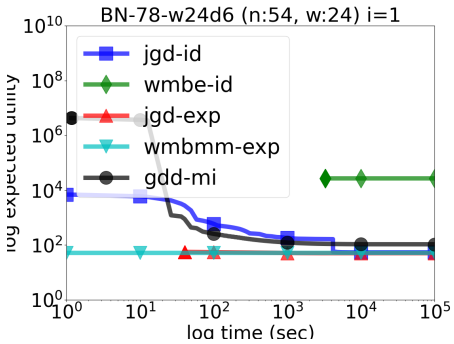


(g) BN-14w57d12 i=15

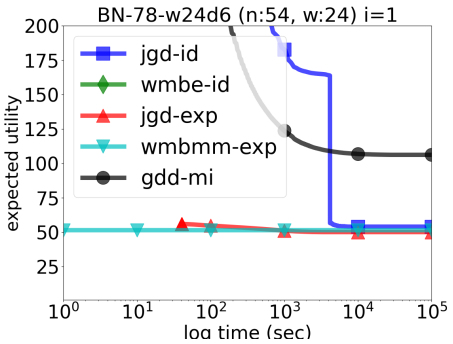


(h) BN-14w57d12 i=15

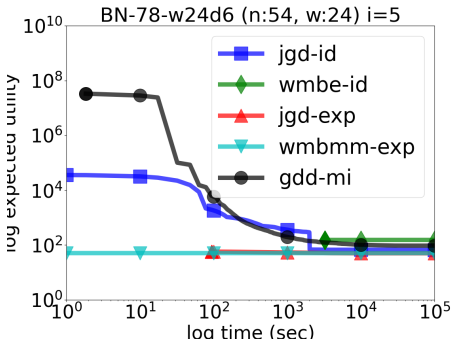
Figure 4.7: Convergence Behavior over Varying i-bounds at BN-14w57d12. The x-axis of is the time in log scale. Figures on the left hand side shows the expected utility in the log scale, and on the right hand side shows the expected utility in the linear scale in a focused region.



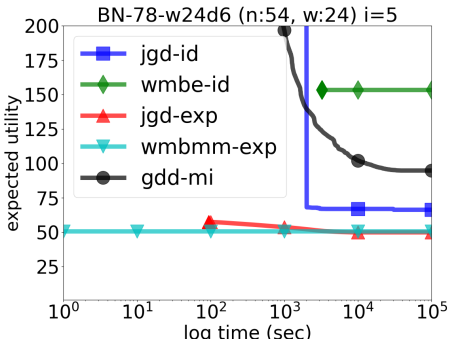
(a) BN-78-w24d6 i=1



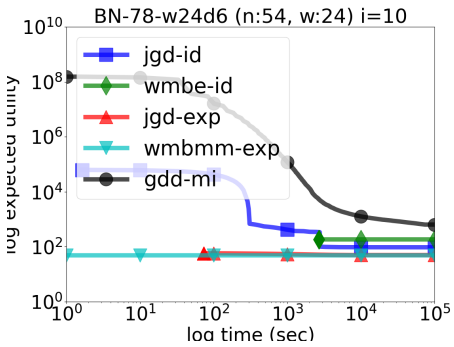
(b) BN-78-w24d6 i=1



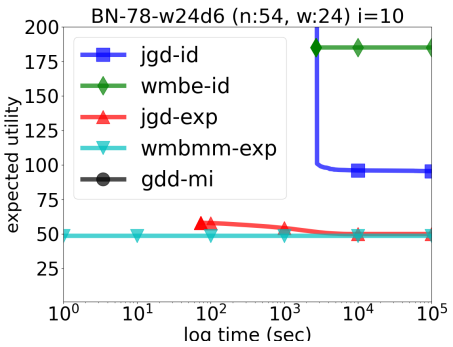
(c) BN-78-w24d6 i=5



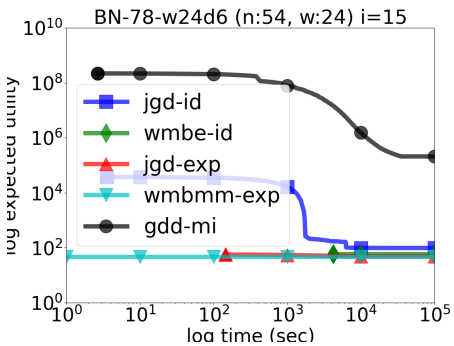
(d) BN-78-w24d6 i=5



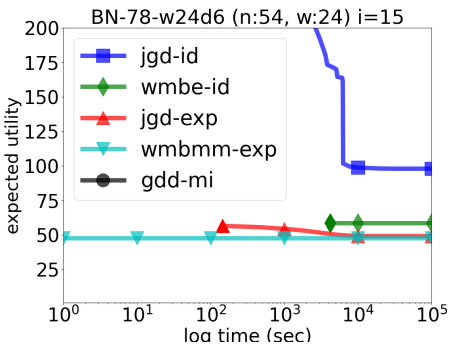
(e) BN-78-w24d6 i=10



(f) BN-78-w24d6 i=10



(g) BN-78-w24d6 i=15



(h) BN-78-w24d6 i=15

Figure 4.8: Convergence Behavior over Varying i-bounds at BN-78-w24d6. The x-axis of is the time in log scale. Figures on the left hand side shows the expected utility in the log scale, and on the right hand side shows the expected utility in the linear scale in a focused region.

Domain	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
All	1	10E+1	0.017	0.0	0.086	**0.708**	0.011	0.005	0.002
n:75.6 f:87.1		10E+2	0.048	0.006	0.504	**0.708**	0.055	0.005	0.002
k:2.4 s:5.9		10E+3	0.112	0.059	0.67	**0.708**	0.105	0.005	0.002
w:25.9		10E+4	0.584	0.096	**0.73**	0.708	0.195	0.005	0.002
		10E+6	0.663	0.181	**0.786**	0.708	0.272	0.005	0.002
	5	10E+1	0.045	0.0	0.019	**0.753**	0.01	0.022	0.066
		10E+2	0.078	0.065	0.457	**0.753**	0.036	0.022	0.066
		10E+3	0.175	0.152	0.686	**0.753**	0.095	0.022	0.066
		10E+4	0.449	0.262	**0.759**	0.753	0.176	0.022	0.066
		10E+6	0.539	0.305	**0.808**	0.753	0.288	0.022	0.066
	10	10E+1	0.062	0.0	0.034	**0.818**	0.008	0.087	0.2
		10E+2	0.151	0.099	0.449	**0.818**	0.027	0.087	0.2
		10E+3	0.252	0.284	0.67	**0.818**	0.071	0.087	0.2
		10E+4	0.46	0.441	0.749	**0.818**	0.108	0.087	0.2
		10E+6	0.468	0.486	0.816	**0.818**	0.226	0.087	0.2
	15	10E+1	0.062	0.025	0.034	**0.798**	0.007	0.184	0.33
		10E+2	0.147	0.099	0.227	**0.861**	0.025	0.184	0.33
		10E+3	0.266	0.283	0.574	**0.861**	0.055	0.184	0.33
		10E+4	0.433	0.489	0.682	**0.861**	0.083	0.184	0.33
		10E+6	0.525	0.631	0.77	**0.861**	0.106	0.184	0.33
	20	10E+1	0.062	0.0	0.019	**0.557**	0.005	0.271	0.478
		10E+2	0.184	0.124	0.137	**0.829**	0.02	0.271	0.478
		10E+3	0.262	0.249	0.335	**0.879**	0.056	0.275	0.478
		10E+4	0.363	0.445	0.568	**0.904**	0.072	0.275	0.478
		10E+6	0.479	0.705	0.614	**0.904**	0.075	0.275	0.478

Table 4.29: Average Quality of Upper Bounds Over All Domains. Table shows the average quality (closer to 1.0, higher the quality) of each algorithm at varying i-bounds and time bounds. i-bd is the i-bound, time is the time bound in seconds, n is the average number of variables, f is the average number of functions, k is the average of the maximum domains size, s is the average of the maximum scope size, w is the average of the constrained induced width.

