UNIVERSITY OF CALIFORNIA,
IRVINE

Exploiting Factor and State Space Symmetry for Inference in Graphical Models

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Nicholas Gallo

Dissertation Committee:
Prof. Alexander Ihler, Chair
Prof. Rina Dechter
Prof. Erik Sudderth

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

I would like to begin by thanking my advisor Alex Ihler for his constant support and help throughout my PhD and while writing this thesis. He always has great suggestions to improve my work, deep insights into graphical models, and a strong focus on fundamental problems. I enjoyed many meetings with him, his good spirit, sense of humor, and his genuine passion for teaching and research.

I would also like to thank Rina Dechter and Erik Sudderth for their time reading my thesis and for serving on my defense and advancement committees. I was fortunuate to have met and interacted with both of them at UCI. I was fortunate to share weekly meetings with Rina and her group for many years. I learned a lot from this collaboration including many new perspectives on graphical models and constraint satisfaction problems. Erik joined the faculty at UCI midway through my PhD and always proved helpful and extremely knowledgeble in my interactions with him. His careful attention to detail during my earlier presentations helped improve my work in this thesis.

My graduate work and this thesis have been supported in part by grants from the National Science Foundation (DMS-0928427, CCF-1331915, IIS-1254071, IIS-1526842, IIS-1816365, and IIS-2008516) and DARPA (contracts W911NF-18-C-0015 and FA8750-14-C-0011).

CURRICULUM VITAE

Nicholas Gallo

EDUCATION

Doctor of Philosophy in Computer Science — **2020**
University of California, Irvine — *Irvine, California*

Bachelor of Science in Computer Science — **2009**
Cornell University — *Ithaca, New York*

RESEARCH EXPERIENCE

Graduate Research Assistant — **2013 – 2020**
University of California, Irvine — *Irvine, California*

PUBLICATIONS

N. Gallo and A. T. Ihler, "Lifted Weighted mini-bucket," *Advances in Neural Information Processing Systems*, pages 10329-10337, 2018b.

N. Gallo and A. Ihler, "Lifted generalized dual decomposition," *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018a.

ABSTRACT OF THE DISSERTATION

Exploiting Factor and State Space Symmetry for Inference in Graphical Models

By

Nicholas Gallo

Doctor of Philosophy in Computer Science

University of California, Irvine, 2020

Prof. Alexander Ihler, Chair

Probabilistic graphical models provide a powerful framework for representing and reasoning about complex systems composed of many small overlapping subsystems. Key problems in graphical models can be formulated as probabilistic inference queries, such as computing the most likely configuration or the marginal probability of a random variable. Since these problems are intractable in general, a large class of approximate inference algorithms that exploit the factorization structure of graphical models have been developed.

In addition to the classic factorization structure, many graphical models also possess *symmetric* structure where groups of objects in the model are indistinguishable. Two types of symmetry arise regularly in practice: first, a model with *state space symmetry* contains factors with groups of states that have identical values; second, a model with *factor symmetry* contains groups of factors that have identical factor tables.

Although efficient inference algorithms for models with perfect symmetry have been well developed, most real problems do not contain perfect symmetry. Often, for example, a model with a symmetric substructure is perturbed by asymmetric evidence factors. Consequently, there is a great need for methods that accurately approximate the desired inference quantities

using symmetric inference terms. While a few methods to address this problem exist, many are heuristic or address only certain aspects of the problem.

The goal of this thesis is to develop more intelligent ways to perform inference in graphical models with approximate symmetry. Our central strategy is to force groups of parameters in a variational inference relaxation to be symmetric. These symmetry groups are iteratively broken in a controlled manner to capture problem asymmetries more accurately; this procedure gives rise to a flexible class of coarse-to-fine inference algorithms. Furthermore, by using intelligently structured parameter symmetries, we are able to construct symmetric high-order inference terms which are often necessary to obtain high accuracy inference estimates. We develop algorithms both for models with state space symmetry and for models with factor symmetry, highlighting the deep similarities between these two classes of problems which has not been widely appreciated before.

Chapter 1

Introduction

Many problems in science and engineering require modeling a large system that is composed of many small interacting subsystems and characterized by uncertainty. Important problem tasks often require analyzing of the global behavior of the system which arises from the complex interactions of its many overlapping subsystems. A classic example is the Ising model [McCoy and Wu, 2014] in statistical physics which models the spins of atoms laid out in a two-dimensional lattice and encodes the probabilistic constraint that two neighboring atoms likely have the same spin. Typical problem tasks include computing the configuration of atomic spins with the lowest energy or computing a physical quantity, such as the entropy, of the system. Another classic example is social network analysis where the system is composed of many unstructured connections between pairs of people (friends) and encodes similar behavior of two friends. For example, an election prediction model may encode the probabilistic constraint that two friends probably vote for the same candidate and the desired task is to predict who each person will vote for or to predict who will win the election.

1.1 Probabilistic Graphical Models (PGMs)

Probabilistic graphical models (PGMs) provide a powerful framework for representing and reasoning about such problems. A PGM is composed of n discrete random variables (RVs)

$(X_1, \ldots, X_n)$ where each RV X_v takes a value in a discrete set $\mathcal{D}_v$ called its domain. The PGM represents the exponentially large joint configuration of RVs (corresponding to all possible combinations of values for the n RVs) efficiently via a collection of small *factors*. A *factor* $\theta_\alpha : \mathcal{D}_\alpha \to \mathbb{R}$ is a function with domain $\mathcal{D}_\alpha = \times_{v \in \alpha} \mathcal{D}_v$, the set of all configurations of a subset of RVs $\alpha \subseteq \mathcal{V}$, that assigns a score to each configuration where a high score encourages the configuration and a low score discourages the configuration. The global joint distribution is a product of local factors,

$$p(X = x;\ \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\ \exp(\boldsymbol{\theta}(x)) \qquad \text{where} \qquad \boldsymbol{\theta}(x)\ =\ \sum_{\alpha \in \mathcal{I}}\ \theta_\alpha(x_\alpha),$$

$\mathcal{I}$ contains all the subsets α associated with the factors, and the partition function, $Z(\boldsymbol{\theta}) = \sum_{x_n}\ \cdots \sum_{x_1}\ \exp(\boldsymbol{\theta}(x))$ normalizes the distribution. In the Ising model, each RV represents the spin of an atom and each factor encourages labelings where pairs of neighboring atoms in the lattice take the same spin. In our election modeling example, each RV represents a person's vote for a candidate and each factor encourages labelings where friends vote for the same candidate.

Many important problem tasks can be formulated as a *probabilistic inference query* on the PGM. The two most popular inference tasks are *MAP inference* and *sum-inference*. MAP inference computes the most likely global configuration $x^* = \arg\max_{x \in \mathcal{D}_\mathcal{V}} \boldsymbol{\theta}(x)$, while sum-inference tasks include computing the partition function $Z(\boldsymbol{\theta})$ and computing the marginal probability $p(x_v) = \sum_{x_{\mathcal{V} \setminus v}} p(x_\mathcal{V})$ of a RV $v \in \mathcal{V}$.

The inference tasks involve aggregating (via maximization or summation) the value of the distribution's exponentially large number of states. Although a naïve computation of the inference quantities is intractable, the factorization structure of the PGM can often be exploited to perform the inference computation more efficiently. Unfortunately, this efficient exact inference computation is also intractable for large problems. Consequently, much work

has been dedicated to the development of efficient *approximate inference* algorithms. For example, some *variational approximate inference algorithms* (which we discuss in more detail later) relax consistencies in the global inference problem to produce a collection of tractable local inference problems (i.e., over a controlled number of RVs). A set of *messages*, representing local approximate inference quantities, are exchanged between the subproblems.

1.2 Symmetric Models

The factorization (or equivalently, graph structure) of a PGM is an important property that enables efficient representation and reasoning about large probability distributions (either exactly or approximatly). However, many PGMs also contain additional structure that can be exploited to attain more efficient reasoning algorithms. In this thesis, we study models that have *symmetric* structure which, broadly, means that the model contains groups of objects whose items are indistinguishable from the others in the group. Two types of symmetry arise in practice:

Factor Symmetry: In this case, groups of factors have identical factor tables, i.e., encode the same relationship but within different sets of variables. For example, in the Ising model, the strength of the interaction between two neighboring atoms is the same throughout the system, and doesn't depend on the absolute location of the atoms (a form of translational invariance). Another class of examples is probabilistic first-order logic networks, such as Markov Logic Networks (MLNs) [Richardson and Domingos, 2006], that encode rules with a "for all" relationship and can be used, for example, to represent the rule in our election modeling example "for all pairs of people (A,B), if A and B are friends, then A and B probably vote for the same candidate".

State Space Symmetry: In this case, groups of states within a factor have identical value.

A classic example is a cardinality potential, whose value depends only on the number of RVs that take a particular state (thus avoiding the exponential cost required to represent *which* RVs take the state). For example, the task of assigning a semantic label to each pixel in an image may encode a rule such as "an image patch that contains a large fraction of `ocean` pixels may also contain a large fraction of `boat` pixels, but it should contain no `car` pixels (with high probability)".

Most algorithms that were designed for exploiting factor symmetry evolved separately from algorithms designed to exploit state space symmetry. Nevertheless, there are deep similarities between approximate inference algorithms for these two classes of models. Much of the presentation in this thesis is aimed at highlighting these similarities, which are not widely noted in the literature. Recognition of these similarities inspired several of the key insights in this thesis (discussed at the end of this introduction chapter) and also paves the way for future work that exploits factor symmetry and state space symmetry simultaneously. These complementary symmetries arise naturally in many problems. For example, a MLN model has considerable factor symmetry, while its factors also take the form of probabilistic truth tables with significant state space symmetry. Similarly, in computer vision problems we may wish to apply counting factor constraints (with state symmetry) to many patches of the image simultaneously (factor symmetry).

1.2.1 Lifted Inference

Now that we have seen how symmetries are useful in modeling, we turn our attention to the problem of performing *lifted inference* that exploits symmetry in inference computations (in this setting, standard inference that does not exploit symmetry is called *ground inference*). The basic idea of lifted inference is to first identify exact symmetries that would arise if a ground inference procedure were run, then perform efficient inference on a *lifted graph* which represents symmetry groups as a single node. In the *ideal* case, the model contains

groups of symmetric objects that perfectly align with each other, and hence the inference quantities also possess these symmetries. For example, in MLN models without evidence we can perform reasoning by exchanging messages represented as probabilistic first-order formulas, rather than passing identical messages associated with each ground object.

Over the past decade, lifted inference procedures have been developed for a large number of popular ground inference algorithms such as belief propagation and variable elimination [e.g., Singla and Domingos, 2008, Poole, 2003]. Unfortunately the ideal case, in which the model is composed of perfectly aligned symmetry groups, rarely occurs in practice. In MLN models, for example, we often observe different information about each individual ground object (e.g., that some pairs of people are friends) and hence no two objects can be treated identically during inference. In this case, lifted inference deteriorates into ground inference, negating all the potential computational gain of the symmetric representation. This effect of breaking model symmetries is sometimes referred to as *shattering*.

1.2.2 Mitigating the Effects of Symmetry Shattering

Because of the shattering effect, in any practical scenario the key question is how to accurately approximate the desired inference quantities using symmetric inference terms. Although a considerable amount of effort has gone into developing solutions to this problem, it still remains largely open.

In particular, many previous works that successfully control the computational cost of the symmetric approximation incur error that is difficult to analyze. For example, projected message passing algorithms compute the exact (relaxed) inference message, then project the message to a tractable functional form (for both factor symmetry [e.g., Singla et al., 2014] and state space symmetry [e.g., Gogate and Domingos, 2013]). Unfortunately, the error incurred by the projection is difficult to analyze, as is the error incurred as it propagates through the

graph [Ihler et al., 2005]. Another class of methods forms an approximation to the model that has more symmetry than the model; exact lifted inference is then performed on this *over-symmetric model* [e.g., Venugopal and Gogate, 2014a]. These methods must choose the approximating structure before inference (which is often difficult to do accurately) and, moreover, it is not clear how inference estimates on the over-symmetric model relate to inference estimates on the original model.

An emerging class of methods performs inference in a coarse-to-fine manner. The basic idea is to iterate between two steps: first, solve a coarse (low-cost) approximation to the inference problem; then use this solution to instantiate a finer problem with higher cost and accuracy. For example, Habeeb et al. [2017] develops a coarse-to-fine inference algorithm for stereo vision. First, pixels are grouped together (via a heuristic metric) and forced to take the same disparity label; then the constrained inference problem is solved by reasoning over pixel groups (rather than individual pixels). A group of pixels is then split into finer groups if the current inference results suggest that pixels in the subgroups should in fact take different disparity labels. The coarse-to-fine inference algorithms are appealing since they control the computational cost of the approximation while also maintaining a concrete connection to the original problem.

1.2.3 Models with No Symmetries

In the framework above, where the model contains symmetric substructures perturbed by asymmetric factors, the utility of incorporating symmetric functions to approximate inference quantities is obvious. However, less obviously, models with no symmetry at all can still benefit from incorporating symmetric inference terms. For example, Gogate and Domingos [2013] use factors with state space symmetry to approximate messages in PGMs with dense factor tables (i.e., factor tables with no symmetry). Another work, Sontag et al. [2009], begins with an inference relaxation on dense factor tables, then tightens it by adding large-

scope factors that use a coarse resolution of each RV domain (a special form of state space symmetry).

In models with no symmetry, we expect symmetric inference terms to be useful if they capture one of the following properties:

Nearly identical value: A set of states has nearly identical value and can be grouped together while incurring small error.

Low value: A set of states has large relative disparity but low value and can be grouped into one "low-scoring" group. Thus, the approximation incurs error only in low probability mass regions of the distribution which have little or no effect on MAP inference or sum-inference queries, which depend mostly (or only) on the most probable states. Note that this case generalizes the popular idea of removing low-scoring states from the problem.

1.3 Our Approach

The central goal of this thesis is to develop more intelligent ways to represent and reason using (approximate) symmetries in PGMs. Our approach operates within a variational inference framework which we believe is ideal for controlling the symmetry in the inference approximation. The central idea of variational inference is to cast the inference problem as an (intractable) optimization problem. In standard PGMs, a tractable approximation to the optimization problem is obtained by restricting the variational parameters to a factored form. For symmetric PGMs, our central idea is to restrict the variational parameters to have a *symmetric* form. This allows us to develop inference approximations with the following two crucial properties:

Coarse-to-Fine Inference: The symmetry restrictions on the variational parameters can be broken in a controlled manner. This lets us generate coarse-to-fine approximations of increasingly fine resolution, and thus of increasing cost and increasing accuracy.

High-Order Terms: In classic (ground) inference, it is often necessary to augment the model factors with high-order inference terms to attain high accuracy. In symmetric problems, the controlled symmetry restrictions on the variational parameters allow us to augment the model with symmetric high-order inference terms and ensure message passing can be carried out without shattering.

The first half our of novel work (in Chapter 4 and Chapter 5) operates on models with symmetric factors perturbed by asymmetric evidence. The second half of our novel work (in Chapter 6) operates on classic PGMs (with no symmetry) and tightens the inference relaxation by adding factors with state space symmetry using a sophisticated grouping of states. As stated above, although these two problem classes are very different, the methods we develop for exploiting and representing symmetry are similar. The following paragraphs provide an outline of and specific contributions of each chapter of the thesis.

Chapter 2: Background: Standard PGMs We begin by providing background on classic PGMs (without symmetry) and inference problems. We present background on variational approximate inference algorithm that are relevant to our later development, such as the generalized dual decomposition algorithm and weighted mini-bucket elimination algorithm for sum-inference problems. Lastly, we discuss anytime inference which interleaves inference on a fixed graph with transitions to a graph with higher cost and accuracy; each of the coarse-to-fine inference algorithms for symmetric problems developed in subsequent chapters is an instance of anytime inference.

Chapter 3: Background: Symmetric PGMs This chapter presents background on PGMs with symmetric structure. We first precisely define symmetry, then describe popular symmetric modeling frameworks such as Markov Logic Networks (MLNs). We then describe lifted inference (for exploiting model symmetries) and the shattering problem (whereby inference breaks model symmetries), first for models with factor symmetries then for models with state space symmetries. We then discuss methods that have been developed for coping with asymmetry; some of the ideas underlying these methods form the basis for our work in subsequent chapters. For example, coarse-to-fine inference allows symmetry groups to be selected dynamically, and models with domain-clustering (a special form of symmetry) allow high-order inference terms to be constructed easily.

Chapter 4: Lifted Generalized Dual Decomposition This chapter presents our first algorithm for performing inference in models with symmetric factors perturbed by asymmetric evidence factors. The central idea is to force variational parameters in a generalized dual decomposition approximation to be symmetric, then gradually break these symmetries to obtain a sequence of coarse-to-fine inference estimates. Our contributions include:

- Strong control over inference symmetries which are gradually *split* in a coarse-to-fine manner. The symmetry groups are selected using preliminary inference information, in contrast to several previous approaches which that symmetry groups to be selected *a priori*.
- The approximation maintains a concrete connection to the original problem while maintaining strong control over the computational cost of the inference structure (most previous approaches maintain only one or the other).
- The generalized dual decomposition allows MAP inference, sum-inference, and mixed-inference (where we sum over a subset of RVs and maximize over another) queries to be

performed within one framework (most previous approaches for coarse-to-fine inference operate only on MAP inference problems).

This chapter is based on work originally published in Gallo and Ihler [2018a].

Chapter 5: Lifted Weighted Mini-Bucket This chapter presents a coarse-to-fine inference algorithm for MLN models that adaptively constructs *symmetric high-order inference terms* by joining symmetric factors with a first-order logic structure. Contributions include:

- A flexible class of bounds that complements the coarse-to-fine resolution *split* operation used in Chapter 4 with a *join* operation that builds a large scope symmetric factor from a collection of small-scope symmetric factors. Reasoning over these higher order terms is often necessary for accurate approximate inference.
- The approximation structure is (nearly) equivalent to the structure of the popular domain-clustering heuristic of Venugopal and Gogate [2014a] (e.g., we produce a clustering of people into K groups and each variable in a FOL-formula ranges over a subset of people, rather than the set of all people). However, rather than creating an over-symmetric model with unknown bias (as in Venugopal and Gogate [2014a]), we perform inference on the original problem with tied variational parameters, again giving both a concrete connection to the original problem and allowing iterative refinement of the approximation.
- Our framework is based on a lifted variant of the weighted mini-bucket elimination algorithm, which typically provides faster convergence than an equivalent decomposition bound (such as used in Chapter 4).

This chapter is based on work originally published in Gallo and Ihler [2018b].

Chapter 6: Adaptive Hierarchical State Space Partitions While the preceding chapters studied inference in models with factor symmetry perturbed by asymmetric evidence factors, this chapter studies the problem of incorporating high-order inference terms with state space symmetry into classic PGM models with fully asymmetric factors. Our basic idea is to recursively build complex state space partitions over increasingly large groups of RVs; these are used as a common basis over which high-order inference terms exchange messages without shattering symmetry. Our contributions include:

- A novel method for constructing complex large-scope state space partitions recursively from sets of small-scope state space partitions. This recursive partition generalizes recursively constructed cardinality potentials used, for example, by Tarlow et al. [2012].
- These state space partitions are used as a basis for factors with domain-symmetry that are introduced to tighten the relaxation. Although these factors have complex form of state space symmetry, their symmetries are perfectly aligned (by construction) and can therefore exchange messages without shattering symmetry.
- The state space partitions are constructed adaptively, in a coarse-to-fine manner. Thus, preliminary inference estimates can be used to select the state space partitions; this is crucial since a good state space partition is problem-dependent and cannot be determined *a priori*.
- Our method generalizes inference structures used previously from disparate lines of work, including standard inference with large-scope factors, coarse univariate high order factors [Sontag et al., 2009], and recursive cardinality models [Tarlow et al., 2012]. Furthermore, our inference structure is very flexible and opens up many avenues for future research.

Chapter 7: Conclusions and Future Directions The final chapter summarizes the contributions of this thesis and suggests directions for future work. These include ideas for fusing factor and state space symmetry, better methods for performing symmetry selection, and applying the symmetry structures developed in this thesis to other inference problems (such as maximum-likelihood learning).

Chapter 2

Probabilistic Graphical Models (PGMs)

A probabilistic graphical model (PGM) is a locally factored probability distribution that efficiently encodes a high-dimensional probability distribution. PGMs are used in a wide variety of science and engineering problems which are naturally represented with a locally factored form. This chapter defines PGMs and typical inference tasks in PGMs. We then provide background on tractable approximate inference algorithms that are necessary in most practical scenarios where exact inference is intractable.

2.1 Probabilistic graphical models (PGMs)

Let X_v be a discrete random variable (RV) taking value $X_v = x_v$ where $x_v \in \mathcal{D}_v$ is a *state* and $\mathcal{D}_v$ is a finite set called the *domain*, or *state space*, of X_v, for $v \in \mathcal{V}$ where $\mathcal{V} = \{1, \dots, n\} =: [n]^+$ and $n \in \mathbb{Z}^+$ is the number of RVs. The set of n RVs $X = (X_1, \dots, X_n) =: (X_v)_{v \in \mathcal{V}}$ collectively takes a *joint state* denoted $x = [x_1, \dots, x_n] \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_n =: \mathcal{D}_\mathcal{V}$ where $\mathcal{D}_\mathcal{V}$ is an exponentially large *global joint domain*.

A probabilistic graphical model (PGM) is a probability density function with a locally fac-

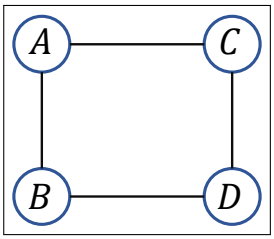
tored structure

$$p(X = x;\ \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\ \exp(\boldsymbol{\theta}(x)) \qquad \text{where} \qquad \boldsymbol{\theta}(x)\ =\ \sum_{\alpha \in \mathcal{I}}\ \theta_\alpha(x_\alpha), \tag{2.1}$$

and each $\alpha \in \mathcal{I}$ is a subset of RVs, i.e., $\alpha \subseteq \mathcal{V}$, that represents the scope $\text{sc}(\theta_\alpha) = \alpha$ of a *model factor* $\theta_\alpha : \mathcal{D}_\alpha \to \mathbb{R}$ where the domain $\mathcal{D}_\alpha := \times_{v \in \alpha} \mathcal{D}_v$ is the set of all configurations of RVs $\alpha \subset \mathcal{V}$. $Z(\boldsymbol{\theta}) = \sum_{x_n} \cdots \sum_{x_1} \exp(\boldsymbol{\theta}(x))$ is the partition function that normalizes the distribution. The cost to represent the PGM is linear in the number of factors and exponential in the largest factor scope, i.e., $O(|\mathcal{I}| \cdot K^a)$ where $a = \max_{\alpha \in \mathcal{I}} |\alpha|$ is the largest size of any model factor scope and $K = \max_{v \in \mathcal{V}} |\mathcal{D}_v|$ is the largest domain size of any RV. This cost is much smaller than the cost to represent the global joint domain, which is exponential in the number of model RVs, i.e., $O(K^n)$.

A PGM is often represented with a graph $G = (N, E)$ where the set of nodes maps to the set of RVs, i.e., $N \leftrightarrow \mathcal{V}$, and there is an undirected edge $e = \{v, v'\} \in E$ between two nodes if and only if both appear together in some factor function, i.e., if $\{v, v'\} \subseteq \alpha$ for some $\alpha \in \mathcal{I}$.

In Figure 2.1a, we show the graphical representation of a PGM on four nodes forming a loop, i.e., with factor scopes $\mathcal{I} = \{\{A, B\}, \{B, D\}, \{C, D\}, \{A, C\}\}$. In this example, each factor table encourages two neighboring RVs to take on the same value (with score $+1.0$, compared to score -1.0 for taking on opposite values); one factor table θ_{AB} is shown in Figure 2.1b. These four factors implicitly and efficiently define a large unnormalized log-probability $\boldsymbol{\theta}(x)$ shown in Figure 2.1c. Note that although the factored and joint form use the same number of parameters in this example (the factored form uses four factor tables of four parameters each, for a total of 16 parameters, while the joint table over four RVs has $2^4 = 16$ entries), in larger problems, the factored form uses exponentially fewer parameters.



(a) MRF representation

x_A	x_B	$\theta_{AB}(x_{AB})$
0	0	+1.0
0	1	-1.0
1	0	-1.0
1	1	+1.0

(b) factor table

x_A	x_B	x_c	x_D	$\boldsymbol{\theta}(x_{ABCD})$
0	0	0	0	+4.0
⋮	⋮	⋮	⋮	⋮
0	1	1	0	0.0
⋮	⋮	⋮	⋮	⋮
1	1	1	1	+4.0

(c) joint pdf table

Figure 2.1: Example graphical model showing (a) a loop on 4 nodes, (b) a single factor table, (c) implicit joint (unnormalized) distribution.

2.1.1 Inference Tasks in Graphical Models

Probabilistic inference refers to the task of answering probabilistic queries about a graphical model. That is, it formalizes questions about the global behavior of the model's many interacting subsystems. Two central probabilistic inference tasks are maximum *a posteriori* (MAP) inference and sum-inference, defined as follows.

MAP Inference MAP inference seeks the most likely configuration of the distribution:

$$x^* = \arg\max_x \boldsymbol{\theta}(x) \qquad \text{and} \qquad \Phi_0(\boldsymbol{\theta}) := \max_x \boldsymbol{\theta}(x) = \boldsymbol{\theta}(x^*) \tag{2.2}$$

Note that the partition function $Z(\boldsymbol{\theta})$ does not affect the solution and thus does not appear in the maximization.

Sum-Inference Computing the partition function $Z(\boldsymbol{\theta})$ is a central sum-inference task for evaluating the (normalized) probability of an observation, or learning the parameters of a graphical model from data. A closely related task is computing the marginal probability of a subset of RVs

$$p(x_\beta) = \sum_{x_{\mathcal{V}\setminus\beta}} p(x_\mathcal{V}) \qquad \text{where} \qquad \beta \subset \mathcal{V} \tag{2.3}$$

and the summation is carried out over joint states of RVs not including β. In many tasks, the marginal probability of each RV i.e., $p(x_v)$ for all $v \in \mathcal{V}$, is required.

2.1.2 Exact Inference Computation

Each inference computation requires aggregating (via maximization or summation) the joint score over an exponentially large number of states. Clearly, a naïve computation is intractable for realistically sized problems. However, in PGMs, the locally factored structure can often be exploited to produce a more efficient computation by rewriting the joint maximization as a sequence of maximizations over each RV. This computation is known as the *variable elimination algorithm*.

We begin with a simple example to illustrate the basic idea of variable elimination. Consider the loop structured PGM in Fig. 2.1a and assume we are interested in computing the log MAP-value $\Phi_0(\boldsymbol{\theta})$. Using the distributive law, this computation can be rewritten as

$$\begin{aligned}
\Phi_0(\boldsymbol{\theta}) &= \max_{x_D} \max_{x_C} \max_{x_B} \max_{x_A} \theta_{AB}(x_{AB}) + \theta_{AC}(x_{AC}) + \theta_{BD}(x_{BD}) + \theta_{CD}(x_{CD}) \qquad (2.4)\\
&= \max_{x_D} \max_{x_C} \theta_{CD}(x_{CD}) + \max_{x_B} \theta_{BD}(x_{BD}) + \Big[\max_{x_A} \theta_{AB}(x_{AB}) + \theta_{AC}(x_{AC})\Big]\\
&= \max_{x_D} \max_{x_C} \theta_{CD}(x_{CD}) + \Big[\max_{x_B} \theta_{BD}(x_{BD}) + m_{BC}(x_{BC})\Big]\\
&= \max_{x_D} \max_{x_C} \theta_{CD}(x_{CD}) + m_{CD}(x_{CD})
\end{aligned}$$

where we perform a sequence of maximizations over individual RVs along the order $\bar{\boldsymbol{o}} = (A, B, C, D)$. The maximization over any RV X_v is said to *eliminate* the RV, and produces a *message* function and which is incorporated into a smaller maximization problem that does not include v. In the example above, the first maximization eliminates X_A producing a message m_{BC}, and the second maximization eliminates X_B producing a message m_{CD}. Each maximization takes place on a factor defined over a group of three RVs, and generates a

Algorithm 1 Bucket (or variable) elimination for computing the log MAP-value $\Phi_0(\boldsymbol{\theta})$ [Dechter, 1999].

Input: Factors of a graphical model $\boldsymbol{F} = \{\theta_\alpha : \alpha \in \mathcal{I}\}$, an elimination order $\boldsymbol{o} = (1, 2, \ldots, n)$.

Output: The log MAP-value $\Phi_0(\boldsymbol{\theta})$.

for $v = 1$ **to** n **do**

`Step 1` **(Join):** Find the set, or *bucket* $\mathcal{B}_v$, of factors whose first RV (under $\boldsymbol{o}$) is v

$$\mathcal{B}_v \leftarrow \{\theta_\beta \in \boldsymbol{F} : \min_{\boldsymbol{o}} \mathrm{sc}(\theta_\beta) = v\}$$

and sum, or *join*, them together

$$\theta_{\mathcal{B}_v}(x_{\mathcal{B}_v}) = \sum_{\theta_\beta \in \mathcal{B}_v} \theta_\beta(x_\beta).$$

Finally, remove the bucket factors from the list, i.e., set $\boldsymbol{F} \leftarrow (\boldsymbol{F} \setminus \mathcal{B}_v)$.

`Step 2` **(Eliminate):** Eliminate RV X_v to create a message factor

$$m_v(x_{m_v}) = \max_{x_v} \theta_{\mathcal{B}_v}(x_{\mathcal{B}_v})$$

whose scope includes all RVs used by factors in bucket $\mathcal{B}_v$ except v, i.e., $\mathrm{sc}(m_v) = \cup_{\theta_\beta \in B_v} \mathrm{sc}(\theta_\beta) \setminus \{v\}$, and add it to the factor list, i.e., set $\boldsymbol{F} \leftarrow (\boldsymbol{F} \cup \{m_v\})$.

end for

return The log MAP-value $\Phi_0(\boldsymbol{\theta}) = \sum_{\theta_\beta \in \boldsymbol{F}} \theta_\beta$. Note that this is a sum of scalar values, each corresponding to the log MAP-value of a connected component of the graph.

message over two RVs. Thus, the variable elimination procedure takes $O(K^3)$ time and is less costly than the $O(K^4)$ time required for the naïve inference computation.

2.1.2.1 Bucket Elimination Algorithm

In general, the variable elimination calculation can be organized as a bucket elimination (BE) procedure [Dechter, 1999] as shown in Algorithm 1. The factor list $\boldsymbol{F}$ is initialized with the model factors, and $\boldsymbol{o}$ is an *elimination order* that dictates the order that RVs are eliminated (without loss of generality we assume that the RVs are eliminated in numeric order, i.e., $\boldsymbol{o} = (1, 2, \ldots, n)$). At the vth iteration of the loop, the factor list $\boldsymbol{F}$ holds the set of factors from the model and previously generated messages that use only RVs equal to or later than v in the elimination order $\boldsymbol{o}$. In `Step 1` (the join step), first the "bucket"

$\mathcal{B}_v$ collects all factors using RV v. These factors are then summed, or *joined*, together to produce a large bucket factor $\theta^m_{\mathcal{B}_v}$. Finally, the bucket factors are removed from the factor list $\boldsymbol{F}$. In `Step 2` (the eliminate step), the RV X_v is eliminated from the joined (bucket) factor and produces a message factor m_v which is added to the factor list $\boldsymbol{F}$.

The computational complexity of the algorithm is dominated by the the cost to represent the (joined) bucket factor. Overall, this requires $O(n \cdot K^{\omega_{\boldsymbol{o}}+1})$ time where $\omega_{\boldsymbol{o}} = \max_{v \in \mathcal{V}} |\text{sc}(m_v)|$ is the size of the largest message generated during elimination along order elimination order $\boldsymbol{o}$, and is known as the *induced width*. Different elimination orders often lead to different induced widths. Since the inference cost is exponential in the induced width along elimination order $\boldsymbol{o}$, it is highly desirable to select an elimination order producing the minimum induced width, i.e., set $\boldsymbol{o} = \arg\min_{\boldsymbol{o}'} \omega_{\boldsymbol{o}'}$. However, in general, finding the optimal elimination order is an NP-complete task. In practice, a number of greedy algorithms have been developed to select a low-width elimination order, such as the min-fill and min-induced-width heuristics (see Koller and Friedman [2009] and references therein).

2.1.3 Chapter Overview

The previous subsections showed how to perform exact inference with the classic bucket elimination algorithm. Although bucket elimination is more efficient than a naïve inference computation, it too is often intractable. This intractability result has motivated a large body of research dedicated to developing tractable approximate inference algorithms. The remainder of this chapter reviews approximate inference algorithms used in this thesis and is organized as follows.

The basic idea of relaxed inference is to perform reasoning over subproblems consisting of a controlled (small) number of RVs. We begin with a discussion of relaxed MAP inference algorithms. In Section 2.2, we review the dual decomposition algorithm which (upper) bounds

the intractable MAP objective as a sum of local MAP objectives. Then, in Section 2.3, we review the mini-bucket elimination algorithm that can be viewed either as a tractable relaxation to the bucket elimination algorithm or as a method for choosing regions for the dual decomposition algorithm. In Section 2.4, we discuss a coordinate-descent-based method to optimize the dual decomposition bound. A symmetric variant of this optimization algorithm is used in our work on *Adaptive Hierarchical State Space Partitions* in Chapter 6.

In Section 2.5, we present variational algorithms for sum-inference problems. Each algorithm is a counterpart to a MAP inference algorithm, but with the maximization operator replaced by a weighted log-sum-exp operator. First, we present the generalized dual decomposition (GDD) which (as its name suggests) has the same structural form as the dual decomposition, but produces an upper bound on the partition function (rather than MAP value). Second, we present the weighted mini-bucket (WMB) elimination algorithm which has the same structural form as the mini-bucket elimination algorithm, but produces an upper bound on the partition function (rather than MAP value). Our *Lifted GDD Algorithm* in Chapter 4 and our *Lifted WMB Algorithm* in Chapter 5 are based on these two ground inference algorithms.

In Section 2.6, we present a framework for anytime inference that iteratively interleaves inference with modification to the approximate inference structure. Anytime inference facilitates the use of preliminary inference estimates to select the inference structure which is often difficult to select *a priori*. Each of our main contributions (in Chapters 4–6) in this thesis is built within an anytime inference paradigm that uses preliminary inference estimates to select inference symmetry groups.

2.2 Dual Decomposition for Approximate MAP Inference

The *dual decomposition algorithm* is an approximate inference algorithm that *relaxes* the equality constraints in the global MAP inference problem. This produces a collection of *decomposed* (independent) MAP inference problems each of which involves a small, user-controlled number of RVs (and hence has controlled computational cost). The basic relaxation (defined on the model factors) bounds the intractable MAP objective as

$$\max_{x_{\mathcal{V}}} \boldsymbol{\theta}(x) \;=\; \max_{x_{\mathcal{V}}} \sum_{\alpha \in \mathcal{I}} \theta_\alpha(x_\alpha) \;\leq\; \sum_{\alpha \in \mathcal{I}} \max_{x_\alpha} \theta_\alpha(x_\alpha)$$

where we applied the *max-sum-swap inequality* $\max_y f(y) + g(y) \;\leq\; \max_y f(y) + \max_y g(y)$ which holds for any two functions f and g with the same domain. That is, the LHS enforces consistency in the maximizing argument, while the RHS does not, i.e., it *relaxes* the consistency requirement.

A more general class of bounds can be defined on a *region graph* $G = (R, S, E)$ where R is a set of region nodes (or *regions*), S is a set of separator nodes (or *separators*), and E is a set of edges connecting each separator to exactly two regions. These terms are defined as follows:

Regions: Each region $r \in R$ is associated with a region model factor θ_r and is labeled with the set of RVs in its scope $\mathrm{sc}(r) := \mathrm{sc}(\theta_r)$. Collectively, the region model factors must represent the original distribution. That is,

$$\boldsymbol{\theta}(x) \;=\; \sum_{\alpha \in \mathcal{I}} \theta_\alpha(x_\alpha) \;=\; \sum_{r \in R} \theta_r(x_r)$$

where we abuse notation by indexing a subset of RVs with a region rather than its corresponding factor scope, i.e., $x_r := x_{\mathrm{sc}(\theta_r)}$.

Separators and Edges: A separator $s \in S$ connects two incident regions $r \in R$ and $r' \in R$ (via a pair of incident edges $\{\{r, s\}, \{s, r'\}\} \subset E$) and serves to exchange a *cost-shifting factor*, or message, δ_s between the regions. (The scope of the cost-shifting factor is equal to the subset of RVs common to both incident regions, i.e., $\text{sc}(\delta_s) = \text{sc}(s) = \text{sc}(r) \cap \text{sc}(r')$.)

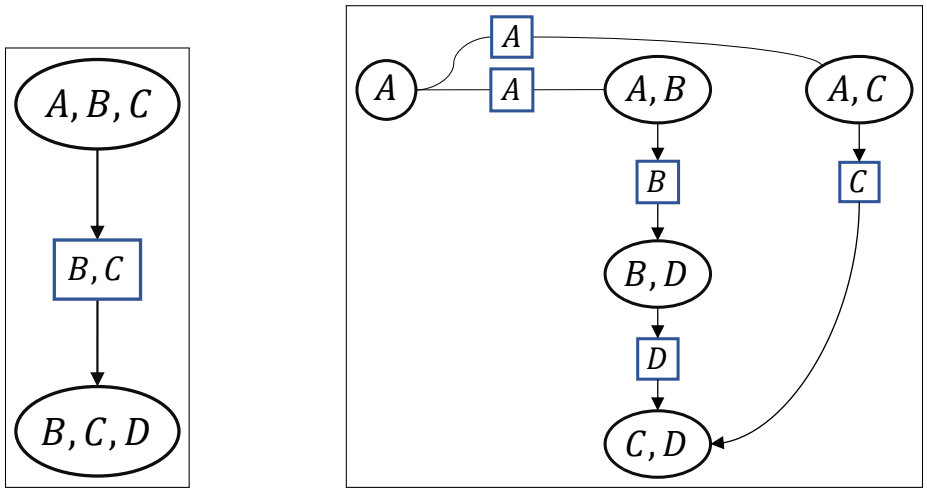
The *cost-shifting factor* is added to one incident region and subtracted from another; this operation is represented by associating cost-shifting factors $\delta_{rs}(x_s) = -\delta_s(x_s)$ and $\delta_{r's}(x_s) = +\delta_s(x_s)$ of opposite sign with the pair of incident edges. Then, each region node $r \in R$ is associated with a *reparameterized factor* equal to the sum of cost-shifting factors on its incident edges

$$\theta_r^{\delta}(x_r) \;=\; \theta_r(x_r) \;+\; \sum_{s \in N(r)} \delta_{rs}(x_s) \tag{2.5}$$

where $N(r) = \{s \in S : \{r, s\} \in E\}$ are the incident separators of region $r \in R$. Clearly, the reparameterization operations leave the model distribution unchanged (since each cost-shifting factor was added to one region and subtracted from another), i.e., $\sum_{r \in R} \theta_r^{\delta}(x_r) = \sum_{r \in R} \theta_r(x_r)$.

For compactness, we will often abuse notation and drop the region subscript on the delta term in the reparameterization equation (2.5), and write it as $\theta_r^{\delta}(x_r) = \theta_r(x_r) + \sum_{s \in N(r)} \delta_s(x_s)$.

We can now define a reparameterized *Dual-Decomposition-objective (DD-objective)* $L_G^{DD}(\boldsymbol{\delta})$ that bounds the intractable MAP objective as a sum of MAP inference problems over local



(a) `iBound` $= 2$, Junction tree

(b) `iBound` $= 1$

Figure 2.2: Mini-bucket elimination (MBE) graph for the loop on four RVs $A-B-D-C-A$ from Figure 2.1. (a) MBE where each message scope has size at most `iBound` $= 2$; this graph is equal to the junction tree and provides the exact inference solution. (b) MBE where each message scope has size at most `iBound` $= 1$.

regions as

$$\max_{x_{\mathcal{V}}} \boldsymbol{\theta}(x) \;=\; \max_{x_{\mathcal{V}}} \sum_{r \in R} \theta_r(x_r) \;=\; \max_{x_{\mathcal{V}}} \sum_{r \in R} \theta_r^{\delta}(x_r)$$
$$\leq \sum_{r \in R} \max_{x_r} \theta_r^{\delta}(x_r) \;=:\; L_G^{DD}(\boldsymbol{\delta}). \tag{2.6}$$

This bound holds for any setting of the cost-shifting factors $\boldsymbol{\delta}$ and we seek the tightest bound

$$\boldsymbol{\delta}^* \;=\; \arg\min_{\boldsymbol{\delta}} \; L_G^{DD}(\boldsymbol{\delta}). \tag{2.7}$$

Before considering an optimization of the DD-objective $L_G^{DD}(\boldsymbol{\delta})$ on a general graph structure, we consider a special case in which a closed form solution can be obtained.

2.2.1 Bucket Elimination on a Region Graph

The bucket elimination algorithm (Algorithm 1) can be represented on a region graph. To

Algorithm 2 Junction tree region graph construction [Dechter and Rish, 1997, Ihler et al., 2012].

Input: A set of regions R corresponding to model factors $\{\theta_r : r \in R\}$, an elimination order $\boldsymbol{o} = (1, 2, \ldots, n)$.

Output: A bucket elimination region graph $G = (R, S, E)$.

for $v = 1$ **to** n **do**

`Step 1` **(Join):** Find the set, or *bucket* $\mathcal{B}_v$, of factors whose first RV (under $\boldsymbol{o}$) is v

$$\mathcal{B}_v \leftarrow \{\theta_r \in R : \min_{\boldsymbol{o}} \mathrm{sc}(\theta_r) = v\}$$

and sum, or *join*, them together

$$\theta_{r_v}(x_{r_v}) = \sum_{r \in \mathcal{B}_v} \theta_r(x_r)$$

Finally, remove the bucket factors from the region graph, i.e., set $R \leftarrow (R \setminus \mathcal{B}_v)$.

`Step 1a`: Reroute the backward separators to the bucket region r_v. That is, for each region $r \in \mathcal{B}_v$ in the bucket and each backward separator $s_\uparrow \in N_\uparrow(r)$ of the region (with neighbors $N(s_\uparrow) = \{r_\uparrow, r\}$), set

$$R \leftarrow (R \setminus r) \qquad \text{and} \qquad N(s_\uparrow) \leftarrow \{r_\uparrow, r_v\}.$$

`Step 2` **(Eliminate):** "Simulate" elimination of RV X_v. That is, create a message region $r_\downarrow$ whose scopes uses all of the RVs in the bucket except v, i.e., $\mathrm{sc}(r_\downarrow) = \mathrm{sc}(r_v) \backslash v$, with value zero, i.e., $\theta_{r_\downarrow} = \mathbf{0}$. Connect the message region to the bucket region with a message separator $s_\downarrow$, i.e., $N(s_\downarrow) = \{r, r_\downarrow\}$, and add them to the graph, i.e., set $R \leftarrow (R \cup \{r_\downarrow\})$ and $S \leftarrow (S \cup \{s_\downarrow\})$.

end for

return A bucket elimination region graph $G = (R, S, E)$.

do this, we separate the algorithm into two parts. First, we construct a region graph, called a *junction tree*, that uses placeholder regions to represent the message factors. Second, we perform the actual message computations on the region graph and show that they minimize the decomposed bound Eq. (2.7).

Algorithm 2 constructs the junction tree. In `Step 1` (the join step), first the "bucket" $\mathcal{B}_v$ collects all regions using RV v (these include region model factors and region message placeholders). Then, these regions are joined to produce a region r_v with factor equal to the sum of region model factors and with scope equal to the scope of the bucket elimination message. In `Step 1a`, we re-route the message regions to the bucket region; this organizes the

Algorithm 3 Exact MAP inference in junction tree

Input: A bucket elimination region graph G, and an elimination order $\boldsymbol{o} = (1, 2, \ldots, n)$.
Output: The set of messages (equivalent to bucket elimination messages) and the exact log MAP-value $\Phi_0(\boldsymbol{\theta})$.

for $v = 1$ **to** n **do**

`Step 1` **(Gather):** Gather the backward messages into the bucket region r_v:

$$\theta^m_{r_v}(x_{r_v}) \;=\; \theta_{r_v}(x_{r_v}) \;+\; \sum_{\substack{s_\uparrow \in N_\uparrow(r) \\ N(s_\uparrow)=\{r, r_\uparrow\}}} m_{r_\uparrow \to s_\uparrow}(x_{s_\uparrow})$$

`Step 2` **(Eliminate):** Eliminate RV v from the bucket region r_v:

$$m_{r_v \to s_\downarrow}(x_{s_\downarrow}) \;=\; \max_{x_v} \; \theta^m_{r_v}(x_{r_v})$$

where $\{s_\downarrow\} = N_\downarrow(r_v)$ is the forward separator.

end for

precise flow of messages on the region graph. In `Step 2` (the eliminate step), we "simulate" the elimination of X_v from the joined (bucket) region r_v to create a *forward region* $r_\downarrow$ and joined to the original region by a *forward separator* $s_\downarrow$. The forward region has the same scope as the true message but has factor value zero.

A simple example of a junction tree on the loop PGM from Figure 2.1a is shown in Figure 2.2a. Note that the symbols used in `Step 1` are based on the orientation shown in the figure where the earlier RVs in the elimination order are organized at the top and elimination passes a message downward. For example, at the first iteration where $v = A$, the bucket region r_v is labeled (A, B, C) and the forward separator $s_\downarrow$ is labeled (B, C).

In Algorithm 3, the bucket elimination message passing is computed on the region graph. In `Step 1` (gather step), we gather and sum the messages into the factor associated with the bucket at RV v. In `Step 2` (message step), we eliminate the RV X_v to produce the message factor.

2.2.1.1 Decomposed form

The last section showed how to represent bucket elimination messages on a region graph. This section shows that the bucket elimination messages provide a closed form solution of the optimal cost-shifting factors $\boldsymbol{\delta}$ in the DD-objective, i.e., we set $\delta_{s_\downarrow}(x_{s_\downarrow}) := m_{r\to s_\downarrow}(x_{s_\downarrow})$ for each separator $s \in S$. To see this note that the reparameterized factor at region r_v associated with bucket v is equal to the accumulated factor (of messages) minus its forward message

$$\theta^{\delta}_{r_v}(x_{r_v}) \;=\; \theta^{m}_{r_v}(x_{r_v}) \;-\; m_{r_v\to s_\downarrow}(x_{s_\downarrow}).$$

Then, the reparameterized factor has maximum value zero

$$\begin{aligned}\max_{x_v}\; \theta^{\delta}_{r_v}(x_{r_v}) \;&=\; \max_{x_v}\; \big(\theta^{m}_{r_v}(x_{r_v}) \;-\; m_{r_v\to s_\downarrow}(x_{s_\downarrow})\big) \\ &=\; \big(\max_{x_v}\; \theta^{m}_{r_v}(x_{r_v})\big) \;-\; m_{r_v\to s_\downarrow}(x_{s_\downarrow}) \;=\; 0\end{aligned}$$

for each state of the message scope $x_{s_\downarrow} \in \mathcal{D}_{s_\downarrow}$. Finally, each non-root region has maximum value 0 and the root node has value equal to the MAP-value. That is,

$$\max_{x_r}\; \theta^{\delta}_{r}(x_r) \;=\; \begin{cases}\Phi_0(\boldsymbol{\theta}) & \text{if r is root} \\ 0 & \text{otherwise}\end{cases} \tag{2.8}$$

and the accumulated value is equal to the MAP value

$$L^{DD}_{G}(\boldsymbol{\delta}) \;=\; \sum_{r\in R}\; \max_{x_r}\; \theta^{\delta}_{r}(x_r) \;=\; \Phi_0(\boldsymbol{\theta}). \tag{2.9}$$

2.3 Mini-Bucket Elimination

Although bucket elimination is more efficient than a naïve computation, it too is often intractable in real problems. The mini-bucket elimination (MBE) algorithm [Dechter and

Rish, 1997] is a relaxation of the bucket elimination algorithm (Algorithm 1) that provides a flexible tradeoff between cost and accuracy. As a simple example to motivate this procedure, consider the exact bucket elimination example in Eq. (2.4) above. The maximization over x_A can be relaxed as

$$\max_{x_A} \theta_{AB}(x_{AB}) + \theta_{AC}(x_{AC}) \;\leq\; \Big[\max_{x_A} \theta_{AB}(x_{AB})\Big] \;+\; \Big[\max_{x_A} \theta_{AC}(x_{AC})\Big] \tag{2.10}$$

using the max-sum-swap inequality. Note that the cost of the relaxed elimination is only $O(K^2)$ rather than $O(K^3)$ for the exact elimination.

In general, the elimination at any variable $v \in \mathcal{V}$ can be bounded by a set of smaller eliminations which produce a message factor using at most `iBound` RVs, where `iBound` is a hyper-parameter that controls the computational cost. To achieve this, the factors in bucket $\mathcal{B}_v$ are partitioned into z_v non-empty sets, or *mini-buckets*, $\{\mathcal{B}_v^i : i \in [z_v]^+\}$. The factors at each mini-bucket $\mathcal{B}_v^i$ are first summed (joined), denoted by a factor $\theta_{\mathcal{B}_v^i}(x_{\mathcal{B}_v^i}) := \sum_{\theta \in \mathcal{B}_v^i} \theta(x_{\mathrm{sc}(\theta)})$. Then, a maximization over X_v eliminates v from this factor and produces a message that is incorporated downstream. Note that mini-bucket elimination is equivalent to exact bucket elimination when `iBound` is at least as large as the induced width $\omega_{\boldsymbol{o}}$.

2.3.1 Mini-Bucket Region Graph

The mini-bucket elimination algorithm is organized on a region graph, as shown in Algorithm 4. The basic structure and notation is similar to what we used to represent bucket elimination on a region graph in Algorithm 2. (Although the original mini-bucket elimination algorithm [Dechter and Rish, 1997] was not developed on a region graph, this connection was made explicit in Ihler et al. [2012].)

In `Step 1` (the join step), first the "bucket" $\mathcal{B}_v$ collects all regions using RV v (these regions include region model factors and message placeholder regions). Then, we partition these

Algorithm 4 Mini-bucket region graph construction [Dechter and Rish, 1997, Ihler et al., 2012].

Input: A set of regions R corresponding to model factors $\{\theta_r : r \in R\}$, an elimination order $\boldsymbol{o} = (1, 2, \ldots, n)$, and a bound on the largest message size $\texttt{iBound} \geq 1$.

Output: A mini-bucket region graph $G = (R, S, E)$.

for $v = 1$ **to** n **do**

`Step 1` **(Join):** Find the set, or *bucket* $\mathcal{B}_v$, of regions whose first RV (under $\boldsymbol{o}$) is v

$$\mathcal{B}_v \leftarrow \{r \in R : \min_{\boldsymbol{o}} \text{sc}(\theta_r) = v\}$$

and partition it into z_v sets, or *mini-buckets*, $\{\mathcal{B}_v^i : i \in [z_v]^+\}$, where each set uses at most $\texttt{iBound} + 1$ RVs. Then, for each $i \in [z_v]^+$, sum, or join, the factors in each mini-bucket together

$$\theta_{r_v^i}(x_{r_v^i}) = \textstyle\sum_{r \in \mathcal{B}_v^i} \theta_r(x_r) \qquad \text{where} \qquad |\text{sc}(\theta_{r_v^i})| \leq \texttt{iBound} + 1$$

where the mini-bucket region r_v^i is associated with mini-bucket $\mathcal{B}_v^i$. Finally, delete the original regions and add the mini-bucket region, i.e., set $R \leftarrow (R \setminus \mathcal{B}_v^i) \cup \{r_v^i\}$.
`Step 1a`: Reroute each backward separator $s_\uparrow \in N_\uparrow(r)$ of a region $r \in \mathcal{B}_v^i$ to its containing mini-bucket region r_v^i. That is, update the neighbors $N(s_\uparrow) = \{r_\uparrow, r\}$ to $N(s_\uparrow) \leftarrow \{r_\uparrow, r_v^i\}$.

`Step 2` **(Eliminate):** "Simulate" the elimination of X_v from each mini-bucket region. That is, for each $i \in [z_v]^+$, create a message region m_v^i with scope $\text{sc}(\theta_{m_v^i}) = \text{sc}(\theta_{r_v^i}) \setminus \{v\}$ connected to the mini-bucket region via separator s_v^i, i.e., $N(s_v^i) = \{r_v^i, m_v^i\}$. That is, set $R \leftarrow R \cup \{m_v^i\}$ and $S \leftarrow S \cup \{s_v^i\}$. Finally, associate the placeholder message region with a factor $\theta_{m_v^i}$ whose value is zero for all states, i.e., $\theta_{m_v^i}(x_{m_v^i}) = 0$ for all $x_{m_v^i} \in \mathcal{D}_{m_v^i}$.

`Step 3` **(Cost-Shift):** Allow cost-shifting among mini-bucket regions by creating a "dummy" univariate region r_v^0 and horizontal separators connecting it to each mini-bucket region. That is, set $R \leftarrow R \cup \{r_v^0\}$ and $S \leftarrow S \cup \{s_{\leftrightarrow;v}^i\}$ where $N(s_{\leftrightarrow;v}^i) = \{r_v^0, r_v^i\}$ for each $i \in [z_v]^+$.

end for

return A mini-bucket region graph $G = (R, S, E)$.

regions into z_v mini-buckets, or sets, consisting of at most $\texttt{iBound}+1$ RVs. Factors in each mini-bucket are joined (and deleted) to create a mini-bucket region which is added to the graph. In `Step 1a`, we re-route the messages to the new joined region.

The mini-bucket partition in `Step 1` is often chosen in a greedy manner by prioritizing joins of larger factors as follows. First, sort the factors in the bucket by decreasing scope size.

Then, starting from the first (largest) factor, join it with the first factor in the sequence such that the resulting scope is smaller than or equal to `iBound`+1. Repeat this step until the end of the list is reached. The result factor represents one mini-bucket region. We then remove the joined factors and repeat this process to create the remaining mini-bucket regions.

In `Step 2` (the eliminate step), we simulate a message passing step by creating a placeholder message region with factor value zero and add it to the graph. At the conclusion of the algorithm, the regions and separators constructed during `Step 1` and `Step 2` constitute a region graph known as a mini-bucket tree.

In `Step 3` (the cost-shift step), we add cost-shifting factors between mini-buckets; this allows the mini-bucket regions to reallocate factors before the elimination step and, in general, can lead to tighter bounds. To do this, we create a cost-shifting term between all mini-buckets via a "dummy" region r_v^0. In addition, we add a cost-shifting separator $s_{\leftrightarrow;v}^i$ to connect a region r_v^i to the "dummy" region r_v^0 in its associated bucket. (The index of the separator signifies that the cost-shifting separator connects horizontally aligned regions in the typical depiction of the graph, as shown in the example discussed in the next paragraph.) Note that cost-shifting through a "dummy" term can represent the same class of bounds as cost-shifting between each pair of mini-bucket factors, but with a more compact representation, i.e., using $O(z_v)$ rather than $O(z_v^2)$ parameters.*

We illustrate the mini-bucket region graph constructed from the loop graph in Figure 2.1. In Figure 2.1a, we show a region graph with `iBound` = 1. In Figure 2.1b, we show a region graph with `iBound` = 2. The latter region graph represents the exact bucket elimination computation.

2.3.1.1 Message Passing on the Mini-Bucket Tree

*Note that if mini-buckets share a larger set of overlapping RVs, then we can perform cost-shifting over that high-order scope $\cap_{i=1}^k \text{sc}(\theta_{\mathcal{B}_v^i})$, as done in Liu and Ihler [2011]. For simplicity, we do not do this here.

Algorithm 5 Exact MAP inference in a mini-bucket tree

Input: A bucket elimination region graph G, and an elimination order $\boldsymbol{o} = (1, 2, \ldots, n)$.
Output: The set of messages (equivalent to bucket elimination messages) and the exact log MAP-value $\Phi_0(\boldsymbol{\theta})$.

for $v = 1$ **to** n **do**
  for $i = 1$ **to** z_v **do**

`Step 1` **(Gather):** Add the (horizontal) cost-shifting factor (associated with separator $s^i_{\leftrightarrow;v}$) to mini-bucket region r^i_v factor

$$\theta^{\delta\leftrightarrow}_{r^i_v}(x_{r^i_v}) \;=\; \theta_{r^i_v}(x_{r^i_v}) \;+\; \delta_{s^i_{\leftrightarrow;v}}(x_v),$$

then add the backward messages

$$\theta^{m}_{r^i_v}(x_{r^i_v}) \;=\; \theta^{\delta\leftrightarrow}_{r^i_v}(x_{r^i_v}) \;+\; \sum_{\substack{s_\uparrow \in N(r^i_v) \\ N(s_\uparrow)=\{r^i_v, r_\uparrow\}}} m_{r_\uparrow \to s_\uparrow}(x_{s_\uparrow})$$

`Step 2` **(Eliminate):** Eliminate RV v from the mini-bucket region r^i_v factor

$$m_{r^i_v \to s_\downarrow}(x_{s_\downarrow}) \;=\; \max_{x_v} \; \theta^{m}_{r^i_v}(x_{r^i_v})$$

where $\{s_\downarrow\} = N_\downarrow(r^i_v)$ is the forward separator.
  end for
end for

In Algorithm 5, we compute the exact MAP value on the mini-bucket tree relaxation. That is, for a fixed set of horizontal separator cost-shifting factors $\boldsymbol{\delta}_{S_\leftrightarrow} = \{\delta_{s_\leftrightarrow} : s_\leftrightarrow \in S_\leftrightarrow\}$, we iterate over each mini-bucket region (in order) and eliminate its first RV to create a message. The algorithm evaluates the mini-bucket objective $L^{MBE}_G(\boldsymbol{\delta}_{S_\leftrightarrow})$ which we write as

$$L^{MBE}_G(\boldsymbol{\delta}_{S_\leftrightarrow}) \;=\; \max_{x_n^{z_n}} \cdots \max_{x_n^1} \;\cdots\; \max_{x_1^{z_1}} \cdots \max_{x_1^1} \sum_{\substack{v\in\mathcal{V} \\ i\in[z_v]^+}} \theta^{\delta\leftrightarrow}_{r^i_v}(x_{r^i_v}). \tag{2.11}$$

where the notation "$\max_{x^i_v}$" indicates elimination of the ith mini-bucket region associated with $v \in \mathcal{V}$, and, as defined in `Step 1`, $\theta^{\delta\leftrightarrow}_{r^i_v}$ is the model factor reparameterized with its horizontal cost-shifting factor. The quality of the bound depends on how cost is allocated among the mini-bucket regions. To obtain the tightest bound, we optimize the mini-bucket

objective over each setting of the horizontal cost-shifting factors

$$\arg\min_{\boldsymbol{\delta}_{S_{\leftrightarrow}}} \; L_G^{MBE}(\boldsymbol{\delta}_{S_{\leftrightarrow}}).$$

One way to perform this optimization is with standard black-box convex optimization tools. Another way to optimize this class of bounds is to use Algorithm 4 to construct a mini-bucket region graph but then optimize the cost-shifting factors as if it were generic region graph, i.e., ignoring the mini-bucket tree substructure and not performing message passing on the mini-bucket tree. For a generic region graph, a block coordinate-descent algorithm is often employed to perform optimization of the cost-shifting factors. We describe this block coordinate-descent algorithm in the next section.

2.4 Optimizing the Dual Decomposition with Block Coordinate-Descent

In the last section, we saw that tree-structured graphs admit a closed form solution to the dual decomposition minimization (2.7). However, for general (loopy) graphs, there is no closed form solution. The DD-objective is convex with respect to its variational parameters and can, in principle, be optimized using any of a variety of iterative convex optimization techniques [Boyd et al., 2004]. However, it is often more efficient to exploit the decomposed form of the objective to perform a *block-coordinate descent* that optimizes a subset of cost-shifting factors in closed form while holding the rest of the cost-shifting factors fixed. When the subgraph is a tree, it can be optimized using the same message passing as the bucket elimination region graph in the last section. A simple special case, pursued by Sontag and Jaakkola [2009], is when the tree is a star-shaped region graph, meaning that it consists of a single region node and its regions. This special case is presented in this section. Our work on *Adaptive Hierarchical State Space Partitions* in Chapter 6 uses updates based on star-shaped region graphs.

Algorithm 6 Block coordinate descent for relaxed MAP inference (optimizing the DD-objective)

Input: A region graph $G = (R, S, E)$.
Output: Optimally reparameterized DD-objective $L_G^{DD}(\boldsymbol{\delta}^*)$.
while $L_G^{DD}(\boldsymbol{\delta})$ has not converged **do**
for each region $r \in R$ **do**
Goal: Form an optimal reparameterization on the star-shaped subgraph

$$\min_{\delta_{S_r^{star}}} \sum_{r \in R_r^{star}} \theta_r^{\delta}(x_r)$$

where r is the root, $R_r^{star} = \cup_{s \in N(r)} N(s)$, and $R_r^{tail} = R_r^{star} \setminus \{r\}$.
`Step 1a`: For each tail region $r' \in R_r^{tail}$, form an outward message (towards root r).

$$m_{r' \to s}(x_s) \;=\; \max_{x_{r' \setminus s}} \; \theta_{r'}^{\delta}(x_{r'})$$

`Step 1b`: For each tail region $r' \in R_r^{tail}$, form an inward message (away from root r).

$$m_{r \to s}(x_s) \;=\; \max_{x_{r \setminus s}} \Big(\theta_r(x_r) + \sum_{\substack{s' \in N(r) \\ \{r, r'\} = N(s')}} m_{r' \to s'}(x_{s'}) \Big)$$

`Step 2`: Compute the optimal cost-shifting factors as a convex combination of message. For each tail separator $s \in N(r)$ where $N(s) = \{r, r'\}$

$$\delta_s^*(x_s) \;=\; \rho_{r'} \cdot m_{r' \to s}(x_s) \;-\; (1 - \rho_{r'}) \cdot m_{r \to s}(x_s)$$

where the weights satisfy $\sum_{r' \in R_r^{tail}} \rho_{r'} \leq 1$. As a default, set $\rho_{r'} = \frac{1}{|R_r^{tail}|+1}$ for each r' to balance cost evenly through all roots.
end for
end while

The star-shaped region updates are as follows. Consider a root region $r \in R$ and its incident separators $S_r^{star} = N(r)$ and star-shaped region neighborhood $R_r^{star} = \cup_{s \in N(r)} N(s)$. Optimizing the block of parameters associated with these separators S_r^{star} only affects the star-graph R_r^{star} and the reparameterization at other regions $R \setminus R_r^{star}$ are fixed. That is,

$$\begin{aligned} \min_{\delta_{S_r^{star}}} L_G^{DD}(\boldsymbol{\delta}) \;&=\; \min_{\delta_{S_r^{star}}} \left(\sum_{r \in R_r^{star}} \theta_r^{\delta}(x_r) \;+\; \sum_{r \in R \setminus R_r^{star}} \theta_r^{\delta}(x_r) \right) \\ &=\; \left(\min_{\delta_{S_r^{star}}} \sum_{r \in R_r^{star}} \theta_r^{\delta}(x_r) \right) \;+\; \sum_{r \in R \setminus R_r^{star}} \theta_r^{\delta}(x_r) \end{aligned}$$

Optimization over the subgraph R_r^{star} can be carried out in closed form by designating any region as a root and directing messages toward it. Selecting any node as the root results in a solution to the decomposed problem, but results in a different reparameterization of the factors. Qualitatively, choosing one node as a root can be seen as "pushing" the cost of the subsystem to that root. This reparameterization affects the neighboring subproblems that are solved next in the sequence.

In general, the effect of the local reparameterization choice on the global convergence rate is not well understood [Tourani et al., 2020]. A common heuristic is to utilize a mixed solution. That is, to solve the local problem with each node as the root and take a convex combination (with equal weights) of the reparameterizations. Clearly, a convex combination of optimal solutions is also optimal.

Algorithm 6 describes the block coordinate descent approach on star-shaped subgraphs. The outer loop iterates over the set of regions in a predefined sequence. The body of the loop computes the optimum reparameterization, as specified in the first line of the loop and implemented in the remaining steps. In `Step 1a` and `Step 1b`, we compute the set of messages directed toward and from each node. In `Step 2` we form a convex combination of the messages.

2.5 Sum-Inference

In the previous sections we discussed mini-bucket elimination and dual decomposition methods for MAP inference problems. In this section, we discuss extensions of these methods for sum-inference problems.

To derive these extensions, the basic idea is to substitute the max-sum-swap inequality in the MAP inference elimination bound for Hölder's inequality. Hölder's inequality [Hardy

and Littlewood, 1952], states that for any K non-negative functions $h_1, \ldots, h_K$,

$$\sum_x \prod_{i=1}^{K} h_i(x) \;\leq\; \prod_{i=1}^{K} \sum_x^{w_i} h_i(x) \tag{2.12}$$

where each weight is non-negative and the weights sum to one, i.e., $w_i \geq 0$ for all $i \in [K]^+$ and $\sum_{i=1}^{K} w_i = 1$. The power-sum operator is defined as

$$\sum_x^{w} h(x) \;=\; \big[\sum_x h(x)^{1/w}\big]^w,$$

which reduces to the standard summation operator $\sum_x h(x)$ when $w = 1$ and the standard maximization operator $\max_x h(x)$ as $w \to 0^+$.

When working with PGMs, it will be convenient to define a log-sum-exp operator and a weighted log-sum-exp operator as

$$\begin{aligned}
\underset{x}{\texttt{lse}}\, \theta(x) \;&=\; \log \sum_x \exp(\theta(x)) \\
\underset{x}{\overset{w}{\texttt{lsew}}}\, \theta(x) \;&=\; \log \sum_x^{w} \exp(\theta(x))
\end{aligned}$$

respectively, for any function θ and non-negative weight w.

2.5.1 Weighted Mini-Bucket Elimination

The weighted mini-bucket algorithm can be broken up into two steps (as we did for the MAP mini-bucket elimination algorithm). First, using Algorithm 4, we construct a region graph whose messages use at most `iBound` RVs (i.e., the graph construction procedure as for MAP inference). Second, we perform message passing on a mini-bucket tree as in Algorithm 5 but

with the max elimination replaced by a weighted log-sum-exp elimination

$$m_{r_v^i \to s_\downarrow}(x_{s_\downarrow}) \;=\; \underset{x_v}{\overset{w_r^i}{\texttt{lsew}}} \;\; \theta^m_{r_v^i}(x_{r_v^i})$$

for each $i \in [z_v]$ where $\{s_\downarrow\} = N_\downarrow(r_v^i)$ is the forward separator and the weights are non-negative and sum to one, i.e., $w_r^i \geq 0$ for each $i \in [z_v]$ and $\sum_{i \in [z_v]} w_r^i = 1$ (where, recall, that z_v is the number of mini-buckets associated with a RV $v \in \mathcal{V}$).

The bound on an exact sum-elimination at RV v is obtained via Hölder's inequality as

$$\log\Big(\sum_{x_v} \prod_{i=0}^{z_v} \exp(\theta^m_{r_v^i}(x_{r_v^i}))\Big) \;\leq\; \sum_{i=0}^{z_v} m_{r_v^i \to s_\downarrow}(x_{s_\downarrow})$$

where the RHS is tractable, with cost exponential in $\texttt{iBound} + 1$, while the computation on the LHS is intractable.

The weighted mini-bucket objective is a function of the horizontal separator cost-shifting factors $\boldsymbol{\delta}_{S_\leftrightarrow}$ and weight $\boldsymbol{w}$ parameters

$$L_G^{WMB}(\boldsymbol{\delta}_{S_\leftrightarrow}, \boldsymbol{w}) \;=\; \underset{x_n}{\overset{w_n^{z_n}}{\texttt{lsew}}} \cdots \underset{x_n}{\overset{w_n^1}{\texttt{lsew}}} \;\cdots\; \underset{x_1}{\overset{w_1^{z_1}}{\texttt{lsew}}} \cdots \underset{x_1}{\overset{w_1^1}{\texttt{lsew}}} \sum_{\substack{v \in \mathcal{V} \\ i \in [z_v]^+}} \theta^{\delta\leftrightarrow}_{r_v^i}(x_{r_v^i}) \tag{2.13}$$

which has the same form as the mini-bucket elimination objective (2.11) for MAP inference, but with max operators replaced by $\texttt{lsew}$ operators.

☐ 2.5.1.1 Gradient and Optimization

The gradient of the WMB-objective $L_G^{WMB}(\boldsymbol{\delta}_{S_\leftrightarrow}, \boldsymbol{w})$ is derived in [Liu and Ihler, 2011] and requires a backward pass that complements the forward pass (elimination). The backward pass aggregates the effect of changing a parameter on mini-buckets later in the elimination order. That is, a change in any parameter affects the outgoing message to its incident bucket,

which then affects the value of that bucket's message, and so on downstream.

For this thesis, we will not need to understand how the gradient is computed; we will simply need compact symbols to describe it and a qualitative understanding of what it represents in terms of the inference problem. The gradient with respect to the region factor can be interpreted as a pseudo-marginal that approximates the true marginal probability (which is equal to the gradient of the log partition function). We denote the pseudo-marginal with the notation

$$\mu_r^\delta(x_r) \;:=\; \frac{\partial\, L_G^{WMB}(\boldsymbol{\delta}_{S_\leftrightarrow}, \boldsymbol{w})}{\partial\, \theta_r^\delta(x_r)}$$

where the sum of these pseudo-marginals is equal to one, i.e., $\sum_{x_r}\; \mu_r^\delta(x_r) = 1$. The gradient with respect to the cost-shifting factor incident to the reigon is a marginalization over a subset of RVs

$$\mu_{r\to s}^\delta(x_s) \;:=\; \frac{\partial\, L_G^{WMB}(\boldsymbol{\delta}_{S_\leftrightarrow}, \boldsymbol{w})}{\partial\, \delta_{rs}(x_s)} \;=\; \sum_{x_{r\setminus s}} \mu_r^\delta(x_r)$$

The gradient of the objective with respect to a horizontal separator $s \in S_\leftrightarrow$ incident to RV-region v and region r is equal to a difference of the incident pseudo-marginals

$$g_s^\delta(x_s) \;:=\; \frac{\partial\, L_G^{WMB}(\boldsymbol{\delta}_{S_\leftrightarrow}, \boldsymbol{w})}{\partial\, \delta_s(x_v)} \;=\; \mu_{r\to s}^\delta(x_s) - \mu_v^\delta(x_v) \tag{2.14}$$

since the cost-shifting factor δ_s appears in the first region with a positive sign and the second with a negative sign (and $x_s = x_v$ since the horizontal separator s has scope $\mathrm{sc}(s) = v$).

When the gradient is zero, the pseudo-marginals in each region are equal. Thus, optimizing the objective can be thought of as enforcing a *weak consistency* such that the multiple copies of each RV (in different regions in the graph) have identical pseudo-marginals.

The gradient with respect to the weight has an interpretation as a conditional entropy. Since the gradient of the weight parameters will not be referenced explicitly in the novel work of this thesis, we simply refer the reader to Liu and Ihler [2011] for details.

Optimization As shown in Liu and Ihler [2011], the WMB-objective is convex with respect to the cost-shifting factors $\boldsymbol{\delta}_{S_{\leftrightarrow}}$ and weights $\boldsymbol{w}$. Liu and Ihler [2011] use a fixed point algorithm to optimize the parameters However, this approach is not guaranteed to converge or monotonically decrease the bound. Instead, in this thesis, we optimize the bound using generic convex optimization procedures [Boyd et al., 2004] that are supplied with the objective function and gradient. In this thesis, we use the `minFunc` toolbox in MATLAB to perform the optimization [Schmidt, 2005].

To maintain positivity of the weights we transform them into log-space and perform an unconstrained optimization. This is a standard practice in convex optimization as it is often more convenient to optimize an unconstrained objective. That is, we use an unconstrained parameter $\eta_r^i \in \mathbb{R}$ which defines a weight

$$w_r^i \;=\; \frac{\exp(\eta_r^i)}{\sum_{i' \in [z_v]} \exp(\eta_r^{i'})}$$

for each $i \in [z_v]$ and each $v \in \mathcal{V}$. We then perform optimization over the objective $L_G^{DD}(\boldsymbol{\delta}, \boldsymbol{\eta})$.

2.5.2 Generalized Dual Decomposition

Ping et al. [2015] developed a *generalized dual decomposition (GDD)* objective for sum-inference problems that extends the dual decomposition bounds for MAP inference problems. The GDD bound is derived from Hölder's inequality and decomposed into a set of tractable *weighted power-sum* inference problems over local regions (as opposed to a set of local MAP inference problems for the dual decomposition bounds).

That is, the GDD bound on the log partition function is

$$\log(Z(\boldsymbol{\theta})) \;=\; \underset{x_{\mathcal{V}}}{\texttt{lse}} \sum_{r \in R} \theta_r(x_r) \tag{2.15}$$

$$\leq \sum_{r \in R} \underset{x_r}{\overset{w_r}{\texttt{lsew}}}\, \theta_r(x_r) \;=:\; L_G^{GDD}(\boldsymbol{\delta}, \boldsymbol{w}) \tag{2.16}$$

where $\boldsymbol{w}_r = (w_r^1, \ldots, w_r^{|r|})$ is a set of nonnegative weights, and the vector (log) power-sum operation

$$\underset{y}{\overset{\boldsymbol{\omega}}{\texttt{lsew}}}\, \theta(y) \;=\; \underset{y_k}{\overset{\omega_k}{\texttt{lsew}}} \cdots \underset{y_1}{\overset{\omega_1}{\texttt{lsew}}}\, \theta(y_1, \ldots, y_k)$$

is a sequence of k power-sum operations where $\boldsymbol{\omega} = (\boldsymbol{\omega}_i)_{i=1}^k$ is a vector of k nonnegative weights, g is a function over k dimensions and the RVs are ordered by the elimination order i.e., $y_1 \prec_{\boldsymbol{o}} \ldots \prec_{\boldsymbol{o}} y_k$. Note that, in general, the vector power-sum operator does not commute. However, when adjacent weights are identical, it does commute.

In the original GDD work [Ping et al., 2015], the weights associated with each RV across all factors are restricted to sum to one, i.e., $\sum_{r \in R: v \in r} \boldsymbol{w}_r^v = 1$, where $\boldsymbol{w}_r^v$ indexes the weight associated with RV $v \in \mathcal{V}$ and region $r \in R$. However, in this thesis we allow the weights to exceed one. (This will allow additional, necessary flexibility for our *Lifted GDD Algorithm* in Chapter 4, which relies on tying weights and will, in general, not be able to force the sum to be exactly one.) To represent this property, we require that the region graph has a univariate factor associated with each RV, i.e., $\mathcal{V} \subset R$, which will manage the weight sum. Each multivariate region $r \in R \setminus \mathcal{V}$ is associated with a weight vector $\boldsymbol{w}_r$ and each univariate region is associated with a weight

$$w_v \;=\; \max(0,\; 1 - w_v^{\Delta}) \qquad \text{where} \qquad w_v^{\Delta} \;=\; \sum_{\substack{r \in R \setminus \mathcal{V} \\ \text{sc}(r) \ni v}} \boldsymbol{w}_r^v \tag{2.17}$$

accumulates the weights associated with RV $v \in \mathcal{V}$ on each multivariate factor. Thus, when

$w_v^{\Delta} \leq 1$ we have that the total weight associated with RV $v \in \mathcal{V}$ sums to one, and otherwise will exceed one. Note that when the total weight associated with v exceeds one, the bound is valid, but loose; when fully optimized, the weights sum to exactly one.

2.5.2.1 Gradient

Optimization of the GDD objective requires its gradient. We use the same notation for the gradient with respect to the GDD-objective $L_G^{GDD}(\boldsymbol{\delta}, \boldsymbol{w})$ as we did for the gradient with respect to the WMB-objective $L_G^{WMB}(\boldsymbol{\delta}_{S_{\leftrightarrow}}, \boldsymbol{w})$ as there will be no ambiguity. That is, the gradient at each region is a pseudo-marginal (the precise computation is given in Ping et al. [2015])

$$\mu_r^{\delta}(x_r) \;:=\; \frac{\partial\, L_G^{GDD}(\boldsymbol{\delta}, \boldsymbol{w})}{\partial\, \theta_r^{\delta}(x_r)}$$

and the sum of these pseudo-marginals is equal to one, i.e., $\sum_{x_r}\; \mu_r^{\delta}(x_r) = 1$. The gradient with respect to the cost-shifting factor is a marginalization over a subset of RVs

$$\mu_{r \to s}^{\delta}(x_s) \;:=\; \frac{\partial\, L_G^{GDD}(\boldsymbol{\delta}, \boldsymbol{w})}{\partial\, \delta_{rs}(x_s)} \;=\; \sum_{x_{r \setminus s}} \mu_r^{\delta}(x_r)$$

The gradient of the objective with respect to a horizontal separator s incident to RV-region v and region r is a difference of incident pseudo-marginals

$$g_s^{\delta}(x_s) \;:=\; \frac{\partial\, L_G^{GDD}(\boldsymbol{\delta}, \boldsymbol{w})}{\partial\, \delta_s(x_s)} \;=\; \mu_{r \to s}^{\delta}(x_s) - \mu_v^{\delta}(x_v) \tag{2.18}$$

since the cost-shifting factor δ_s appears in the first region with a positive sign and the second with a negative sign (and $x_s = x_v$ since the separator s has scope $\mathrm{sc}(s) = v$).

Algorithm 7 Anytime inference for dense tabular factors

Input: An initial region graph G^0, a function to select a higher cost and higher accuracy graph `Transition`, a number of inference sweeps on each graph I, and the available memory and computational cost τ_{mem} and τ_{cpu}.

Output: Inference summary as pairs of objective evaluation and iteration cost $\{(L_{t;i},\ c_{t;i}) : t \in [m'],\ i \in [I]^+\}$.

Initialize: $t \leftarrow 0$

while $\left(\sum_{t' \in [t-1]} \sum_{i \in [I]^+} c_{t';i} \leq \tau_{CPU}\right) \quad \wedge \quad \neg\texttt{isJTree}(G^t)$ **do**

`Step 1` **(Inference):** Inference hot-started at the old parameterization:

$$(\delta^{t;I},\ L_{t;\cdot}, c_{t;\cdot}) = \texttt{Infer}(G^t, \delta^{t-1;I};\ I)$$

`Step 2` **(Transition):** Transition to a graph with higher cost and accuracy:

$$G^{t+1} \ = \ \texttt{Transition}(G^t,\ \tau_{mem})$$

`Step 3`: $t \leftarrow t + 1$;

end while

2.5.2.2 Relation to Weighted-Mini-Bucket

The decomposed GDD-objective represents the same class of bounds as the tree sub-structured WMB-objective. GDD was originally proposed [Ping et al., 2015] to address convergence issues with the weighted mini-bucket optimization used in Liu and Ihler [2011], which are especially severe in marginal-MAP problems. However, for summation problems WMB optimization is often quicker since its forward pass computes a closed form solution to the large subset of variational parameters along the WMB tree.

2.6 Anytime Inference

The quality of an approximate inference relaxation depends heavily on its choice of inference structure. For PGMs with dense factor tables (this chapter), we must choose the subsets of RVs (region scope) used in the relaxation; in our novel work in this thesis (Chapters 4–6), the selection of symmetric groups of factors is the fundamental choice.

However, often, a high quality inference structure cannot be selected *a priori*. Instead, it is desirable to use preliminary inference estimates to select the terms that appear most promising to improve the inference objective. For decomposition bounds in PGMs, several works join groups of factors such that the MAP dual decomposition bound is decreased as much as possible [e.g., Batra et al., 2011, Sontag et al., 2012]. For sum-inference tasks, Forouzan and Ihler [2015] performs a greedy joining procedure within a weighted mini-bucket elimination framework; the basic idea is to select mini-buckets by joining regions within a bucket that improves a local upper bound on the (tree-structured) WMB bound as much as possible. For symmetric PGMs (introduced in the next chapter), coarse-to-fine methods such as [Sarkhel et al., 2015, Habeeb et al., 2017] and our novel work in Chapters 4–6, break inference symmetries that appear to improve the inference objective as much as possible.