4.5 Average Quality of Upper Bounds

We next summarize the performance of our upper bounding schemes by reporting average quality at varying time bounds and i-bounds. Namely, we compute $\frac{1}{N_{\text{domain}}} \sum_{k=1}^{N_{\text{domain}}} \frac{\text{ub}_k^*}{\text{ub}_k}$, where N_{domain} is the total number of instances in a domain, ub_k^* is the best known upper bound of the k-th problem instance, and ub_k is the upper bound of the k-th problem instance generated by each algorithm.

Table 4.29 shows the average quality of upper bounds over all domains. Consistent with our earlier tables and plots we see that JGD-EXP was the best performing algorithm when it was given i-bound 1 or 5 and time bound longer than 1,000 seconds. When allowed higher memory

resources, that correspond to higher i-bounds WMBMM-EXP generated superior bounds than others. For i-bound=10, both JGD-EXP and WMBMM-EXP computed comparable upper bounds with similar relative distance from the average quality, 0.816 from JGD-EXP and 0.818 from WMBMM-EXP. Among all possible combinations of i-bounds and time bounds, algorithm WMBMM-EXP with i-bound 20 and time bound 10,000 seconds was the best performing configuration, and the same algorithm with i-bound 15 and 10 are the second and third best configurations. Then, JGD-EXP with i-bound 10, 5, and 1 with the time bound 1,000,000 seconds follow.

Table 4.30 shows the average quality in `FH-MDP` and `FH-POMDP` domains. In `FH-MDP` domain, we see a similar trend, but the best configuration was WMBE-ID with i-bound 15 and a time bound 1,000,000 seconds. We clearly observe that JGD-EXP was the overall best in terms of average quality of upper bounds when we limit the i-bound to 1 and 5, and WMBMM-EXP gradually takes over as we allow higher i-bounds. We see the same trend in `FH-POMDP` domain. We presents average quality of upper bounds in `RAND` and `BN` domains in Table 4.31, which shows the same trends.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
fh-mdp	1	10E+1	0.041	0.0	0.048	**0.583**	0.004	0.0	0.001
n:105.7 f:134.1		10E+2	0.084	0.0	0.122	**0.583**	0.048	0.0	0.001
k:3.1 s:7.1		10E+3	0.155	0.152	0.473	**0.583**	0.137	0.0	0.001
w:25.9		10E+4	0.463	0.28	**0.631**	0.583	0.465	0.0	0.001
		10E+6	0.779	0.619	**0.841**	0.583	0.77	0.0	0.001
	5	10E+1	0.086	0.0	0.0	**0.645**	0.0	0.0	0.1
		10E+2	0.1	0.1	0.113	**0.645**	0.016	0.0	0.1
		10E+3	0.101	0.185	0.485	**0.645**	0.113	0.0	0.1
		10E+4	0.154	0.562	**0.659**	0.645	0.321	0.0	0.1
		10E+6	0.507	0.734	**0.841**	0.645	0.723	0.0	0.1
	10	10E+1	0.086	0.0	0.0	**0.738**	0.0	0.0	0.2
		10E+2	0.1	0.1	0.073	**0.738**	0.006	0.0	0.2
		10E+3	0.196	0.2	0.453	**0.738**	0.074	0.0	0.2
		10E+4	0.196	0.609	0.613	**0.738**	0.16	0.0	0.2
		10E+6	0.208	0.79	**0.842**	0.738	0.606	0.0	0.2
	15	10E+1	0.086	0.1	0.0	**0.734**	0.0	0.001	0.4
		10E+2	0.1	0.1	0.07	**0.81**	0.003	0.001	0.4
		10E+3	0.196	0.2	0.26	**0.81**	0.041	0.001	0.4
		10E+4	0.295	0.394	0.458	**0.81**	0.088	0.001	0.4
		10E+6	0.457	**0.959**	0.691	0.81	0.182	0.001	0.4
	20	10E+1	0.086	0.0	0.0	0.387	0.0	0.002	**0.4**
		10E+2	0.1	0.1	0.07	**0.773**	0.004	0.002	0.4
		10E+3	0.196	0.2	0.195	**0.873**	0.061	0.002	0.4
		10E+4	0.295	0.3	0.313	**0.873**	0.079	0.002	0.4
		10E+6	0.394	0.8	0.361	**0.873**	0.079	0.002	0.4
fh-pomdp	1	10E+1	0.01	0.0	0.153	**0.606**	0.014	0.003	0.002
n:55.9 f:73.5		10E+2	0.024	0.0	0.302	**0.606**	0.028	0.003	0.002
k:2.4 s:5.5		10E+3	0.082	0.012	0.551	**0.606**	0.033	0.003	0.002
w:28.0		10E+4	0.394	0.015	**0.611**	0.606	0.035	0.003	0.002
		10E+6	0.395	0.015	**0.624**	0.606	0.035	0.003	0.002
	5	10E+1	0.016	0.0	0.076	**0.69**	0.026	0.013	0.045
		10E+2	0.076	0.063	0.299	**0.69**	0.056	0.013	0.045
		10E+3	0.211	0.095	0.611	**0.69**	0.073	0.013	0.045
		10E+4	0.328	0.097	0.682	**0.69**	0.084	0.013	0.045
		10E+6	0.328	0.097	**0.697**	0.69	0.086	0.013	0.045
	10	10E+1	0.082	0.0	0.075	**0.82**	0.022	0.082	0.225
		10E+2	0.224	0.2	0.331	**0.82**	0.055	0.082	0.225
		10E+3	0.284	0.262	0.622	**0.82**	0.082	0.082	0.225
		10E+4	0.385	0.262	0.703	**0.82**	0.093	0.082	0.225
		10E+6	0.387	0.263	0.738	**0.82**	0.096	0.082	0.225
	15	10E+1	0.082	0.0	0.075	**0.723**	0.018	0.229	0.289
		10E+2	0.237	0.2	0.287	**0.9**	0.051	0.229	0.289
		10E+3	0.325	0.318	0.478	**0.9**	0.08	0.229	0.289
		10E+4	0.384	0.36	0.618	**0.9**	0.089	0.229	0.289
		10E+6	0.384	0.36	0.713	**0.9**	0.097	0.229	0.289
	20	10E+1	0.082	0.0	0.075	**0.488**	0.015	0.237	0.427
		10E+2	0.289	0.2	0.193	**0.788**	0.045	0.237	0.427
		10E+3	0.396	0.3	0.314	**0.888**	0.078	0.252	0.427
		10E+4	0.471	0.4	0.396	**0.988**	0.083	0.252	0.427
		10E+6	0.471	0.468	0.427	**0.988**	0.085	0.252	0.427

Table 4.30: Average Quality of Upper Bounds in FH-MDP and FH-POMDP Domains. Table shows the average quality (closer to 1.0, higher the quality) of each algorithm at varying i-bounds and time bounds. i-bd is the i-bound, time is the time bound in seconds, n is the average number of variables, f is the average number of functions, k is the average of the maximum domains size, s is the average of the maximum scope size, w is the average of the constrained induced width.