Anytime inference provides a general framework for incrementally improving an inference approximation (see Algorithm 7). It alternates two steps of (`Step 1`) performing iterative inference on a fixed region graph with (`Step 2`) transitioning to a new graph structure with higher cost and higher accuracy (where the cost must be below the memory bound τ_{mem}); ideally preliminary inference estimates are used to select the transition. The anytime inference framework outputs inference estimates at varying cost/accuracy tradeoffs. This in itself is useful in many practical settings where the computational resources are not known *a priori* and inference estimates may be queried at any time. In Algorithm 7, the bound on the inference objective and the computational cost incurred by the inference iteration are recorded as a pair $(L_{t;i}, c_{t;i})$ for the tth graph in the transition sequence and the ith inference sweep on that graph (in this thesis, we perform a fixed number of inference iterations on each graph specified by a hyperparameter I). The algorithm is terminated when computational resources (specified by τ_{CPU}) are exhausted or when an exact inference structure (here, the junction tree) is reached.

When upper bound frameworks are used for inference (such as dual decomposition for MAP

inference or WMB or GDD for sum-inference), the anytime inference algorithm monotonically improves the bound. Consequently, the quality of two anytime inference algorithms can be compared by plotting their reported cost-accuracy tradeoffs (where lower value of the inference objective is better). Each of the novel works in this thesis, Chapters 4–6, report results in this form.

Chapter 3

Symmetric PGMs

In the previous chapter, we discussed inference in standard PGMs with dense factor tables. In this chapter, we discuss classes of *symmetric* PGMs where groups of objects in the model are indistinguishable. Two types of symmetry arise regularly in practice: first, a model with *state space symmetry* contains factors with groups of states that have identical values; second, a model with *factor symmetry* contains groups of factors that have identical factor tables.

Symmetry provides another type of structure (in addition to the factorization structure of the PGM) that can be exploited to perform inference efficiently. In this chapter, we review *lifted inference* which, broadly, refers to any inference technique that exploits exact model symmetry. Since exact symmetry rarely arises in practice, there is a great need for methods that perform inference in models with *approximate symmetry*. We review methods that have been developed for coping with asymmetry, and highlight some of their shortcomings that motivate our novel work in subsequent chapters.

Although our focus in this chapter is to describe existing methods and approximations for reasoning about models with symmetry or approximate symmetry, some of the conceptual *frameworks* in which we describe these ideas are novel to our work, and are included here in order to be consistent with, and highlight the parallels and similarities to our contributions

in subsequent chapters. As one example, while our discussion highlights the deep similarities between factor symmetry and state space symmetry and their respective approximate inference methods, these connections are not widely noted within the literature. Where possible, we also describe many existing inference algorithms within the framework and using the notation of decomposition inference methods, which serves to clarify their relationships both to each other and to our proposed methods in subsequent chapters.

3.1 Introduction

In the previous chapter, we saw how the factored structure of PGMs leads to efficient inference algorithms. In this chapter, we study PGMs which, in addition to the factorization structure, possess a *symmetric* structure which leads to even more efficient inference algorithms. In particular, we study two types of symmetry that may occur in a graphical model. First, a model with *state space symmetry* contains factors with groups of states that have identical values; for example, a "cardinality factor" constrains the number of `on` RVs in a factor, e.g., to model a binary image prior [Tarlow et al., 2012]. Second, a model with *factor symmetry* contains groups of factors that have identical factor tables; an important class of models with this property is probabilistic first-order logic networks, such as Markov Logic Networks (MLNs) [Richardson and Domingos, 2006], that encode rules with a "for all" relationship. For example, when modeling voting patterns in a social network, we might use a rule such as, "for all pairs of people (A,B), if A and B are friends, then A and B are more likely to vote for the same candidate". In Section 3.2, we formally define models with state space symmetry and factor symmetry, and in Section 3.3, we formally define MLNs.

3.1.1 Lifted Inference

Lifted inference algorithms encompass a broad class of algorithms that exploit symmetry during inference; in this setting, standard inference that does not exploit symmetry is called *ground inference*. The basic idea of lifted inference is to first identify exact symmetries that would arise if a ground inference procedure were run, then perform inference more efficiently on a *lifted graph* which represents each symmetry group using a single node. In the *ideal* case, the model contains groups of symmetric objects that perfectly align with each other, and hence the inference quantities also possess these symmetries. For example, in MLN models without evidence, we can reason about entire groups of problem objects as if they were a single object, e.g., in the social network example, we can (roughly) reason about the group of all people as if it were a single person.

Over the past decade, lifted inference procedures for most popular ground inference algorithms have been developed. These include lifted variants of belief propagation [Singla and Domingos, 2008], tree-reweighted inference [Bui et al., 2014], generalized BP [Apsel, 2016, Smith et al., 2016], high-order MAP inference [Mladenov et al., 2014b], dual convex entropy methods [Mladenov and Kersting, 2015], relax-compensate-recover [Van den Broeck et al., 2012], and linear programming [Mladenov et al., 2012, 2014a]. For exact (unrelaxed) inference, lifted variable elimination algorithms have been developed [Poole, 2003, Braz et al., 2005, Milch et al., 2008, Taghipour et al., 2012]; another work that performs unrelaxed infernce is Bui et al. [2012] that identifies exact (non-relaxed) symmetries via graph automorphism algorithms which define inference symmetry groups. Lastly, lifted variants of Markov chain Monte Carlo algorithms such as Gibbs sampling [Venugopal and Gogate, 2012] and importance sampling [Venugopal and Gogate, 2014b] have also been developed.

Unfortunately this ideal case, where the model is composed of perfectly aligned symmetry groups, rarely occurs in practice. In MLN models, for example, we often observe different

information about each individual ground object (e.g., that some pairs of people are friends) and hence no two objects can be treated identically during inference. In this case, lifted inference deteriorates into ground inference, negating all potential computational gain of the symmetric representation. This effect of breaking model symmetries is sometimes referred to as *shattering*.

In Section 3.4, we present lifted inference and the shattering problem for models with factor symmetries. Our presentation is couched within a decomposition-based inference framework that we will make use of on in later chapters. In Section 3.5, we present lifted inference and the shattering problem for models with state space symmetries. To do so, we develop a graphical depiction of the *states* of the factors, and a corresponding lifted graph, which will be helpful in developing our methods for *Adaptive Hierarchical State Space Partitions* in Chapter 6. Lastly, our presentation is meant to highlight the deep similarities between inference with state space symmetry and factor symmetry. This connection is not widely noted in the literature but leads to important insights for handling asymmetry in this thesis.

3.1.2 Coping with Asymmetry

Because of the shattering effect, in any practical scenario, the key question is whether or not useful inference quantities can be accurately approximated with symmetric terms. A considerable amount of effort has gone into solving this problem; however, it remains still largely open. Broadly, the algorithms that have been proposed to cope with asymmetry fall into a few categories, such as **projected message passing** methods, which compute the (asymmetric) inference messages, then project them back into a more tractable (symmetric) functional form, or **over-symmetric approximate models**, which create a perturbed model that has more symmetry than the original. Both these approaches control computational cost, but incur error which is difficult to analyze. An emerging class of methods are **coarse-to-fine** approximations, which solve a coarse approximation to the inference prob-

Concept	State Space Symmetry	Factor Symmetry
Dense representation	Dense factor table.	Ground PGM.
Symmetric functional form	Quantized factor table. (Section 3.2.1)	Lifted RV and Lifted Factor. (Section 3.2.2)

(a) Model Representation

Concept	State Space Symmetry	Factor Symmetry
Projected Message Passing	ADD message projection [Gogate and Domingos, 2013].	Lifted Approx. BP [Singla et al., 2014].
Over-Symmetric Approximate Model	Cluster RV domains to define coarse tabular factors. (Section 3.7.1)	Cluster logical domain defines small MLN coarse evidence. (Section 3.7.2)
Coarse-to-Fine Small to Large scope	Split RV domains, join to form large regions. (Chapter 6)	Split logical domains, join to form large lifted factors. (Chapter 5)

(b) Inference: Coping with Asymmetries

Figure 3.1: Tables illustrate similarity between models with state space symmetry and models with factor symmetry. The top table specifies concepts dealing with model representation. The bottom table specifies methods for coping with inference asymmetry. Each table cell contains a brief description and a reference or Section number where further details are discussed.

lem, then use this solution to instantiate a finer-resolution approximation with higher cost but higher accuracy. These methods are appealing since they control the computational cost, while also maintaining a concrete connection to the original problem. Interestingly, these same classes of methods have been used, in separate communities, for coping with both factor asymmetry and state space asymmetry. However, their similarity has not previously been widely appreciated; we highlight this similarity in Table 3.1.

In Section 3.6, we discuss these methods in more detail and highlight some of their relative strengths and weaknesses; we also present desiderata for an ideal framework, which guides the development of our novel algorithms in subsequent chapters. In Section 3.7, we discuss *domain clustering* in detail since it forms the basis for our work in later chapters.

3.1.3 Aggregate Symmetry

Finally, in Section 3.8 we discuss an important identity whereby symmetric groups of factors are joined (summed) together to form a large "counting" factor, i.e., a factor with significant state space symmetry. This identity is well-known in the MLN literature; in the context of our work, it illustrates a deep connection between factor symmetry and state-space symmetry when higher-order variable interactions are included in the inference procedure. This connection inspired, and is then generalized by, our contributions on *Adaptive Hierarchical State Space Partitions* in Chapter 6.

3.2 Symmetric Models

A set of objects is called *symmetric* if each element in the set is indistinguishable from the others in the set. In this setting, a *lifted* object is said to be associated with the symmetric set of *ground* objects. In this thesis, we study two different types of symmetric sets of objects: first, where *states* within a factor are symmetric, and second, where *factors* (in the global model) are symmetric. We discuss each of these two cases in turn.

3.2.1 State Space Symmetry

A *Lifted-State-Space-factor (LSS-factor)* $\bar{\theta}_\alpha : \bar{\mathcal{D}}_\alpha \to \mathbb{R}$ represents state space symmetry in an associated ground factor $\theta_\alpha : \mathcal{D}_\alpha \to \mathbb{R}$. The domain of the LSS-factor is a set of *lifted states* $\bar{\mathcal{D}}_\alpha$ where each lifted state $y_\alpha \in \bar{\mathcal{D}}_\alpha$ is associated with a set of ground states $\mathrm{gr}(y_\alpha) \subset \mathcal{D}_\alpha$; collectively, the sets of ground states $\mathrm{gr}(\bar{\mathcal{D}}_\alpha) := \{\mathrm{gr}(y_\alpha) : y_\alpha \in \bar{\mathcal{D}}_\alpha\}$ partition the factor's domain $\mathcal{D}_\alpha$. The ground factor assigns the same value to all ground states associated with the same lifted state, i.e.,

$$(\forall\ y_\alpha \in \bar{\mathcal{D}}_\alpha,\ x_\alpha \in \mathrm{gr}(y_\alpha)), \qquad \theta_\alpha(x_\alpha)\ =\ \bar{\theta}_\alpha(y_\alpha). \tag{3.1}$$

x_1	x_2	x_3	$\#x_{1:3}$	$\theta(x)$
0	0	0	0	a
0	0	1	1	b
0	1	0	1	b
0	1	1	2	c
1	0	0	1	b
1	0	1	2	c
1	1	0	2	c
1	1	1	3	d

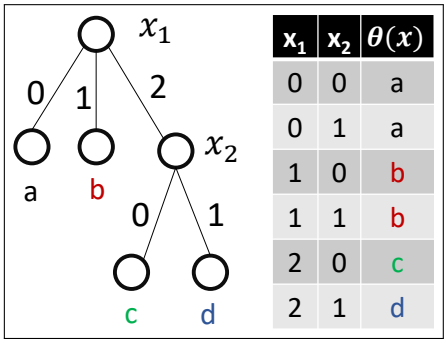
(a) Counting factor

x_1	x_2	x_3	$\sigma_\alpha(x_\alpha)$	$\theta_\alpha(x_\alpha)$
F	F	F	T	ω
F	F	T	T	ω
F	T	F	T	ω
F	T	T	T	ω
T	F	F	T	ω
T	F	T	F	0
T	T	F	F	0
T	T	T	T	ω

(b) Probabilistic logic

$$\theta(x) = \left[\begin{array}{cc|c} a & a & c \\ a & a & c \\ \hline b & b & d \end{array}\right]$$

(c) MD-LSS-factor symmetry



x_1	x_2	$\theta(x)$
0	0	a
0	1	a
1	0	b
1	1	b
2	0	c
2	1	d

(d) Algebraic Decision Diagram

Figure 3.2: Example factors with state space symmetry. (a) Counting factor whose value depends on the number of `on` states, $\#x_{1:n} = \sum_{i=1}^{n} x_i$, (b) Probabilistic logic factor associated with logical sentence $\sigma_\alpha(x_\alpha) = x_1 \Rightarrow (x_2 \Leftrightarrow x_3)$ (with scope $\alpha = \{1, 2, 3\}$) and weight ω. (c) Ground factor associated with a MD-LSS-factor in which states within each RV are treated identically (here, states 0 and 1 of each RV are identical). This gives rise to state space symmetry defined over hypercubes. (d) Algebraic Decision Diagram where each joint state's value is obtained by traversing the tree to the leaves, at each node deciding which branch to go down by testing a single RV for the value labeled on its edge.

3.2.1.1 Examples

Factors with state space symmetry are sometimes called quantized [Gogate and Domingos, 2013] or coarse [Sontag et al., 2009] factors in previous work. Such factors have been used to efficiently represent structured model factors and/or approximate inference terms. Some examples, illustrated in Figure 3.2, include:

(a) Counting factors over binary RVs, whose value depends only on the number of `on` RVs

in a group. Some tasks that use counting factors include worker scheduling problems (say, to encode a requirement that at least three workers are assigned to a given task), and computer vision problems with binary variables like foreground/background segmentation (to constraint the fraction of pixels in a patch that can be assigned to the foreground) [Tarlow et al., 2012].

(b) Probabilistic logic extends standard (hard) logic by associating a logical truth-table with a weight ω; this defines a factor table with value ω if the logical sentence evaluates to `True` and value 0 if the logical sentence evaluates to `False`. Markov Logic Networks [Richardson and Domingos, 2006] provide a popular language for expressing probabilistic logic, and are defined Section 3.3 and used throughout this thesis.

(c) A multidimensional-LSS-factor (MD-LSS-factor) is defined by symmetry in the states of each RV; this partitions the factor into axis-parallel hypercubes of equal value. In the depicted example, state groups $\{0, 1\}$ and $\{2\}$ are identical in both RVs. Factors of this form are used to represent discretizations of continuous problems [Isard et al., 2009] or to represent a grouping of states in discrete problems with large RV domains (e.g., [Sontag et al., 2009] groups states representing molecular bond angles in a protein design problem). Factors of this form give rise to efficient lifted inference computations, which we discuss in Section 3.7.1. The central idea of our work on *Adaptive Hierarchical State Space Partitions* in Chapter 6 is to build high-order inference terms that exploit this type of symmetry.

(d) An Algebraic Decision Diagram (ADD) is a tree-structured graph where the path from the root to a leaf consists of a set of tests on the value of one RV within a joint state. All joint states associated with a leaf node are assigned the value associated with the leaf. In the example shown, the state $(X_1, X_2) = (2, 1)$ is assigned to the leaf in the bottom right and has value d. Factors of this form were used by Gogate and Domingos [2013] to represent the result of approximate message passing operations.

3.2.2 Factor Symmetry

A *lifted factor* $\bar{\theta}_{\bar{\alpha}}$ is equal to the sum of identical ground factors

$$\bar{\theta}_{\bar{\alpha}}(\bar{x}_{\bar{\alpha}}) \;=\; \sum_{\alpha \in \mathrm{gr}(\bar{\alpha})} \theta_{\bar{\alpha}}(x_\alpha) \tag{3.2}$$

where its *lifted scope symbol* $\bar{\alpha}$ is associated a set of ground scopes $\mathrm{gr}(\bar{\alpha}) \subseteq \mathcal{P}(\mathcal{V})$, and its template factor $\theta_{\bar{\alpha}} : \mathcal{D}_{\bar{\alpha}} \to \mathbb{R}$ is equal to all associated ground factors, i.e.,

$$(\forall \alpha \in \mathrm{gr}(\bar{\alpha}),\;\; x_\alpha \in \mathcal{D}_{\bar{\alpha}}) \qquad \theta_\alpha(x_\alpha) \;=\; \theta_{\bar{\alpha}}(x_\alpha) \tag{3.3}$$

where each ground factor has domain equal to the template domain, i.e., $\mathcal{D}_{\bar{\alpha}} = \mathcal{D}_\alpha$ for all $\alpha \in \mathrm{gr}(\bar{\alpha})$. Note the subtle difference in notation: a lifted factor has a bar over its base symbol $\bar{\theta}$ and a bar over its subscript index $\bar{\alpha}$, while a template factor has no bar over its base symbol θ but has a bar over its subscript index $\bar{\alpha}$, and a ground factor has no bar over either its base symbol θ or its subscript index α.

A model defined via lifted factors has a probability density function $\exp(\boldsymbol{\theta}(x))$ that can be defined solely in terms of the lifted factors, i.e.,

$$\boldsymbol{\theta}(x) \;=\; \sum_{\alpha \in \mathcal{I}} \theta_\alpha(x_\alpha) \tag{3.4}$$

$$\;=\; \sum_{\bar{\alpha} \in \bar{\mathcal{I}}} \sum_{\alpha \in \mathrm{gr}(\bar{\alpha})} \theta_{\bar{\alpha}}(x_\alpha) \;=\; \sum_{\bar{\alpha} \in \bar{\mathcal{I}}} \bar{\theta}_{\bar{\alpha}}(\bar{x}_{\bar{\alpha}}) \tag{3.5}$$

where $\bar{\mathcal{I}}$ is a set of lifted scope symbols.

It will sometimes be useful to define a set of lifted RVs $\bar{\mathcal{V}}$ which represent identically behaving ground RVs. That is, each lifted RV $\bar{v} \in \bar{\mathcal{V}}$ is associated with a set of ground RVs $\mathrm{gr}(\bar{v}) \subseteq \mathcal{V}$ and a template domain $\mathcal{D}_{\bar{v}}$ that is equal to the domain of each ground RV, i.e., $\mathcal{D}_v = \mathcal{D}_{\bar{v}}$

for each $v \in \text{gr}(\bar{v})$. Collectively, the set of groundings associated with lifted RVs, $\text{gr}(\bar{\mathcal{V}}) := \{\text{gr}(\bar{v}) : \bar{v} \in \bar{\mathcal{V}}\}$, partition the set of ground RVs $\mathcal{V}$.

3.2.2.1 Examples

We now describe some example applications of PGMs with symmetric factors.

Computer Vision Many low-level computer vision problems are represented with a grid-structured MRF which captures dependence between neighboring pixels, where each pixel is associated with a RV. Often, the relationship between pixels is translationally invariant and thus is represented by a single, identical (i.e., symmetric) factor function between any neighboring pair of pixels. For example, in stereo vision, the goal is to estimate the depth of the object at each pixel using a pair of images taken by adjacent cameras. Objects that are nearer to the camera will appear to "shift" between the left and right images by a greater horizontal distance than objects that are far from the camera. A basic model might compute a local disparity score for each pixel, by measuring the visual similarity of a patch on the left to a patch on the right for each possible shift, giving a univariate potential for each RV. Then, a translationally invariant smoothness term enforces a constraint that two neighboring pixels are likely to have the same or similar disparity. This model is a PGM with distribution

$$\boldsymbol{\theta}(x) \;=\; \sum_{v \in \mathcal{V}} \theta_v(x_v) \;+\; \sum_{e \in \text{gr}(\bar{\mathcal{E}})} \theta_{\bar{\mathcal{E}}}(x_e) \tag{3.6}$$

where the first term represents a local disparity score and the second term represents the translationally invariant smoothness term and $\text{gr}(\bar{\mathcal{E}})$ is the set of all neighboring pixels in the image. Often the template factor $\theta_{\bar{\mathcal{E}}}(x_e)$ is a function of difference in disparity $g_{\bar{\mathcal{E}}}(x_v - x_{v'})$ where $e = (v, v')$, and so may also have some state-space symmetry in addition to the factor symmetry from translational invariance.

Ising Model The Ising model [e.g., McCoy and Wu, 2014] is a classical model in statistical mechanics. An Ising model consists of discrete (binary) RVs that represent magnetic dipole moments of atomic "spins" that can be in one of two states (+1 or -1). The connectivity of the graph defines which atoms can interact with each other; for example, in the basic two-dimensional Ising lattice, each spin interacts with four neighbors. In attractive Ising models, neighboring spins that agree have a lower energy than those that disagree. This is represented with a graphical model of the same form as Eq. (3.6) above.

Probabilistic First-Order Logic First-order logic (FOL) is widely used in mathematics, philosophy and artificial intelligence to represent knowledge efficiently using "for all" statements [Russell and Norvig, 2002]. Probabilistic first-order logic (probabilistic FOL) uses a FOL-like representation to define a PGM with symmetric factors. A classic example is modeling the smoking habits of people within a social network [Richardson and Domingos, 2006]. This *friend-smoker-MLN* is shown in Figure 3.3. Figure 3.3a shows the RVs used by the model (friendship relations between all pairs of people, smoking attribute of each person, cancer attribute of each person) while Figure **??** shows the weighted FOL-formulas used by the model, e.g., $(\forall\ l_1 \neq l_2 \in \mathrm{gr}(\bar{\Delta}_{\mathrm{ppl}}))\ \mathtt{Fr}(l_1, l_2) \Rightarrow (\mathtt{Sm}(l_1) \Leftrightarrow \mathtt{Sm}(l_2))$ encodes the rule that if two people are friends, then they probably both smoke or both do not smoke (for completeness, in Figure 3.4a we show a ground probabilistic logic factor associated with this sentence). Figure 3.3c shows a lifted region graph (which will be discussed in detail later) that is used to efficiently represent and reason about FOL-formulas and Figure 3.3d shows the associated ground region graph which explicitly represents the rules for each person or group of people.

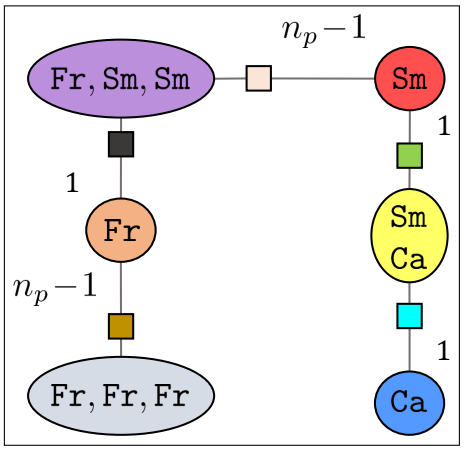
In practice, models contain both (symmetric) statements in probabilistic FOL and (asymmetric) statements about each individual, called *evidence*, e.g., the observation that Ana and Bob are friends. In Figure 3.4b we show an example set of evidence observations. The evidence, as we will see in subsequent sections, breaks the symmetry present in the FOL-

English	Predicate	Argument Type	Ground RVs
Friendship Relation	$\texttt{Fr}$	$(\bar{\Delta}_{\text{ppl}}, \bar{\Delta}_{\text{ppl}})$	$\{\texttt{Fr}(L_1, L_2) : \ L_1 \neq L_2 \in \text{gr}(\bar{\Delta}_{\text{ppl}})\}$
Smoke Attribute	$\texttt{Sm}$	$(\bar{\Delta}_{\text{ppl}})$	$\{\texttt{Sm}(L_1) : \ L_1 \in \text{gr}(\bar{\Delta}_{\text{ppl}})\}$
Cancer Attribute	$\texttt{Ca}$	$(\bar{\Delta}_{\text{ppl}})$	$\{\texttt{Ca}(L_1) : \ L_1 \in \text{gr}(\bar{\Delta}_{\text{ppl}})\}$

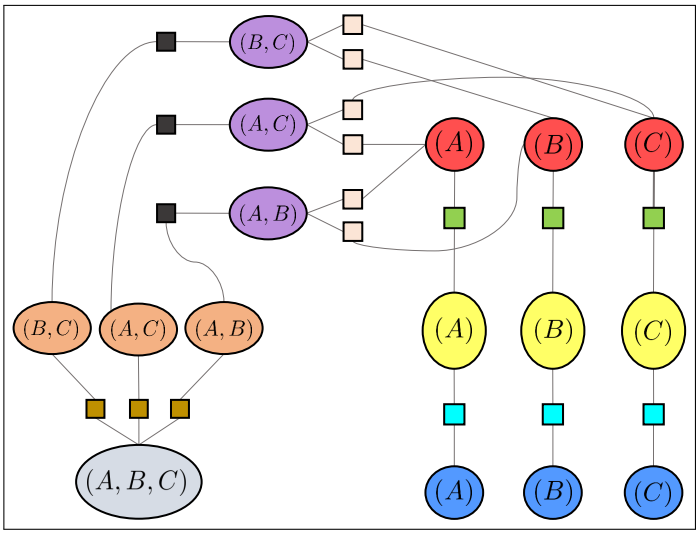
(a) Predicates and associated sets of ground RVs.

English	First-order logic	Weight
Friends of friends are probably friends.	$(\forall\ l_1 \neq l_2 \neq l_3 \in \text{gr}(\bar{\Delta}_{\text{ppl}}))\ \ (\texttt{Fr}(l_1, l_2) \wedge \texttt{Fr}(l_2, l_3)) \Rightarrow \texttt{Fr}(l_3, l_1)$	$+2.2$
Friends probably both smoke or don't smoke.	$(\forall\ l_1 \neq l_2 \in \text{gr}(\bar{\Delta}_{\text{ppl}}))\ \ \texttt{Fr}(l_1, l_2) \Rightarrow (\texttt{Sm}(l_1) \Leftrightarrow \texttt{Sm}(l_2))$	$+1.6$
A person who smokes probably gets cancer.	$(\forall\ l_1 \in \text{gr}(\bar{\Delta}_{\text{ppl}}))\ \ \texttt{Sm}(l_1) \Rightarrow \texttt{Ca}(l_1)$	$+0.5$
Two arbitrary people are probably not friends.	$(\forall\ l_1 \neq l_2 \in \text{gr}(\bar{\Delta}_{\text{ppl}}))\ \ \texttt{Fr}(l_1, l_2)$	-6.1
Friendship is symmetric.	$(\forall\ l_1 \neq l_2 \in \text{gr}(\bar{\Delta}_{\text{ppl}}))\ \ \texttt{Fr}(l_1, l_2) \nLeftrightarrow \texttt{Fr}(l_2, l_1)$	$-\inf$

(b) Weighted first-order formulas.



(c) lifted region graph



(d) ground region graph

Figure 3.3: Example friend-smoker-MLN [Richardson and Domingos, 2006] (a) Predicates and ground RVs, and (b) Weighted first-order formulas. The bottom pair of figures show the associated (c) lifted region graph where each lifted region corresponds to a FOL-formula and is labeled with the predicates used by the formula, and (d) ground region graph where each ground region corresponds to a grounding of the lifted region of the same color. Each ground region is labeled only with its logical constants for compactness. For example, the yellow region labeled "A" corresponds to the grounding $\texttt{Sm}(A) \Rightarrow \texttt{Ca}(A)$ and the purple region labeled "A,B" corresponds to the ground sentence $\texttt{Fr}(A, B) \Rightarrow (\texttt{Sm}(A) \Leftrightarrow \texttt{Sm}(B))$.

$\boldsymbol{x}_{\texttt{Fr}(\boldsymbol{A},\boldsymbol{B})}$	$\boldsymbol{x}_{\texttt{Sm}(\boldsymbol{A})}$	$\boldsymbol{x}_{\texttt{Sm}(\boldsymbol{B})}$	$\boldsymbol{\sigma}_{\bar{\alpha}}(\boldsymbol{x}_{\boldsymbol{\alpha}})$	$\boldsymbol{\theta}_{\bar{\alpha}}(\boldsymbol{x}_{\boldsymbol{\alpha}})$
F	F	F	T	+1.6
⋮	⋮	⋮	⋮	⋮
T	T	T	F	0
T	T	T	T	+1.6

(a) Template factor representing friendship and smoking habits

Evidence Formulas
$x_{\texttt{Fr}(\text{Ana,Bob})} \Leftrightarrow \texttt{True}$
$x_{\texttt{Fr}(\text{Bob,Cathy})} \Leftrightarrow \texttt{True}$
$x_{\texttt{Sm}(\text{Ana})} \Leftrightarrow \texttt{True}$
$x_{\texttt{Sm}(\text{Bob})} \Leftrightarrow \texttt{False}$
$x_{\texttt{Ca}(\text{Bob})} \Leftrightarrow \texttt{True}$
⋮
$x_{\texttt{Ca}(\text{Cathy})} \Leftrightarrow \texttt{False}$

(b) Example Evidence

Figure 3.4: Details for friend-smoker-MLN model. Panel (a) shows the template factor table associated with the formula $\sigma_{\bar{\alpha}}(x_{\texttt{Fr}(A,B)}, x_{\texttt{Sm}(A)}, x_{\texttt{Sm}(B)}) = x_{\texttt{Fr}(A,B)} \Rightarrow (x_{\texttt{Sm}(A)} \Leftrightarrow x_{\texttt{Sm}(B)})$ with weight +1.6. Panel (b) shows an example of observed evidence associated with the MLN.

formulas and prevents lifted inference from being applied directly.

Other languages In the next section, we give a detailed overview probabilistic FOL represented using Markov Logic Networks (MLNs) [Richardson and Domingos, 2006] which will be used in subsequent chapters of this thesis. We emphasize that MLNs simply provide a convenient language to represent problems with repetitive model structure. That is, other languages that represent repetitive structure using a combination of first-order-logic and probability (e.g., Blog [Milch et al., 2007], IBAL [Pfeffer, 2001], Problog [De Raedt et al., 2007], PSL [Kimmig et al., 2012]) could be used instead and should not fundamentally alter the strategies for approximate inference developed in this thesis.

3.3 Markov Logic Networks

In this section, we discuss Markov Logic Networks (MLNs) [Richardson and Domingos, 2006] which express a symmetric model using probabilistic first-order logic. One of the central motivations for MLNs is to unify first-order logic with probability theory, a long-standing goal in artificial intelligence with applications to many domains [Russell and Norvig, 2002].

We begin in Section 3.3.1, by discussing (hard) logic which provides the foundation for many knowledge representation systems. In Section 3.3.1.1, we discuss *relational logic models* which index variables in real-world problems in a special way. In particular, many logical systems are used to model a set of objects in the world (e.g., the set of all people), and each variable in logical model represents an *attribute* of an object (e.g., Ana smokes, or Bob has cancer), or a *relationship* between objects (e.g., Ana and Cathy are friends). In Section 3.3.2, we discuss first-order logic (FOL) which efficiently represents a logical rule that is applied over a range of objects, corresponding to a "for all" sentence in English. Finally, in Section 3.3.3, we discuss Markov Logic Networks (MLNs) which are structurally identical to a hard first-order logic (FOL) networks, but the hard logical rules are replaced by probabilistic rules which encourage, rather than require, individual logical sentences to be `True`.

3.3.1 Hard Logic

Logic forms the basis for knowledge representation and inference in many problems [Russell and Norvig, 2002]. There are two propositions with fixed meaning: `True` (or `T`) is the always-true proposition and `False` (or `F`) is the always-false proposition. A *logical sentence* is constructed from simpler sentences, using parentheses and *logical connectives*, which consist of negations $\neg$, conjunctions $\wedge$, disjunctions $\vee$, implication $\Rightarrow$, and equivalence $\Leftrightarrow$ [Russell and Norvig, 2002]. A logical sentence over a set of variables $\alpha \subset \mathcal{V}$ is associated with a *truth table* $\sigma_\alpha : \{\mathtt{T}, \mathtt{F}\}^{|\alpha|} \rightarrow \{\mathtt{T}, \mathtt{F}\}$ that assigns value `T` or `F` to each input. An example logical sentence is $\sigma_\alpha(x_\alpha) = x_1 \Rightarrow (x_2 \Leftrightarrow x_3)$ which means that "if x_1 is true then x_2 and x_3 are both `True` or both `False`", where $\alpha = \{1, 2, 3\}$ is the index of variables used. The truth table for this formula is shown in Figure 3.2b.

A ground (propositional) logic network is a collection of logical sentences. The truth value of the network is equal to the conjunction of truth value of the logical sentences. That is, $\boldsymbol{\sigma}(x) = \bigwedge_{\alpha \in \mathcal{I}} \sigma_\alpha(x_\alpha)$ where $\mathcal{I}$ is the set of scopes of each logical sentence. A typical task

in a logic network is determining satisfiability, that is, determining if any configuration x satisfies all formulas simultaneously [Russell and Norvig, 2002]. Note that a logic network is a special case of a constraint network [Dechter et al., 2003] where each constraint is defined by a logical formula.

3.3.1.1 Relational Logic Models

A logical description is often used to model problems whose variables are named in a special way. In particular, the model contains sets of auxiliary objects and the variables of the problem are defined by attributes associated with each object or by relationships between groups of objects [Russell and Norvig, 2002]. The objects often represent *nouns* in the problem definition, e.g., a set of people, or a set of items that can be purchased, the *attributes* associated with an object often represents a *verb* or *adjective* in the problem definition, e.g., whether or not a person smokes or whether or not he has cancer, and the *relations* among objects often represent *verbs* in the problem definition, e.g., if two people are friends.

Formally, the auxiliary objects and the problem variables are defined as follows:

Auxiliary Objects: $\bar{\boldsymbol{\Delta}}$ is a set of *lifted logical constants* where each $\bar{\Delta} \in \bar{\boldsymbol{\Delta}}$ is associated with a set of ground logical constants $\text{gr}(\bar{\Delta})$ which represent *objects* in the problem. For example, in a social network there is one set of constants $\bar{\boldsymbol{\Delta}} = \{\bar{\Delta}_{\text{ppl}}\}$ where $\text{gr}(\bar{\Delta}_{\text{ppl}}) = \{\text{Ana, Bob, Cathy}\}$ represents the set of all people in the problem.

Predicates: $\boldsymbol{P}$ is a set of predicates where each predicate $\bar{\boldsymbol{p}} \in \boldsymbol{P}$ is associated with a vector (ordered set with repeats) of lifted logical constants $\bar{\Delta}_{\bar{\boldsymbol{p}}} = (\bar{\Delta}_{\bar{\boldsymbol{p}};1}, \ldots, \bar{\Delta}_{\bar{\boldsymbol{p}};A_{\bar{\boldsymbol{p}}}})$ where $\bar{\Delta}_{\bar{\boldsymbol{p}};i} \in \bar{\boldsymbol{\Delta}}$ is the *type* of the ith argument of the predicate for each $i \in [A_{\bar{\boldsymbol{p}}}]^+$ where $A_{\bar{\boldsymbol{p}}}$ is the *arity* of the predicate. The predicate $\bar{\boldsymbol{p}}$ is associated with ground RVs whose

index is any tuple of distinct ground logical constants of the appropriate type

$$\mathcal{V}_{\bar{\boldsymbol{p}}} \;=\; \{\bar{\boldsymbol{p}}(l_1, \ldots, l_{A_{\bar{\boldsymbol{p}}}}) :\; l_i \in \mathrm{gr}(\bar{\Delta}_{p;i}),\; \texttt{all-diff}(l_1, \ldots, l_{A_{\bar{\boldsymbol{p}}}})\}.$$

where $\texttt{all-diff}(l_1, \ldots, l_k)$ is true if and only if $l_i \neq l_j$ for all i, j.*

An example is the friendship predicate `Fr` which is associated with two copies of people $(\bar{\Delta}_{\text{ppl}}, \bar{\Delta}_{\text{ppl}})$.

A (hard) logic network is defined on the set of variables associated with each predicate, i.e., $\mathcal{V} = \cup_{\bar{\boldsymbol{p}} \in \boldsymbol{P}} \mathcal{V}_{\bar{\boldsymbol{p}}}$. Each formula represents a logical relationship between objects, e.g.,

$$\sigma_\alpha(x_\alpha) \;=\; x_{\texttt{Fr}(\text{Ana,Bob})} \Rightarrow (x_{\texttt{Sm}(\text{Ana})} \Leftrightarrow x_{\texttt{Sm}(\text{Ana})})$$

has scope $\alpha = \{\texttt{Fr}(\text{Ana}, \text{Bob}),\; \texttt{Sm}(\text{Ana}),\; \texttt{Sm}(\text{Bob})\}$ and the formula in English states that "If Ana and Bob are friends, then either both smoke or neither smokes".

3.3.2 First-Order Logic

A First-order logic (FOL) formula applies a propositional logic formula to all objects (or tuples of objects) in the model. An example FOL-formula is

$$(\forall\; l_1 \neq l_2 \in \mathrm{gr}(\bar{\Delta}_{\text{ppl}}))\;\; x_{\texttt{Fr}(l_1, l_2)} \Rightarrow (x_{\texttt{Sm}(l_1)} \Leftrightarrow x_{\texttt{Sm}(l_2)}) \tag{3.7}$$

which states that "For all pairs of people, if the people are friends, then either both smoke or neither smokes". This one FOL-formula efficiently represents $n_p \cdot (n_p - 1)$ propositional formulas for each possible substitution of a pair of people for the logical variables (LVs) (l_1, l_2) in the FOL-formula, where $n_p = |\mathrm{gr}(\bar{\Delta}_{\text{ppl}})|$ is the number of people in the model.

*To model a relationship with duplicate ground logical constants, we can use another predicate without duplicates. For example, rather than $\texttt{Fr}(\text{Ana}, \text{Ana})$ we can use $\texttt{FrSelf}(\text{Ana})$. However, we will not deal with problems of this form in this thesis.

For our work later, we will need a slightly different representation of a FOL-formula that makes the sets of variables in its propositional representation explicit. A FOL-formula is associated with symbol $\bar{\alpha}$, sets of ground logical constants $\text{gr}_{\text{LV}}(\bar{\alpha})$, a predicate-index template $\rho_{\bar{\alpha}}$, and a template truth table $\sigma_{\bar{\alpha}}$ where

Sets of Auxiliary Objects: The set $\text{gr}_{\text{LV}}(\bar{\alpha})$ contains tuples of k_{LV} logical constants where each tuple represents the set of objects (e.g., two people) that participate in the formula. This set could, in theory, contain any tuples of logical constants, but, often, $\bar{\alpha}$ is associated with a set of object types $\bar{\text{sc}}_{\bar{\boldsymbol{\Delta}}}(\bar{\alpha}) = (\bar{\Delta}_{\bar{\alpha};1}, \ldots, \bar{\Delta}_{\bar{\alpha};k_{LV}})$, known as its *logical scope*, and

$$\text{gr}_{\text{LV}}(\bar{\alpha}) \;=\; \{(l_1, \ldots, l_{k_{LV}}) : l_i \in \text{gr}(\bar{\Delta}_{\bar{\alpha};i}),\; \texttt{all-diff}(l_1, \ldots, l_{k_{LV}})\}.$$

is the set of all combinations of elements of the appropriate type, with no repeat elements.

Set of Variable Scopes: The predicate-index template $\rho_{\bar{\alpha}}$ is used to convert each tuple of logical constants $l \in \text{gr}_{\text{LV}}(\bar{\alpha})$ into a set of ground variable as

$$\text{gr}(\bar{\alpha}) = \{\{\bar{\boldsymbol{p}}(l_{\bar{\boldsymbol{I}}_1}, \ldots, l_{\bar{\boldsymbol{I}}_{|\bar{\boldsymbol{I}}|}}) :\; (\bar{\boldsymbol{p}}, \bar{\boldsymbol{I}}) \in \rho_{\bar{\alpha}}\} :\; l \in \text{gr}_{\text{LV}}(\bar{\alpha})\}$$

where each element $(\bar{\boldsymbol{p}}, \bar{\boldsymbol{I}}) \in \rho_{\alpha}$, consists of a predicate $\bar{\boldsymbol{p}} \in \boldsymbol{P}$ and a vector of local indices $\bar{\boldsymbol{I}} := (\bar{\boldsymbol{I}}_1, \ldots, \bar{\boldsymbol{I}}_{|\bar{\boldsymbol{I}}|})$ where $\bar{\boldsymbol{I}}_i \in [k_{LV}]^+$ is a valid integer index into a tuple of k_{LV} logical constants. We denote $k_{RV} := |\rho_{\bar{\alpha}}|$ as the number of variables in each set.

Template Truth Table: The template truth table $\sigma_{\bar{\alpha}} : \{\texttt{T}, \texttt{F}\}^{k_{RV}} \rightarrow \{\texttt{T}, \texttt{F}\}$ is applied to each set of ground variables such that the FOL formula

$$\bar{\sigma}_{\bar{\alpha}}(x_{\bar{\alpha}}) \;=\; \bigwedge_{\alpha \in \text{gr}(\bar{\alpha})} \sigma_{\bar{\alpha}}(x_{\alpha})$$

is `True` if and only if each setting of the ground variables is `True`.

As an example to illustrate this notation, the FOL-sentence in Eq. (3.7) uses types $(\bar{\Delta}_{\text{ppl}}, \bar{\Delta}_{\text{ppl}})$ and predicate-index template $\rho_{\bar{\alpha}} = \{(\texttt{Fr}, (1,2)),\ (\texttt{Sm}, (1)),\ (\texttt{Sm}, (2))\}$. One set of ground logical constants $l = (\text{Ana}, \text{Bob})$ is associated with variables $\alpha = \{\texttt{Fr}(\text{Ana}, \text{Bob}),\ \texttt{Sm}(\text{Ana}),\ \texttt{Sm}(\text{Bob})\}$ since $\texttt{Fr}(l_1, l_2) = \texttt{Fr}(\text{Ana}, \text{Bob})$, $\texttt{Sm}(l_1) = \texttt{Sm}(\text{Ana})$, and $\texttt{Sm}(l_2) = \texttt{Sm}(\text{Bob})$.

First-Order Knowledge Base (with Evidence) A first-order knowledge base [Genesereth and Nilsson, 2012] consists of a set of sentences in first-order logic (FOL) as well as sentences in propositional logic. The sentences in propositional logic are often known as the *evidence* and are associated with a single individual, e.g., we may observe that "Ana smokes" or "Bob does not have cancer". Given a first-order knowledge base, the goal is often to determine the configurations that satisfy all FOL-formulas and all propositional formulas. A familly of *lifted inference* algorithms solve this efficiently by reasoning using a first-order representation as much as possible [Russell and Norvig, 2002].

3.3.3 Probabilistic Logic

Although first-order logic (FOL) provides a powerful language for knowledge representation, its utility is limited by the fact that it can only represent hard `True` and `False` values for each formula; consequently, if even one formula evaluates to `False`, the entire network (i.e., the conjunction of FOL-formulas and evidence formulas) evaluates to `False`. *Probabilistic logic networks* remedy this problem by penalizing rather than prohibiting configurations.

A probabilistic first-order logic (FOL) factor is a lifted factor (Eq. (3.2))

$$\bar{\theta}_{\bar{\alpha}}(x_{\bar{\alpha}}) = \sum_{\alpha \in \text{gr}(\bar{\alpha})} \theta_{\bar{\alpha}}(x_{\alpha}) \tag{3.8}$$

where the template factor

$$\theta_{\bar{\alpha}}(\bar{x}_{\bar{\alpha}}) \;=\; \begin{cases} \omega_{\bar{\alpha}} & \text{if } \sigma_{\bar{\alpha}}(\bar{x}_{\bar{\alpha}}) = \texttt{True} \\ 0 & \text{if } \sigma_{\bar{\alpha}}(\bar{x}_{\bar{\alpha}}) = \texttt{False} \end{cases}$$

is a probabilistic logic factor that assigns weight of $\omega_{\bar{\alpha}}$ to a `True` value and a weight of 0 to a `False` value of the template configuration $\bar{x}_{\bar{\alpha}}$.

3.3.3.1 Markov Logic Networks (MLNs)

A Markov Logic Network (MLN) $\mathcal{M} = (\bar{\boldsymbol{\Delta}}, \boldsymbol{P}, \bar{\mathcal{I}}^{FOL}, \boldsymbol{\omega}, \boldsymbol{E})$ is a tuple where the typed logical constants $\bar{\boldsymbol{\Delta}}$ and predicates $\boldsymbol{P}$ define relational RVs, $\bar{\mathcal{I}}^{FOL}$ defines probabilistic first-order logic (FOL) formulas associated with weights $\omega_{\bar{\alpha}}$ for each $\bar{\alpha} \in \bar{\mathcal{I}}^{FOL}$, and $\boldsymbol{E} = \{\boldsymbol{E}_{\bar{\boldsymbol{p}}} : \bar{\boldsymbol{p}} \in \boldsymbol{P}\}$ where $E_{\bar{\boldsymbol{p}}} \subset \mathcal{V}_{\bar{\boldsymbol{p}}}$ is a subset of RVs with observed *evidence*.

A MLN has a (log-unnormalized) probability distribution equal to the sum of all probabilistic FOL-formulas and evidence

$$\boldsymbol{\theta}(x) \;=\; \sum_{\bar{\alpha} \in \bar{\mathcal{I}}^{FOL}} \bar{\theta}_{\bar{\alpha}}(x_{\alpha}) \;+\; \sum_{\bar{\boldsymbol{p}} \in \boldsymbol{P}} \left(\bar{\theta}_{\tilde{E}_{\bar{\boldsymbol{p}}}}(\bar{x}_{\tilde{E}_{\bar{\boldsymbol{p}}}}) \;+\; \sum_{v \in \boldsymbol{E}_{\bar{\boldsymbol{p}}}} \theta_v(x_v) \right) \tag{3.9}$$

where $\bar{\theta}_{\tilde{E}_{\bar{\boldsymbol{p}}}}(\bar{x}_{\tilde{E}_{\bar{\boldsymbol{p}}}}) = \sum_{v \in \tilde{E}_{\bar{\boldsymbol{p}}}} \theta_{\tilde{E}_{\bar{\boldsymbol{p}}}}(x_v)$ and $\tilde{E}_{\bar{\boldsymbol{p}}} := \mathcal{V}_{\bar{\boldsymbol{p}}} \setminus \boldsymbol{E}_{\bar{\boldsymbol{p}}}$ is the set of RVs associated with predicate $\bar{\boldsymbol{p}}$ without observed evidence. The univariate factors associated with these RVs are either unobserved or they have a default problem-dependent value, e.g., in some problems it is appropriate to assume that a pair of people with no friendship observation are not friends.

3.4 Lifted Inference – Factor Symmetry

Now that we have seen how models with symmetric factors are represented, we turn our attention to the problem of exploiting symmetry during inference, a process known as *lifted inference*. Lifted inference algorithms first identify groups of objects that can be treated identically. This is often done by "simulating" a ground inference algorithm, i.e., by reasoning about which computations are identical rather than performing them numerically. Each group of symmetric regions (or separators) is then represented as a single lifted region (or lifted separator) in a *lifted region graph*, and numerical inference is performed on the lifted region graph. Lifted inference produces the same answer as ground inference but with computational cost proportional to the, ideally much smaller, size of the lifted region graph rather than the size of the ground region graph.

In general, lifted inference is applicable to most algorithms that have been developed for standard ground inference in PGMs, including tree-structured message passing algorithms, decomposition bounds, MCMC sampling, and other strategies (some of these works were mentioned in this chapter's introduction). In this chapter, we present and discuss lifted inference from the point of view of MAP inference using decomposition bounds. This choice gives a concrete connection to algorithms that will be developed later in the thesis, however, many of the ideas for exploiting and identifying symmetry are applicable more broadly to other classes of inference algorithms (such as tree-structured message passing algorithms). In Section 3.4.1, we discuss lifted inference computations with known symmetry groups. Then, in Section 3.4.2, we present a standard iterative coloring algorithm to detect symmetries.

3.4.1 Lifted Objective

Consider a region graph $G = (R, S, E)$ and an associated lifted region graph $\bar{G} = (\bar{R}, \bar{S}, \bar{E})$ defined by a set of *lifted region nodes (lifted regions)* $\bar{R}$, *lifted separator nodes (lifted*

separators) $\bar{S}$, and *lifted edges* $\bar{E}$. Each lifted object represents a set of symmetric ground objects, giving rise to lifted versions of our usual graphical relationships:

Lifted Regions: A lifted region $\bar{r} \in \bar{R}$ is associated with a set of ground regions $\text{gr}(\bar{r}) \subset R$ whose reparameterized factors are each equal to a common *template reparameterized factor* $\bar{\theta}^{\bar{\delta}}_{\bar{r}}$, i.e., $\bar{\theta}^{\bar{\delta}}_{\bar{r}} = \theta^{\delta}_{r}$ for each $r \in \text{gr}(\bar{r})$.

Collectively, the sets of ground regions associated with the lifted regions $\text{gr}(\bar{R}) := \{\text{gr}(\bar{r}) : \bar{r} \in \bar{R}\}$ partition the set of ground regions R.

Lifted Separators: A lifted separator $\bar{s} \in \bar{S}$ is associated with a set of ground separators $\text{gr}(\bar{s}) \subset S$ whose cost-shifting factors are each equal to a common template cost-shifting factor $\delta_{\bar{s}}$, i.e., $\delta_{\bar{s}} = \delta_s$ for each $s \in \text{gr}(\bar{s})$.

Collectively, the sets of ground separators associated with the lifted separators $\text{gr}(\bar{S}) := \{\text{gr}(\bar{s}) : \bar{s} \in \bar{S}\}$ partition the set of ground separators S.

Lifted Edges: A lifted edge $\bar{e} = (\bar{r}, \bar{s}) \in \bar{E}$ connects a lifted region $\bar{r} \in \bar{R}$ to a lifted separator $\bar{s} \in \bar{S}$ and represents all ground edges between the associated ground regions $\text{gr}(\bar{r})$ and ground separators $\text{gr}(\bar{s})$, i.e., $\text{gr}(\bar{e}) = E \cap (\text{gr}(\bar{r}) \times \text{gr}(\bar{s}))$. The lifted edge $\bar{e}$ represents a symmetric relation between a number of ground regions and ground separators; we define a *weight* $M_{\bar{e}}$ that captures the number of such relations represented:

$$M_{\bar{e}} = |N_{\bar{s}}(r)| \qquad \text{where} \qquad N_{\bar{s}}(r) := N(r) \cap \text{gr}(\bar{s}). \tag{3.10}$$

Here, $N_{\bar{s}}(r)$ is the set of neighbors of ground region $r \in \text{gr}(\bar{r})$ associated with the lifted separator $\bar{s}$, so that the lifted edge weight is the number of neighbors of each ground region $r \in \text{gr}(\bar{r})$ associated with the lifted edge $\bar{e}$.

Collectively, the sets of ground edges associated with the lifted edges $\text{gr}(\bar{E}) := \{\text{gr}(\bar{e}) : \bar{e} \in \bar{E}\}$ partition the set of ground edges E. Furthermore, we denote the neighbors

in the lifted region graph as $\bar{N}(\bar{r}) = \{\bar{s} : (\bar{r}, \bar{s}) \in \bar{E}\}$ for a lifted region $\bar{r} \in \bar{R}$, and $\bar{N}(\bar{s}) = \{\bar{r} : (\bar{r}, \bar{s}) \in \bar{E}\}$ for a lifted separator $\bar{s} \in \bar{S}$.

A lifted region graph defined in this way has identical reparameterized factors for each ground region $r \in \text{gr}(\bar{r})$ associated with any lifted region $\bar{r} \in \bar{R}$ as

$$\begin{aligned}
\theta_r^{\delta}(x_r) \;&=\; \theta_r(x_r) \;+\; \sum_{s \in N(r)} \delta_s(x_s) \\
&=\; \theta_r(x_r) \;+\; \sum_{\bar{s} \in \bar{N}(\bar{r})} \sum_{s \in N_{\bar{s}}(r)} \delta_s(x_s) \\
&=\; \theta_{\bar{r}}(\bar{x}_{\bar{r}}) \;+\; \sum_{\bar{s} \in \bar{N}(\bar{r})} M_{(\bar{r},\bar{s})} \cdot \bar{\delta}_{\bar{s}}(\bar{x}_{\bar{s}}) \;=:\; \bar{\theta}_{\bar{r}}^{\bar{\delta}}(\bar{x}_{\bar{r}})
\end{aligned} \tag{3.11}$$

where the second summation groups cost-shifting factors with identical value, and $x_r = \bar{x}_{\bar{r}}$, meaning that the index into the ground factor equals the index into the template factor.

Symmetries in the reparameterized factors can be exploited to compute the ground-DD-objective efficiently as

$$\begin{aligned}
L_G^{DD}(\boldsymbol{\delta}) \;&=\; \sum_{r \in R} \max_{x_r} \; \theta_r^{\delta}(x_r) \;=\; \sum_{\bar{r} \in \bar{R}} \sum_{r \in \text{gr}(\bar{r})} \max_{x_r} \; \theta_r^{\delta}(x_r) \\
&= \sum_{\bar{r} \in \bar{R}} |\text{gr}(\bar{r})| \cdot \max_{\bar{x}_{\bar{r}}} \; \bar{\theta}_{\bar{r}}^{\bar{\delta}}(\bar{x}_{\bar{r}}) \;=:\; \bar{L}_{\bar{G}}^{DD}(\boldsymbol{\delta}_{\bar{S}})
\end{aligned} \tag{3.12}$$

since the maximization at each $r \in \text{gr}(\bar{r})$ is identical and hence is computed just once on a template and then scaled by the number of corresponding ground regions $|\text{gr}(\bar{r})|$. The term $\bar{L}_{\bar{G}}^{DD}(\boldsymbol{\delta}_{\bar{S}})$ is known as the *lifted-DD-objective* and makes this computation mechanism explicit.

3.4.1.1 Optimizing the Lifted Objective

The lifted-DD-objective (Eq. (3.12)) could, in theory, be optimized for any symmetry groups of the inference parameters, i.e., for any grouping of the ground separators $\text{gr}(\bar{S})$. However, for an arbitrary grouping, the optimized lifted-DD-objective (Eq. (3.12)) produces an upper bound

$$\min_{\boldsymbol{\delta}_{\bar{S}}} \bar{L}^{DD}_{\bar{G}}(\boldsymbol{\delta}_{\bar{S}}) \;\geq\; \min_{\boldsymbol{\delta}} L^{DD}_{G}(\boldsymbol{\delta}) \tag{3.13}$$

on the exact ground solution. Our *Lifted GDD Algorithm* in Chapter 4 takes up the idea of controlling the symmetry groups to perform approximate inference with varying cost/accuracy tradeoffs. In this chapter, however, we are concerned with the classic case that identifies symmetry groups $\text{gr}(\bar{S})$ such that the optimized lifted-DD-objective (Eq. (3.12)) provides a solution to the ground-DD-objective (Eq. (2.6)), i.e.,

$$\min_{\boldsymbol{\delta}_{\bar{S}}} \bar{L}^{DD}_{\bar{G}}(\boldsymbol{\delta}_{\bar{S}}) \;=\; \min_{\boldsymbol{\delta}} L^{DD}_{G}(\boldsymbol{\delta}) \tag{3.14}$$

The symmetry groups used during lifted inference are often depicted by a *coloring* of the ground region graph. That is, each node in the lifted region graph is assigned a color, and each of its corresponding nodes in the ground region graph is assigned the same color. Using the coloring terminology, the symmetry groups that solve the exact inference problem (3.14) are known as a *stable coloring* of the ground region graph. An example model and coloring is shown in Figure 3.5; this example will be discussed in further detail in the next subsection where we discuss the standard coloring algorithm used to detect exact inference symmetries.

3.4.2 Finding a Stable Coloring

One way to identify a coloring that satisfies this ground optimality property is to "simulate" a ground inference algorithm; but rather than performing the actual numeric computation,

we simply reason about and keep track of which regions and separators have identical computations. Each inference operation propagates asymmetries through the region graph, forcing the ground regions and ground separators to be grouped into finer groups – the so-called *shattering* phenomenon. When no further regrouping occurs, the simulated inference is terminated and the computed coloring represents symmetries $\bar{S}$ in the exact ground-DD-objective in Eq. (3.14).

For concreteness and to connect to the algorithms developed in the sequel, in this section, we simulate a (block coordinate) gradient descent algorithm.[†] That is, we iteratively simulate a gradient step on the block of ground separators associated with a single lifted separator $\bar{s}$. This results in a recoloring of separators with identical ground gradient.

The symmetries in the ground DD gradient are represented as follows:

Lifted-Gradient-Separator: A *lifted-gradient-separator* $\bar{s}_{(\bar{r},\bar{r}')}$ is associated with a lifted separator $\bar{s} \in \bar{S}$ and two of its incident lifted regions $\{\bar{r}, \bar{r}'\} \subset \bar{N}(\bar{s})$. It represents each ground separator that is associated with the lifted separator $\bar{s}$ and has incident regions associated with the lifted regions $\bar{r}$ and $\bar{r}'$, i.e., $\text{gr}(\bar{s}_{(\bar{r},\bar{r}')}) := \{s \in \text{gr}(\bar{s}) : N(s) \in \text{gr}(\bar{r}) \times \text{gr}(\bar{r}')\}$. Each associated ground separator $s \in \text{gr}(\bar{s}_{(\bar{r},\bar{r}')})$ with incident regions $r \in \text{gr}(\bar{r})$ and $r' \in \text{gr}(\bar{r}')$ is associated with gradient factors that are each equal to a common template gradient factor

$$\begin{aligned} g^{\delta}_{s}(x_s) &= \mu^{\delta}_{r\to s}(x_s) - \mu^{\delta}_{r'\to s}(x_s) \\ &= \mu^{\delta}_{\bar{r}\to \bar{s}}(\bar{x}_{\bar{s}}) - \mu^{\delta}_{\bar{r}'\to \bar{s}}(\bar{x}_{\bar{s}}) := g^{\delta}_{\bar{s}_{(\bar{r},\bar{r}')}}(\bar{x}_{\bar{s}}) \end{aligned}$$

[†]Note that, since we are reasoning about which elements *must* remain symmetric (same color), and which symmetries may be broken (changing their color), the precise ground inference algorithm that we choose to simulate has no effect on the final stable coloring; in fact, the stable coloring can be computed entirely in graph theoretic terms, i.e., without reference to probabilistic inference. We discuss both of these points at the end of this section.

where $x_s = \bar{x}_s$, meaning that the index into the ground factor equals the index into the template factor.

The set of lifted-gradient-separators associated with at least one ground separator is denoted $\bar{s}_{gradAll} := \{\bar{s}_{(\bar{r},\bar{r}')} : \text{gr}(\bar{s}_{(\bar{r},\bar{r}')}) \neq \emptyset\}$.

Note that a lifted-gradient-separator $\bar{s}_{(\bar{r},\bar{r}')}$ represents information about the ground separators that is not contained in the lifted region graph. Although it would be possible to develop a graphical depiction to represent the lifted-gradient-separator $\bar{s}_{(\bar{r},\bar{r}')}$, this level of detail will not be necessary.

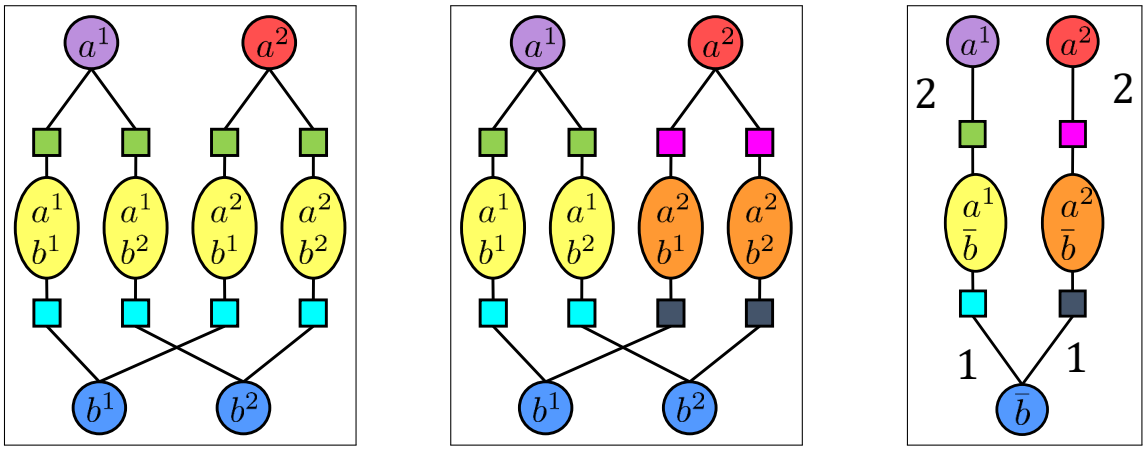
Now, the simulated gradient descent step at a lifted separator $\bar{s} \in \bar{S}$ affects symmetries in one of two ways:

Stable: The lifted separator $\bar{s} \in \bar{S}$ has exactly two incident lifted regions, i.e., $\bar{N}(\bar{s}) = \{\bar{r}, \bar{r}'\}$, and is associated with one lifted-gradient-separator, i.e., $\bar{s}_{gradAll} = \{\bar{s}_{(\bar{r},\bar{r}')}\}$. In this case, each ground separator $s \in \text{gr}(\bar{s})$ has identically colored incident regions and identical gradient. Thus, a gradient step does not change the separator coloring, and the ground region sub-graph is said to have a *locally stable coloring*.

Non-Stable: The lifted separator $\bar{s} \in \bar{S}$ has more than two incident lifted regions and is associated with more than one lifted-gradient-separators, i.e., $|\bar{s}_{gradAll}| > 1$. Thus, a gradient step recolors ground separators according to their lifted-gradient-separator group. That is, we replace the lifted separator with lifted-gradient-separator

$$\bar{S} \leftarrow (\bar{S} \setminus \{\bar{s}\}) \cup \bar{s}_{gradAll}$$

The ground separator recoloring induces a recoloring of the ground regions to ensure that each region in the color group has identically colored neighborhood as in Eq. (3.11).



(a) First coloring iteration (b) Second coloring iteration (c) Stable lifted graph

Figure 3.5: Illustration of stable coloring algorithm for models with factor symmetry. The left panel shows the original coloring of factors with identical model factors. The middle panel shows the result of one iteration of the coloring algorithm which is a stable coloring. The right panel shows the lifted region graph associated with the stable coloring.

The simulated block coordinate descent algorithm iteratively chooses a non-stable lifted separator $\bar{s} \in \bar{S}$ and splits it into groups with identical gradient, and terminates when each lifted separator is stable. In this case, the region graph is said to have a stable coloring (meaning that each node of the same color has identically colored neighborhood). Optimization of the lifted-DD-objective can then be performed entirely on the lifted region graph at a much cheaper cost than optimization of the ground-DD-objective on the ground region graph.

Example An example of the iterative recoloring algorithm along with the stable coloring and its associated lifted region graph is shown in Figure 3.5. The model contains two sets of RVs $\text{gr}(\bar{a}) = \{a^1, a^2\}$ and $\text{gr}(\bar{b}) = \{b^1, b^2\}$ with distinct univariate evidence on a^1 and a^2 (colored purple and red, respectively, in Figure 3.5a), with identical evidence on b^1 and b^2 (both colored blue), and each pair of RVs from group $\bar{a}$ and $\bar{b}$ is connected by a factor with identical value (all colored yellow). This is the initial coloring of the ground region graph.

A gradient step splits the green separators into two groups. The separators with one yellow and one purple neighbor remain green; the separators with one yellow and one red neighbor

become magenta. These newly colored separators induce a recoloring of the regions: The yellow regions (a^2, b^1) and (a^2, b^2) incident to a magenta separator are recolored orange. This is shown in Figure 3.5b, and corresponds to a stable coloring of the graph; notice that each node of the same color has an identically colored neighborhood, e.g., each purple region has two green neighbors, each yellow region has one green and one cyan neighbor, and so forth.

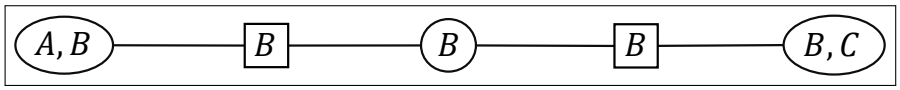
The associated lifted region graph is shown in Figure 3.5c. Each node in the lifted region graph corresponds to the ground nodes with the same color. The edge weights indicate the number of neighbors of each color, e.g., each purple has two green neighbors. For compactness, an unlabeled edge is sometimes used to indicate a weight on one, e.g., each yellow has one green and one cyan neighbor and neither of these are labeled.

3.5 Lifted Inference – State Space Symmetry

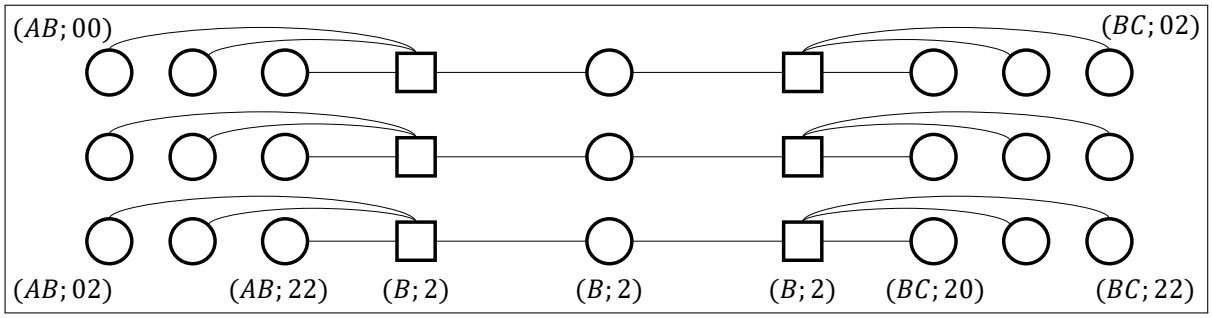
In this section, we show how to perform lifted inference using factors with *state space symmetry*. At a high level, lifted inference using factors with state space symmetry is identical to lifted inference using factor symmetry developed in the last section, but with groups of symmetric factors replaced by groups of symmetric states. In Section 3.5.1, we develop a state space region graph that allows us to visualize individual states. In Section 3.5.2, we show how to perform lifted inference with symmetric groups of states on this graph. In Section 3.5.3, we show how to detect symmetries.

3.5.1 State Space Graph

A standard region graph $G = (R, S, E)$ represents only the scope of factors and hence does not provide the necessary precision for discussing operations on states. To represent states explicitly, we associate the region graph G with a *State-Space-Graph (SS-Graph)* $\mathcal{G} = (\mathcal{R}, \mathcal{S}, \mathcal{E})$ where $\mathcal{R}$ is a set of SS-region nodes (*SS-regions*), $\mathcal{S}$ is a set of SS-separator



(a) Region graph



(b) Associated SS-Graph

Figure 3.6: (a) A simple region graph and (b) an associated state space region graph (SS-Graph) where each node represents a state.

nodes (*SS-separators*), and $\mathcal{E}$ is a set of *SS-edges*. The SS-Graph $\mathcal{G}$ has a block structure isomorphic to the region graph G, defined as follows.

Each region $r \in R$ is associated with a block of SS-regions $\mathcal{R}_r$ that represents the set of states associated with the region r, i.e., $\mathcal{R}_r \leftrightarrow \mathcal{D}_r$. Each separator $s \in S$ is associated with a block of SS-separators $\mathcal{S}_s$ that represents the set of states associated with the separator s, i.e., $\mathcal{S}_s \leftrightarrow \mathcal{D}_s$. Each edge $e = (r, s) \in E$ is associated with a block of SS-edges $\mathcal{E}_e$ between the SS-regions $\mathcal{R}_r$ and SS-separators $\mathcal{S}_s$. A SS-edge connects a SS-region $x_r \in \mathcal{R}_r$ to a SS-separator $x_s \in \mathcal{S}_s$ if and only if x_s is a subconfiguration of x_r.

An example of a SS-Graph is shown in Figure 3.6. Figure 3.6a shows a region graph $G = (R, S, E)$ on three RVs $\mathcal{V} = \{A, B, C\}$. Figure 3.6b shows an associated SS-Graph $\mathcal{G} = (\mathcal{R}, \mathcal{S}, \mathcal{E})$ assuming each RV $v \in \mathcal{V}$ has domain $\mathcal{D}_v = \{(v; 0), (v; 1), (v; 2)\}$. The left block of SS-regions $\mathcal{R}_r$ is associated with region $r = (A, B)$. Some of the SS-regions are labeled, e.g., $(AB; 00)$ represents the event that X_A and X_B take value 0.

3.5.2 Lifted Objective

A factor's state space symmetry is represented by a coloring of the associated SS-nodes in the SS-Graph $\mathcal{G}$. Collectively, these colorings are represented on a *Lifted-State-Space-Graph (LSS-Graph)* $\bar{\mathcal{G}} = (\bar{\mathcal{R}}, \bar{\mathcal{S}}, \bar{\mathcal{E}})$ defined by a set of *LSS-region-nodes (LSS-regions)* $\bar{\mathcal{R}}$, *LSS-separator-nodes (LSS-separators)* $\bar{\mathcal{S}}$, and *LSS-edges* $\bar{\mathcal{E}}$. The lifted objects represent sets of symmetric ground objects as follows:[‡]

LSS-regions: Each region $r \in R$ is associated with a block of LSS-regions $\bar{\mathcal{R}}_r$. Each LSS-region $y_r \in \bar{\mathcal{R}}_r$ is associated with a set of ground SS-regions $\text{gr}(y_r)$.

Collectively, $\text{gr}(\bar{\mathcal{R}}_r) := \{\text{gr}(y_r) : y_r \in \bar{\mathcal{R}}_r\}$ partitions the set of ground SS-regions $\mathcal{R}_r$ associated with region r and is associated with a LSS-factor $\bar{\theta}_r : \bar{\mathcal{R}}_r \to \mathbb{R}$.

LSS-separators: Each separator $s \in S$ is associated with a block of LSS-separators $\bar{\mathcal{S}}_s$. Each LSS-separator $y_s \in \bar{\mathcal{S}}_s$ is associated with a set of ground SS-separators $\text{gr}(y_s)$.

Collectively, $\text{gr}(\bar{\mathcal{S}}_s) := \{\text{gr}(y_s) : y_s \in \bar{\mathcal{S}}_s\}$ partitions the set of ground SS-separators $\mathcal{S}_s$ associated with separator s and is associated with a LSS-factor $\bar{\delta}_s : \bar{\mathcal{S}}_s \to \mathbb{R}$.

LSS-edges: Each edge $e = (r, s) \in E$ is associated with a block of LSS-edges $\bar{\mathcal{E}}_e$ between region $r \in R$ and separator $s \in S$.

A LSS-Graph defined in this way is associated with symmetric reparameterized factors which are written as

$$\begin{aligned} \theta_r^{\delta}(x_r) \;&=\; \theta_r(x_r) \;+\; \sum_{s \in N(r)} \delta_{rs}(x_s) \\ &=\; \bar{\theta}_r(y_r) \;+\; \sum_{s \in N(r)} \bar{\delta}_s(y_s) \;=\; \bar{\theta}_r^{\bar{\delta}}(y_r) \end{aligned} \tag{3.15}$$

[‡]Note how the definition of the LSS-Graph parallels the definition of the lifted region graph in Section 3.4.1.

for each $x_r \in \text{gr}(y_r)$ and each $y_r \in \bar{\mathcal{R}}_r$.

Symmetries in the reparameterized factors can be exploited to compute the ground-DD-objective efficiently as

$$\begin{aligned} L_G^{DD}(\boldsymbol{\delta}) &= \sum_{r \in R} \max_{x_r} \ \theta_r^{\delta}(x_r) \\ &= \sum_{r \in R} \max_{y_r \in \bar{\mathcal{R}}_r} \ \bar{\theta}_r^{\bar{\delta}}(y_r) \ =: \ \bar{L}_G^{DD}(\boldsymbol{\delta}_{\bar{S}}) \end{aligned} \tag{3.16}$$

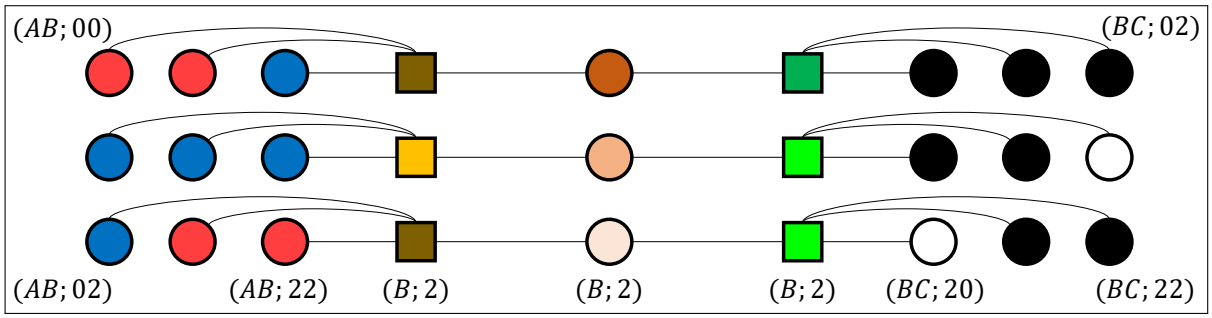
where the lifted maximization (second line) takes place over the lifted regions, since each corresponding ground state has identical value.

Example To depict state space symmetry, we color the LSS-region $y_r \in \bar{\mathcal{R}}_r$ and its associated ground SS-nodes in the set $\text{gr}(y_r)$ the same color. Figure 3.7a shows an example coloring where the region (A, B) on the left has factor function $\theta_{AB} = \begin{bmatrix} r & r & u \\ u & u & u \\ u & r & r \end{bmatrix}$ with states of value r colored red and value u colored blue, and the region (B, C) on the right has factor function $\theta_{BC} = \begin{bmatrix} b & b & b \\ b & b & w \\ w & b & b \end{bmatrix}$ with states of value b colored black and value w colored white.

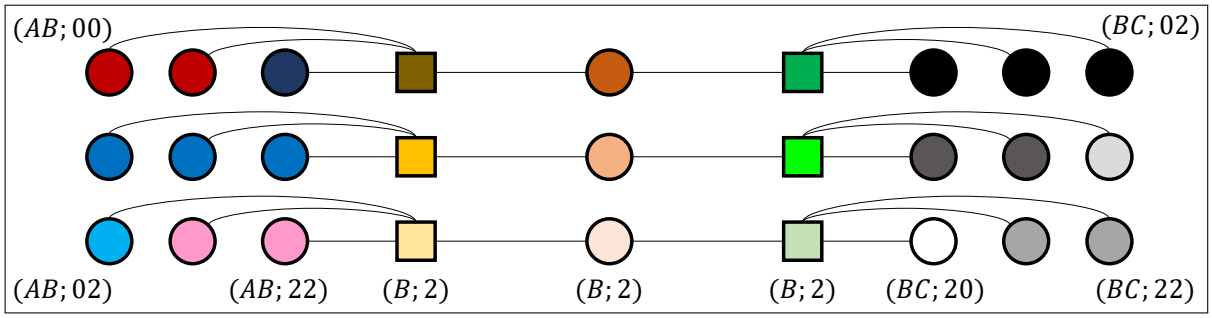
3.5.3 Finding a Stable Coloring

To detect state space symmetries, we use a coloring algorithm similar to the coloring algorithm for detecting factor symmetries presented in Section 3.4.2. We give only a brief, informal overview to illustrate the similarity between the two algorithms since the state space stable coloring algorithm will not be explicitly required in the sequel.

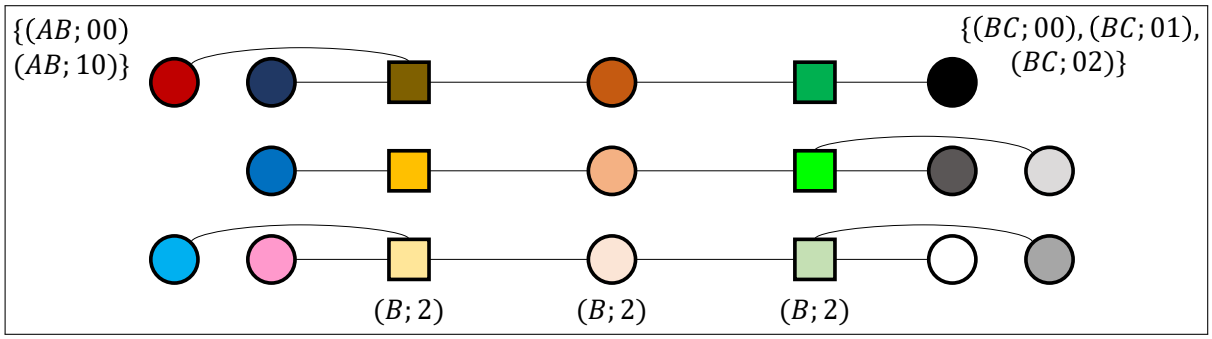
At each iteration of the coloring algorithm, we consider a separator $s \in S$ and incident regions $r \in R$ and $r' \in R$. For each LSS-separator $y_s \in \bar{\mathcal{S}}_s$, we recolor its associated ground SS-separators $\text{gr}(y_s)$ that have identical gradients. Two ground SS-separators have identical gradient if and only if they have identically colored neighborhood. First, symmetry in the



(a) First coloring iteration.



(b) Second coloring iteration, stable coloring.



(c) LSS-Graph.

Figure 3.7: Detecting stable coloring (symmetry partition) of SS-Graph $\mathcal{G}$. (a) One iteration of state space color propagation, (b) Two iterations producing stable coloring, (c) Associated LSS-Graph $\bar{\mathcal{G}}$

max-marginals can be identified as

$$\begin{aligned}\mu^{\delta}_{r\to s}(x_s) &= \sum_{x_{r\setminus s}} \mu^{\delta}_{r}(x_r) \\ &= \sum_{y_s\in\bar{\mathcal{N}}(y_r)} \bar{\mu}^{\bar{\delta}}_{r}(y_r) =: \bar{\mu}^{\bar{\delta}}_{r\to s}(y_s)\end{aligned}$$

for each $x_s \in \mathrm{gr}(y_s)$ and each $y_s \in \bar{\mathcal{S}}_s$ where $\bar{\mathcal{N}}(y_r)$ is the set of neighbors of y_r in the

LSS-Graph. Then, the ground gradient has symmetry groups defined by identically colored neighborhoods in *both* incident regions. That is,

$$\begin{aligned} g_s^{\delta}(x_s) &= \mu_{r\to s}^{\delta}(x_s) - \mu_{r'\to s}^{\delta}(x_s) \\ &= \bar{\mu}_{r\to s}^{\bar{\delta}}(y_s) - \bar{\mu}_{r'\to s}^{\bar{\delta}}(y_s) := \bar{g}_{s_{(y_r,y_{r'})}}^{\bar{\delta}}(y_s) \end{aligned}$$

for each $y_s \in \mathrm{gr}(s_{(y_r,y_{r'})})$ and $\mathrm{gr}(s_{(y_r,y_{r'})})$ is a set of SS-separators with identically colored neighborhood.

The recoloring is iterated until no more changes take place. The final coloring is called the *stable coloring* of the SS-Graph $\mathcal{G}$.

Example An example illustrating the gradient symmetry and the state space stable coloring algorithm is shown in Figure 3.7. An initial coloring is shown in Figure 3.7a. At the second iteration, we recolor the separators with identical (ignoring counts) colored neighborhood. For example, consider the dark yellow SS-separators. The top one has a red, blue, and *dark* orange neighbor, while the bottom one has a red, blue, and *light* orange neighbor.

These are split into separate colors in the next iteration in Figure 3.7b. The incident rows of each SS-region group are then split (corresponding to newly reparameterized factors θ^{δ}). This figure represents a stable coloring and is associated with the LSS-Graph $\bar{\mathcal{G}}$ in Figure 3.7c. Note how the initial model has two symmetry groups for each pairwise factor (A, B) and (B, C) but when they are combined to solve the global inference problem, they each *shatter* into five symmetry groups. In general, shattering can fully reduce the model factor symmetries to ground symmetries, leaving are no symmetries to exploit during inference.

Relation to Binarized Graphical Model It is worth drawing a connection between a SS-Graph and a "binarized" graphical model [Koller and Friedman, 2009], i.e., where we replace each variable X_v with a set of variables, one for each value of X_v, and a "one-hot" constraint

which ensures that exactly one such variable is active. A binarized model would have the same structure as an SS-Graph, but where the nodes represent RVs and factors instead of states, and there is an additional factor encoding the one-hot constraint (this constraint does not affect the symmetry detection at all). With this equivalence, symmetry detection and exploitation on the binarized graphical model could, in theory, proceed using standard lifted inference techniques for factor symmetries. Although we do not do this, we presented this connection to provide the reader with another perspective for establishing the qualitative equivalence between methods for detecting and exploiting state space symmetry and methods for detecting and exploiting factor symmetry.

3.6 Coping with Asymmetry

In the previous two sections, we saw how to perform lifted inference in models with exact symmetry, first for models with factor symmetries, then for models with state space symmetries. As we saw in those sections, exact inference symmetries rarely exist due to the shattering effect, whereby inference operations break unaligned symmetry groups into finer groups. In many models, this shattering process is complete and exact lifted inference is forced to ground the model, negating all the potential efficiency of the symmetric representation.