Instance	i-bd	time (sec)	JGD -ID	WMBE -ID	JGD -EXP	WMBMM -EXP	GDD -MI	WMBMM -MMAP	MBE -ID
rand	1	10E+1	0.016	0.0	0.143	**0.821**	0.025	0.01	0.004
n:56.2 f:56.2		10E+2	0.048	0.025	0.82	**0.821**	0.064	0.01	0.004
k:2.0 s:3.0		10E+3	0.105	0.051	**0.823**	0.821	0.093	0.01	0.004
w:20.4		10E+4	0.774	0.051	**0.823**	0.821	0.105	0.01	0.004
		10E+6	0.775	0.051	**0.823**	0.821	0.106	0.01	0.004
	5	10E+1	0.08	0.0	0.0	**0.826**	0.012	0.057	0.117
		10E+2	0.121	0.096	0.808	**0.826**	0.057	0.057	0.117
		10E+3	0.319	0.273	0.817	**0.826**	0.114	0.057	0.117
		10E+4	0.711	0.273	0.819	**0.826**	0.136	0.057	0.117
		10E+6	0.713	0.273	0.819	**0.826**	0.148	0.057	0.117
	10	10E+1	0.079	0.0	0.062	**0.828**	0.01	0.201	0.325
		10E+2	0.281	0.096	0.803	**0.828**	0.045	0.201	0.325
		10E+3	0.413	0.443	0.81	**0.828**	0.115	0.201	0.325
		10E+4	0.733	0.582	0.812	**0.828**	0.148	0.201	0.325
		10E+6	0.747	0.582	0.812	**0.828**	0.152	0.201	0.325
	15	10E+1	0.079	0.0	0.062	**0.829**	0.01	0.336	0.461
		10E+2	0.252	0.096	0.437	**0.829**	0.045	0.336	0.461
		10E+3	0.487	0.462	0.807	**0.829**	0.098	0.336	0.461
		10E+4	0.637	0.692	0.811	**0.829**	0.137	0.336	0.461
		10E+6	0.764	0.692	0.811	**0.829**	0.123	0.336	0.461
	20	10E+1	0.079	0.0	0.0	**0.729**	0.005	0.51	0.704
		10E+2	0.347	0.196	0.284	**0.829**	0.03	0.512	0.704
		10E+3	0.455	0.494	0.633	**0.829**	0.084	0.512	0.704
		10E+4	0.548	0.781	0.808	**0.829**	0.124	0.512	0.704
		10E+6	0.705	**0.896**	0.811	0.829	0.124	0.512	0.704
BN	1	10E+1	0.0	0.0	0.0	**0.822**	0.0	0.008	0.0
n:84.6 f:84.6		10E+2	0.037	0.0	0.771	**0.822**	0.082	0.008	0.0
k:2.0 s:8.0		10E+3	0.107	0.021	**0.835**	0.822	0.155	0.008	0.0
w:29.2		10E+4	0.704	0.038	**0.854**	0.822	0.176	0.008	0.0
		10E+6	0.704	0.038	**0.854**	0.822	0.177	0.008	0.0
	5	10E+1	0.0	0.0	0.0	**0.852**	0.0	0.019	0.003
		10E+2	0.015	0.0	0.607	**0.852**	0.015	0.019	0.003
		10E+3	0.07	0.053	0.83	**0.852**	0.079	0.019	0.003
		10E+4	0.605	0.116	**0.875**	0.852	0.162	0.019	0.003
		10E+6	0.607	0.116	**0.875**	0.852	0.196	0.019	0.003
	10	10E+1	0.0	0.0	0.0	**0.886**	0.0	0.066	0.052
		10E+2	0.0	0.0	0.59	**0.886**	0.001	0.066	0.052
		10E+3	0.115	0.232	0.796	**0.886**	0.012	0.066	0.052
		10E+4	0.524	0.31	0.868	**0.886**	0.032	0.066	0.052
		10E+6	0.53	0.31	0.873	**0.886**	0.053	0.066	0.052
	15	10E+1	0.0	0.0	0.0	**0.905**	0.0	0.172	0.169
		10E+2	0.0	0.0	0.115	**0.905**	0.0	0.172	0.169
		10E+3	0.056	0.154	0.751	**0.905**	0.001	0.172	0.169
		10E+4	0.417	0.512	0.84	**0.905**	0.017	0.172	0.169
		10E+6	0.494	0.512	0.865	**0.905**	0.023	0.172	0.169
	20	10E+1	0.0	0.0	0.0	**0.625**	0.0	0.335	0.38
		10E+2	0.0	0.0	0.0	**0.925**	0.0	0.335	0.381
		10E+3	0.0	0.0	0.198	**0.925**	0.0	0.335	0.381
		10E+4	0.14	0.3	0.754	**0.925**	0.002	0.335	0.381
		10E+6	0.348	0.655	0.858	**0.925**	0.014	0.335	0.381

Table 4.31: Average Quality of Upper Bounds in RAND and BN Domains. Table shows the average quality (closer to 1.0, higher the quality) of each algorithm at varying i-bounds and time bounds. i-bd is the i-bound, time is the time bound in seconds, n is the average number of variables, f is the average number of functions, k is the average of the maximum domains size, s is the average of the maximum scope size, w is the average of the constrained induced width.

4.6 Case Study on SysAdmin Planning Domain

We next present results from a case study on bounding the MEU task that solves probabilistic planning domain problems. The problem domain we experimented is `SysAdmin` MDP and POMDP instances introduced by Guestrin et al. [2003].

The `Sysadmin` domain asks for finding the MEU and the optimal policy of the problem is as follows. A system administrator maintains a network of s computers. In this network, each machine is is connected to a subset of other machines. The network topology defines a `Sysadmin` problem instance. we model the behavior of each machine as either working or not working by binary random variables X_i, where i ranges over all the machines. The system administrator receives a certain amount of reward for each working machine. The task of the system administrator is to decide which machine to reboot in order to maximize the total sum of rewards. If a machine is rebooted, it will be working with a high probability at the next time step. Every machine has a small probability of failure at each time step. However, if a neighboring machine fails, this probability increases dramatically. These failure probabilities define the transition probability $P(\mathbf{X}_{t+1}|\mathbf{X}_t, a)$, where $\mathbf{X}_t$ is the set of binary random variables at time step t, a is the choice of machine to reboot, and $\mathbf{X}_{t+1}$ is the resulting state of the machines in the next time step.