Consequently, the practicality of lifted inference hinges upon our ability to intelligently identify, represent, and exploit *approximate symmetry* during inference. In the following subsections we discuss the broad classes of methods that have been developed previously to cope with asymmetry, highlighting the strengths and weaknesses of each approach. (Recall that these methods are, as summarized in Table 3.1, projected message passing, over-symmetric model approximation, and coarse-to-fine inference.) We then detail the criteria that an ideal lifted inference procedure would have and briefly discuss our strategies in subsequent

chapters of this thesis to remedy the shortcomings of previous approaches.

3.6.1 Projected Message Passing

The basic idea of projected message passing is to approximate each exact message computation with a tractable symmetric function. Specifically, at each iteration, the exact message is computed; then an approximate message is then constructed by minimizing a distance between the exact message and an approximation from a tractable class, such as a function using a fixed number of symmetry groups. Projected message passing is a popular approximation in both models with state space symmetry and factor symmetry as well as models with other parametric (non-symmetric) structure:

State Space Symmetry: Gogate and Domingos [2013] present a junction graph belief propagation algorithm where each message is represented with an algebraic decision diagram (ADD). (An example of an ADD is shown in Figure 3.2d.) At each iteration, the product of messages with different ADD structures produces an intractably large ADD which must be (approximately) projected back to a tractable ADD (with a fixed number of nodes).

Factor Symmetry: In MLN models, Singla et al. [2014] presents a lifted belief propagation algorithm that represents symmetry in messages as a union of hypercubes over logical domain objects. Each hypercube represents, for example, an identical belief for all friendship RVs between two subsets of people. At each iteration, each message is (approximately) projected to a representation with a small set of hypercubes.

Parameteric Structure: There is a large body of work on projected message passing using factors with other (non-symmetric) structure in classic PGMs. For example, many works [e.g., Sudderth et al., 2005, Minka, 2001, Ramakrishnan et al., 2011] approximate continuous messages with Gaussian distributions or mixtures of Gaussians. Message

updates result in a function outside the desired family, which is then re-approximated by a new Gaussian or Gaussian mixture. In Sudderth et al. [2005], each iteration produces a message that is a mixture of a large number of Gaussians, which is then approximated using a smaller, fixed size mixture distribution.

An important advantage of the class of projected message passing methods is that they select the symmetry (or other) structure using preliminary inference estimates, in contrast to several methods described in the sequel, which require the approximate symmetry groups to be selected *a priori*. However, projected message passing methods also suffer from the following problems:

Lossy Messages: The projection step introduces error into the approximation. This error is often difficult to quantify, both at one iteration and in its global effect as it propagates through the graph [Ihler et al., 2005].

Fixed Graph: Inference is usually performed on a fixed graph structure, that is, no higher-order terms are introduced (by joining regions) after inference has begun. Introducing such terms on the fly is difficult, for two primary reasons. First, joining would shatter the current symmetry into an uncontrolled number of groups, which would likely have to be projected back onto a tractable representation. Second, it is not clear how error introduced by previous lossy message passing operations would affect the join.

3.6.2 Domain Clustering – Over-Symmetric Approximate Model

Since shattering causes many difficulties in inference, it is useful to study a class of symmetric models on which shattering does not occur. In this class of models, symmetry is defined on the *domains* of the model's fundamental objects and lifted inference is structurally equivalent to ground inference on a compressed model. Models of this form with both state space symmetry and factor symmetry arise as follows:

State Space Symmetry: The states of each RV are partitioned into symmetry groups, meaning that states in each group are indistinguishable within all factors in the model. In this case, each lifted factor has the form of a small table and lifted inference operations are structurally equivalent to standard ground inference operations using tables, but where an element of each table axis represents a group of ground states rather than an individual state.

MLN Models (Factor Symmetry): The domains of auxiliary objects are partitioned into symmetry groups, meaning that objects within each group are indistinguishable throughout the model. In the friend-smoker-MLN, for example, each person in a group has identical attribute (smoking and cancer) evidence, and identical friendship relationships with all of each other group's members, i.e., between any two groups, all pairs of people are either friends or not friends. Collectively, the lifted factors have (nearly) the same structure as a ground MLN with a small number of auxiliary objects, in which each element represents a group of ground objects rather than an individual object.

Again, of course, in realistic models this precise structure will not exist. In practice, a popular strategy is to approximate the original model with an *over-symmetric model* with domain symmetry and perform inference on the over-symmetric model:

State Space Symmetry: For a model with continuous factors, Isard et al. [2009] approximates each factor with a grid-structured piecewise constant factor. Continuous belief propagation on this over-symmetric model is equivalent to (weighted) belief propagation on a set of discrete tabular factors, where each element in a table corresponds to a contiguous range of the underlying continuous-valued variables. The resulting discretization can be made adaptive, in order to concentrate higher resolution (smaller size) cells in areas of high probability mass.

MLN Models (Factor Symmetry): The auxiliary objects in a MLN are clustered into groups where the evidence is approximately identical, e.g., *most* pairs of people within two groups are friends (or most are not friends), and the original evidence is replaced with *over-symmetric* approximate evidence, in which all evidence associated with a group is identical. Venugopal and Gogate [2014a] directly constructs a MLN of reduced size (e.g., where formulas relate objects representing groups of people, rather than an individual person) and performs inference on the reduced MLN. Another work, Van den Broeck and Darwiche [2013] performs exact inference (for a restricted class of MLNs) using the over-symmetric evidence.

The advantages of this family of approximations include:

Controlled Cost: The number of clusters is a hyper-parameter which can be made arbitrarily small to produce an approximation with controlled cost.

Generality (High-Order Inference Terms): Since the approximation is (nearly) identical to a small model, any inference technique can be used to solve the over-symmetric approximation. In particular, it can be possible to join symmetric inference groups to construct high-order inference terms without shattering symmetry.

However, there are a number of disadvantages as well:

Biased Inference Estimates: It is difficult to quantify how well the inference estimates on the over-symmetric model approximate inference quantities on the original model. In particular, the model approximation introduces bias both due to the representation of asymmetric evidence as a single symmetry group, and also due to the model compression which substitutes individual objects for groups of objects in each formula (e.g., the same formula used to constrain friendship of three individual people in the original

model is used to constrain friendship among three *groups* of people in the compressed model). These two sources of bias are discussed in further detail in Section 3.7.2.

A-Priori Symmetry Choice: For MLN models, Venugopal and Gogate [2014a] and Van den Broeck and Darwiche [2013] build the over-symmetric approximation *a priori*, without using inference information and without the ability to improve the approximation in an anytime fashion. To use a finer clustering, these works must restart inference, and cannot easily leverage the coarse approximation results to improve their performance.

However, for state space symmetry, Isard et al. [2009] does select the partition adaptively, using preliminary inference estimates. Thus, although selecting symmetry groups *a priori* is not fundamental to this class of methods, many works have not successfully incorporated adaptive symmetry selection, which presents additional complications.

In Section 3.7, we present details of domain clustering which will be needed for subsequent chapters.

3.6.3 Coarse-to-Fine Inference

An emerging class of methods performs inference in a coarse-to-fine manner [Sarkhel et al., 2015, Habeeb et al., 2017] by alternating the following two steps. First, restrict subsets of inference parameters to be identical; this creates a coarse (low-cost) inference problem which is (approximately) solved. Then, break a set of symmetry restrictions to create a finer problem with higher cost and higher accuracy; instantiate this problem with the solution to the coarse problem. This generates a sequence of inference estimates of monotonically improving accuracy.

A concrete example of this procedure is the work of Habeeb et al. [2017] which develops a coarse-to-fine inference algorithm for stereo vision. First, pixels are grouped together (via a

heuristic metric) and forced to take the same disparity label; then the constrained inference problem is solved by reasoning over pixel groups (rather than individual pixels). A group of pixels is then split into a finer grouping if it is believed that pixels in the groups should in fact take different disparity labels.

The coarse-to-fine inference algorithms are appealing for the following reasons:

Adaptive: The symmetry groups are chosen adaptively, using preliminary inference estimates (as opposed to methods that require symmetry groups to be selected *a priori*).

Lossless: Each restricted inference problem is solved exactly. This is in contrast to lossy projected message passing methods (which also select the symmetry groups adaptively) where each message passing operation incurs error and the converged solution does not solve any easily specifiable approximate inference problem.

However, current coarse-to-fine inference algorithms have a number of shortcomings. First, only MAP inference algorithms have been developed so far; furthermore, only configuration-based MAP inference algorithms have been developed (i.e., not using variational approximations like the dual decomposition). Second, to our knowledge, no coarse-to-fine inference algorithm exploiting state space symmetry has been developed; previous works exploit factor symmetries (Sarkhel et al. [2015] does so for stereo vision problems, while Habeeb et al. [2017] does so for MLN models).

3.6.4 Ideal Framework and Our Work

An ideal lifted inference procedure that exploits approximate symmetries would have the following properties:

Controlled Cost: Perform approximate inference with a user-controlled degree of symmetry, and hence a user controlled computational cost.

Accuracy Guarantees: Maintain a concrete connection to the original problem, e.g., a bound on the inference objective or inference error.

Adaptive Selection: Since symmetries cannot be determined *a priori*, we must adaptively choose symmetry groups using preliminary inference estimates.

Anytime Improvement: The approximation quality should monotonically improve with additional resources such as time or memory. For example, a modified inference structure, e.g., from splitting symmetry groups, should be able to utilize the current inference information (i.e., inference should not have to be restarted on the new structure).

Exact Solution: Given sufficient resources, anytime inference should terminate at the exact (non-relaxed) inference solution. Ideally, the anytime inference procedure should traverse approximations using factors of increasingly large scope.

Inference Queries: We should be able to perform a variety of inference queries, including MAP inference, sum-inference, marginal MAP inference (where we sum over a subset of RVs and maximize over another set), using the same strategies for controlling and introducing symmetry.

Many existing methods satisfy some but not all of these desiderata. Methods that exploit exact inference symmetry are guaranteed to produce the same answer as ground inference but have an uncontrolled cost (since the degree of symmetry a function of the model itself and cannot be reduced). On the other hand, many of the methods for exploiting approximate symmetry (discussed in previous sections) have controlled cost but provide no guarantees on the approximation quality. The coarse-to-fine inference algorithm discussed in the last section satisfies many of these properties, but has been developed only for MAP inference for factor symmetries and does not incorporate high-order inference terms.

3.6.4.1 Our Work

Our work in this thesis is aimed at addressing the shortcomings of previous approaches. Each of our novel works in subsequent chapters is developed within a coarse-to-fine inference framework, which allows us to adaptively select the symmetry groups, control computational cost, and maintain a concrete connection to the ground inference problem (i.e., not introduce irrecoverable bias). Furthermore, much of our work exploits and introduces substructures with domain symmetry which allow us to adaptively build symmetric high-order inference terms. In summary:

State Space Symmetry: Rather than creating an over-symmetric model, in Sontag et al. [2009], the original model factors are left in place and the inference relaxation is tightened with large-scope domain-symmetric factors (i.e., large scope factors are represented at a coarse resolution). Our work *Adaptive Hierarchical State Space Partitions* in Chapter 6 can be seen as a generalization of this basic structure that incorporate more sophisticated state space symmetry over the joint space of large collections of RVs. Furthermore, the inference structure is selected adaptively, adding factors of increasingly large size and increasingly fine state space symmetry.

MLNs (Factor Symmetry): Our *Lifted GDD Algorithm* in Chapter 4 performs coarse-to-fine inference in symmetric models perturbed by evidence; inference is performed on (low-order) model factors. Our *Lifted WMB Algorithm* in Chapter 5 introduces high-order inference terms into the relaxation by reasoning with symmetry groups defined by a partition of the logical domains. That is, the inference relaxation is structurally equivalent to a small MLN (just like Venugopal and Gogate [2014a]) but maintains a concrete connection to the ground problem (unlike Venugopal and Gogate [2014a] which creates an over-symmetric model).

Furthermore, our frameworks are capable of handling a variety of inference algorithms. Our

Lifted GDD Algorithm in Chapter 4 and *Lifted WMB Algorithm* in Chapter 5 operate within a sum-inference framework but should be easy to extend to MAP inference and marginal MAP inference. Our work on *Adaptive Hierarchical State Space Partitions* in Chapter 6 operates within a MAP inference framework, but should be able to be extended to sum-inference (as we discuss in future work section).

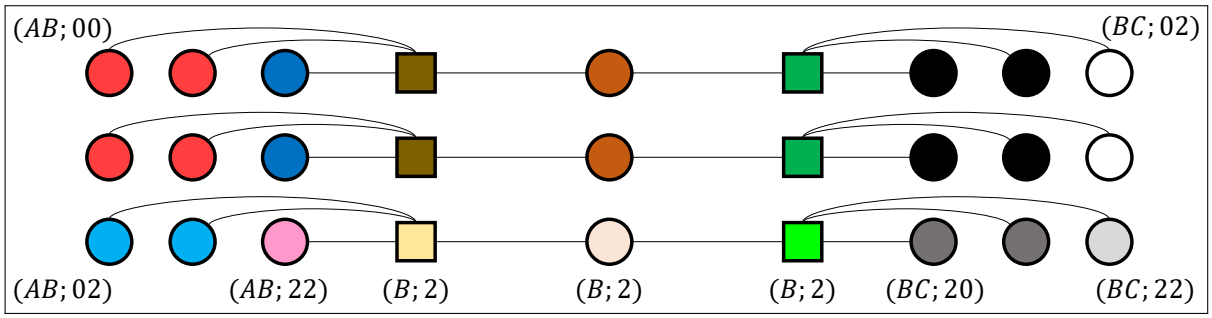
3.7 Domain Symmetry

In this section, we present details of domain clustering, first by clustering the domains of RVs to induce state space symmetry, then by clustering the domains of logical constants in MLN models to induce factor symmetry. These details will be needed for our development in subsequent chapters.

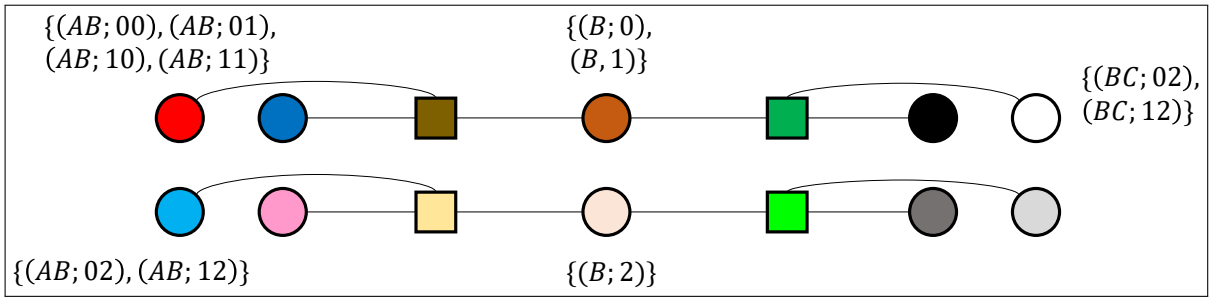
3.7.1 RV Domain Symmetry – MS-LSS-Factors

In this section, we discuss models with state space symmetry defined by partitions of each RV domain, meaning that a group of states associated with each RV behaves identically throughout the model. In models with this type of symmetry, message passing and joining inference operations can be performed without shattering the model factors' state space symmetry. The notation developed in this section will be used in our work on *Adaptive Hierarchical State Space Partitions* in Chapter 6.

For each RV $v \in \mathcal{V}$, let its domain $\mathcal{D}_v$ be partitioned as $\text{gr}(\bar{D}_v) := \{\text{gr}(y_v) : y_v \in \bar{D}_v\}$ where $\bar{D}_v$ is a set of *lifted domain states*. A *multidimensional LSS-factor (MD-LSS-factor)* $\bar{\theta}_\alpha : \bar{\mathcal{D}}_\alpha \to \mathbb{R}$ with scope $\alpha \subset \mathcal{V}$ has domain that is a table of all combinations of lifted domain states associated with RVs in its scope, i.e., $\bar{\mathcal{D}}_\alpha := \times_{v \in \alpha} \bar{D}_v$. Each element of the domain $y_\alpha \in \bar{\mathcal{D}}_\alpha$ is associated with a hypercube $\text{gr}(y_\alpha) = \times_{v \in \alpha} \text{gr}(y_v)$ equal to the Cartesian product of its



(a) Colors indicating state space symmetry with stable coloring



(b) Lifted graph representing state space symmetry groups (labeled)

Figure 3.8: (a) SS-graph graph $\mathcal{R}$ associated factors with *domain symmetry* groups $\text{gr}(\bar{\mathcal{D}}_v) = \{\{0,1\},\{2\}\}$ at each RV (meaning states 0 and 1 of each RV behave identically in each factor). (b) Associated LSS-Graph $\bar{\mathcal{G}}$ which is isomorphic to a standard tabular SS-Graph but with axes corresponding to subsets of domain elements (rather than individual elements). The tabular form allows (nearly) standard inference operations to be performed.

associated ground domain sets. Collectively, these sets of ground states

$$\text{gr}(\bar{\mathcal{D}}_\alpha) \;:=\; \{\text{gr}(y_\alpha) : y_\alpha \in \bar{\mathcal{D}}_\alpha\} \;=\; \times_{v\in\alpha}\, \text{gr}(\bar{D}_v) \tag{3.17}$$

partition the factor's state space $\mathcal{D}_\alpha$ into a grid of hypercubes with axes defined by the RV-domain partitions $\{\text{gr}(\mathcal{D}_v) : v \in \alpha\}$. In other words, each pair of ground joint states is in the same hypercube $\{x_\alpha, x_\alpha{}'\} \subset \text{gr}(y_\alpha)$ if and only if each of their elements is in the same RV-domain partition, i.e., $\{x_v, x_v{}'\} \subset \text{gr}(y_v)$ for each $v \in \alpha$.

A region graph $G = (R, S, E)$ defined by MD-LSS-factors can be represented with a coloring of the SS-Graph $\mathcal{G} = (\mathcal{R}, \mathcal{S}, \mathcal{E})$ where each block of SS-regions and SS-separators has a coarse tabular structure. This represents a stable coloring of the SS-Graph $\mathcal{G}$ since the nodes of each

color class have identically colored neighborhood (as we illustrate with an example shortly). Consequently, no shattering occurs during inference.

A SS-Graph with symmetries of this form can be represented as a LSS-Graph $\bar{\mathcal{G}} = (\bar{\mathcal{R}}, \bar{\mathcal{S}}, \bar{\mathcal{E}})$ with a small tabular structure.

Example An example is shown in Figure 3.8a. Each RV domain $\mathcal{D}_v$ is partitioned into $\bar{D}_v = \{\{(v;0),(v;1)\},\{(v;2)\}\}$ for each of the RVs $v \in \{A, B, C\}$. For region $r = (A, B)$, the domain $\mathcal{D}$ is partitioned as $\text{gr}(\bar{\mathcal{D}}_{\text{sc}(r)}) = \{\{(AB;00),\ (AB;01),\ (AB;10),\ (AB;11)\}, \{(AB;02),\ (AB;12)\},\ \{(AB;20),\ (AB;21)\},\ \{(AB;22)\}\}$ corresponding to colors red, cyan, blue, and pink, respectively. The LSS-Graph is shown in Figure 3.8b.[§]

3.7.2 Logical Domain Clustering

One strategy for handling asymmetry in a MLN $\mathcal{M} = (\bar{\boldsymbol{\Delta}}, \boldsymbol{P}, \bar{\mathcal{I}}^{FOL}, \boldsymbol{\omega}, \boldsymbol{E})$ pursued by Venugopal and Gogate [2014a], is to cluster its domain elements into groups that are expected to have similar marginal probability for each attribute and relationship predicate (e.g., cluster people into groups expected to have the same likelihood of smoking, cancer, and friendship relations with each other person).

These cluster centers are then substituted for ground auxiliary objects in the original MLN to form a cluster-MLN $\mathcal{M}' = (\bar{\boldsymbol{\Delta}}', \boldsymbol{P}, \bar{\mathcal{I}}^{FOL}, \boldsymbol{\omega}, \boldsymbol{E}')$ where $\bar{\boldsymbol{\Delta}}'$ represents a set of groups of domain elements, and $\boldsymbol{E}'$ is the average evidence on each cluster group. In other words, we form an MLN with the same formulas, but rather than ranging over individual elements, the logical variables range over individual groups.

[§]This example has two properties that are not required in general: (i) domain symmetry at each RV need not group the same states at each RV, i.e., we need not group states 0 and 1 at each RV, and (ii) the domain partition may group noncontiguous labels, e.g., $\bar{\mathcal{D}}_A = \{\{(A;0),(A;2)\},\{(A;1)\}\}$ is also valid but obviously leads to different factor symmetry.

An example FOL-formula is,

$$(\forall\ l_1 \neq l_2 \in \mathrm{gr}(\bar{\Delta}'_{\mathrm{ppl}}))\ \ \mathtt{Fr}(l_1, l_2) \Rightarrow (\mathtt{Sm}(l_1) \Leftrightarrow \mathtt{Sm}(l_2)) \tag{3.18}$$

with the same weight +1.6 as the model formula. In English, this states that "If two groups of people are friends, then the two groups probably either both smoke or both don't smoke".

The cluster-MLN is associated with a clustered-model $\boldsymbol{\theta}_{\mathcal{M}'}$. Inference is performed on the model $\boldsymbol{\theta}_{\mathcal{M}'}$ to compute marginal probabilities $p'(x_{v'}) = \sum_{x_{\mathcal{V}'\setminus v'}} \exp(\boldsymbol{\theta}_{\mathcal{M}'})/Z(\boldsymbol{\theta}_{\mathcal{M}'})$ of each cluster-RV. Each ground RV associated with the cluster-RV is assigned the same marginal probability in the original model, i.e., $p(x_v) = p'(x_{v'})$ for each $v \in \mathrm{gr}(v')$.

3.7.2.1 Sources of Bias

The primary advantage of this approach is its flexibility: we have full control over the computational cost of inference (by controlling the number of clusters in the cluster-MLN $\mathcal{M}'$) and we can use any inference algorithm (exact or relaxed) to compute the inference quantities. The primary disadvantage of this approach is that the cluster-MLN and its inference quantities bears no concrete connection to the original MLN and its inference quantities. The bias is introduced in two forms:

Evidence bias: One source of bias is the approximation of the evidence with a single value. If each block of RVs is associated with identical evidence, this is not a source of error, i.e., if $\theta_{v'} = \theta_v$ for each $v \in \mathrm{gr}(v')$. If *most* of the RVs in the block have identical evidence, we expect little error to be incurred, but there are no guarantees.

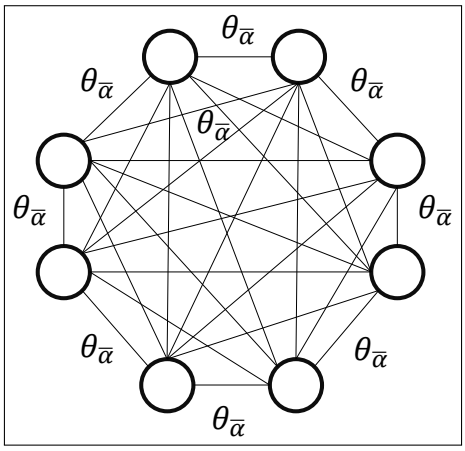
Size bias: The formulas in the cluster-MLN are identical to those in the original MLN. However, it does not seem appropriate to take a rule that was learned to relate individual elements and apply it to relate clusters of elements For example, it seems reasonable

that a ground factor (in the clustered MLN) penalizing groups for not obeying a rule should incur larger penalty if it relates large groups of people rather than small groups of people. One possible heuristic to fix this is to scale each factor $\theta_{\bar{\alpha}'}$ over a set of clusters by the number of underlying ground clusters, i.e., to set $\theta_{\bar{\alpha}'} = \theta_{\bar{\alpha}} \cdot |\mathrm{gr}(\bar{\alpha}')|$. However, this heuristic choice is not pursued by Venugopal and Gogate [2014a] or any follow-up work; furthermore, it too would lack accuracy guarantees.

3.7.2.2 Related work

In a related work, Van den Broeck and Darwiche [2013] shows that exact inference can be computed efficiently in a restricted class of MLNs when the evidence has a small Boolean rank. Using this result, they propose performing approximate inference by first approximating the model evidence with low-rank evidence, then performing exact inference on the model with over-symmetric approximate evidence. The primary advantage of this approach is that it produces exact inference estimates when the evidence is exactly of this form. That is, it does not suffer from the "size biasedness" problem of Venugopal and Gogate [2014a] mentioned above. The primary shortcoming of this approach is that it is only applicable to a subset of MLN formulas while the method of Venugopal and Gogate [2014a] presented above is applicable to any general MLNs.

Fixing the Biasedness Problem Van den Broeck and Niepert [2014] proposes a method to fix the bias introduced by the over-symmetric evidence approximation of Van den Broeck and Darwiche [2013]. The basic idea is use the over-symmetric approximation to define the proposal distribution of a Markov chain Monte Carlo (MCMC) inference algorithm. The advantage of this method is that the MCMC procedure is asymptotically unbiased. Furthermore, if the asymmetries in the evidence are small, then the proposal matches the original distribution well and is expected to converge quickly. However, even though this



(a) Complete symmetric graph

(b) Bipartite symmetric graph

Figure 3.9: Two graphs with symmetric factors whose product is a large factor with state space symmetry. (a) Complete graph with symmetric potentials is equivalent to a counting factor over all RVs, (b) Bipartite graph between RVs groups $\mathcal{V}_1$ (top) and $\mathcal{V}_2$ (bottom) with symmetric potentials is equivalent to a joint counting factor over RV groups $\mathcal{V}_1$ and $\mathcal{V}_2$.

method is expected to converge quickly, there are no guarantees on convergence rate and, moreover, the convergence is difficult to diagnose.

3.8 Aggregate Symmetry

The previous sections showed how to perform exact (relaxed) lifted inference and how to cope with model asymmetries. In Section 3.8.1, we present an identity where the a complete graph with factor symmetry aggregates to a large counting factor; this implies that exact (non-relaxed) lifted inference can be performed efficiently. This identity provides an important relationship between the two types of symmetry we have seen so far (factor symmetry and state space symmetry) and inspired our work on *Adaptive Hierarchical State Space Partitions* in Chapter 6. Our work in that chapter constructs large scope factors with (arbitrary) state space symmetry from small scope factors with finer (or no) state space symmetry, generalizing the counting-factor identity and expanding the utility of this basic idea to a large class of problems.

3.8.1 Joint Counting Factor Aggregation

A *LSS-counting-factor (LSS$^{\#}$-factor)* $\bar{\theta}^{\#}_{\mathcal{V}'} : \bar{\mathcal{D}}^{\#}_{\mathcal{V}'} \to \mathbb{R}$ over a subset of RVs $\mathcal{V}' \subset \mathcal{V}$ is a LSS-factor where

$$\text{gr}(y^{\#i}_{\mathcal{V}'}) \;=\; \{(x_v)_{v\in\mathcal{V}'} : \sum_{v\in\mathcal{V}'} x_v = i\} \tag{3.19}$$

is the set of joint states of binary RVs $\mathcal{V}'$ with exactly i `on` RVs and the factor's domain

$$\bar{\mathcal{D}}^{\#}_{\mathcal{V}'} \;=\; \{y^{\#i}_{\mathcal{V}'} : \; i \in [|\mathcal{V}'|]\} \tag{3.20}$$

is a set of symbols for each count. We will sometimes abuse notation to index a counting factor with an integer count, i.e., we will write $\bar{\theta}^{\#}_{\mathcal{V}'}(i) := \bar{\theta}^{\#}_{\mathcal{V}'}(y^{\#i}_{\mathcal{V}'})$. Recall that we saw an example of a factor table associated with a counting factor on three RVs in Figure 3.2a.

Symmetric Complete Graph Consider the symmetric complete graph in Figure 3.9a corresponding to a single lifted factor with ground scopes $\text{gr}(\bar{\alpha}) = \{\{v, v'\} : v \neq v' \in \mathcal{V}\}$ and template $\theta_{\bar{\alpha}} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$ for any a, b, c. The sum of these symmetric factors is equivalent to a LSS$^{\#}$-factor

$$\sum_{\alpha\in\text{gr}(\bar{\alpha})} \theta_{\bar{\alpha}}(x_\alpha) \;=\; \bar{\theta}^{\#}_{\mathcal{V}'}(\sum_{v\in\mathcal{V}'} x_v)$$

where

$$\bar{\theta}^{\#}_{\mathcal{V}'}(y^{\#i}_{\mathcal{V}'}) \;=\; \binom{n-i}{2}\cdot a + \binom{i}{1}\binom{n-i}{1}\cdot b + \binom{i}{2}\cdot c. \tag{3.21}$$

This holds since for any configuration $x_{\mathcal{V}}$ with i `on` RVs and $n-i$ `off` RVs there are $\binom{n-i}{2}$ pairs with two `off` RVs, $\binom{i}{1}\binom{n-i}{1}$ pairs with one `on` RV and one `off` RV, and $\binom{i}{2}$ pairs with two `on` RVs. The factor template $\theta_{\bar{\alpha}}$ assigns values a,b, and c, respectively, to these pairs.

Since this model can be written as a compact LSS$^{\#}$-factor, exact inference can be performed efficiently by iterating over the groups of symmetric states rather than the exponentially

large group of ground states. That is,

$$\begin{aligned}\Phi_0(\boldsymbol{\theta}) &= \max_{i=0}^{n} \bar{\theta}^{\#}_{\mathcal{V}'}(y^{\#i}_{\mathcal{V}'}) \qquad \text{and} \\ \log(Z(\boldsymbol{\theta})) &= \log \sum_{i=0}^{n} \binom{n}{i} \cdot \exp\left(\bar{\theta}^{\#}_{\mathcal{V}'}(y^{\#i}_{\mathcal{V}'})\right)\end{aligned}$$

where $\binom{n}{i}$ is the number of configurations with i `on` RVs. Each of these quantities can be computed in $O(|\mathcal{V}'|)$ time rather than $O(2^{|\mathcal{V}'|})$ time it would take to iterate over the ground states. This identity is sometimes called the *binomial rule* in the lifted inference literature [Gogate and Domingos, 2012, Jha et al., 2010].

Symmetric Bipartite Graph Consider the symmetric bipartite graph shown in Figure 3.9b where each RV in groups $\mathcal{V}_1$ (top) and $\mathcal{V}_2$ (bottom) are connected. The RVs in each group are indistinguishable and the joint distribution depends only on the number of `on` RVs in each group. The distribution can be written as a joint LSS$^{\#}$-factor $\bar{\theta}^{\#}_{\mathcal{V}_1\mathcal{V}_2} : \bar{\mathcal{D}}^{\#}_{\mathcal{V}_1} \times \bar{\mathcal{D}}^{\#}_{\mathcal{V}_2} \to \mathbb{R}$ whose domain ranges over all counts of `on` RVs in each group $\mathcal{V}_1$ and $\mathcal{V}_2$. That is,

$$\begin{aligned}\text{gr}(y^{\#ij}_{\mathcal{V}_1\mathcal{V}_2}) &= \text{gr}(y^{\#i}_{\mathcal{V}_1}) \times \text{gr}(y^{\#j}_{\mathcal{V}_2}) \\ &= \{(x_v)_{v\in\mathcal{V}} : \sum_{v\in\mathcal{V}_1} x_v = i, \ \sum_{v\in\mathcal{V}_2} x_v = j\}\end{aligned}$$

is the set of ground states with i `on` RVs in $\mathcal{V}_1$ and j `on` RVs in $\mathcal{V}_2$ where $y^{\#ij}_{\mathcal{V}_1\mathcal{V}_2} := (y^{\#i}_{\mathcal{V}_1}, \ y^{\#j}_{\mathcal{V}_2})$ is a tuple of count symbols for both groups. The factor template is $\theta_{\bar{\alpha}} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ for any a, b, c, d. The factor value is

$$\begin{aligned}\bar{\theta}^{\#}_{\mathcal{V}_1\mathcal{V}_2}(y^{\#ij}_{\mathcal{V}_1\mathcal{V}_2}) &= ((n_1 - i)\cdot(n_2 - j))\cdot a \ + \ ((n_1 - i)\cdot j)\cdot b \ + \\ &\quad (i\cdot(n_2 - j))\cdot c \ + \ (i\cdot j)\cdot d\end{aligned}$$

with i `on` RVs in $\mathcal{V}_1$ and j `on` RVs in $\mathcal{V}_2$ which holds since the first term is the number of `off` RVs in group one and `off` RVs in group two, the second term is the number of `off` RVs in group one and `on` RVs in group two, the third term is the number of `on` RVs in group one and `off` RVs in group two, and the fourth term is the number of `on` RVs in group one and `on` RVs in group two. The factor template $\theta_{\bar{\alpha}}$ assigns values a,b,c, and d, respectively, to these pairs.

Since this model can be written as a compact LSS$^{\#}$-factor, exact inference can be performed efficiently by iterating over the groups of symmetric states rather than the exponentially larger group of ground states. That is,

$$\begin{aligned}
\Phi_0(\boldsymbol{\theta}) &= \max_{j=0}^{n_2} \max_{i=0}^{n_1} \bar{\theta}^{\#}_{\mathcal{V}_1\mathcal{V}_2}(y^{\#i}_{\mathcal{V}_1}, y^{\#i}_{\mathcal{V}_2}) \qquad \text{and} \\
\log(Z(\boldsymbol{\theta})) &= \log \sum_{j=0}^{n_2} \sum_{i=0}^{n_1} \binom{n_1}{i}\binom{n_2}{j} \cdot \exp\left(\bar{\theta}^{\#}_{\mathcal{V}_1\mathcal{V}_2}(y^{\#i}_{\mathcal{V}_1}, y^{\#j}_{\mathcal{V}_2})\right)
\end{aligned}$$

where $\binom{n_1}{i}\binom{n_2}{j}$ is the number of configurations with i `on` RVs in $\mathcal{V}_1$ and j `on` RVs in $\mathcal{V}_2$. Each of these quantities can be computed in $O(|\mathcal{V}_1| \cdot |\mathcal{V}_2|)$ time rather than the $O(2^{|\mathcal{V}_1|+|\mathcal{V}_2|})$ time it would take to iterate over the ground states. This identity is sometimes called the *generalized binomial rule* in the lifted inference literature [Gogate and Domingos, 2012, Jha et al., 2010].

3.8.2 Other Identities

We note that the counting-factor identity is a special case of a more general class of MLN models where the repetitive factor structure can be exploited to perform exact (non-relaxed) inference efficiently. Other examples include rules for computing the partition function of a symmetric transitivity formula (e.g., the friendship rule used in the symmetric friend-smoker MLN) and an encoding of the classic "birthday paradox" in FOL [Kazemi et al., 2016]. When such rules are available, they produce an exponential reduction in the inference cost;

consequently, developing a more complete set of rules and is an active area of research.

Although we are rarely interested in purely symmetric models, these specialized inference rules can sometimes be exploited within sub-models [e.g., Milch et al., 2008, Gogate and Domingos, 2012]. For example, Milch et al. [2008] performs a lifted variable elimination algorithm but at each bucket checks if accumulated bucket factor can be represented as a counting factor; if so, the factors are implicitly aggregated and eliminated to produce a counting factor message. We do not discuss more general identities or methods for incorporating them into exact lifted inference procedures as this is not the focus of our thesis.

Chapter 4

Lifted Generalized Dual Decomposition

In this chapter, we develop a coarse-to-fine lifted inference framework for models with large symmetric groups of factors perturbed by asymmetric evidence factors. Our framework is based on the technique of generalized dual decomposition (GDD) for sum-inference queries. Our key idea is to artificially impose symmetry restrictions on the inference relaxation which are incrementally split to allow the approximation to represent model asymmetries more accurately. In contrast to the majority of lifted inference frameworks that exploit exact symmetry, our lifted GDD algorithm has strong control over the computational cost of the approximation. In contrast to heuristic frameworks that control cost, our lifted GDD algorithm maintains a concrete connection to the ground inference problem. We verify the utility of our approach empirically, demonstrating increasing gains for increasingly large models, and demonstrating the benefit of using preliminary (over-symmetric) inference estimates to select the split of symmetry groups.

This chapter is based on our work originally published in Gallo and Ihler [2018a].

4.1 Introduction

Standard lifted inference procedures identify and exploit exact symmetries in inference computations. However, in general, real problems contain no exact symmetries; instead they often contain a symmetric substructure perturbed by asymmetric evidence factors. In web-based collective classification, for example, no two web pages have the exact same link structure or local classification score (since each web page has distinct text). In models of this form, lifted inference techniques that exploit exact symmetry deteriorate to ground inference, negating all potential efficiency of the symmetric representation.

Many approximate inference algorithms that have been developed to exploit this symmetric substructure provide control over the computational cost of the approximation but do not provide accuracy guarantees. For example, in MLN models, Singla et al. [2014] executes a lifted loopy belief propagation algorithm that projects messages to a symmetric form with controlled cost; however, the error incurred by this procedure is difficult to analyze. Another body of work constructs an over-symmetric approximate model (i.e., a model with more symmetries that approximates the original model in some way) on which inference is performed [Venugopal and Gogate, 2014a, Van den Broeck and Darwiche, 2013] (see Section 3.7.2 for details on these methods). The degree of symmetry in the approximate model is a user-specified parameter that controls the computational cost. However, it is not clear how the inference estimates on the approximate model relate to desired inference quantities on the original model. Additionally, models of this class are often forced to select the approximate symmetry groups *a priori* using the initial model potentials which may not represent the final inference quantities well. Ideally, preliminary inference estimates should be used to select the symmetry groups used in the inference approximation.

An emerging class of methods [Sarkhel et al., 2015, Habeeb et al., 2017], known as coarse-to-fine inference, has controlled computational cost but, unlike the aforementioned approaches,

also maintains a concrete connection to the ground problem. The basic idea of coarse-to-fine inference is to set up and solve a restricted inference problem with artificially imposed symmetry. The symmetry restrictions produce a problem whose solution has low accuracy but is obtained at a low cost. One of the symmetry restrictions is then split to produce a problem whose solution has higher accuracy but is obtained at a higher cost. These two steps – of solving a restricted problem, then choosing and splitting a symmetry restriction – are iterated until an exact solution is obtained or until computational resources run out. This coarse-to-fine procedure is used, for example, in Habeeb et al. [2017] to perform stereo depth perception. First, groups of pixels are forced to take the same label and MAP inference is performed using this restriction. This MAP inference problem is more efficient than the original problem since it requires reasoning over groups of pixels rather than individual pixels. To make the solution more accurate, a set of pixels in a symmetry group is split into its own group if it is believed that the solution can be improved by assigning it its own disparity label.

In this chapter, we develop a coarse-to-fine inference procedure for sum-inference queries based on the generalized dual decomposition (GDD) method [Ping et al., 2015] (see Section 2.5.2 for a discussion of GDD). Like other coarse-to-fine approaches, we maintain full control over the inference cost while maintaining a concrete connection to the ground problem (in contrast to lifted inference approaches that exploit exact symmetries with no control over cost and in contrast to heuristic methods that control cost with no accuracy guarantees). However, in contrast to the coarse-to-fine approaches mentioned above [Sarkhel et al., 2015, Habeeb et al., 2017], our approach is based on enforcing symmetry on variational parameters in the GDD algorithm, rather than enforcing symmetry in the RV configurations. This allows us to perform sum-inference queries and a variational optimization that produces a monotonically improving upper bound on the partition function.

Selecting the split of symmetry groups is a critical problem; in this chapter, we develop

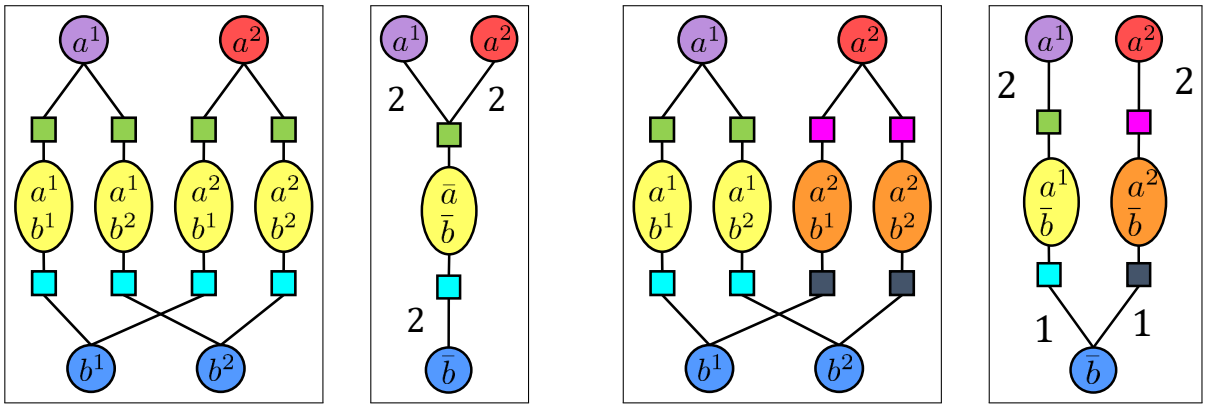
a cheap yet effective approach that clusters groups of parameters with similar gradients. Similar gradients suggest that these parameters can be treated equivalently, at least at the current point in the optimization process, while significant gradient differences suggest that the objective could be improved significantly by allowing the parameters to evolve independently. We empirically demonstrate the utility of this symmetry selection over purely structural choices.

Chapter Organization We organize the chapter as follows. In Section 4.2, we introduce a new over-symmetric generalized dual decomposition (GDD) inference objective and develop a modified lifted region graph that is capable of representing restricted symmetry sets (in contrast to the lifted region graph used in Section 3.4 which represents exact symmetries). In Section 4.3, we present an anytime coarse-to-fine inference algorithm that interleaves splitting (where we group parameters with similar gradient) with inference on the over-symmetric objective developed in the preceding section. In Section 4.4, we present experimental results demonstrating the utility of our approach.

4.2 Over-symmetric Lifted Inference

Classic lifted inference techniques exploit exact symmetries in a ground inference problem. As we discussed in Section 3.4, for decomposition bounds represented on a ground region graph $G = (R, S, E)$, the inference symmetries can be represented compactly on a lifted region graph $\bar{G} = (\bar{R}, \bar{S}, \bar{E})$. An example from that section is repeated here in Figure 4.1. In Figure 4.1c, we show a stable coloring of the ground region graph, and in Figure 4.1d, we show its associated lifted region graph.

In most practical problems, which contain no inference symmetries, lifted inference deteriorates into ground inference, negating all potential efficiency of the symmetric representation.



(a) Initial ground coloring (b) Nonstable lifted graph (c) Stable ground coloring (d) Stable lifted graph

Figure 4.1: Nonstable and stable colorings of a ground region graph and their corresponding lifted region graphs. (a) Ground graph colored by model symmetries and (b) corresponding nonstable symmetric lifted region graph – this chapter presents the first lifted inference algorithm that is able to represent inference computations on this structure. (c) Stable (symmetric) coloring and (d) corresponding stable lifted graph – classic lifted inference algorithms can only reason with exact symmetries of this form.

In this chapter, we remedy this problem by forcing groups of variational parameters to be symmetric, i.e., we force groups of ground separators in the ground region graph to be colored identically. Then, groups of ground regions are colored identically if and only if they have identically colored separator neighbors since each region is associated with an identical reparameterized factor (the region symmetries are the same as in Eq. (3.11) when we dealt with exact inference symmetries).

Roughly speaking, our method amounts to utilizing symmetries that would typically be traversed during the symmetry detection phase of an exact lifted inference procedure (see Section 3.4 for an example of the iterative symmetry detection algorithm) to form a valid variational objective and produce inference estimates. Inference computations are represented on an *over-symmetric* lifted region graph where each lifted region represents a symmetric set of ground regions, each lifted separator represents a symmetric set of ground separators, and each lifted edge $\bar{e} = (\bar{r}, \bar{s}) \in \bar{E}$ corresponds to the set of ground edges $\text{gr}(\bar{e}) = \bar{E} \cap (\text{gr}(\bar{s}) \times \text{gr}(\bar{r}))$ between ground separators associated with lifted separator $\bar{s} \in \bar{S}$

and ground regions associated with lifted region $\bar{r} \in \bar{R}$. However, unlike the case of exact symmetry, in this over-symmetric case, each lifted separator may have more than two incident lifted regions in the lifted region graph. For example, for the initial model symmetries in Figure 4.1a, we are able to produce an inference estimate represented on an *over-symmetric* lifted region graph in Figure 4.1b.

4.2.1 Lifted Generalized Dual Decomposition

This section discusses how to perform over-symmetric sum-inference using the GDD bound presented in Section 2.5.2. The basic idea is to tie parameter *weights* $\boldsymbol{w}$ in addition to tying the cost-shifting factors $\boldsymbol{\delta}$. These symmetries can be exploited in the GDD bound (2.16) on the log partition function as

$$\begin{aligned} L_G^{GDD}(\boldsymbol{\delta}, \boldsymbol{w}) &= \sum_{r \in R} \underset{x_r}{\overset{\boldsymbol{w}_r}{\texttt{lsew}}}\, \theta_r^{\delta}(x_r) \\ &= \sum_{\bar{r} \in \bar{R}} \sum_{r \in \mathrm{gr}(\bar{r})} \underset{x_r}{\overset{\boldsymbol{w}_r}{\texttt{lsew}}}\, \theta_r^{\delta}(x_r) \\ &= \sum_{\bar{r} \in \bar{R}} |\mathrm{gr}(\bar{r})| \cdot \underset{\bar{x}_{\bar{r}}}{\overset{\boldsymbol{w}_{\bar{r}}}{\texttt{lsew}}}\, \bar{\theta}_{\bar{r}}^{\bar{\delta}}(\bar{x}_{\bar{r}}) \;=:\; \bar{L}_{\bar{G}}^{GDD}(\boldsymbol{\delta}_{\bar{S}}, \boldsymbol{w}_{\bar{S}}) \end{aligned} \tag{4.1}$$

where the *lifted-GDD-objective* $\bar{L}_{\bar{G}}^{GDD}(\boldsymbol{\delta}_{\bar{S}}, \boldsymbol{w}_{\bar{S}})$ makes the symmetry exploitation explicit.

4.2.1.1 Optimizing the Lifted Objective

We use a black-box convex optimization procedure to optimize the lifted objective Eq. (4.1) using its objective and gradient information. The gradient of the over-symmetric objective is equal to the sum of gradients of associated ground parameters (by the chain rule). That

is, letting $\bar{g}^{\delta}_{\bar{s}}(\bar{x}_{\bar{s}}) := \frac{\partial \bar{L}}{\partial \delta_{\bar{s}}(\bar{x}_{\bar{s}})}$ be shorthand for the gradient,

$$\begin{aligned}
\bar{g}^{\delta}_{\bar{s}}(\bar{x}_{\bar{s}}) &= \sum_{s \in \mathrm{gr}(\bar{s})} g^{\delta}_{s}(x_s) \\
&= \sum_{\bar{s}_{(\bar{r},\bar{r}')} \in \bar{s}_{gradAll}} \; \sum_{s \in \mathrm{gr}(\bar{s}_{(\bar{r},\bar{r}')})} g^{\delta}_{s}(x_s) \\
&= \sum_{\bar{s}_{(\bar{r},\bar{r}')} \in \bar{s}_{gradAll}} |\mathrm{gr}(\bar{s}_{(\bar{r},\bar{r}')})| \cdot g^{\delta}_{\bar{s}_{(\bar{r},\bar{r}')}}(\bar{x}_{\bar{s}})
\end{aligned}$$

where the second line groups ground separators with identical gradients, the third line lifts the computation by performing the template computation and scaling it by the number of identical ground terms, and $x_s = \bar{x}_{\bar{s}}$, meaning that the index into the ground factor equals the index into the template factor. (Recall from Section 3.4.2 that $\bar{s}_{gradAll}$ corresponds to sets of ground separators associated with $\bar{s}$ and with identical gradient.)

In practice, we utilize unrestricted parameters $\boldsymbol{\eta}_{\bar{S}} = \{\eta_{\bar{s}} : \bar{s} \in \bar{S}\}$ and transform the weights by defining $w_{\bar{s}} = \epsilon + \exp(\bar{\eta}_{\bar{s}})$ where $\epsilon > 0$ is a small number. This transformation ensures positivity of the weights and allows us to perform unrestricted optimization over $\boldsymbol{\eta}_{\bar{S}}$. We also relax the sharp $w^0_v = \max(0, w^{\Delta}_{\bar{v}})$ defining the univariate weight (from Eq. (2.17)) to $w^0_{\bar{v}} = a(w^{\Delta}_{\bar{v}})$ where

$$a(x) = \epsilon + t \cdot \log(1 + \exp(x/t)) \tag{4.2}$$

for a small $t > 0$. Since $a(x) \geq \max(0, x)$, our relaxation produces larger univariate weights and hence a valid upper bound on the true GDD bound (and hence on the log partition function).

Note that the global descent we use in this work does not take full advantage of the decomposed form of the objective. It may be possible to perform the optimization more efficiently by performing a block coordinate descent on the lifted objective, analogous to the strategy employed by Ping et al. [2015] for the ground GDD objective. However, we leave this

direction to future work.

4.3 Coarse-to-Fine Anytime Inference

Having defined a tractable over-symmetric lifted inference objective and optimization procedure, we now show how to adaptively select the symmetry groups in a coarse-to-fine manner, producing approximations that can trade off quality versus time. We first address the question of how to select the split in Section 4.3.1. We then present an anytime inference algorithm that interleaves inference updates with splitting operations in Section 4.3.2.

4.3.1 Gradient Clustering

A key component of any coarse-to-fine process is the concept of *splitting*, in which we relax some of the over-symmetric assumptions that make the procedure efficient but inexact. In this section, we develop our procedure for splitting a symmetric group of ground separators into two finer groups. We first present the basic concepts, then show how to exploit symmetries within the computations.

To judge the quality of a partition, we measure the distance of its associated ground gradients to an over-symmetric projection of those gradients. When the distance is zero, the over-symmetric and unrestricted gradients are identical and the over-symmetric restriction incurs no error (locally). Similarly, when the distance is small, we expect the error incurred by the over-symmetric restriction to be small.

For any lifted separator $\bar{s} \in \bar{S}$, the distance of the gradients associated with its ground separators $\mathrm{gr}(\bar{s})$ to their over-symmetric projection is

$$d(\bar{s},\ \gamma_{\bar{s}}) = \sum_{s \in \mathrm{gr}(\bar{s})} ||g_s^\delta - \gamma_{\bar{s}}||_2^2, \qquad \text{where} \qquad \gamma_{\bar{s}}(\bar{x}_{\bar{s}}) = \frac{1}{|\mathrm{gr}(\bar{s})|} \sum_{s \in \mathrm{gr}(\bar{s})} g_s^\delta(\bar{x}_{\bar{s}}) \tag{4.3}$$

is the average gradient factor, i.e., the factor that is nearest to the ground gradients in L^2-norm, and $\bar{x}_{\bar{s}}$ ranges over all values in the domain.

Now that we have a way to measure the quality of a partition, we are interested in splitting $\bar{s}$ into lifted separators $\bar{s}'_1$ and $\bar{s}'_2$ (with associated ground separators $\{\mathrm{gr}(\bar{s}'_1), \mathrm{gr}(\bar{s}'_2)\}$ that partition $\mathrm{gr}(\bar{s})$), such that the over-symmetric projected gradient is as close as possible to the ground gradient. That is, we seek

$$\underset{\{\mathrm{gr}(\bar{s}'_1),\, \mathrm{gr}(\bar{s}'_2)\}}{\arg\min} \quad d(\bar{s}'_1,\ \gamma_{\bar{s}'_1}) \ +\ d(\bar{s}'_2,\ \gamma_{\bar{s}'_2}). \tag{4.4}$$

Note that the partitions $\mathrm{gr}(\bar{s}'_1)$ and $\mathrm{gr}(\bar{s}'_2)$ associated with the lifted separators appear implicitly in the definitions of $d(\bar{s}'_1,\ \gamma_{\bar{s}'_1})$ and $d(\bar{s}'_2,\ \gamma_{\bar{s}'_2})$.

4.3.1.1 Lifted Computation

The above computations can be *lifted* by exploiting symmetry in the ground gradients. For a lifted separator $\bar{s}$, recall that $\mathrm{gr}(\bar{s}_{gradAll})$ partitions ground separators $\mathrm{gr}(\bar{s})$ into groups of identical gradient. Using this gradient symmetry, we rewrite the projected distance computation (4.3) as

$$d(\bar{s}, \gamma_{\bar{s}}) \ =\ \sum_{\bar{s}_{(\bar{r},\bar{r}')} \in \bar{s}_{gradAll}} |\mathrm{gr}(\bar{s}_{(\bar{r},\bar{r}')})| \cdot ||g^{\delta}_{\bar{s}_{(\bar{r},\bar{r}')}} - \gamma_{\bar{s}_{(\bar{r},\bar{r}')}}||_2^2 \qquad \text{where} \tag{4.5}$$

$$\gamma_{\bar{s}} \ =\ \frac{1}{\mathrm{gr}(\bar{s})} \sum_{\bar{s}_{(\bar{r},\bar{r}')} \in \bar{s}_{gradAll}} |\mathrm{gr}(\bar{s}_{(\bar{r},\bar{r}')})| \cdot g^{\delta}_{\bar{s}_{(\bar{r},\bar{r}')}}. \tag{4.6}$$

By a symmetry argument, we can conclude that the optimal split (4.4) groups identical ground gradients together. Thus, we perform a search only over a split $\{\bar{s}'_1,\ \bar{s}'_2\}$ of the

gradient separators $\bar{s}_{gradAll}$

$$\arg\min_{\{\bar{s}'_1,\, \bar{s}'_2\}} \; d(\bar{s}'_1, \gamma_{\bar{s}'_1}) \; + \; d(\bar{s}'_2, \gamma_{\bar{s}'_2}). \tag{4.7}$$

The problem (4.7) corresponds to a standard weighted clustering problem. Here, the gradient templates $\{g_{\bar{s}_{(\bar{r},\bar{r}')}} : \bar{s}_{(\bar{r},\bar{r}')} \in \bar{s}_{gradAll}\}$ act as the "data" and the term $|\mathrm{gr}(\bar{s}_{(\bar{r},\bar{r}')})|$ appearing in Eq. (4.5) and Eq. (4.6) weights the data by the number of associated identical ground gradients.

Finding the optimal clustering requires a search over a number of splits exponential in the number of data points. This optimization is generally intractable, but many efficient and accurate approximations, such as K-means [MacQueen et al., 1967, Friedman et al., 2001], have been developed and can be employed here. When the model contains only binary RVs, which is typical in probabilistic logic and MLNs, an additional simplification arises. In this case, each ground gradient vector is length two, but defined by a single scalar value (the other element is redundant since the ground gradients are pseudo-marginals that sum to one). Thus, the clustering is performed on scalar data, and the optimal clustering can be found efficiently, simply by sorting the gradient values and evaluating the sum of squared differences for all possible splits efficiently via dynamic programming.

Selecting a Separator to Split Previously, we discussed how to partition the ground separators associated with a single lifted separator. But we also need to select which lifted separator to split. A natural choice is to select the lifted separator whose split produces the largest *change* in gradient distance

$$\arg\min_{\bar{s}\in\bar{S}} \; d(\bar{s}'_1,\; \gamma_{\bar{s}'_1}) \; + \; d(\bar{s}'_2,\; \gamma_{\bar{s}'_2}) \; - \; d(\bar{s},\; \gamma_{\bar{s}}). \tag{4.8}$$

Note that this criterion is biased toward choosing lifted separators with large ground sets. To see this, consider the case of binary variables (so that each gradient can be represented as a scalar quantity); then, the maximum value of the distance (4.3) is proportional to the ground set size: $d(\bar{s}) \in [0,\ 4 \cdot |\mathrm{gr}(\bar{s})|]$. The upper bound results from the fact that the sum ranges over $|\mathrm{gr}(\bar{s})|$ terms, each of which is bounded by 4, since each gradient is a difference of pseudo-marginals and hence $g_s^\delta \in [-1, 1]$. Similarly, since their average is in the same range, the difference $g_s^\delta - \gamma_{\bar{s}} \in [-2, 2]$, and hence its square is in the range $[0, 4]$.

4.3.1.2 Baseline Splitting Methods

In our experiments (Section 4.4), we compare our gradient-based split selection to a number of simple split selection techniques that do not use inference estimates. One method, labeled `Syntactic` in the experiments, selects a separator group uniformly at random and splits it. This corresponds to the split method used in the standard stable coloring algorithm (which is not concerned with the quality of any intermediate partitions). Another choice, labeled `Largest` in the experiments, splits the largest separator group into two random partitions of roughly equal size. Intuitively, this seems reasonable if no other information is available.

4.3.2 Coarse-to-Fine Anytime Inference

In this section, we show how to perform coarse-to-fine inference which interleaves inference with splitting modifications. The sequence of refinements eventually terminates at the stable coloring which represents a solution to the ground problem (i.e., the solution obtained by standard lifted inference algorithms which detect exact symmetries before performing inference). This procedure is summarized in Algorithm 8.

An important practical difficulty is deciding how to allocate computational work between inference and splitting. An ideal system would predict the benefit of spending a fixed amount

Algorithm 8 Coarse-to-Fine anytime inference

Input: A region graph G, an initial lifted region graph $\bar{G}$, a number of full inference sweeps I, a multiplicative splitting cost increment β.

$t \leftarrow 0$

while $\bar{G}$ is not *stable* **do**

`Step 1` (Inference): Perform I inference sweeps hot-started at the old parameterization:

$$[\bar{\boldsymbol{\delta}}^t, \bar{\boldsymbol{w}}^t, L_{t;\cdot}, c_{t;\cdot}] \;=\; \texttt{infer}(G^t,\; \bar{\boldsymbol{\delta}}^{t-1},\; \bar{\boldsymbol{w}}^{t-1};\; I)$$

`Step 2` (Split): Sequentially split groups until inference cost increases by β:

Compute optimal clustering and scores for each $\bar{s} \in \bar{S}$ (via Eq. (4.5))

$\tau_{old} \leftarrow \texttt{Cost}(\bar{G})$ // Old inference cost

while $(\texttt{Cost}(\bar{G}) \;\leq\; \beta \cdot \tau_{old}) \;\;\wedge\;\; (\bar{G}$ not stable$)$ **do**

Select the lifted separator $\bar{s} \in \bar{S}$ with minimum split score (via Eq. (4.8))

$\bar{S} \;\leftarrow\; (\bar{S} \setminus \bar{s}) \;\cup\; \{\bar{s}'_1, \bar{s}'_2\}$ // Replace lifted separator $\bar{s}$ and ground groups

Split $\bar{R}$ with finer $\bar{S}$.

Compute clustering scores for new lifted separators $\bar{s}'_1$ and $\bar{s}'_2$ (via Eq. (4.5))

end while

`Step 3`: $t \leftarrow t + 1$

end while

$[\bar{\boldsymbol{\delta}}, \bar{\boldsymbol{w}}] = \texttt{infer}(G^t, \bar{\boldsymbol{\delta}}^t, \bar{\boldsymbol{w}}^t)$ // Inference at stable coloring

of work on each operation and use the prediction to choose the more effective operation. However, assessing the (long-term) benefit of either operation is difficult. Consequently, in this work we pursue a less ambitious approach that uses a fixed cost allocation between inference and splitting. The allocation is controlled by two metaparameters, $I \in \mathbb{N}^+$ and $\beta > 1$ where I is the number of inference sweeps to perform before iteratively performing splits (as in Section 4.3.1) until the inference cost increases by a factor of β. Note that after a split, only the scores associated with incident nodes need to be updated (the final line of the loop in `Step 2`) since the others remain unchanged until more inference is performed.

An inference summary, consisting of the inference costs $c_{t;\cdot}$ and objective values $L_{t;\cdot}$ is produced by an inference call on the tth graph structure in `Step 1`. In the experiments section,

the anytime performance of the algorithm is reported by plotting these inference summaries.

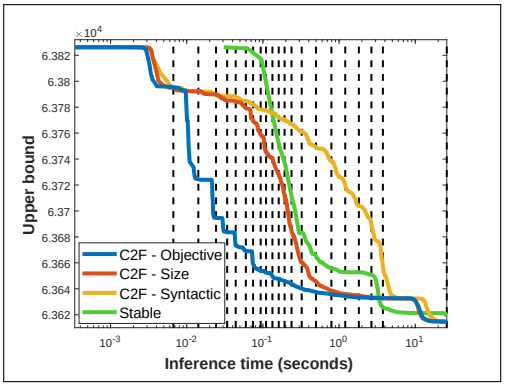
Cost to Recolor the Graph Care must be taken to update the ground separators associated with the split efficiently (the line $\bar{S} \leftarrow (\bar{S} \setminus \bar{s}) \cup \{\bar{s}'_1, \bar{s}'_2\}$). We implement the update operation by recoloring the smaller ground group, leaving the larger ground group its original color. Recoloring only the smaller group guarantees that each node changes color at most $\log_2(|R| + |S|)$ times, and consequently, the total cost of all recolorings is $O((|R| + |S|) \cdot \log_2(|R| + |S|))$, equal to the cost of the standard stable coloring algorithm [Berkholz et al., 2013].

When evaluating the timing performance of our model, we measure the cost of performing inference, but not the cost of identifying symmetries, which requires touching the ground model. Reporting results in this way is common in experiments on relaxed lifted inference in prior work [Mladenov et al., 2014a, Van den Broeck and Niepert, 2014, Venugopal and Gogate, 2014a], as well as for exact lifted variational inference approaches [Bui et al., 2012, 2014, Mladenov and Kersting, 2015] which assume problem symmetries are available before inference. Works such as Singla et al. [2014], Kersting et al. [2010] note that identifying (approximate) symmetries on the ground graph can significantly outweigh the cost of inference on a compact lifted graph. Although we do not address this potential issue in this chapter, in Chapter 5 we build approximations using MLN models and FOL-formulas that do not explicitly represent the symmetric sets of ground factors.

4.4 Experiments

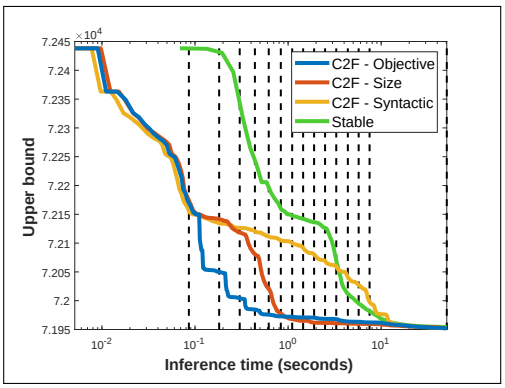
This section provides an empirical illustration of our coarse-to-fine lifted GDD algorithm. We demonstrate excellent any-time performance on problems with distinct soft evidence on every RV compared to exact lifted variational approaches which are forced to ground

Weight	Formula
b	$(\forall x \neq y)\ V(x) \Leftrightarrow V(y)$
u_x	$(\forall x)\ V(x)$



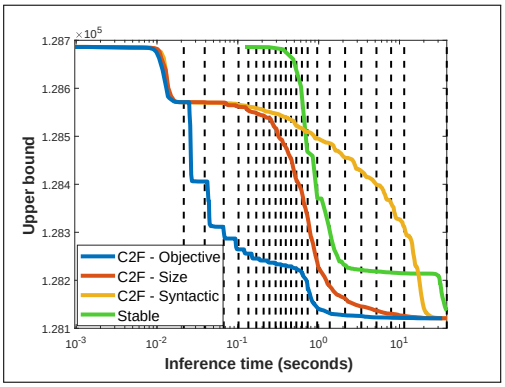
(a) Complete Graph

Weight	Formula
b	$(\forall\, x \neq y)\ \ L(x,y) \wedge (C(y) \Leftrightarrow C(x))$
u_x	$(\forall x)\ C(x)$
l_{xy}	$(\forall x \neq y)\ L(x,y)$



(b) Binary Collective Classification

Weight	Formula
b	$(\forall\, x \neq y)\ \ Q_1(x) \Leftrightarrow Q_2(y)$
b	$(\forall\, x \neq y)\ \ Q_2(x) \Leftrightarrow Q_3(y)$
b	$(\forall\, x \neq y)\ \ Q_3(x) \Leftrightarrow Q_1(y)$
u_x^i	$(\forall x)\ Q_i(x), \quad i \in \{1,2,3\}$



(c) Clique Cycle

Figure 4.2: Comparison of anytime inference at stable coloring (exact model symmetries) with our coarse-to-fine lifted GDD algorithm for three different splitting methods. Black dashed vertical lines indicate coarse-to-fine transitions for objective-based splitting (blue curve); others transition at approximately the same point. In general, the objective-based splitting (blue) method is superior to the methods that do not utilize inference information to select the split (red and yellow curves). The coarse-to-fine methods generally perform better than the stable coloring (green curve) that utilizes exact model symmetries, which are equivalent to the ground model in this case since models contain distinct univariate evidence.

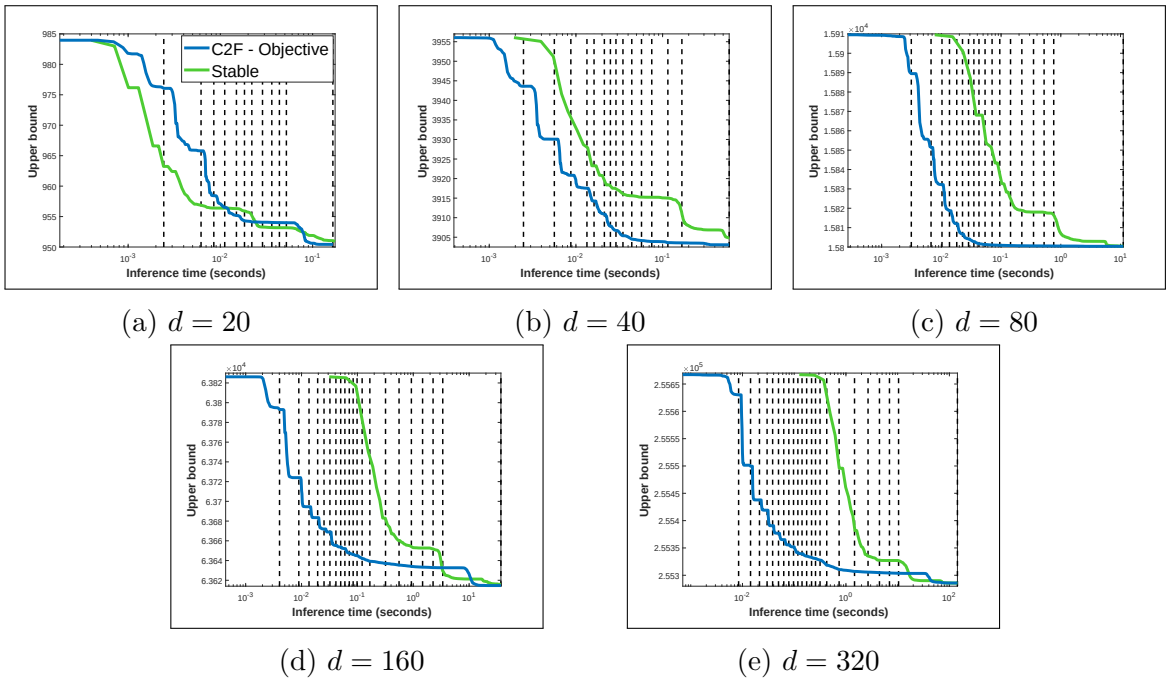
the entire problem. We demonstrate the superiority of selecting the split using objective information (via gradient clustering) over purely syntactic selection approaches.

4.4.1 Models and Methodology

Models The left panes of Figure 4.2 show the types of models we use for evaluation. Similar models, without evidence, were used in Mladenov and Kersting [2015] and Bui et al. [2014] to evaluate the performance of lifted reasoning on marginalization tasks. In all models, we use the pairwise strength parameter $b = 2.5$. The terms with base symbol u specify simulated evidence, which is drawn uniformly on $[-\zeta, \zeta]$ where $\zeta = \log((1 - 10^{-2})/10^{-2})$; this ensures that each marginal probability is in the range $[0.01, 0.99]$ when pairwise terms are omitted. For the collective classification problem, we also choose a random 10% of the x, y values associated with links $L(x, y)$ and set l_{xy} to a uniformly sampled value in the range $[0, 5.0]$. This quantity can be interpreted as the strength of a link between two web pages; in practice, this strength value might be obtained, for example, from a sentiment score on the text surrounding the link.

Note that, due to the presence of random evidence, these models contain *no* exact symmetries. Thus, standard lifted inference techniques which identify exact symmetries are forced to ground the model.

Parameter settings We set $\epsilon = 10^{-3}$, $t = 10^{-2}$ in the definition of the softening function in Eq. (4.2) for the softened objective lifted GDD objective. We perform a LBGFS black box optimization with rank 20 Hessian correction. To balance the work allocation between inference and model refinement, we perform $I = 30$ inference iterations, and interleave splitting refinement steps that increase the model cost by a factor of $\beta = 1.25$ (i.e., twenty five percent higher cost).



(a) $d = 20$ (b) $d = 40$ (c) $d = 80$

(d) $d = 160$ (e) $d = 320$

Figure 4.3: Scalability comparison. Panels (a)-(e) show the bound vs. time on instances of complete graphs with evidence, while varying the logical variable domain size d in the MLN specification (corresponding to the number of ground variables in the model). C2F exhibits increasing advantage as the model grows larger.

4.4.2 Results

Figure 4.2 compares our coarse-to-fine procedure with several baselines on each of our three test models. The blue curve ("C2F - objective") performs splitting using gradient clustering as discussed in Section 4.3.1. The yellow ("C2F - syntactic") and red ("C2F - Size") curves perform syntactic and size-based splitting; these methods do not use inference information to select the split and are described in Section 4.3.1.2. The green ("Stable") curve performs exact lifted inference on the graph's stable coloring; since our test models contain no symmetry, the stable coloring is simply the ground region graph.

We see that objective-based refinement significantly outperforms the other methods, with size-based refinement performing better than syntactic splitting. Objective-based splitting provides orders of magnitude speed-up over the stable coloring for similar quality results,

and all methods provide better early performance than the stable coloring, which must touch the entire model to even provide an initial bound.

Figure 4.3 demonstrates the scalability of our approach. Using the complete graph, we vary the number of model RVs, ranging from $d = 20$ to $d = 320$. As before, each RV is given random evidence u_x, resulting in a model with no exact symmetries. Each panel shows the running time for our coarse-to-fine method ("C2F - Objective", blue) and the stable coloring method (green). As the model size becomes large, our method improves its bound more quickly than the stable coloring approach does, resulting in increasingly superior anytime performance.

4.5 Conclusions and Future work

In this chapter, we developed a coarse-to-fine lifted inference framework for symmetric models perturbed by asymmetric evidence factors. Our method works by imposing over-symmetric constraints on the variational parameters of a GDD approximation, then, guided by preliminary inference estimates, selects the symmetry split that best represents the problem at hand. Our method provides orders of magnitude speed-up on our benchmarks, with increasing advantages on larger models.

Although this chapter focuses on sum-inference problems, the GDD framework is also applicable to MAP and marginal MAP problems (in which one subset of variables is marginalized and another maximized). Thus, it should be easy to extend our methods for controlling symmetry in the sum-inference objective to these other inference objectives.

Chapter 5

Lifted Weighted Mini-Bucket

In the preceding chapter, we developed a coarse-to-fine inference algorithm for symmetric models perturbed by asymmetric evidence. That algorithm's primary shortcoming is that it reasons over model factors and does not form high-order inference terms which are often necessary to obtain accurate inference estimates. In this chapter, we develop a coarse-to-fine inference algorithm that incorporates high-order inference terms while maintaining control over the degree of symmetry in the relaxed inference structure. Experimental results demonstrate the utility of high-order inference terms in (approximately) symmetric models, especially in models with strong repulsive potentials.

This chapter is based on work originally published in Gallo and Ihler [2018b].

5.1 Introduction

Computing inference quantities is a central problem in probabilistic graphical models. However, in many models, an exact inference computation is intractable and relaxed inference algorithms that reason over factors with a controlled number of RVs are necessary (as we discussed in Chapter 2). In some models, such as those with attractive factors (factors which encourage neighboring RVs to take the same value), accurate inference estimates may

be obtainable with a fully relaxed inference structure. However, in many other models, such as those with repulsive factors (which encourage neighboring RVs to take different values), inference on the fully relaxed structure produces inaccurate inference estimates, and it is necessary to reason over high-order terms (larger sets of RVs) in order to ensure consistency among the various sub-configurations.

In models with (approximately) symmetric structure, it is desirable to introduce symmetric high-order inference terms which can can be exploited during inference. Although introducing symmetric high-order inference terms is straightforward in principle, in practice, a number of difficulties arise. Most previous works that attempt to address these difficulties fall into one of two classes.

The first class of methods performs lifted inference with high-order inference terms assuming the model possesses exact symmetry (although this is often unrealistic). For *exact inference*, the lifted variable elimination algorithm [Poole, 2003, Braz et al., 2005] identifies elimination operations that are identical then performs each (exponentially large) computation one time; works such as Milch et al. [2008], Taghipour et al. [2012] extend the basic variable elimination algorithm to exploit counting factors that arise when symmetric factors are joined and eliminated (see the discussion in Section 3.8.1). For *relaxed inference*, the goal is to identify groups of high-order inference terms that can be treated identically; works such as Mladenov et al. [2014b] use high-order terms for MAP inference, while works such as Van den Broeck et al. [2012], Smith et al. [2016], Apsel [2016] use high-order terms for sum-inference.

The second class of methods is applicable to models with approximate (rather than exact) symmetry. The basic idea is to form an over-symmetric approximation to the model on which high-order inference terms can be easily incorporated. The method of Venugopal and Gogate [2014a] (reviewed in Section 3.6.2) forms a small MLN by clustering the logical constants into groups that are expected to have approximately identical marginals – for example, in a social

network we might heuristically form groups of people with similar attribute and relationship evidence. Any standard inference algorithm, including high-order relaxed inference and variable elimination for exact inference, can then be applied to the small over-symmetric model. Another work, Van den Broeck and Darwiche [2013], performs exact inference on the original model using an over-symmetric approximation to the evidence (which is chosen such that the exact inference computation can be performed efficiently). Lastly, Sen et al. [2009] allows a group of factors that are identical up to a user-defined error to be replaced with a symmetric group of factors (each with value equal to the group's average value). This creates an over-symmetric model on which a mini-bucket inference algorithm that exploits exact symmetry can be applied. These methods suffer, primarily, from two shortcomings. First, inference on the over-symmetric model produces biased inference estimates whose error is difficult to quantify, and second, the over-symmetric model must be constructed *a priori*. This latter restriction prevents preliminary inference estimates from being used to select the approximate symmetry groups.

In short, the first class of methods introduces high-order inference terms into a symmetric model but incurs no error, while the second class of methods incurs error but is able to introduce high-order inference terms into an asymmetric model. An emerging class of methods is coarse-to-fine inference algorithms that provides a principled (rather than heuristic) way to reason in models with approximate symmetry [Sarkhel et al., 2015, Habeeb et al., 2017]. We introduced a coarse-to-fine inference algorithm in the last chapter; however, that algorithm performed inference with lifted model factors and was not able to build symmetric high-order inference terms.

In this chapter, we develop a coarse-to-fine inference algorithm that incorporates high-order inference terms by joining several low-order lifted factors. Although performing a lifted join is straightforward in principle, in practice an important difficulty arises. Namely, although a group of lifted factors may possess low-order symmetry necessary for efficient lifted inference,

this group does not necessarily possess high-order symmetry and joining will, in general, produce high-order terms with no symmetry (this is an example of the shattering effect which occurs when symmetries do not line up, as discussed in Section 3.4). In this chapter, we focus on MLN models since their FOL-formulas can (often) be joined without breaking symmetry; that is, the FOL-formulas with low-order symmetry required by low-order relaxed inference also possess high-order symmetry needed to perform the lifted join operation.

All-in-all, our coarse-to-fine inference algorithm with high-order inference terms for MLN models interleaves inference with one of the following two modifications to the inference structure:

Splitting: This modification *splits* the symmetry of the variational parameters which allows the approximation to represent asymmetric evidence more accurately (this is the same basic idea as the algorithm in the previous chapter). In a MLN model, we split a group of logical constants, which in turn splits all lifted factors that use that group. For example, a group of people may be split into a two finer groups of people, and all lifted factors using the original group are then split into lifted factors using the finer groups.

Joining: This modification *joins* groups of lifted factors to create a high-order lifted factor which allows the approximation to resolve inconsistencies present in the low-order relaxation. In a MLN model, joining two lifted factors produces a lifted factor with the same form as a FOL-formula whose variables range over all subsets of objects in the operand factors. For example, joining a lifted factor over two groups of people $\bar{\Delta}_{\text{ppl};1}$ and $\bar{\Delta}_{\text{ppl};2}$ with a lifted factor over two groups of people $\bar{\Delta}_{\text{ppl};2}$ and $\bar{\Delta}_{\text{ppl};3}$ produces a lifted factor over all three groups of people.

Our framework allows the flexibility to decide how best to invest additional computation: either in resolving differences within the current symmetry groups (splitting), or in resolving

inconsistencies in the joint values of several lifted variables (joining). At a high level, we expect joining to be essential in repulsive models which produce strong local inconsistencies, while we expect joining to provide little benefit in attractive models which produce weak local inconsistencies. In this latter case, it is better to utilize small scope regions which can be split to finer symmetry groups than large scope regions can be (for the same computational cost) and thus capture problem asymmetry better. Our empirical results confirm this expected behavior.

Chapter Overview This chapter is organized as follows. We begin in Section 5.2 by introducing a modification to the lifted region graph presented in the last chapter. The modification incorporates additional structure that explicitly represents symmetry in groups of RVs; this additional structure will be necessary for our lifted join operations later in the chapter. In Section 5.3, we present a coarse-to-fine inference algorithm for MLN models where the relaxation is incrementally refined by splitting sets of logical constants; we present a gradient-based clustering algorithm to select the domain split. In Section 5.4, we show how two FOL-formulas with disjoint logical domains can be joined to produce a larger FOL-formula. Collectively, these sections define a complete algorithm for coarse-to-fine *decomposed* inference with high-order inference terms. In Section 5.5, we present a lifted variant of the weighted-mini-bucket algorithm [Liu and Ihler, 2011] which represents the same class of bounds as the decomposed form, but provides a good way to select the high-order regions (via relaxed variable elimination operations) and an efficient way to pass messages. In Section 5.6, we present empirical results justifying the utility of our method.

5.2 Lifted GDD with RV Symmetry

In Section 5.2.1, we introduce a modification to the lifted region graph presented in the last chapter that explicitly represents symmetry within groups of RVs. This class of relax-

ations requires additional symmetry restrictions (compared to the last chapter) that facilitate symmetry-preserving split and join operations in MLN models, as we will see in the following sections. In Section 5.2.2, we see how the RV symmetry can be used to define a *symmetric elimination order*; we exploit this property in the next section to select the elimination order (in addition to the other inference parameters) in a coarse-to-fine fashion.

5.2.1 Lifted Region Graph

Consider a lifted region graph $\bar{G} = (\bar{R}, \bar{S}, \bar{E})$ with a factor graph topology. That is, each lifted separator $\bar{s} \in \bar{S}$ is incident to one lifted RV $\bar{v} \in \bar{\mathcal{V}}$ and one lifted region $\bar{r} \in \bar{R} \setminus \bar{\mathcal{V}}$ where the set of lifted RVs $\bar{\mathcal{V}}$ associated with groups of ground RVs $\mathrm{gr}(\bar{\mathcal{V}}) := \{\mathrm{gr}(\bar{v}) : \bar{v} \in \bar{\mathcal{V}}\}$ that partition the set of model RVs $\mathcal{V}$.