The `SysAdmin` problem domain was also used in the ICAPS 2011 international planning competition [Sanner et al., 2011]. These probabilistic planning problems are defined in Relational Dynamic Influence Diagram (RDDL) language [Sanner, 2010]. We translated the problem into influence diagrams to allow applying our bounding algorithm.

4.6.1 Experiment Setting

We experimented with 10 SysAdmin MDP and POMDP instances that are used in the ICAPS 2011 international planning competition [Sanner et al., 2011]. Since the original SysAdmin domain by Guestrin et al. [2003] defines finite horizon MDP instances, the finite horizon POMDP instances are generated by modifying a SysAdmin MDP domain by introducing additional variables to capture the observation state of the servers.

Translation to Influence Diagrams We can model finite horizon MDP/POMDP instances in influence diagrams given a fixed time horizon. In our case study, we varied the time horizon from 3 to 10 steps for all problem instances and altogether solved 80 instances in `SysAdmin-MDP` and `SysAdmin-POMDP` instances.

Algorithms In Sections 4.5, we concluded that WMBMM-EXP with i-bound 20 is the overall best algorithm that can quickly generate high quality upper bounds compared with other direct decomposition bounding algorithms. Therefore, we used WBMMM-EXP to generate the upper bounds of the MEU for all `SysAdmin` the translated instances. We evaluated the sampling based online planning algorithms SOGBOFA [Cui and Khardon, 2016] and SNAP [Cui and Khardon, 2019] for both `SysAdmin-MDP` and `SysAdmin-POMDP`. For both online planners, we ran each problem instance 40 times and computed the average of the accumulated rewards over 40 runs. We gave 2 second time limit for each stage of decision making, in each run, for both MDP and POMDP using the same parameter used in [Cui and Khardon, 2019]. Note that this case study does not attempt to solve online planning problems as done by online planning algorithms, which have different objectives. The upper bounds generated by algorithm WMBMM-EXP can be viewed as a heuristic upper bounds of the finite horizon offline planning problems. However, the sampling-based online planners compute Monte-Carlo estimates of the lower bounds of the MEU that can be generated by the optimal online-policy functions.

Performance Measure We measure the performance of the upper bounds (UB) computed by algorithm WMBMM-EXP by the gap between the lower bounds (LB) computed by the online planners, namely, gap $= 1 - \frac{\text{LB}}{\text{UB}}$. The closer the value to 0.0, the better the performance. In the extreme, if both lower bound LB and upper bound UB are the same (e.g., because we found the optimal MEU), the gap is zero. On the other hand, if the upper bound UB is infinity or the lower bound LB is zero, the gap is one.

4.6.2 Case Study Results

Tables 4.32 and 4.33 report the results of evaluating WMBMM-EXP against the online planner SOGBOFA [Cui and Khardon, 2016] for solving factored MDP problems. We see that the problem instances translated to IDs have 79 variables and the constrained induced is 24 for the `mdp1-s10-t3` instance, which is the easiest one in the benchmark. The total number of variables and the constrained induced width are 1,100 and 113 at for the most challenging instance `mdp10-s50-t10`. We see that the induced width increases as the SysAdmin-MDP instances introduce more variables by modeling a large number of machines, yet the increased time horizon doesn't influence the constrained induced width. We used the online planner results as a proxy for the MEU. We observe that the gap between the upper and lower bounds ranges between 4 percent and 34 percent. for the same instance, as the increased time horizon induced larger gaps.

Table 4.34 and 4.35 report the results of evaluating WMBMM-EXP against the online planner SNAP [Cui and Khardon, 2019] for solving factored POMDP problems. Solving a POMDP domain is considered more difficult than solving MDP due to the partial observability. We can see that the problem statistics translated to IDs reflect such difficulties in the constrained induced width, where the value ranges between 60 to 1,000. Although both `SysAdmin-MDP` and `SysAdmin-POMDP` model the same number of machines, `SysAdmin-POMDP` problem instances have a larger number of variables, ranging between 119 to 1620.

Instance	c	d	p	u	k	s	w	utime (sec)	ub wmbmm	ltime (sec)	lb sogbofa	gap $(\frac{ub-lb}{ub})$
mdp1-s10-t3	49	30	49	60	3	6	24	23	30.021	240	28.488	5%
mdp1-s10-t4	62	40	62	80	3	6	24	27	39.552	320	37.775	4%
mdp1-s10-t5	75	50	75	100	3	6	24	42	48.764	400	46.106	5%
mdp1-s10-t6	88	60	88	120	3	6	25	45	58.783	480	55.856	5%
mdp1-s10-t7	101	70	101	140	3	6	24	48	69.343	560	64.625	7%
mdp1-s10-t8	114	80	114	160	3	6	23	65	78.575	640	74.138	6%
mdp1-s10-t9	127	90	127	180	3	6	25	56	89.386	720	81.856	8%
mdp1-s10-t10	140	100	140	200	3	6	23	73	97.365	800	90.806	7%
mdp2-s10-t3	49	30	49	60	3	7	25	29	30.489	240	28.513	6%
mdp2-s10-t4	62	40	62	80	3	7	26	38	39.947	320	37.650	6%
mdp2-s10-t5	75	50	75	100	3	7	27	49	50.541	400	46.063	9%
mdp2-s10-t6	88	60	88	120	3	7	26	51	60.090	480	54.938	9%
mdp2-s10-t7	101	70	101	140	3	7	26	67	69.737	560	63.506	9%
mdp2-s10-t8	114	80	114	160	3	7	27	79	80.100	640	70.919	11%
mdp2-s10-t9	127	90	127	180	3	7	26	89	89.539	720	80.306	10%
mdp2-s10-t10	140	100	140	200	3	7	26	92	100.190	800	87.000	13%
mdp3-s20-t3	92	60	92	120	3	8	44	89	62.963	240	57.388	9%
mdp3-s20-t4	116	80	116	160	3	8	45	93	84.298	320	75.275	11%
mdp3-s20-t5	140	100	140	200	3	8	44	125	105.231	400	91.450	13%
mdp3-s20-t6	164	120	164	240	3	8	44	150	126.496	480	109.213	14%
mdp3-s20-t7	188	140	188	280	3	8	45	168	145.925	560	123.306	16%
mdp3-s20-t8	212	160	212	320	3	8	45	204	168.171	640	139.856	17%
mdp3-s20-t9	236	180	236	360	3	8	45	255	188.524	720	157.019	17%
mdp3-s20-t10	260	200	260	400	3	8	45	225	210.388	800	168.225	20%
mdp4-s20-t3	92	60	92	120	3	10	50	55	63.812	240	57.313	10%
mdp4-s20-t4	116	80	116	160	3	10	46	84	83.977	320	75.375	10%
mdp4-s20-t5	140	100	140	200	3	10	46	121	105.736	400	91.231	14%
mdp4-s20-t6	164	120	164	240	3	10	47	152	126.122	480	108.475	14%
mdp4-s20-t7	188	140	188	280	3	10	47	211	147.848	560	122.369	17%
mdp4-s20-t8	212	160	212	320	3	10	47	175	167.536	640	138.169	18%
mdp4-s20-t9	236	180	236	360	3	10	47	228	190.204	720	151.563	20%
mdp4-s20-t10	260	200	260	400	3	10	47	247	211.048	800	167.750	21%
mdp5-s30-t3	138	90	138	180	3	7	67	115	96.756	240	86.069	11%
mdp5-s30-t4	174	120	174	240	3	7	67	128	130.025	320	110.956	15%
mdp5-s30-t5	210	150	210	300	3	7	67	196	160.059	400	134.563	16%
mdp5-s30-t6	246	180	246	360	3	7	67	211	193.490	480	158.538	18%
mdp5-s30-t7	282	210	282	420	3	7	67	256	226.253	560	181.450	20%
mdp5-s30-t8	318	240	318	480	3	7	67	334	256.302	640	205.013	20%
mdp5-s30-t9	354	270	354	540	3	7	67	217	289.938	720	225.119	22%
mdp5-s30-t10	390	300	390	600	3	7	67	417	322.090	800	240.981	25%