Each lifted RV $\bar{v}$ is associated with univariate evidence factors associated with its set of ground RVs. With this representation, the lifted GDD bound (4.1) of the last chapter can be written as

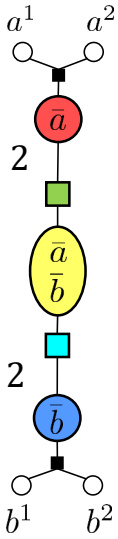
$$\begin{aligned} \bar{L}^{GDD}_{\bar{G}}(\boldsymbol{\delta}_{\bar{S}}, \boldsymbol{w}_{\bar{S}}) \;&=\; \sum_{\bar{r} \in (\bar{R} \setminus \bar{\mathcal{V}})} |\mathrm{gr}(\bar{r})| \cdot \operatorname*{\mathtt{lsew}}_{x_r}^{w_{\bar{r}}} \bar{\theta}^{\bar{\delta}}_{\bar{r}}(x_{\bar{r}}) \\ &\quad + \sum_{\bar{v} \in \bar{\mathcal{V}}} \sum_{v \in \mathrm{gr}(\bar{v})} \operatorname*{\mathtt{lsew}}_{x_v}^{w^0_{\bar{v}}} \Big(\theta_v(x_v) + \delta^0_{\bar{v}}(x_v) \Big) \end{aligned} \tag{5.1}$$

where the first line is the lifted computation of factor power-sum terms, the second line incorporates evidence at each lifted RV, and

$$\delta^0_{\bar{v}} \;=\; -\sum_{\bar{s} \in \bar{N}(\bar{v})} \delta_{\bar{s}} \qquad \text{and} \qquad w^0_{\bar{v}} \;=\; \max(1 - \sum_{\bar{s} \in \bar{N}(\bar{v})} w_{\bar{s}},\; 0)$$

are the sums of the cost-shifting factors (and weights) incident to the lifted RV $\bar{v}$.

An example lifted region graph using the lifted RV representation is shown in the embedded figure on the right. The model contains two lifted RVs labeled $\bar{a}$ and $\bar{b}$ and associated with ground RVs $\text{gr}(\bar{a}) = \{a^1, a^2\}$ and $\text{gr}(\bar{b}) = \{b^1, b^2\}$, respectively. Each ground RV is associated with a univariate evidence factor, shown as a white node; each ground RV is connected to its containing lifted RV via a black square node (note that this node is not a separator in the graph, even though, like a separator, it is represented as a square). Note that this lifted region graph represents the same set of parameter tying constraints as the graph in Figure 4.1b from the last chapter.



5.2.2 Lifted Elimination Order

A lifted elimination order $\bar{\boldsymbol{o}} = (\bar{v}_1, \ldots, \bar{v}_{\bar{n}})$ is an order on lifted RVs $\bar{\mathcal{V}}$ that corresponds to all ground elimination orders in which a ground RV associated with a lifted RV appears before a ground RV associated with a later lifted RV. For example, a lifted elimination order $\bar{\boldsymbol{o}} = (\bar{v}_1, \bar{v}_2)$ where the lifted RVs are associated with ground RVs $\text{gr}(\bar{v}_1) = \{1, 2\}$ and $\text{gr}(\bar{v}_2) = \{3, 4\}$ is associated with the set of ground elimination orders $\{(1, 2, 3, 4), (2, 1, 3, 4), (1, 2, 4, 3), (2, 1, 4, 3)\}$, each of which orders ground RVs associated with $\bar{v}_1$ before ground RVs associated with $\bar{v}_2$. As another example, consider the friend-smoker-MLN with the problem's predicates ordered as $\bar{\boldsymbol{o}} = (\texttt{Fr}(\bar{\Delta}_{\text{ppl}}, \bar{\Delta}_{\text{ppl}}), \texttt{Sm}(\bar{\Delta}_{\text{ppl}}), \texttt{Ca}(\bar{\Delta}_{\text{ppl}}))$. One possible ground elimination order is

$$\Big(\texttt{Fr}(A, B), \texttt{Fr}(C, A), \texttt{Fr}(A, C), \texttt{Fr}(B, A), \texttt{Fr}(B, C), \texttt{Fr}(C, B), \\ \texttt{Sm}(A), \texttt{Sm}(C), \texttt{Sm}(B), \ \texttt{Ca}(C), \texttt{Ca}(B), \texttt{Ca}(A)\Big)$$

which orders the friendship ground RVs, then the smoke ground RVs, then the cancer ground RVs for people $\{A, B, C\}$.

We want to ensure that the lifted GDD objective (Eq. 5.1) evaluated on lifted elimination order $\bar{\boldsymbol{o}}$ is equal to any ground computation carried out along a ground elimination order consistent with the lifted elimination order $\bar{\boldsymbol{o}}$. To do this, we restrict all weights in a lifted region associated with the same lifted RV to be identical; this means that the power-sum operator commutes (as we discussed in Section 2.5.2) along all permutations of the ground RVs. For example, consider a ground region $r \in R$ with scope $\text{sc}(r) = \{\texttt{Fr}(A, B), \texttt{Sm}(A), \texttt{Sm}(B)\}$ (corresponding to the rule "if A and B are friends then probably both smoke or both don't smoke"). By restricting the weight parameters associated with the smoking predicate to be identical, i.e., $w^{\bar{r}}_{\texttt{Sm}} = w^{r}_{\texttt{Sm}(A)} = w^{r}_{\texttt{Sm}(B)}$, the decomposed inference terms

$$\underset{x_{\texttt{Fr}(A,B)}}{\overset{w^{\bar{r}}_{\texttt{Fr}}}{\texttt{lsew}}} \; \underset{x_{\texttt{Sm}(A)}}{\overset{w^{\bar{r}}_{\texttt{Sm}}}{\texttt{lsew}}} \; \underset{x_{\texttt{Sm}(B)}}{\overset{w^{\bar{r}}_{\texttt{Sm}}}{\texttt{lsew}}} \; \theta^{\delta}_{r}(x_r) \;=\; \underset{x_{\texttt{Fr}(A,B)}}{\overset{w^{\bar{r}}_{\texttt{Fr}}}{\texttt{lsew}}} \; \underset{x_{\texttt{Sm}(B)}}{\overset{w^{\bar{r}}_{\texttt{Sm}}}{\texttt{lsew}}} \; \underset{x_{\texttt{Sm}(A)}}{\overset{w^{\bar{r}}_{\texttt{Sm}}}{\texttt{lsew}}} \; \theta^{\delta}_{r}(x_r)$$

corresponding to an elimination over order $(\texttt{Fr}(A, B), \texttt{Sm}(A), \texttt{Sm}(B))$ on the LHS, and over order $(\texttt{Fr}(A, B), \texttt{Sm}(B), \texttt{Sm}(A))$ on the RHS (where smoke RVs are swapped), are identical.

5.3 Anytime Inference: Logical-Domain Splitting

This section presents a coarse-to-fine anytime inference algorithm for MLN models with asymmetric evidence, which is codified in Algorithm 9. Each step of the algorithm splits a set of logical constants (such as a set of people) into two sets. Each FOL-formula using that set is then split into a set of FOL-formulas whose LVs range over the smaller sets. Thus, in contrast to the coarse-to-fine algorithm in the last chapter (Algorithm 8) which splits a single lifted separator, we perform a structured split that preserves the FOL-structure of the formula. Maintaining this structure is crucial for joining operations (`Step 3` of the algorithm) which require a FOL-formula representation, as we will discuss in the next section.

The rest of this chapter discusses the split operation and the selection of the split. In

Algorithm 9 Coarse-to-Fine anytime inference with joins for MLN models

Input: A region graph G, an initial lifted region graph, a number of full inference sweeps I, a multiplicative splitting cost increment β, a maximum separator size `iBound`.

$t \leftarrow 0$

while $\bar{G}$ is not *stable* **do**

`Step 1` (Inference): Perform I sweeps of inference hot-started at the old parameterization:

$$[\bar{\boldsymbol{\delta}}^t,\ \bar{\boldsymbol{w}}^t, L_{t;\cdot}, c_{t;\cdot}] \ = \ \texttt{infer}(\bar{G}^t,\ \bar{\boldsymbol{\delta}}^{t-1},\ \bar{\boldsymbol{w}}^{t-1};\ I)$$

`Step 2` (Split): Sequentially split groups until inference cost increases by β:

Set $\tau_{old} \leftarrow \texttt{Cost}(\bar{G})$

repeat

`Step 2a` (Gradient Clustering): Select $\bar{\Delta} \in \bar{\boldsymbol{\Delta}}$ and a split $\bar{\Delta}' = \{\bar{\Delta}'_1, \bar{\Delta}'_2\}$ via gradient clustering (Alg. 10).

`Step 2b` (Structure Split): Split lifted logical domains, separators, regions, RVs, and elimination order:

$$\begin{aligned}
\bar{\boldsymbol{\Delta}} &\leftarrow (\bar{\boldsymbol{\Delta}} \setminus \bar{\Delta}) \cup \bar{\Delta}' \\
\bar{S} &\leftarrow (\bar{S} \setminus \bar{S}^{\bar{\Delta}}) \cup \bar{S}^{\bar{\Delta}'} \\
\bar{R} &\leftarrow (\bar{R} \setminus \bar{R}^{\bar{\Delta}}) \cup \bar{R}^{\bar{\Delta}'} \\
\bar{\mathcal{V}} &\leftarrow (\bar{\mathcal{V}} \setminus \bar{\mathcal{V}}^{\bar{\Delta}}) \cup \bar{\mathcal{V}}^{\bar{\Delta}'} \\
\bar{\boldsymbol{o}} &\leftarrow \text{Split}(\bar{\boldsymbol{o}},\ \bar{\mathcal{V}}^{\bar{\Delta}},\ \bar{\mathcal{V}}^{\bar{\Delta}'})
\end{aligned}$$

until $\texttt{Cost}(\bar{G}) \ \geq \ \beta \cdot \tau_{old}$

`Step 3` (Join): Perform a sequence of joins with maximum factor size `iBound`+1.

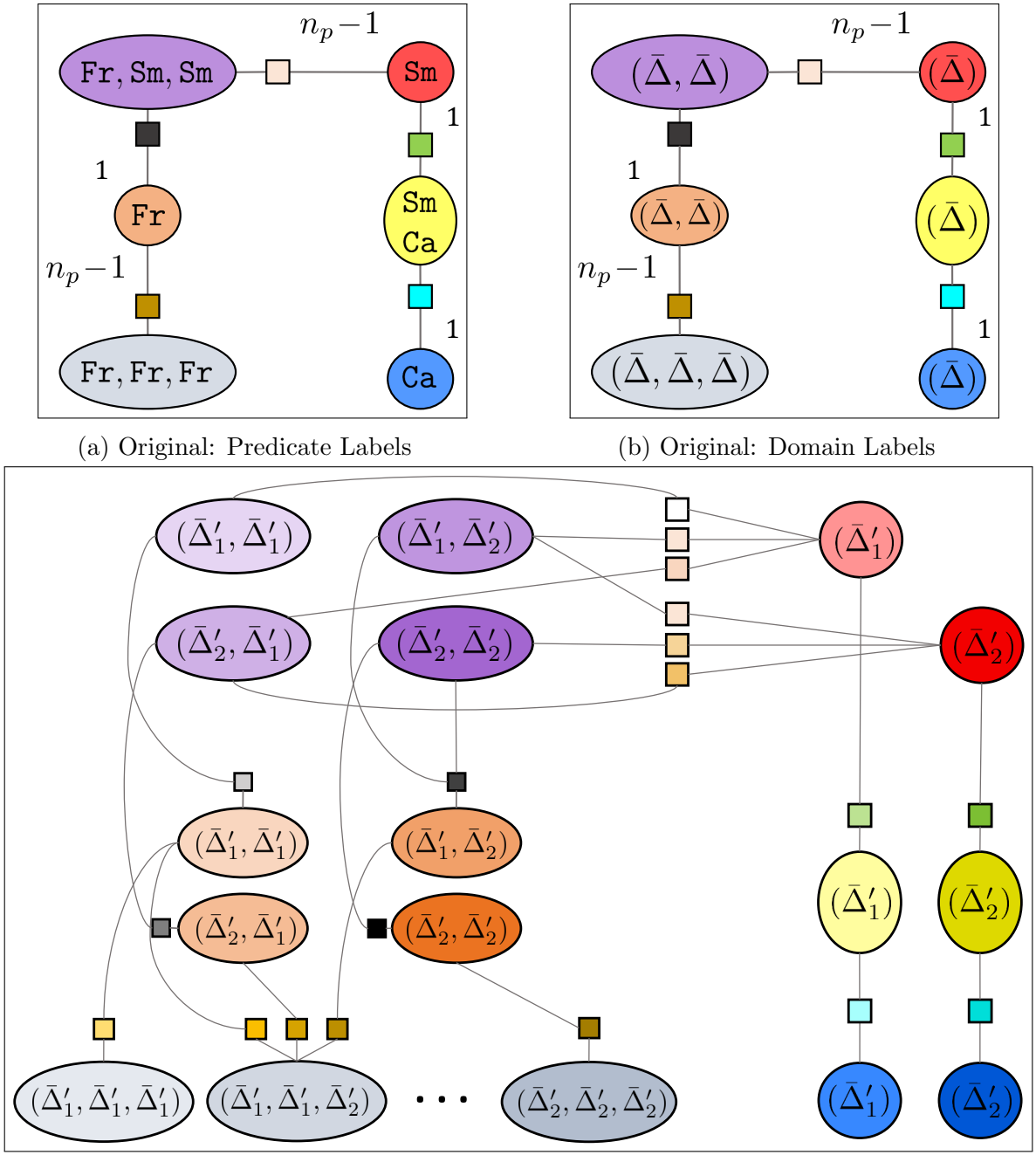
`Step 4`: $t \leftarrow t + 1$

end while

Section 5.3.1, we discuss the structure of the lifted region graph after a given domain split. In Section 5.3.2, we show how to select the split via a structured gradient clustering algorithm, similar to the unstructured gradient clustering algorithm we presented in the last chapter.

5.3.1 Splitting the Graph Structure

In this section, we describe the effect of splitting a lifted logical constant $\bar{\Delta}$ into two lifted logical constants $\bar{\Delta}'_1$ and $\bar{\Delta}'_2$ where the associated ground sets $\{\text{gr}(\bar{\Delta}'_1), \text{gr}(\bar{\Delta}'_2)\}$ partition



(a) Original: Predicate Labels

(b) Original: Domain Labels

(c) Split Domain

Figure 5.1: Splitting the domain $\bar{\Delta}$ (corresponding to the set of all people) of a friend-smoker-MLN into $\bar{\Delta}'_1$ and $\bar{\Delta}'_2$ (corresponding to two subsets of people). (a) The original lifted region graph labeled with predicate symbols. (b) The original lifted region graph labeled with domain symbols. (c) The lifted region graph after the split; the friends-transitivity formula (gray node) splits into eight groups (one for each combination of three copies of $\bar{\Delta}'_1$ and $\bar{\Delta}'_2$) which are abbreviated with dots.

the original set of ground logical constants $\mathrm{gr}(\bar{\Delta})$. We first give an example of how a single FOL-formula splits. In the friend-smoker-MLN, if we split the set of people $\bar{\Delta} = \bar{\Delta}_{\mathrm{ppl}}$, the FOL-formula relating friendship and smoking habits among all pairs of people

$$(\forall l_1 \in \mathrm{gr}(\bar{\Delta}),\ l_2 \in \mathrm{gr}(\bar{\Delta}) \setminus l_1)\ \ x_{\mathtt{Fr}(l_1,l_2)} \Rightarrow (x_{\mathtt{Sm}(l_1)} \Leftrightarrow x_{\mathtt{Sm}(l_2)})$$

is split into FOL-formulas

$$\begin{aligned}
&(\forall l_1 \in \mathrm{gr}(\bar{\Delta}'_1),\ l_2 \in \mathrm{gr}(\bar{\Delta}'_1) \setminus l_1)\ \ x_{\mathtt{Fr}(l_1,l_2)} \Rightarrow (x_{\mathtt{Sm}(l_1)} \Leftrightarrow x_{\mathtt{Sm}(l_2)}) \\
&(\forall l_1 \in \mathrm{gr}(\bar{\Delta}'_1),\ l_2 \in \mathrm{gr}(\bar{\Delta}'_2))\ \ x_{\mathtt{Fr}(l_1,l_2)} \Rightarrow (x_{\mathtt{Sm}(l_1)} \Leftrightarrow x_{\mathtt{Sm}(l_2)}) \\
&(\forall l_1 \in \mathrm{gr}(\bar{\Delta}'_2),\ l_2 \in \mathrm{gr}(\bar{\Delta}'_1))\ \ x_{\mathtt{Fr}(l_1,l_2)} \Rightarrow (x_{\mathtt{Sm}(l_1)} \Leftrightarrow x_{\mathtt{Sm}(l_2)}) \\
&(\forall l_1 \in \mathrm{gr}(\bar{\Delta}'_2),\ l_2 \in \mathrm{gr}(\bar{\Delta}'_2) \setminus l_1)\ \ x_{\mathtt{Fr}(l_1,l_2)} \Rightarrow (x_{\mathtt{Sm}(l_1)} \Leftrightarrow x_{\mathtt{Sm}(l_2)})
\end{aligned}$$

where the first and fourth FOL-formulas relate pairs of people within the same subgroup (for each split subgroup $\bar{\Delta}'_1$ and $\bar{\Delta}'_2$) and the second and third FOL-formulas relate pairs of people between subgroups (for each direction).

In general, a lifted factor $\bar{\theta}_{\bar{\alpha}}$ with $q_{\bar{\alpha}}$ copies of $\bar{\Delta}$ in its lifted LV-scope $\mathrm{sc}_{\boldsymbol{\Delta}}(\bar{\alpha})$ is split into

$$\bar{\alpha}^{\bar{\Delta}'} = \{\bar{\alpha}^{\bar{\Delta}'}_{\boldsymbol{j}} : \boldsymbol{j} \in \{1,2\}^{q_{\bar{\alpha}}}\}$$

where $\bar{\alpha}^{\bar{\Delta}'}_{\boldsymbol{j}}$ substitutes all copies of the lifted domain $\bar{\Delta}$ for the split domains indexed by $\boldsymbol{j} = (j_1, \ldots, j_{q_{\bar{\alpha}}})$, i.e.,

$$\begin{aligned}
\bar{\mathrm{sc}}_{\bar{\boldsymbol{\Delta}}}(\bar{\alpha}) &= (\ldots,\ \bar{\Delta}, \ldots, \bar{\Delta},\ \ldots) \\
\bar{\mathrm{sc}}_{\bar{\boldsymbol{\Delta}}}(\bar{\alpha}^{\bar{\Delta}'}_{\boldsymbol{j}}) &= (\ldots,\ \bar{\Delta}'_{j_1}, \ldots, \bar{\Delta}'_{j_{q_{\bar{\alpha}}}},\ \ldots)
\end{aligned} \tag{5.2}$$

and its factor template is equal to the the original factor template i.e., $\theta_{\bar{\alpha}} = \theta_{\bar{\alpha}^{\bar{\Delta}'}_{\boldsymbol{j}}}$ for all $\boldsymbol{j}$.

5.3.1.1 Splitting the Lifted Region Graph

In `Step 2b`, we split all lifted regions, lifted RVs, and lifted separators that use the lifted logical constant $\bar{\Delta}$ ($\bar{R}^{\bar{\Delta}}$, $\bar{\mathcal{V}}^{\bar{\Delta}}$, and $\bar{S}^{\bar{\Delta}}$, respectively) into new lifted regions, lifted RVs, and lifted separators that use the split lifted logical constants $\bar{\Delta}'_1$ and $\bar{\Delta}'_2$ ($\bar{R}^{\bar{\Delta}'}$, $\bar{\mathcal{V}}^{\bar{\Delta}'}$, and $\bar{S}^{\bar{\Delta}'}$, respectively). These terms are denoted as

$$\begin{aligned}
\bar{R}^{\bar{\Delta}} &= \{\bar{r} \in \bar{R} : \bar{\Delta} \in \bar{\mathrm{sc}}_{\bar{\boldsymbol{\Delta}}}(\bar{r})\} \qquad \text{and} \qquad \bar{R}^{\bar{\Delta}'} = \cup_{\bar{r} \in \bar{R}^{\bar{\Delta}}} \bar{r}^{\bar{\Delta}'}, \\
\bar{\mathcal{V}}^{\bar{\Delta}} &= \{\bar{v} \in \bar{\mathcal{V}} : \bar{\Delta} \in \bar{\mathrm{sc}}_{\bar{\boldsymbol{\Delta}}}(\bar{v})\} \qquad \text{and} \qquad \bar{\mathcal{V}}^{\bar{\Delta}'} = \cup_{\bar{v} \in \bar{\mathcal{V}}^{\bar{\Delta}}} \bar{v}^{\bar{\Delta}'}, \\
\bar{S}^{\bar{\Delta}} &= \{\bar{s} \in \bar{S} : \bar{\Delta} \in \bar{\mathrm{sc}}_{\bar{\boldsymbol{\Delta}}}(\bar{s})\} \qquad \text{and} \qquad \bar{S}^{\bar{\Delta}'} = \cup_{\bar{s} \in \bar{S}^{\bar{\Delta}'}} \bar{s}^{\bar{\Delta}'},
\end{aligned}$$

where each lifted region, lifted RV, and lifted separator splits as follows:

Lifted Region: A lifted region $\bar{r} \in \bar{R}^{\bar{\Delta}}$ associated with $q_{\bar{r}}$ copies of $\bar{\Delta}$ is associated with all possible combinations of splits

$$\bar{r}^{\bar{\Delta}'} = \{\bar{r}'_{\boldsymbol{j}} : \boldsymbol{j} \in \{1,2\}^{q_{\bar{r}}}\}$$

where $\bar{r}'_{\boldsymbol{j}}$ substitutes all copies of the lifted domain $\bar{\Delta}$ for the split domains indexed by $\boldsymbol{j} = (j_1, \ldots, j_{q_{\bar{r}}})$; the ground scopes associated with the lifted region are defined as in Eq. (5.2) above.

Lifted RV: A lifted RV $\bar{v} \in \bar{\mathcal{V}}^{\bar{\Delta}}$ associated with $q_{\bar{v}}$ copies of $\bar{\Delta}$ is associated with all possible combinations of splits

$$\bar{v}^{\bar{\Delta}'} = \{\bar{v}'_{\boldsymbol{j}} : \boldsymbol{j} \in \{1,2\}^{q_{\bar{v}}}\}$$

where $\bar{v}'_{\boldsymbol{j}}$ substitutes all copies of the lifted domain $\bar{\Delta}$ for the split domains indexed by $\boldsymbol{j} = (j_1, \ldots, j_{q_{\bar{v}}})$; again, the ground scopes are defined as in Eq. (5.2).

Lifted Separator A lifted separator $\bar{s} \in \bar{S}^{\bar{\Delta}}$ incident to lifted RV $\bar{v} \in \bar{\mathcal{V}}^{\bar{\Delta}}$ and lifted region

$\bar{r} \in \bar{R}^{\bar{\Delta}}$ splits into a set of lifted separators $\bar{s}^{\bar{\Delta}'}$ where each lifted separator $\bar{s}' \in \bar{s}^{\bar{\Delta}'}$ is incident to a split lifted RV $\bar{v}' \in \bar{v}^{\bar{\Delta}'}$ and a split lifted region $\bar{r}' \in \bar{r}^{\bar{\Delta}'}$. The lifted separator $\bar{s}'$ is associated with all ground separators incident to a ground region and a ground RV from both groups, i.e.,

$$\mathrm{gr}(\bar{s}') = \{s \in \mathrm{gr}(\bar{s}) :\ v \in \mathrm{gr}(\bar{v}'),\ r \in \mathrm{gr}(\bar{r}'), \quad \text{where} \quad N(s) = \{v, r\}\}$$

In Figure 5.1, we illustrate the effect of splitting one domain and all corresponding FOL-formulas in the friend-smoker-MLN. The top two subfigures show the lifted region graph where lifted regions and lifted RVs are labeled in two different ways: in the top left they are labeled with predicate symbols, and in the top right they are labeled with logical domain symbols. The bottom figure shows the effect of splitting $\bar{\Delta}$ into $\bar{\Delta}'_1$ and $\bar{\Delta}'_2$ on the lifted region graph. Each lifted region (or lifted RV) with q copies of $\bar{\Delta}$ splits into 2^q lifted regions (or lifted RVs). Note that the distinct univariate evidence associated with each lifted RV (as in the embedded figure at the end of Section 5.2.1) is not shown on this graph for compactness.

5.3.1.2 Splitting the Lifted Elimination Order

Having described how the structure of the graph is altered by splitting a domain $\bar{\Delta}$, we now discuss how the lifted elimination order will be adapted to account for the split. To do so, we replace each lifted RV $\bar{v} \in \bar{\mathcal{V}}^{\bar{\Delta}}$ with (any arbitrary) ordered sequence of its split lifted RVs $\bar{v}^{\bar{\Delta}'}$ in the lifted elimination order $\bar{\boldsymbol{o}}$.

For example, if the lifted elimination order is $\bar{\boldsymbol{o}} = (\mathtt{Fr}(\bar{\Delta}_{\mathrm{ppl}}, \bar{\Delta}_{\mathrm{ppl}}), \mathtt{Sm}(\bar{\Delta}_{\mathrm{ppl}}), \mathtt{Ca}(\bar{\Delta}_{\mathrm{ppl}}))$, before the split, then, a possible lifted elimination order after the split is

$$\bar{\boldsymbol{o}}' = \Big(\mathtt{Fr}(\bar{\Delta}'_1, \bar{\Delta}'_1), \mathtt{Fr}(\bar{\Delta}'_2, \bar{\Delta}'_2), \mathtt{Fr}(\bar{\Delta}'_2, \bar{\Delta}'_1), \mathtt{Fr}(\bar{\Delta}'_1, \bar{\Delta}'_2), \mathtt{Sm}(\bar{\Delta}'_1), \mathtt{Sm}(\bar{\Delta}'_2),\ \mathtt{Ca}(\bar{\Delta}'_2), \mathtt{Ca}(\bar{\Delta}'_1)\Big).$$

Algorithm 10 Logical domain splitting for MLN Models

Input: A lifted logical constant $\bar{\Delta} \in \bar{\mathbf{\Delta}}$, a set of split lifted separators $\bar{S}^{\bar{\Delta}'}$, a number of iterations M.

Output: A set of domain partitions $\{\text{gr}(\bar{\Delta}'_1),\ \text{gr}(\bar{\Delta}'_2)\}$ that approximately minimizes the gradient-clustering objective

$$\min_{\gamma_{\bar{S}^{\bar{\Delta}'}}} \min_{\{\text{gr}(\bar{\Delta}'_1),\ \text{gr}(\bar{\Delta}'_2)\}} d(\bar{S}^{\bar{\Delta}'},\ \gamma_{\bar{S}^{\bar{\Delta}'}})$$

`Step 0` (Initialize): Randomly initialize the domain partition $\{\text{gr}(\bar{\Delta}'_1),\ \text{gr}(\bar{\Delta}'_2)\}$.

for `iter` $= 1$ **to** M **do**

`Step 1` (Cluster Center): Compute the gradient cluster center factor for each $\bar{s}' \in \bar{S}^{\bar{\Delta}'}$ incident to lifted RV $\bar{v}'$ and lifted region $\bar{r}'$

$$\gamma_{\bar{s}'} = \frac{\sum_{v \in \text{gr}(\bar{v}')} g^{\delta}_{\bar{s}'_{(v,\bar{r}')}}}{|\text{gr}(\bar{v}')|}$$

`Step 2` (Assignment): Assign each logical constant to its nearest cluster center. That is, for each $i \in \{1, 2\}$, set

$$\text{gr}(\bar{\Delta}'_i) \ \leftarrow\ \{l \in \text{gr}(\bar{\Delta}) :\ i^*_l = i\}$$

where the nearest cluster center for logical constant $l \in \text{gr}(\bar{\Delta})$ is

$$i^*_l \ =\ \arg\min_{i \in \{1,2\}}\ \bar{d}_l(\bar{S}^{\bar{\Delta}'_i},\ \gamma_{\bar{S}^{\bar{\Delta}'_i}}).$$

end for

We expect the choice of the lifted elimination order to have a potentially large impact on the bound quality. However, in this work, we use the simple and expedient approach of selecting a random permutation of each split lifted RV. A potential direction for future work is to identify more intelligent ways of ordering the new sets of lifted variables.

5.3.2 Choosing the Domain Split

As in our refinement procedure from the previous chapter (in Section 4.3.1), we elect to judge the quality of a partition by measuring the distance of its associated ground gradients to an over-symmetric projection of those gradients. When the distance is zero, the over-

symmetric and unrestricted gradients are identical and the symmetric restriction incurs no error. Similarly, when the distance is small, we expect the inference error to be small.

The distance for a single split-lifted separator $\bar{s}' \in \bar{S}^{\bar{\Delta}'}$, incident to lifted RV $\bar{v}'$ and lifted region $\bar{r}'$, is

$$d(\bar{s}',\ \gamma_{\bar{s}'}) = \sum_{s \in \text{gr}(\bar{s}')} ||g_s^{\delta} - \gamma_{\bar{s}'}||_2^2 \ = \ \sum_{v \in \text{gr}(\bar{v}')} \ \sum_{s \in N(v) \cap \text{gr}(\bar{r}')} ||g_s^{\delta} - \gamma_{\bar{s}'}||_2^2$$
$$= \ M_{\bar{s}'_{(\bar{v}',\bar{r}')}} \cdot \sum_{v \in \text{gr}(\bar{v}')} ||g^{\delta}_{\bar{s}'_{(v,\bar{r}')}} - \gamma_{\bar{s}'}||_2^2 \ =: \ \bar{d}(\bar{s}',\ \gamma_{\bar{s}'}) \qquad (5.3)$$

where $M_{\bar{s}'_{(v,\bar{r}')}} = M_{\bar{s}'_{(\bar{v}',\bar{r}')}}$ is the number of separator neighbors of each ground RV $v \in \text{gr}(\bar{v}')$, and the calculation in the second line exploits symmetry in the computation in the first line.

A notable change from the last chapter is that in this chapter, the separator sets are not split directly, but instead are split *indirectly* via the domain split operation creating new domains $\{\text{gr}(\bar{\Delta}'_1),\ \text{gr}(\bar{\Delta}'_2)\}$, as detailed in Section 5.3.1. In this case, the clustering is obtained by the following optimization

$$\min_{\gamma_{\bar{S}^{\bar{\Delta}'}}} \ \min_{\{\text{gr}(\bar{\Delta}'_1),\ \text{gr}(\bar{\Delta}'_2)\}} \ d(\bar{S}^{\bar{\Delta}'},\ \gamma_{\bar{S}^{\bar{\Delta}'}}) \qquad (5.4)$$

where $\gamma_{\bar{S}^{\bar{\Delta}'}} = \{\gamma_{\bar{s}'} : \bar{s}' \in \bar{S}^{\bar{\Delta}'}\}$ is the set of all cluster center factors, and

$$d(\bar{S}^{\bar{\Delta}'},\ \gamma_{\bar{S}^{\bar{\Delta}'}}) \ = \ \sum_{\bar{s}' \in \bar{S}^{\bar{\Delta}'}} d(\bar{s}',\ \gamma_{\bar{s}'})$$

is the sum of distances associated with lifted separators associated with the split domains $\bar{\Delta}'_1$ and $\bar{\Delta}'_2$.

5.3.2.1 Clustering Algorithm

The clustering optimization defined by Eq. (5.4) is carried out iteratively, by alternating between minimizing over the cluster centers $\gamma_{\bar{S}^{\bar{\Delta}'}}$ and the assignment of ground constants to

clusters $\{\text{gr}(\bar{\Delta}'_1),\ \text{gr}(\bar{\Delta}'_2)\}$. This alternating minimization procedure is codified in Algorithm 10, and resembles the standard K-Means clustering algorithm [MacQueen et al., 1967] for $K = 2$ cluster centers. This algorithm assumes that each lifted factor has exactly one copy of the lifted logical domain symbol $\bar{\Delta}$. This allows each ground constant to be assigned to a domain cluster center independently (as we will see shortly). The general case, where a lifted factor uses more than one copy of $\bar{\Delta}$ is more difficult as it couples the constants in each cluster. Our strategy in the general case, as we discuss in the next subsection, is to solve a relaxed clustering problem where each lifted factor uses a single copy of $\bar{\Delta}$ (and is thus amenable to the clustering algorithm presented in this section).

We now describe each step of the algorithm. In `Step 1`, for a fixed clustering $\{\text{gr}(\bar{\Delta}'_1),\ \text{gr}(\bar{\Delta}'_2)\}$, each element of the cluster center factors $\gamma_{\bar{S}^{\bar{\Delta}'}}$ is optimized by setting it to the mean value of the associated ground gradients.

In `Step 2`, the assignment step, for a fixed set of cluster center factors $\gamma_{\bar{S}^{\bar{\Delta}'_1}}$ and $\gamma_{\bar{S}^{\bar{\Delta}'_2}}$, the clustering $\{\text{gr}(\bar{\Delta}'_1),\ \text{gr}(\bar{\Delta}'_2)\}$ is optimized by associating each logical constant $l \in \text{gr}(\bar{\Delta})$ with the nearest set of cluster center factors. Here, we defined $\bar{S}^{\bar{\Delta}'_i} = \{\bar{s}' \in \bar{S}^{\bar{\Delta}'_i} :\ \bar{\Delta}'_i \in \bar{\text{sc}}_{\bar{\mathbf{\Delta}}}(\bar{s}')\}$ as the set of lifted separators and $\gamma_{\bar{S}^{\bar{\Delta}'_i}} = \{\gamma_{\bar{s}'} : \bar{s}' \in \bar{S}^{\bar{\Delta}'_i}\}$ as the set of cluster center factors associated with the ith domain split for $i \in \{1, 2\}$. This distance of logical constant l to its associated lifted separators is

$$\bar{d}_l(\bar{S}^{\bar{\Delta}'_i},\ \gamma_{\bar{S}^{\bar{\Delta}'_i}}) \ = \ \sum_{\bar{s}' \in \bar{S}^{\bar{\Delta}'_i}} \bar{d}_l(\bar{s}',\ \gamma_{\bar{s}'}) \tag{5.5}$$

where

$$\begin{aligned} d_l(\bar{s}',\ \gamma_{\bar{s}'}) \ &= \ \sum_{s \in N^{\bar{s}'}_{LV}(l)} ||g^{\delta}_s - \gamma_{\bar{s}'}||^2_2 \ = \ \sum_{v \in N^{\bar{v}}_{LV}(l)} \ \sum_{s \in N(v) \cap \text{gr}(\bar{r}')} ||g^{\delta}_s - \gamma_{\bar{s}'}||^2_2 \\ &= \ M_{\bar{s}'_{(\bar{v}',\bar{r}')}} \cdot \sum_{v \in N^{\bar{v}}_{LV}(l)} ||g^{\delta}_{\bar{s}'_{(v,\bar{r}')}} - \gamma_{\bar{s}'}||^2_2 \ =: \ \bar{d}_l(\bar{s}',\ \gamma_{\bar{s}'}) \end{aligned}$$

is the distance associated with ground separators $N^{\bar{s}'}_{LV}(l) = \{s \in \text{gr}(\bar{s}') : l \in \text{sc}_{\boldsymbol{\Delta}}(s)\}$ that are associated with lifted separator $\bar{s}' \in \bar{S}^{\bar{\Delta}'_i}$ and use logical constant l. In these equations, $\bar{s}'$ is incident to lifted RV $\bar{v}'$ and lifted region $\bar{r}'$. Furthermore, the computation in the second line exploits symmetry in the computation in the first line where we defined $N^{\bar{v}}_{LV}(l) = \{v \in \text{gr}(\bar{v}) : l \in \text{sc}_{\boldsymbol{\Delta}}(v)\}$ as the set of ground RVs associated with lifted RV $\bar{v}$ that use logical constant l; note that $N^{\bar{v}}_{LV}(l)$ is used to index symmetric gradients.

Our algorithm's alternating computation of the cluster center with the assignment of each logical constant to a cluster center has (nearly) the same form as the standard K-means algorithm [MacQueen et al., 1967] with $K = 2$ where each logical constant plays the role of a data point and the gradient associated with a logical constant's attribute or relational predicate plays the role of a feature. We illustrate this similarity via an example.

Example Consider a MLN with predicates $\texttt{Sm}(\bar{\Delta}_{\text{ppl}})$, $\texttt{Ca}(\bar{\Delta}_{\text{ppl}})$, and $\texttt{Buy}(\bar{\Delta}_{\text{ppl}}, \bar{\Delta}_{\text{items}})$ representing smoking attributes, cancer attributes, and the relations that a person buys an item, respectively. This example is like the friend-smoker-MLN with an additional predicate `Buy` introduced to illustrate a relational predicate with a single copy of the split domain $\bar{\Delta}_{\text{ppl}}$, while removing the friend predicate, which would comprise self-relational term that we defer handling until the next subsection. For the example, let the ground sets of people $\text{gr}(\bar{\Delta}_{\text{ppl}}) = \{A, B, C, D\}$ and items $\text{gr}(\bar{\Delta}_{\text{items}}) = \{1, 2, 3\}$.

Our goal is to select a split of the domain of people into two groups. As in standard K-Means, we organize the data (here, the gradients) as a matrix g_{All} where columns are data points and rows are features. The columns (from left to right) represent people A, B, C, D, and the rows (from top to bottom) represent the `Sm` attribute, the `Ca` attribute, and each `Buy` relationship with the three items. The data matrix is shown in Figure 5.2a.

The data matrix g_{All} is partitioned by solid vertical lines that represent one possible clustering of the people into groups $\{A, B\}$ and $\{C, D\}$, and by dashed horizontal lines that

$$g_{All} = \left[\begin{array}{cc|cc} g_{\texttt{Sm}(A)} & g_{\texttt{Sm}(B)} & g_{\texttt{Sm}(C)} & g_{\texttt{Sm}(D)} \\ \hline g_{\texttt{Ca}(A)} & g_{\texttt{Ca}(B)} & g_{\texttt{Ca}(C)} & g_{\texttt{Ca}(D)} \\ \hline g_{\texttt{Buy}(A,1)} & g_{\texttt{Buy}(B,1)} & g_{\texttt{Buy}(C,1)} & g_{\texttt{Buy}(D,1)} \\ g_{\texttt{Buy}(A,2)} & g_{\texttt{Buy}(B,2)} & g_{\texttt{Buy}(C,2)} & g_{\texttt{Buy}(D,2)} \\ g_{\texttt{Buy}(A,3)} & g_{\texttt{Buy}(B,3)} & g_{\texttt{Buy}(C,3)} & g_{\texttt{Buy}(D,3)} \end{array}\right]$$

(a) Gradient data matrix

$$\gamma_{All} = \left[\begin{array}{cc|cc} \gamma_{\texttt{Sm1}} & \gamma_{\texttt{Sm1}} & \gamma_{\texttt{Sm2}} & \gamma_{\texttt{Sm2}} \\ \hline \gamma_{\texttt{Ca1}} & \gamma_{\texttt{Ca1}} & \gamma_{\texttt{Ca2}} & \gamma_{\texttt{Ca2}} \\ \hline \gamma_{\texttt{Buy1}} & \gamma_{\texttt{Buy1}} & \gamma_{\texttt{Buy2}} & \gamma_{\texttt{Buy2}} \\ \gamma_{\texttt{Buy1}} & \gamma_{\texttt{Buy1}} & \gamma_{\texttt{Buy2}} & \gamma_{\texttt{Buy2}} \\ \gamma_{\texttt{Buy1}} & \gamma_{\texttt{Buy1}} & \gamma_{\texttt{Buy2}} & \gamma_{\texttt{Buy2}} \end{array}\right]$$

(b) Cluster center matrix

Figure 5.2: MLN-gradient clustering represented as a standard (e.g., K-Means) clustering problem. (a) A data matrix of ground gradients g_{All} with data points (columns) and features (rows). The columns represent people A, B, C, D (from left to right), and the rows represent the `Sm` attribute, the `Ca` attribute, and each `Buy` relationship with three items (from top to bottom). The solid vertical lines represent one possible clustering of the people into groups $\{A, B\}$ and $\{C, D\}$ and the dashed horizontal lines separate predicate groups. (b) The set of cluster center factors γ_{All} associated with each gradient in the data matrix.

separate features associated with different predicates. (Note that although we cluster adjacent columns in our example, other non-adjacent clusterings are possible, e.g., into $\{A, C\}$ and $\{B, D\}$.)

This clustering problem is associated with cluster center factors $\{\gamma_{\texttt{Sm}i}, \gamma_{\texttt{Ca}i}, \gamma_{\texttt{Buy}i}\}$ for each cluster group $i \in \{1, 2\}$. Each ground term in the data matrix g_{All} is associated with a cluster center factor collected in γ_{All} (shown in Figure 5.2b). Note that each block of elements (demarcated by the vertical lines grouping people and horizontal lines grouping predicates) has the same cluster center factor. In particular, the block of relational gradients associated with the `Buy` feature has $\gamma_{\texttt{Buy1}}$ replicated three times (for all items) rather than having a distinct value for each item.