Table 4.32: Case Study Results from SysAdmin-MDP Domains 1. Table shows the upper bound generated by algorithm WMBMM-EXP(20), and the lower bound generated by online planner SOGBOFA [Cui and Khardon, 2016]. c is the number of chance variables, d is the number of decision variables, p is the number of probability functions, u is the number of utility functions, k is the maximum domain size, s is the maximum scope size, w is the constrained induced width, utime is the time in seconds for generating upper bounds, and ltime is the time in seconds for generating lower bounds.

Instance	c	d	p	u	k	s	w	utime (sec)	ub wmbmm	ltime (sec)	lb sogbofa	gap $(\frac{ub-lb}{ub})$
mdp6-s30-t3	135	90	135	180	3	10	67	105	97.103	240	85.919	12%
mdp6-s30-t4	170	120	170	240	3	10	67	136	130.633	320	110.513	15%
mdp6-s30-t5	205	150	205	300	3	10	67	105	163.202	400	134.075	18%
mdp6-s30-t6	240	180	240	360	3	10	68	184	196.229	480	156.488	20%
mdp6-s30-t7	275	210	275	420	3	10	68	242	228.402	560	179.719	21%
mdp6-s30-t8	310	240	310	480	3	10	68	291	261.418	640	202.200	23%
mdp6-s30-t9	345	270	345	540	3	10	68	304	294.124	720	221.438	25%
mdp6-s30-t10	380	300	380	600	3	10	68	361	328.251	800	239.000	27%
mdp7-s40-t3	181	120	181	240	3	8	87	134	128.957	240	113.619	12%
mdp7-s40-t4	228	160	228	320	3	8	88	179	172.620	320	147.944	14%
mdp7-s40-t5	275	200	275	400	3	8	88	213	215.441	400	180.113	16%
mdp7-s40-t6	322	240	322	480	3	8	88	262	260.180	480	209.581	19%
mdp7-s40-t7	369	280	369	560	3	8	88	293	303.037	560	238.450	21%
mdp7-s40-t8	416	320	416	640	3	8	88	367	345.449	640	265.875	23%
mdp7-s40-t9	463	360	463	720	3	8	88	238	389.022	720	287.938	26%
mdp7-s40-t10	510	400	510	800	3	8	88	273	433.281	800	312.813	28%
mdp8-s40-t3	178	120	178	240	3	9	90	183	130.898	240	113.594	13%
mdp8-s40-t4	224	160	224	320	3	9	91	236	174.219	320	146.619	16%
mdp8-s40-t5	270	200	270	400	3	9	91	276	218.239	400	178.906	18%
mdp8-s40-t6	316	240	316	480	3	9	91	354	262.556	480	207.025	21%
mdp8-s40-t7	362	280	362	560	3	9	91	372	307.060	560	234.281	24%
mdp8-s40-t8	408	320	408	640	3	9	91	475	352.263	640	256.281	27%
mdp8-s40-t9	454	360	454	720	3	9	91	342	396.280	720	282.231	29%
mdp8-s40-t10	500	400	500	800	3	9	92	622	439.447	800	300.056	32%
mdp9-s50-t3	227	150	227	300	3	8	107	217	161.292	240	142.931	11%
mdp9-s50-t4	286	200	286	400	3	8	108	241	218.210	320	184.250	16%
mdp9-s50-t5	345	250	345	500	3	8	109	298	274.907	400	222.925	19%
mdp9-s50-t6	404	300	404	600	3	8	109	353	329.248	480	261.319	21%
mdp9-s50-t7	463	350	463	700	3	8	109	49406	383.290	560	296.269	23%
mdp9-s50-t8	522	400	522	800	3	8	109	530	438.786	640	328.550	25%
mdp9-s50-t9	581	450	581	900	3	8	108	370	496.466	720	355.263	28%
mdp9-s50-t10	640	500	640	1000	3	8	109	129	547.757	800	385.263	30%
mdp10-s50-t3	218	150	218	300	3	11	112	149	162.368	240	142.731	12%
mdp10-s50-t4	274	200	274	400	3	11	112	184	217.515	320	183.650	16%
mdp10-s50-t5	330	250	330	500	3	11	113	257	273.332	400	221.450	19%
mdp10-s50-t6	386	300	386	600	3	11	113	19741	327.268	480	257.394	21%
mdp10-s50-t7	442	350	442	700	3	11	113	28013	383.312	560	291.988	24%
mdp10-s50-t8	498	400	498	800	3	11	113	41748	439.826	640	316.600	28%
mdp10-s50-t9	554	450	554	900	3	11	113	34739	494.662	720	345.844	30%
mdp10-s50-t10	610	500	610	1000	3	11	113	58270	549.867	800	364.569	34%

Table 4.33: Case Study Results from SysAdmin-MDP Domains 2. Table shows the upper bound generated by algorithm WMBMM-EXP(20), and the lower bound generated by online planner SOGBOFA [Cui and Khardon, 2016]. c is the number of chance variables, d is the number of decision variables, p is the number of probability functions, u is the number of utility functions, k is the maximum domain size, s is the maximum scope size, w is the constrained induced width, utime is the time in seconds for generating upper bounds, and ltime is the time in seconds for generating lower bounds.

Instance	c	d	p	u	k	s	w	utime (sec)	ub wmbmm	ltime (sec)	lb snap	gap $(\frac{ub-lb}{ub})$
pomdp1-s10-t3	79	30	79	60	3	6	140	25	34.527	240	28.300	18%
pomdp1-s10-t4	102	40	102	80	3	6	160	58	47.723	320	37.125	22%
pomdp1-s10-t5	125	50	125	100	3	6	180	107	61.946	400	46.375	25%
pomdp1-s10-t6	148	60	148	120	3	6	200	168	74.433	480	55.175	26%
pomdp1-s10-t7	171	70	171	140	3	6	60	221	88.583	560	63.125	29%
pomdp1-s10-t8	194	80	194	160	3	6	80	293	104.376	640	71.000	32%
pomdp1-s10-t9	217	90	217	180	3	6	100	439	113.964	720	81.575	28%
pomdp1-s10-t10	240	100	240	200	3	6	120	680	127.463	800	90.200	29%
pomdp2-s10-t3	79	30	79	60	3	7	140	39	34.858	240	28.625	18%
pomdp2-s10-t4	102	40	102	80	3	7	160	64	48.901	320	37.600	23%
pomdp2-s10-t5	125	50	125	100	3	7	180	123	62.902	400	46.300	26%
pomdp2-s10-t6	148	60	148	120	3	7	401	180	77.965	480	55.850	28%
pomdp2-s10-t7	171	70	171	140	3	7	120	289	92.126	560	63.625	31%
pomdp2-s10-t8	194	80	194	160	3	7	160	390	105.696	640	71.725	32%
pomdp2-s10-t9	217	90	217	180	3	7	200	541	119.560	720	81.250	32%
pomdp2-s10-t10	240	100	240	200	3	7	120	716	135.210	800	89.250	34%
pomdp3-s20-t3	155	60	155	120	3	7	281	196	68.619	240	57.375	16%
pomdp3-s20-t4	200	80	200	160	3	7	320	335	96.697	320	74.025	23%
pomdp3-s20-t5	245	100	245	200	3	7	360	1205	124.742	400	92.125	26%
pomdp3-s20-t6	290	120	290	240	3	7	400	2104	152.390	480	107.650	29%
pomdp3-s20-t7	335	140	335	280	3	7	120	2372	176.213	560	122.075	31%
pomdp3-s20-t8	380	160	380	320	3	7	160	3516	204.288	640	137.025	33%
pomdp3-s20-t9	425	180	425	360	3	7	200	9539	231.791	720	151.500	35%
pomdp3-s20-t10	470	200	470	400	3	7	240	7900	256.678	800	168.725	34%
pomdp4-s20-t3	152	60	152	120	3	8	280	274	72.953	240	56.675	22%
pomdp4-s20-t4	196	80	196	160	3	8	320	603	101.970	320	74.625	27%
pomdp4-s20-t5	240	100	240	200	3	8	360	1096	131.511	400	90.375	31%
pomdp4-s20-t6	284	120	284	240	3	8	601	2410	161.957	480	108.750	33%
pomdp4-s20-t7	328	140	328	280	3	8	180	3223	191.321	560	122.850	36%
pomdp4-s20-t8	372	160	372	320	3	8	241	4170	219.081	640	138.550	37%
pomdp4-s20-t9	416	180	416	360	3	8	301	5662	253.065	720	152.625	40%
pomdp4-s20-t10	460	200	460	400	3	8	240	7502	280.402	800	164.978	41%
pomdp5-s30-t3	225	90	225	180	3	10	421	947	106.395	240	85.900	19%
pomdp5-s30-t4	290	120	290	240	3	10	481	2388	144.423	320	111.425	23%
pomdp5-s30-t5	355	150	355	300	3	10	541	2857	185.908	400	136.050	27%
pomdp5-s30-t6	420	180	420	360	3	10	600	6992	229.701	480	159.750	30%
pomdp5-s30-t7	485	210	485	420	3	10	180	10074	273.717	560	182.975	33%
pomdp5-s30-t8	550	240	550	480	3	10	240	14616	311.690	640	199.825	36%
pomdp5-s30-t9	615	270	615	540	3	10	300	33306	354.370	720	221.150	38%
pomdp5-s30-t10	680	300	680	600	3	10	361	34181	395.413	800	235.050	41%

Table 4.34: Case Study Results from SysAdmin-MDP Domains 1. Table shows the upper bound generated by algorithm WMBMM-EXP(20), and the lower bound generated by online planner SNAP [Cui and Khardon, 2019]. c is the number of chance variables, d is the number of decision variables, p is the number of probability functions, u is the number of utility functions, k is the maximum domain size, s is the maximum scope size, w is the constrained induced width, utime is the time in seconds for generating upper bounds, and ltime is the time in seconds for generating lower bounds.

Instance	c	d	p	u	k	s	w	utime (sec)	ub wmbmm	ltime (sec)	lb snap	gap $(\frac{ub-lb}{ub})$
pomdp6-s30-t3	225	90	225	180	3	9	420	1297	110.580	240	85.475	23%
pomdp6-s30-t4	290	120	290	240	3	9	480	1820	155.242	320	110.550	29%
pomdp6-s30-t5	355	150	355	300	3	9	540	5622	198.051	400	136.350	31%
pomdp6-s30-t6	420	180	420	360	3	9	801	7690	244.680	480	156.625	36%
pomdp6-s30-t7	485	210	485	420	3	9	240	13814	288.301	560	178.975	38%
pomdp6-s30-t8	550	240	550	480	3	9	320	18621	331.830	640	197.100	41%
pomdp6-s30-t9	615	270	615	540	3	9	400	25551	382.746	720	214.725	44%
pomdp6-s30-t10	680	300	680	600	3	9	360	44681	420.173	800	229.125	45%
pomdp7-s40-t3	301	120	301	240	3	8	561	2122	137.679	240	114.125	17%
pomdp7-s40-t4	388	160	388	320	3	8	640	6278	195.345	320	147.500	24%
pomdp7-s40-t5	475	200	475	400	3	8	721	10556	247.624	400	180.625	27%
pomdp7-s40-t6	562	240	562	480	3	8	800	20195	304.780	480	208.225	32%
pomdp7-s40-t7	649	280	649	560	3	8	240	25459	361.472	560	235.250	35%
pomdp7-s40-t8	736	320	736	640	3	8	320	64929	414.487	640	263.550	36%
pomdp7-s40-t9	823	360	823	720	3	8	400	67101	473.587	720	287.725	39%
pomdp7-s40-t10	910	400	910	800	3	8	480	101257	526.012	800	304.575	42%
pomdp8-s40-t3	295	120	295	240	3	11	560	2927	145.588	240	113.850	22%
pomdp8-s40-t4	380	160	380	320	3	11	640	12636	207.960	320	147.900	29%
pomdp8-s40-t5	465	200	465	400	3	11	720	15902	267.996	400	179.475	33%
pomdp8-s40-t6	550	240	550	480	3	11	300	27842	327.941	480	204.675	38%
pomdp8-s40-t7	635	280	635	560	3	11	400	71118	387.330	560	233.100	40%
pomdp8-s40-t8	720	320	720	640	3	11	501	50991	447.594	640	261.150	42%
pomdp8-s40-t9	805	360	805	720	3	11	600	68209	508.135	720	280.800	45%
pomdp8-s40-t10	890	400	890	800	3	11	480	88919	568.487	800	298.825	47%
pomdp9-s50-t3	371	150	371	300	3	10	300	7660	177.597	240	141.575	20%
pomdp9-s50-t4	478	200	478	400	3	10	400	20747	244.415	320	182.775	25%
pomdp9-s50-t5	585	250	585	500	3	10	501	27369	315.351	400	223.000	29%
pomdp9-s50-t6	692	300	692	600	3	10	600	69085	384.860	480	257.350	33%
pomdp9-s50-t7	799	350	799	700	3	10	701	107660	454.910	560	290.425	36%
pomdp9-s50-t8	906	400	906	800	3	10	800	224727	528.582	640	323.575	39%
pomdp9-s50-t9	1013	450	1013	900	3	10	900	207883	599.923	720	348.325	42%
pomdp9-s50-t10	1120	500	1120	1000	3	10	701	231511	668.883	800	372.750	44%
pomdp10-s50-t3	371	150	371	300	3	10	400	6249	180.257	240	141.950	21%
pomdp10-s50-t4	478	200	478	400	3	10	500	14333	253.043	320	184.925	27%
pomdp10-s50-t5	585	250	585	500	3	10	600	36078	331.108	400	221.775	33%
pomdp10-s50-t6	692	300	692	600	3	10	200	66848	409.456	480	258.525	37%
pomdp10-s50-t7	799	350	799	700	3	10	60	121312	480.291	560	290.300	40%
pomdp10-s50-t8	906	400	906	800	3	10	80	116597	563.324	640	321.425	43%
pomdp10-s50-t9	1013	450	1013	900	3	10	100	290003	633.134	720	346.500	45%
pomdp10-s50-t10	1120	500	1120	1000	3	10	300	244446	707.226	800	375.900	47%

Table 4.35: Case Study Results from SysAdmin-MDP Domains 2. Table shows the upper bound generated by algorithm WMBMM-EXP(20), and the lower bound generated by online planner SNAP [Cui and Khardon, 2019]. c is the number of chance variables, d is the number of decision variables, p is the number of probability functions, u is the number of utility functions, k is the maximum domain size, s is the maximum scope size, w is the constrained induced width, utime is the time in seconds for generating upper bounds, and ltime is the time in seconds for generating lower bounds.

Summary Our results show that the gap between the upper and the lower bounds is larger compared with `SysAdmin-MDP` problems, ranging between 18 percent to 47 percent. We see that the upper bound for the most challenging problem instance `pomdp10-s50-t10` is still within a factor of 2 of the unknown MEU.

4.7 Conclusion

We presented empirical evaluation results of our proposed algorithms JGD-ID, WMBE-ID, JGD-EXP, and WMBMM-EXP along with earlier translation based approaches and non-variational bounding algorithm MBE-ID. In the experiment, we evaluated algorithms in four synthetic benchmark sets including factored finite horizon MDP/POMDP domains, a random network instances, and instances obtained by converting BNs. When problem instances are easy or moderately difficult, the bounding schemes JGD-ID and WMBE-ID that are extending the valuation algebra generated tight upper bounds. However, as the constrained induced width grows in more challenging problem instances, the bounding schemes JGD-EXP and WMBMM-EXP dominated all other approaches. Comparing JGD-EXP and WMBMM-EXP, JGD-EXP generated the tightest upper bounds when algorithms were given low i- bounds, 1 or 5 and given enough time for iterative algorithms to converge. On the other hand, WMBMM-EXP is the best performing algorithm when using higher i-bounds, 15 or 20 or we are limiting time bounds shorter than 1,000 seconds. We next presented the result of a case study on `SysAdmin MDP/POMDP` from probabilistic planning domains. We see that our bounding algorithms were able to generate upper bounds within a factor of 2 from unknown MEUs in the worst case, and the overall upper bounds are close to the lower bounds within 4 percent to 47 percent gap from the lower bounds estimated by online planners, which shows a potential of applying our bounding algorithms as a heuristic generator for probabilistic planning.

Chapter 5

Conclusion

The main topic we addressed in this dissertation was the development of algorithms that generate upper bounds on the maximum expected utility in influence diagrams.

We first considered a bounding scheme for the MEU task that utilizes the valuation algebra framework. Our bounding method generalizes two powerful variational decomposition bounds, namely, generalized dual decomposition bounds and weighted mini-bucket bounds to the maximum expected utility task. Specifically, we extend the powered summation operation and the decomposition bounds to valuation algebra, provide two optimization formulations yielding two types of message passing architectures, called JGD-ID and WMBE-ID.

We next focused on another bounding scheme for influence diagrams, that does not use the valuation algebra. In this scheme, we utilized the Jensen's inequality applied to exponential functions and developed a bounding method that allows for the reuse of powerful algorithms designed for the marginal MAP task. This was enabled through an MEU formulation that uses an exponential representation of the utility functions and lead to another two message passing algorithms, called JGD-EXP and WBMMM-EXP.

Directions for Future Research

The work presented in this dissertation opens up new directions for future research of inference and search algorithms for the MEU task in influence diagrams.

Short-term Research

Inference algorithms for the MEU task The approach presented in this dissertation extends variational decomposition bounds to the task of maximizing the expected utility in influence diagrams. In bounding schemes using valuation algebra, our reparameterization scheme leads to non-convex optimization problems. The empirical evaluation results show that the naive first-order optimization algorithm often fails to tighten the upper bounds, or it needs a longer time to converge. In future work, we can adopt more advanced optimization routines that recently became available to mitigate the issue by optimizing a large number of non-convex settings.

Another orthogonal direction is to explore bounding schemes under new model decomposition frameworks [Pralet et al., 2006, Lee, 2020], which capture the local structure of influence diagrams more tightly and possibly leading to more efficient algorithms.

Search algorithms for the MEU task One major bottleneck in advancing heuristic search algorithms for the MEU task has been the lack of good enough, yet efficient heuristic generators. It appears that for other inference tasks, AND/OR search augmented with partition-based heuristic is able to guide the performance of the various search strategies quite well [Dechter and Mateescu, 2007, Marinescu and Dechter, 2009, Otten and Dechter, 2012, Flerova et al., 2016, Otten and Dechter, 2017, Marinescu et al., 2014, 2017, 2018a]. In a similar vein, one could adopt the bounding schemes presented in this dissertation to generate heuristics for search, in solving influence diagrams. In particular, algorithm WMBMM-EXP can be

well-suited for generating admissible heuristics for the AND/OR search spaces [Dechter and Mateescu, 2007, Marinescu, 2010], as the same type of algorithm was successfully applied to the marginal MAP as shown in [Marinescu et al., 2018b]. Indeed, we demonstrated the tightness of the bounds by algorithm WMBMM-EXP on large scale planning instances in the case study showing its promise.

Long-term Research

Limited memory influence diagrams [Lauritzen and Nilsson, 2001] offer a graphical model framework for sequential decision-making under imperfect recall. Namely, a decision-maker forgets the history while making decisions. When multiple agents are competing or co-operating to achieve their goals, multi-agent influence diagrams [Koller and Milch, 2003, Detwarasiti and Shachter, 2005] or networks of influence diagrams [Gal and Pfeffer, 2008] provide a graphical model for the inference task, maximizing the maximum expected utility of participating agents. Although graphical models are available for modeling various decision-making scenarios, we rarely see the works that address the inference task defined over more complex influence diagrams, even for exact algorithms. As a long-term research agenda, we plan to extend the graphical model inference frameworks to those more complex influence diagrams modeling more practical and general decision-making scenarios.

Bibliography

M. Botvinick and M. Toussaint. Planning as inference. *Trends in Cognitive Sciences*, 16(10): 485–488, 2012.

C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

J. F. Carriger and M. G. Barron. Minimizing risks from spilled oil to ecosystem services using influence diagrams: The deepwater horizon spill response. *Environmental Science & Technology*, 45(18):7631–7639, 2011.

D. Chandler. *Introduction to Modern Statistical Mechanics*. Mechanics. Oxford University Press, Oxford, UK, 1987.

H. Cui and R. Khardon. Online symbolic gradient-based optimization for factored action mdps. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 3075–3081, 2016.

H. J. Cui and R. Khardon. Sampling networks and aggregate simulation for online pomdp planning. In *Advances in Neural Information Processing Systems*, pages 9218–9228, 2019.

A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational intelligence*, 5(2):142–150, 1989.

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

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

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

R. Dechter and R. Mateescu. AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2-3):73–106, 2007.

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

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

I. Ekeland. On the variational principle. *Journal of Mathematical Analysis and Applications*, 47(2):324–353, 1974.

T. Everitt, R. Kumar, V. Krakovna, and S. Legg. Modeling AGI safety frameworks with causal influence diagrams. *arXiv preprint arXiv:1906.08663*, 2019.

N. Flerova, R. Marinescu, and R. Dechter. Searching for the M best solutions in graphical models. *Journal of Artificial Intelligence Research*, 55:889–952, 2016.

Y. Gal and A. Pfeffer. Networks of influence diagrams: a formalism for representing agents' beliefs and decision-making processes. *Journal of Artificial Intelligence Research*, 33:109–147, 2008.

R. Gillespie. Principles of mathematical analysis. *The Mathematical Gazette*, 39(329):258–259, 1955.

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

J. M. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. *Unpublished Manuscript*, 1970.

E. A. Hansen and R. Zhou. Anytime heuristic search. *Journal of Artificial Intelligence Research*, 28(1):267–297, 2007.

G. Hardy, J. Littlewood, and G. Pólya. *Inequalities*. Cambridge Mathematical Library. Cambridge University Press, 1952.

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

R. A. Howard and J. E. Matheson. Influence diagrams. *The Principles and Applications of Decision Analysis*, 2:719–763, 1984.

A. Ihler, N. Flerova, R. Dechter, and L. Otten. Join-graph based cost-shifting schemes. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, pages 397–406, 2012.

F. Jensen, F. V. Jensen, and S. L. Dittmer. From influence diagrams to junction trees. In *Proceedings of the 10th International Conference on Uncertainty in Artificial Intelligence*, pages 367–373, 1994.

J. L. W. V. Jensen et al. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30:175–193, 1906.

H. J. Kappen, V. Gómez, and M. Opper. Optimal control as a graphical model inference problem. *Machine Learning*, 87(2):159–182, 2012.

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

K. Kask, A. Gelfand, L. Otten, and R. Dechter. Pushing the power of stochastic greedy ordering schemes for inference in graphical models. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 54–60. AAAI Press, 2011.

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

N. Koenig and M. J. Matarić. Robot life-long task learning from human demonstrations: A Bayesian approach. *Autonomous Robots*, 41(5):1173–1188, 2017.

J. Kohlas and P. P. Shenoy. Computation in valuation algebras. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, pages 5–39. Springer, 2000.

J. Kohlas and N. Wilson. Semiring induced valuation algebras: Exact and approximate local computation algorithms. *Artificial Intelligence*, 172(11):1360–1399, 2008.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45(1):181–221, 2003.

N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3): 531–552, 2010.

R. E. Kopp. Pontryagin maximum principle. In *Mathematics in Science and Engineering*, volume 5, pages 255–279. Elsevier, 1962.

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

S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

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

S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society Series B-Methodological*, 50:415–448, 1988.

J. Lee. Submodel decomposition for solving limited memory influence diagrams (student abstract). In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020.

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

Q. Liu and A. Ihler. Bounding the partition function using Hölder's inequality. In *Proceedings of the 28th International Conference on Machine Learning*, pages 849–856, 2011.

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

R. Marinescu. A new approach to influence diagrams evaluation. In *Research and Development in Intelligent Systems XXVI*, pages 107–120. Springer, 2010.

R. Marinescu and R. Dechter. AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1457–1491, 2009.

R. Marinescu, R. Dechter, and A. Ihler. AND/OR search for marginal MAP. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, pages 563–572, 2014.

R. Marinescu, J. Lee, A. T. Ihler, and R. Dechter. Anytime best+ depth-first search for bounding marginal map. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3775–3782, 2017.

R. Marinescu, R. Dechter, and A. T. Ihler. Stochastic anytime search for bounding marginal map. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 5074–5081, 2018a.

R. Marinescu, J. Lee, R. Dechter, and A. Ihler. AND/OR search for marginal map. *Journal of Artificial Intelligence Research*, 63:875–921, 2018b.

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

D. D. Mauá. Equivalences between maximum a posteriori inference in bayesian networks and maximum expected utility computation in influence diagrams. *International Journal of Approximate Reasoning*, 68(C):211–229, 2016.

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

T. P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, page 362–369, 2001.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

T. D. Nielsen and F. V. Jensen. Sensitivity analysis in influence diagrams. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 33(2):223–234, 2003.

T. D. Nielsen and F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer Science & Business Media, 2009.

D. Nilsson and M. Hohle. Computing bounds on expected utilties for optimal policies based on limited information. *Dinar Research Report*, 2001.

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

M. Opper and D. Saad. *Advanced Mean Field Methods: Theory and Practice*. MIT press, 2001.

L. Otten and R. Dechter. Anytime AND/OR depth-first search for combinatorial optimization. *AI Communications*, 25(3):211–227, 2012.

L. Otten and R. Dechter. AND/OR branch-and-bound on a computational grid. *Journal of Artificial Intelligence Research*, 59:351–435, 2017.

D. K. Owens, R. D. Shachter, and R. F. Nease Jr. Representation and analysis of medical decision problems with influence diagrams. *Medical Decision Making*, 17(3):241–262, 1997.

J. Pearl. Heuristics: Intelligent search strategies for computer problem solving. *Addision Wesley*, 1984.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

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

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

R. T. Rockafellar and R. J.-B. Wets. *Variational Analysis*, volume 317. Springer Science & Business Media, 2009.

S. J. Russell and P. Norvig. *Artificial Intelligene - A Modern Approach*. Pearson Education London, 2010.

S. Sanner. Relational dynamic influence diagram language RDDL: Language description. *Unpublished Manuscript*, 2010.

S. Sanner, S. Yoon, and T. Walsh. *ICAPS 2011 International Probabilistic Planning Competition*, 2011. `http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/`.

P. P. Shenoy. A valuation-based language for expert systems. *International Journal of Approximate Reasoning*, 3(5):383–411, 1989.

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

T. Sommestad, M. Ekstedt, and L. Nordstrom. Modeling security of power communication systems using defense graphs and influence diagrams. *IEEE Transactions on Power Delivery*, 24(4):1801–1808, 2009.

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

E. Todorov. Optimal control theory. *Bayesian Brain: Probabilistic Approaches to Neural Coding*, pages 269–298, 2006.

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

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudo-moment matching. In *Proceedings of the 9th International Conference on Artificial Intelligence and Statistics*, 2003.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems*, pages 689–695, 2001.

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