As is evident from this organization, this problem has the same form as a standard clustering problem with four data points (columns) and five features (rows), but, the value of the last three dimensions of the cluster center must be identical (since they are associated with the same relational predicate).

$$\left[\begin{array}{cc|cc} \times & g_{\mathtt{Fr}(A,B)} & g_{\mathtt{Fr}(A,C)} & g_{\mathtt{Fr}(A,D)} \\ g_{\mathtt{Fr}(B,A)} & \times & g_{\mathtt{Fr}(B,C)} & g_{\mathtt{Fr}(B,D)} \\ \hline g_{\mathtt{Fr}(C,A)} & g_{\mathtt{Fr}(C,B)} & \times & g_{\mathtt{Fr}(C,D)} \\ g_{\mathtt{Fr}(D,A)} & g_{\mathtt{Fr}(D,B)} & g_{\mathtt{Fr}(D,C)} & \times \end{array}\right]$$

(a) Unrelaxed clustering.

$$\left[\begin{array}{cccc} \times & g_{\mathtt{Fr}(A,B)} & g_{\mathtt{Fr}(A,C)} & g_{\mathtt{Fr}(A,D)} \\ g_{\mathtt{Fr}(B,A)} & \times & g_{\mathtt{Fr}(B,C)} & g_{\mathtt{Fr}(B,D)} \\ \hline g_{\mathtt{Fr}(C,A)} & g_{\mathtt{Fr}(C,B)} & \times & g_{\mathtt{Fr}(C,D)} \\ g_{\mathtt{Fr}(D,A)} & g_{\mathtt{Fr}(D,B)} & g_{\mathtt{Fr}(D,C)} & \times \end{array}\right]$$

(b) Relaxaed clustering: Split row but not column.

$$\left[\begin{array}{cc|cc} \times & g_{\mathtt{Fr}(A,B)} & g_{\mathtt{Fr}(A,C)} & g_{\mathtt{Fr}(A,D)} \\ g_{\mathtt{Fr}(B,A)} & \times & g_{\mathtt{Fr}(B,C)} & g_{\mathtt{Fr}(B,D)} \\ g_{\mathtt{Fr}(C,A)} & g_{\mathtt{Fr}(C,B)} & \times & g_{\mathtt{Fr}(C,D)} \\ g_{\mathtt{Fr}(D,A)} & g_{\mathtt{Fr}(D,B)} & g_{\mathtt{Fr}(D,C)} & \times \end{array}\right]$$

(c) Relaxaed clustering: Split column but not row.

Figure 5.3: Grouping of ground gradients associated with a FOL-formula $\mathtt{Fr}(\bar{\Delta}, \bar{\Delta})$ relating people in the same group (each element of each table corresponds to a pair of people and the cross "$\times$" means no entry, i.e., no person has a friendship relation with himself). In all cases, the solid lines demarcate an RV clustering associated with the clustering of people into groups $\{A, B\}$ and $\{C, D\}$. The subfigures show (a) Unrelaxed clustering on both dimensions into four groups; (b) Relaxed clustering that splits only the rows (a row represents out-links of a person's friendship); and (c) Relaxed clustering that splits only the columns (a column represents out-links of a person's friendship).

5.3.3 Self-Relations

When a lifted RV uses one copy of the lifted logical constant $\bar{\Delta}$, the assignment step can be performed independently for each ground logical constant (as we saw above). That is, we can compute the distance incurred by assigning logical constant $A \in \mathrm{gr}(\bar{\Delta})$ to the split sets $\mathrm{gr}(\bar{\Delta}'_1)$ and $\mathrm{gr}(\bar{\Delta}'_2)$ and choose the best one, as we saw in the last section.

However, when a lifted RV uses two copies of the lifted logical constant $\bar{\Delta}$ (representing a relationship between objects in the same domain), the assignment step can *not* be performed independently for each ground logical constant. For example, consider a friendship relation $\mathtt{Fr}(\bar{\Delta}, \bar{\Delta})$ between all pairs of people of the group $\bar{\Delta} = \bar{\Delta}_{\mathrm{ppl}}$. In this case, the split of $\bar{\Delta}$ produces four lifted RVs (as shown in Figure 5.1). If the logical constant $A \in \mathrm{gr}(\bar{\Delta})$ is assigned to the first domain cluster $\bar{\Delta}'_1$, then for each $l \in \mathrm{gr}(\bar{\Delta})$, $\mathtt{Fr}(A, l)$ is assigned to one

of two clusters

$$\texttt{Fr}(A, l) \in \text{gr}(\texttt{Fr}(\bar{\Delta}_1', \bar{\Delta}_1')) \quad \text{if} \quad l \in \text{gr}(\bar{\Delta}_1')$$
$$\texttt{Fr}(A, l) \in \text{gr}(\texttt{Fr}(\bar{\Delta}_1', \bar{\Delta}_2')) \quad \text{if} \quad l \in \text{gr}(\bar{\Delta}_2')$$

Consequently, we cannot perform the assignment of logical constant A without knowing which domain cluster the other logical constants are assigned to.

To obtain a problem admitting an independent assignment, we relax the gradient split by not splitting the dimension associated with the second copy of $\bar{\Delta}$. That is, we force $\texttt{Fr}(\bar{\Delta}, \bar{\Delta})$ to split only on its first copy of $\bar{\Delta}$ into $\texttt{Fr}(\bar{\Delta}_1', \bar{\Delta})$ and $\texttt{Fr}(\bar{\Delta}_2', \bar{\Delta})$. This split has the same form as a relational predicate with only a single copy of the splitting domain, e.g., the `Buy` predicate. (Note that we only relax the clustering problem; after the clustering is selected, the lifted region graph is split along both copies of $\bar{\Delta}$, as detailed in Section 5.3.1.)

We depict the effect of the domain clustering relaxation in Figure 5.3. Each subfigure illustrates the ground gradients for all pairs of people laid out in a matrix; the solid lines demarcate the partition of gradients associated with the partition of the domain of four people $\{A, B, C, D\}$ into two sets $\{A, B\}$ and $\{C, D\}$. The first subfigure depicts the unrelaxed clustering that partitions each dimension of the table into these two groups (the top left block represents links between people in group one, the top right block represents links from people in group one to group two, the bottom left block represents links from people in group two to group one, and the bottom right block represents links between people in group one). The second subfigure depicts the relaxed clustering discussed above where only the first dimension splits (hence only rows of the data matrix are partitioned). The third subfigure depicts a clustering relaxation obtained by instead splitting only the second dimension (hence only columns of the data matrix are partitioned).

$$g_{All} = \left[\begin{array}{cc|cc} g_{\texttt{Sm}(A)} & g_{\texttt{Sm}(B)} & g_{\texttt{Sm}(C)} & g_{\texttt{Sm}(D)} \\ \hline g_{\texttt{Ca}(A)} & g_{\texttt{Ca}(B)} & g_{\texttt{Ca}(C)} & g_{\texttt{Ca}(D)} \\ \hline g_{\texttt{Buy}(A,1)} & g_{\texttt{Buy}(B,1)} & g_{\texttt{Buy}(C,1)} & g_{\texttt{Buy}(D,1)} \\ g_{\texttt{Buy}(A,2)} & g_{\texttt{Buy}(B,2)} & g_{\texttt{Buy}(C,2)} & g_{\texttt{Buy}(D,2)} \\ g_{\texttt{Buy}(A,3)} & g_{\texttt{Buy}(B,3)} & g_{\texttt{Buy}(C,3)} & g_{\texttt{Buy}(D,3)} \\ \hline \times & g_{\texttt{Fr}(A,B)} & g_{\texttt{Fr}(A,C)} & g_{\texttt{Fr}(A,D)} \\ g_{\texttt{Fr}(B,A)} & \times & g_{\texttt{Fr}(B,C)} & g_{\texttt{Fr}(B,D)} \\ g_{\texttt{Fr}(C,A)} & g_{\texttt{Fr}(C,B)} & \times & g_{\texttt{Fr}(C,D)} \\ g_{\texttt{Fr}(D,A)} & g_{\texttt{Fr}(D,B)} & g_{\texttt{Fr}(D,C)} & \times \end{array}\right] \qquad \gamma_{All} = \left[\begin{array}{cc|cc} \gamma_{\texttt{Sm1}} & \gamma_{\texttt{Sm1}} & \gamma_{\texttt{Sm2}} & \gamma_{\texttt{Sm2}} \\ \hline \gamma_{\texttt{Ca1}} & \gamma_{\texttt{Ca1}} & \gamma_{\texttt{Ca2}} & \gamma_{\texttt{Ca2}} \\ \hline \gamma_{\texttt{Buy1}} & \gamma_{\texttt{Buy1}} & \gamma_{\texttt{Buy2}} & \gamma_{\texttt{Buy2}} \\ \gamma_{\texttt{Buy1}} & \gamma_{\texttt{Buy1}} & \gamma_{\texttt{Buy2}} & \gamma_{\texttt{Buy2}} \\ \gamma_{\texttt{Buy1}} & \gamma_{\texttt{Buy1}} & \gamma_{\texttt{Buy2}} & \gamma_{\texttt{Buy2}} \\ \hline \times & \gamma_{\texttt{Fr1}} & \gamma_{\texttt{Fr2}} & \gamma_{\texttt{Fr2}} \\ \gamma_{\texttt{Fr1}} & \times & \gamma_{\texttt{Fr2}} & \gamma_{\texttt{Fr2}} \\ \gamma_{\texttt{Fr1}} & \gamma_{\texttt{Fr1}} & \times & \gamma_{\texttt{Fr2}} \\ \gamma_{\texttt{Fr1}} & \gamma_{\texttt{Fr1}} & \gamma_{\texttt{Fr2}} & \times \end{array}\right]$$

(a) Gradient data matrix (b) Cluster center matrix

Figure 5.4: Concatenation of data matrix with smoking, cancer, and buy predicates from Figure 5.2 with data matrix of friendship predicates from Figure 5.3. Vertical line demarcates a clustering into $\{A, B\}$ an $\{C, D\}$ and we use the relaxed partition of self-relational friendship predicate that only partitions columns.

Putting it Together We can now depict a clustering for a general problem with multiple predicates and self-relational lifted RVs. We augment the example in the last section (with smoking, cancer, and buy predicates) with the relaxed clustering on the friendship predicate discussed in this section. We concatenate these two data matrices as shown in Figure 5.4. Thus, our clustering algorithm, Algorithm 10, acts like K-means (with $K = 2$) on this data matrix where the cluster centers associated with each predicate (in between the dashed horizontal lines) are restricted to be equal.

Finally, we note that previous work that pre-clusters the MLN domains [Venugopal and Gogate, 2014a] encounters a similar with self-relational predicates in MLN models and also copes with the problem by performing clustering on a relaxed problem.

5.4 Lifted Join

In ground inference problems, reasoning over relaxations formed from the original model factors often produces inaccurate inference estimates. These estimates can often be improved

by joining a group of model factors to produce a large-scope factor, allowing the approximation to reason about the consistency of larger subsets of variables at a time. Analogously, lifted inference relaxations that use regions defined by the model factors (as in our *Lifted GDD Algorithm* in Chapter 4) can be similarly limited and produce inaccurate inference estimates. To improve the approximation quality, it is desirable to join groups of lifted factors to produce a large-scope lifted factor.

More specifically, we want to join two lifted factors $\bar{\theta}_{\bar{\alpha}}$ and $\bar{\theta}_{\bar{\beta}}$ to produce a third lifted factor $\bar{\theta}_{\bar{j}}$ that is equivalent, i.e.,

$$\sum_{j \in \mathrm{gr}(\bar{j})} \theta_{\bar{j}}(x_{\bar{j}}) \;=\; \sum_{\alpha \in \mathrm{gr}(\bar{\alpha})} \theta_{\bar{\alpha}}(x_{\alpha}) \;+\; \sum_{\beta \in \mathrm{gr}(\bar{\beta})} \theta_{\bar{\beta}}(x_{\beta}). \tag{5.6}$$

That is, we would like for a single lifted factor, with a single template, to represent the sum. Unfortunately, in general this equivalence is not possible; instead, the sum of our two lifted factors may shatter, producing an uncontrolled number of new lifted factors.

However, in the special case that the model consists of FOL-formulas, it is often possible to perform a lifted join without shattering. In this case, the resultant factor is a FOL-formula using a set of logical domains used by each factor. We illustrate the joining of FOL-formulas with two examples.

In the simplest case, each formula is an attribute over the same logical domain, and the lifted join corresponds to a set of identical non-overlapping joins. For example, consider a problem with attributes Smokes (`Sm`), Has Cancer (`Ca`), and Regularly Exercises (Ex), and model factors that consist of pairwise relations among all three pairs (e.g., a person who smokes probably has cancer, a person who regularly exercises is unlikely to smoke, a person who regularly exercises is unlikely to get cancer). The lifted join corresponds to a set of

identical ground joins which closes the loop at each ground group, so that

$$\theta_{\bar{j}}(x_{\texttt{Sm}(l_1)}, x_{\texttt{Ca}(l_1)}, x_{\text{Ex}(l_1)}) \;=\; \theta_{\bar{\alpha}}(x_{\texttt{Sm}(l_1)}, x_{\texttt{Ca}(l_1)}) \;+\; \theta_{\bar{\beta}}(x_{\texttt{Ca}(l_1)}, x_{\text{Ex}(l_1)}) \;+\; \theta_{\bar{\gamma}}(x_{\texttt{Sm}(l_1)}, x_{\text{Ex}(l_1)})$$

for each logical constant $l_1 \in \text{gr}(\bar{\Delta}_{\text{ppl}})$. In other words, for each person, we join three factors, and these groups of three factors are identical for each person.

More generally, to join two formulas that contain multiple logical variables, we must match the logical variables in one formula to logical variables in the other formula. Consider joining two FOL-formulas relating friendship and smoking habits over disjoint groups of people with ground factor scopes

$$\begin{aligned}
\text{gr}(\bar{\alpha}) &= \{\{\texttt{Fr}(l_1, l_2), \texttt{Sm}(l_1), \texttt{Sm}(l_2)\} : l_1 \in \text{gr}(\bar{\Delta}_{\text{ppl};1}),\; l_2 \in \text{gr}(\bar{\Delta}_{\text{ppl};2})\} \\
\text{gr}(\bar{\beta}) &= \{\{\texttt{Fr}(l_2, l_3), \texttt{Sm}(l_2), \texttt{Sm}(l_3)\} : l_2 \in \text{gr}(\bar{\Delta}_{\text{ppl};2}),\; l_3 \in \text{gr}(\bar{\Delta}_{\text{ppl};3})\}
\end{aligned}$$

Joining these two formulas produces a formula over all three groups of people:

$$\begin{aligned}
\text{gr}(\bar{j}) = \{\{\texttt{Fr}(l_1, l_2), \texttt{Fr}(l_2, l_3),\, \texttt{Sm}(l_1), \texttt{Sm}(l_2), \texttt{Sm}(l_3)\} :& \\
l_1 \in \text{gr}(\bar{\Delta}_{\text{ppl};1}),\; l_2 \in \text{gr}(\bar{\Delta}_{\text{ppl};2}),\; l_3 \in \text{gr}(\bar{\Delta}_{\text{ppl};3})\}&
\end{aligned}$$

and a template factor

$$\begin{aligned}
\theta_{\bar{j}}(x_{\texttt{Fr}(l_1,l_2)}, x_{\texttt{Sm}(l_1)}, x_{\texttt{Sm}(l_2)}, x_{\texttt{Fr}(l_2,l_3)}, x_{\texttt{Sm}(l_3)}) \;=\; & \theta_{\bar{\alpha}}(x_{\texttt{Fr}(l_1,l_2)}, x_{\texttt{Sm}(l_1)}, x_{\texttt{Sm}(l_2)}) \cdot \frac{1}{|\text{gr}(\bar{\Delta}_{\text{ppl};3})|} \\
+\; & \theta_{\bar{\beta}}(x_{\texttt{Fr}(l_2,l_3)}, x_{\texttt{Sm}(l_2)}, x_{\texttt{Sm}(l_3)}) \cdot \frac{1}{|\text{gr}(\bar{\Delta}_{\text{ppl};1})|}
\end{aligned}$$

at each ground factor $j \in \text{gr}(\bar{j})$. That is, each of the operand factors is scaled by the number of people in the group of people it does not use. In this example, each pair of people (l_2, l_3) from groups two and three appears in $|\text{gr}(\bar{\Delta}_{\text{ppl};1})|$ three-tuples (l_1, l_2, l_3), each corresponding

to some person l_1 in group one. Thus, the total collection of $\bar{\theta}_{\bar{\beta}}$'s ground factors in $\text{gr}(\bar{j})$ will be equivalent to the original total collection in the grounding $\text{gr}(\bar{\beta})$.

Self-Relations A more difficult case arises when certain logical variables (LVs) in a formula range over the same domain, i.e., two people from the same group. Unlike the case above, there is no longer a unique way to match logical variables between formulas. There are two possible ways to resolve this. First, we can select a single matching and form a join; second, we can select all possible matchings and form all possible joins. In the first approach, we are faced with the difficulty of selecting one of the matchings – although any one is technically valid it is not clear how this affects the bound. The second approach seems appealing since it does not require such a choice; however, it is more complicated and can also result in excessively large-scope factors.

In this chapter, we only consider joins of formulas with one copy of any LV ranging over any set of objects (as discussed in the last section); we leave the development of rules that allow joining of formulas where multiple LVs range over a single set of objects to future work. Note that although many MLNs contain formulas with LVs over the same domain, our splitting operation will typically create a number of formulas with LVs that all range over different domains. For example, in Figure 5.1, the original FOL-formula with scope $\{\texttt{Fr}(\bar{\Delta}, \bar{\Delta}), \texttt{Sm}(\bar{\Delta}), \texttt{Sm}(\bar{\Delta})\}$ contains two LVs over the set of all people $\bar{\Delta} = \bar{\Delta}_{\text{ppl}}$ and thus cannot be joined by our rule. When $\bar{\Delta}$ is split into $\bar{\Delta}'_1$ and $\bar{\Delta}'_2$, however, this formula is split into multiple formulas, a number of which use a single LV over each group of people, e.g., formulas defined on $(\bar{\Delta}'_1, \bar{\Delta}'_2)$ and $(\bar{\Delta}'_2, \bar{\Delta}'_1)$. Thus, the limitation on joins with matching variables becomes less restrictive as the domains are successively refined through splitting.

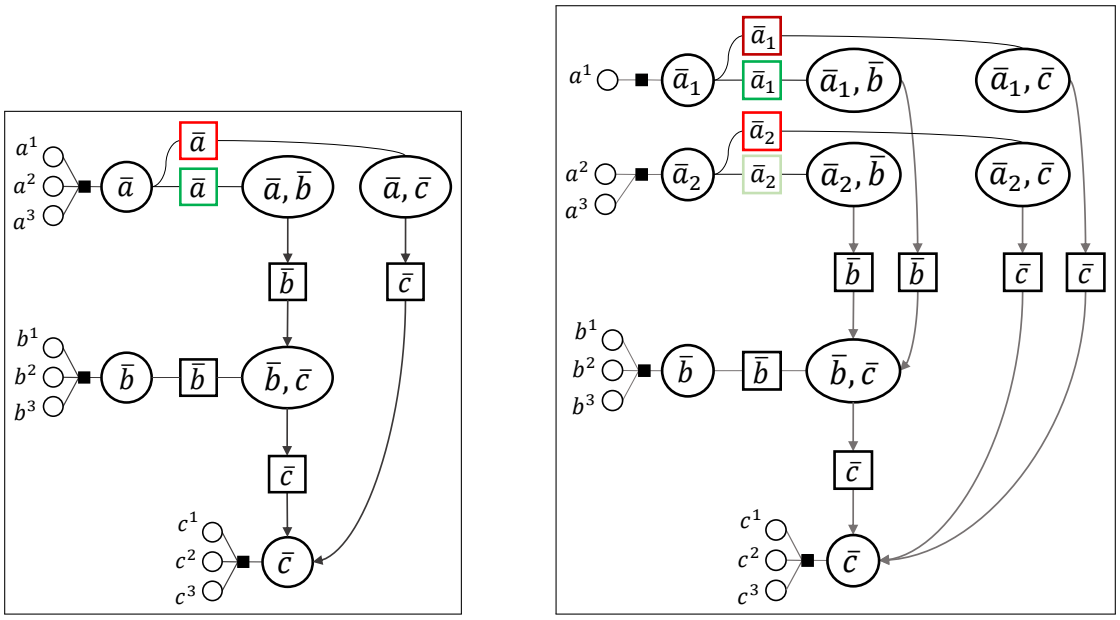
5.4.1 Impact of Lifted Join on Later Splits

The lifted join operation presented in the last section produces a single lifted factor with a predictable and controlled cost (by controlling the number of RVs in the template). However, the compactness of the resultant lifted factor can often conceal a loss of flexibility in controlling the cost incurred by later splitting modifications.

The problem arises when the lifted join operation implicitly produces a much larger number of ground terms. In our second example in the last section we joined two lifted factors each over two groups of people (hence associated with $O(n^2)$ ground factors) to produce a lifted factor that is associated with all $O(n^3)$ ground factors associated with three people, one from each group, where each group has n people. This increase in ground representation size does not have an immediate impact on lifted inference cost. However, as we continue, splitting will eventually reach the ground model, and each of these $O(n^3)$ factors will be represented explicitly.

On the other hand, suppose that instead of joining the two lifted factors, we split them repeatedly until we reach a set of $O(n^2)$ ground factors, and we seek to join these ground factors. There are a total of $O(n^3)$ possible triples (which use one logical constant from each group) that could be added to tighten the relaxation. One possibility is to add all such triples, which would produce the relaxation above. A more flexible approach would select only the subset of the $O(n^3)$ terms that improve the approximation appreciably. If only a small subset is selected, then we would produce a high quality bound at a cost much smaller than $O(n^3)$.

Clearly, the relative quality of each approximation depends on the problem. However, it is also clear that we do not want to join "too early" as it causes each subsequent splits to produce a larger number of lifted regions, incurring a correspondingly higher cost. We leave the question of selecting between split and join operations to future work. In our



(a) Initial LWMB graph

(b) LWMB graph after split

Figure 5.5: A LWMB graph and a split of the graph. (a) A LWMB graph on three lifted RVs with lifted elimination order $\bar{\boldsymbol{o}} = (\bar{a}, \bar{b}, \bar{c})$. Each lifted RV is associated with three ground RVs, $\text{gr}(\bar{a}) = \{a^1, a^2, a^3\}$, $\text{gr}(\bar{b}) = \{b^1, b^2, b^3\}$, $\text{gr}(\bar{c}) = \{c^1, c^2, c^3\}$; each ground RV is associated with asymmetric evidence connected via a solid square node to the region associated with its lifted RV. (b) Splitting the lifted RV $\bar{a}$ into $\bar{a}_1$ and $\bar{a}_2$, associated with ground RVs $\text{gr}(\bar{a}_1) = \{a^1\}$ and $\text{gr}(\bar{a}_2) = \{a^2, a^3\}$. The split is selected using gradients on the two horizontal separators labeled $\bar{a}$ and colored red and green in panel (a).

experiments, we take the simple expedience of performing joins at predefined times. It is also worth noting that this issue, of creating lifted factors with an excessively large number of ground factors, could be avoided if we had more precise control over the approximation. More specifically, if we could perform splitting operations on individual lifted factors, rather than globally on all lifted factors using $\bar{\Delta}$ as we currently do, then we could control the cost by selecting only a promising subset of lifted factors to split.

5.5 Lifted Weighted Mini-Bucket (LWMB)

In the previous sections, we described an anytime lifted GDD algorithm for MLN models that interleaves inference with splits (to represent asymmetric evidence more accurately) and

joins (which produce formulas over larger groups of objects). Our initial implementation utilized this exact framework. However, in our initial experiments, the inference algorithm was slow to converge. To remedy this problem, we developed a *lifted weighted mini-bucket* (LWMB) algorithm which represents the same class of bounds, but with a tree-structured (rather than a decomposed) form which allows a closed form message passing sequence over a large tree-structured set of regions (this difference in convergence speed was discussed in Section 2.5.2.2 for ground inference).

Although the modifications necessary to present the anytime LWMB algorithm are fairly straightforward, they are notationally complex and would repeat (with only slight modification) several of algorithms already presented. Consequently, we simply describe the required modifications, using an example in Figure 5.5 to illustrate the concepts concretely. In our example, the model contains lifted RVs $\bar{a}$, $\bar{b}$, and $\bar{c}$ that are eliminated along lifted elimination order $\bar{\boldsymbol{o}} = (\bar{a}, \bar{b}, \bar{c})$, and the model contains lifted factors labeled $(\bar{a}, \bar{b})$, $(\bar{b}, \bar{c})$, and $(\bar{a}, \bar{c})$. Each lifted RV is associated with three ground RVs, $\text{gr}(\bar{a}) = \{a^1, a^2, a^3\}$, $\text{gr}(\bar{b}) = \{b^1, b^2, b^3\}$, and $\text{gr}(\bar{c}) = \{c^1, c^2, c^3\}$. In the left pane of Figure 5.5a, we show a LWMB graph constructed with `iBound` $= 1$; in the right pane we show the LWMB after a split of the lifted RV $\bar{a}$ into $\bar{a}_1$ and $\bar{a}_2$, associated with ground RVs $\text{gr}(\bar{a}_1) = \{a^1\}$ and $\text{gr}(\bar{a}_2) = \{a^2, a^3\}$. For LWMB, the variational parameters used to select the split appear on the lifted horizontal separators, which are colored in red and green in the left pane. These are split to form the red and dark red, and green and light green separators, respectively, in the right pane.

5.5.1 Constructing the LWMB Graph

To construct the LWMB graph, we use a lifted variant of the ground mini-bucket construction algorithm (Algorithm 4) that, essentially, substitutes lifted objects for the ground objects – notationally, this amounts putting a bar over the ground objects (the ground region graph, the ground elimination order, etc.). We now describe the LWMB algorithm and its

modifications to various steps of Algorithm 4.

The lifted region graph is constructed by simulating a relaxed lifted variable elimination procedure along lifted elimination order $\bar{\boldsymbol{o}} = (\bar{v}_1, \ldots, \bar{v}_{\bar{n}})$ where $\bar{n}$ is the number of lifted RVs.

The ground join-step (`Step 1` of Algorithm 4) is replaced by a lifted join-step where, at the iteration associated with lifted RV $\bar{v}$, we first collect the set of lifted factors whose earliest lifted RV (under the lifted elimination order $\bar{\boldsymbol{o}}$) is $\bar{v}$. We then partition these lifted factors into groups such that the lifted factors in each group can be joined into a resultant lifted factor that uses at most `iBound`+1 RVs, and, furthermore, we prohibit FOL-formulas from joining if any of their LVs range over the same set of logical constants, as required by our joining rule in Section 5.4.

The simulated ground elimination step (`Step 2` of Algorithm 4) is replaced by a simulated lifted elimination step. For each lifted region $\bar{r}$ whose earliest lifted RV is $\bar{v}$, we eliminate all copies of $\bar{v}$ which produces a lifted region using all lifted RVs except $\bar{v}$. This message region is added to the graph as a lifted region and connected via a lifted separator to $\bar{r}$.

Finally, in the cost-shift step (`Step 3` of Algorithm 4) we augment the graph with cost-shifting terms to reallocate cost between lifted regions associated with the current elimination. To do this, we add a lifted RV $\bar{v}$ to the region graph and connect it to each lifted mini-bucket region. Lastly, all evidence factors associated with ground RVs $\text{gr}(\bar{v})$ are linked to $\bar{v}$ in the graph.

Within the context of the anytime inference algorithm in Algorithm 9, the LWMB graph construction replaces the join step in `Step 3` to construct high-order inference terms.

5.5.2 Optimizing the LWMB Objective

The lifted WMB objective is

$$\bar{L}_{\bar{G}}^{WMB}(\bar{\boldsymbol{\delta}}_{\bar{S}_{\leftrightarrow}}, \boldsymbol{w}) = \underset{\bar{x}_{\bar{r}_{\bar{n}}}}{\overset{\bar{w}_{\bar{r}_{\bar{n}}}^{z_{\bar{r}_{\bar{n}}}}}{\texttt{lsew}}} \cdots \underset{\bar{x}_{\bar{r}_{\bar{n}}}}{\overset{\bar{w}_{\bar{r}_{\bar{n}}}^{1}}{\texttt{lsew}}} \cdots \underset{\bar{x}_{1}}{\overset{\bar{w}_{1}^{z_1}}{\texttt{lsew}}} \cdots \underset{\bar{x}_{1}}{\overset{\bar{w}_{1}^{1}}{\texttt{lsew}}} \sum_{\substack{\bar{v}\in\bar{\mathcal{V}} \\ i\in[z_{\bar{v}}]^+}} \bar{\theta}^{\bar{\delta}_{\leftrightarrow}}_{\bar{r}_{\bar{v}}^{i}}(\bar{x}_{\bar{r}_{\bar{v}}^{i}}) + \sum_{\bar{v}\in\bar{\mathcal{V}}} \sum_{v\in\mathrm{gr}(\bar{v})} \underset{x_v}{\overset{w_{\bar{v}}^{0}}{\texttt{lsew}}} \Big(\theta_v(x_v) + \delta_{\bar{v}}^{0}(x_v)\Big) \tag{5.7}$$

where the first line computes the forward message pass and is notationally identical to the ground WMB objective Eq. (2.13) but with bars over terms to index lifted, rather than ground, objects, and the second line is the collection of evidence terms associated with each lifted RV (note that this term also appeared in our lifted GDD bound (5.1)).

We then seek the tightest such bound

$$\underset{\bar{\boldsymbol{\delta}}_{\bar{S}_{\leftrightarrow}}, \bar{\boldsymbol{w}}}{\arg\min}\ \bar{L}_{\bar{G}}^{WMB}(\bar{\boldsymbol{\delta}}_{\bar{S}_{\leftrightarrow}}, \boldsymbol{w}).$$

As in the ground case, we perform a black box optimization using the objective evaluation and gradient (not shown) where the weights are transformed into log-space to maintain positivity.

5.5.3 Splitting

We select the split using gradient clustering on the variational parameters, as in the decomposed case in Section 5.3.2. However, for the LWMB algorithm, the variational parameters are located only on the horizontal separators $\bar{\boldsymbol{\delta}}_{\bar{S}_{\leftrightarrow}}$ and thus, only these separators (and not the message separators) are used in the clustering. That is, in Algorithm 10, when we select the split of logical domain $\bar{\Delta}$, rather than splitting *all* lifted separators $\bar{S}^{\bar{\Delta}}$ associated $\bar{\Delta}$, we split only lifted horizontal separators $\bar{S}^{\bar{\Delta}}_{\leftrightarrow} := \{\bar{s} \in \bar{S}_{\leftrightarrow} : \bar{\Delta} \in \bar{\mathrm{sc}}_{\bar{\mathbf{\Delta}}}(\bar{s})\}$ associated with $\bar{\Delta}$.

Otherwise, the clustering algorithm is identical.

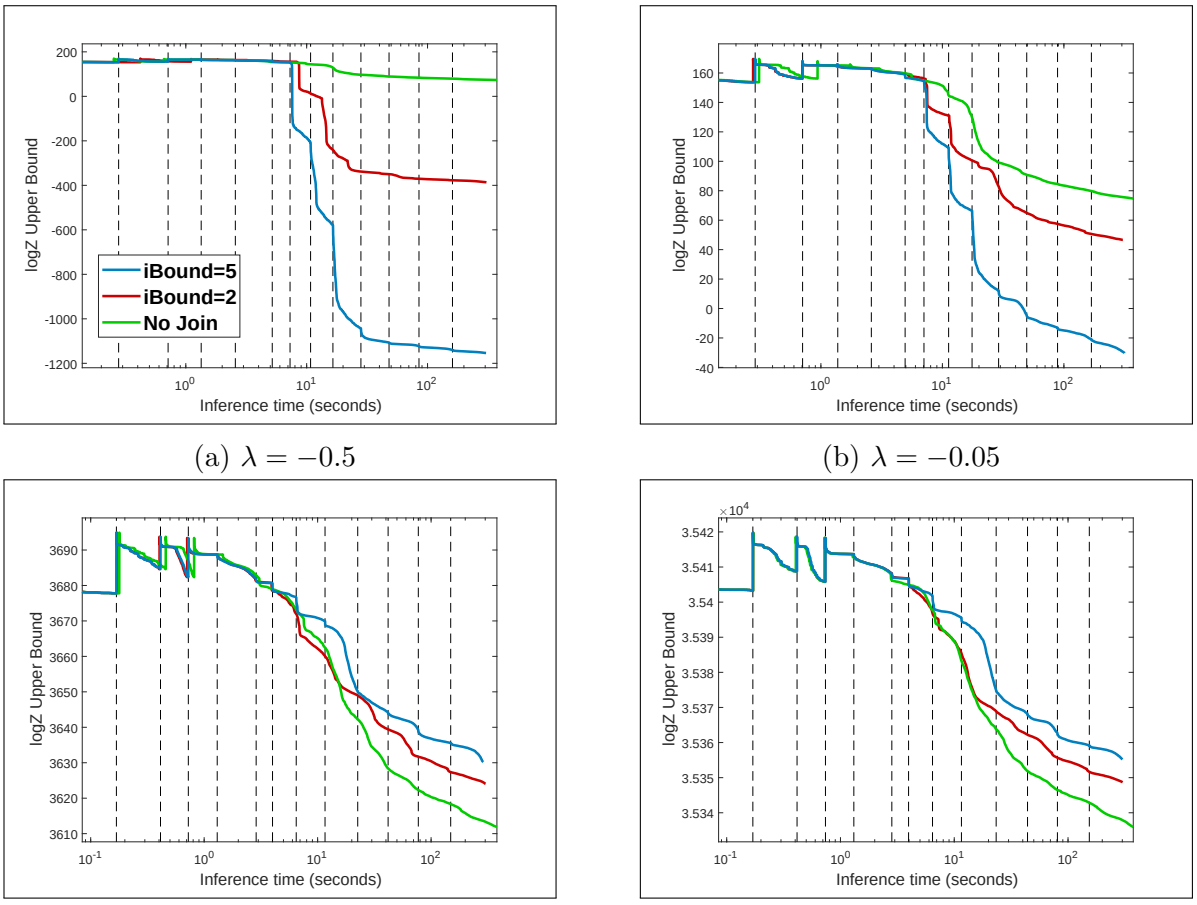
Modifying the LWMB structure after the split requires some extra work. We must split the lifted regions and lifted separators as we would for the decomposed case (the structure split step, `Step 2b`, in Algorithm 9), but also must maintain a tree structure. This can be accomplished by simply executing the LWMB graph construction (detailed in Section 5.5.1) on the decomposed regions where we disallow any joins. A more economical strategy, which we take, only modifies the LWMB graph on regions that split. That is, if a region does not split, then keep its forward separator structure the same; if it does split, then generate a new message region by a simulated elimination step.

5.6 Experiments

This section provides an empirical illustration of our anytime LWMB algorithm. We demonstrate the utility of relaxations that incorporate both split and join modifications. We show that in models with strong low-order inconsistencies, relaxations with high `iBound` are significantly more accurate. In contrast, in models with weak low-order inconsistencies, relaxations with high `iBound` can be less accurate. This is because they perform fewer splits than lower `iBound` relaxations do for the same cost. By performing more splits, the lower `iBound` approximation is able to represent asymmetric evidence more accurately, giving better results when this is the dominant source of approximation error.

5.6.1 Models

We consider a standard collective classification MLN with one FOL-formula $(\forall x \neq y \in \text{gr}(\bar{\Delta}_{\text{page}}))\ L(x,y) \wedge (C(x) \Leftrightarrow C(y))$ with weight λ where C is a `Class` and L is a `Link` predicate. If $\lambda < 0$, a hard `True` observation on the link $L(x,y)$ induces a repulsive potential between $C(x)$ and $C(y)$. Otherwise, if $\lambda > 0$, it induces an attractive potential. $\text{gr}(\bar{\Delta}_{\text{page}})$ is



(a) $\lambda = -0.5$

(b) $\lambda = -0.05$

(c) $\lambda = +0.05$

(d) $\lambda = +0.5$

Figure 5.6: Anytime LWMB results on simulated collective classification examples, illustrating the tradeoff between using different values of scope size `iBound`. (a)-(b) The models use repulsive $\lambda < 0$ potentials. Large `iBound` is important for resolving low-order inconsistencies and consequently significantly outperforms relaxations with lower `iBound`. (c-d) The models use attractive $\lambda > 0$ potentials. Low order inconsistencies are weak and large `iBound` regions do not produce a significant impact, but incur high cost. Consequently, for the same cost, large `iBound` relaxations underperform smaller `iBound` relaxations (which invest more time in splitting and representing evidence more accurately). In all plots, dashed black lines indicate a batch domain split (Step 2 in Algorithm 9) for the blue curve (split transitions for other curves occur at similar locations) in all subfigures.

an arbitrary set of $|\mathrm{gr}(\bar{\Delta}_{\mathrm{page}})| = 512$ web pages.

We generate models with synthetic observed evidence. The evidence is defined by a perturbed symmetric domain clustering as follows. Let $\boldsymbol{\Delta}' = \{\Delta'_k : k \in [K]^+\}$ be a partition of $\bar{\Delta}_{\mathrm{page}}$ into $K = 16$ random clusters of equal size. Let $q : \mathrm{gr}(\bar{\Delta}_{\mathrm{page}}) \to \boldsymbol{\Delta}'$ map each web page to its

cluster center.

Class: The evidence associated with $C(i)$ is a univariate factor $\theta_{C(i)}(x_{C(i)}) = [0; c_i]$ where $c_i \sim \bar{c}_{q^{-1}(i)} + \mathcal{N}(0, 0.4)$ and $\bar{c}_k \sim \mathcal{N}(0, 2)$ for $k \in [K]^+$. That is, each domain element i in cluster Δ'_k has a shared value $\bar{c}_k$ perturbed by an asymmetric term.

Link: The evidence associated with $L(i, j)$ is generated as follows: each pair of clusters $[k]^+ \times [k']^+$ is associated with $\bar{e}_{(k,k')} =$ `True` with probability 0.25 and `False` otherwise. Then, each $L(i, j)$ has evidence $\bar{e}_{(q^{-1}(i),q^{-1}(j))}$ with probability 0.75 and its negation otherwise. That is, each of the K^2 blocks has `True` evidence with each other block with probability 0.25; the individual evidence in each block is then flipped with probability 0.25.

5.6.2 Selecting Split or Join

When refining our approximations, we select a domain split using gradient clustering as discussed in Section 5.3.2. Lifted joins are performed by prioritizing the join of large-scope regions, as is common in ground mini-bucket [Dechter and Rish, 1997]; future work will look into selecting the join using preliminary inference information, as Forouzan and Ihler [2015] does for ground inference. As discussed in Section 5.4.1, it is difficult to select the appropriate time to perform a join, given its impact on the cost of future splits. In this work, we use a fixed schedule for splits and join operations, so that we perform ten domain splits then a global lifted join by building a LWMB-Graph with pre-specified `iBound`.

5.6.3 Inference

We perform inference on a LWMB relaxation by numerical optimization over its variational parameters. We use the `minFunc` toolbox in MATLAB as a black-box optimizier to perform

the optimization [Schmidt, 2005] using conjugate gradients and allowing a maximum of 1000 function evaluations.

5.6.4 Timing

We report time performing inference and not time updating the inference structure. Code for the former was written in C++ while code for the latter was written in MATLAB and thus provides unreliable timing. However, in a real system where the latter is written in C++, it should consume a negligible amount of time. This is because it touches the factor tables one time, compared to I times for I inference iterations. Our timing approach is consistent with other works on lifted inference, which commonly report only inference time (e.g.,Mladenov et al. [2014a], Van den Broeck and Niepert [2014] and our work in Chapter 4). However, those works can actually incur significant overhead during symmetry detection, i.e., by touching the entire ground model. In contrast, our LWMB algorithm with FOL-formulas never instantiates the full ground model, and so should be far more efficient if inference structure time is accounted for.

5.6.5 Results

Figure 5.6 shows results for (a) a strongly repulsive potential model, with $\lambda = -0.5$; (b) a weakly repulsive potential model, $\lambda = -0.05$; (c) a weakly attractive model, $\lambda = +0.05$; and (d) a strongly attractive model, $\lambda = +0.5$. In the repulsive cases (a) and (b), higher `iBound` relaxations significantly outperform lower `iBound` relaxations. This is because the repulsive models produce strong low-order inconsistencies which are (partially) resolved by higher order factors. In the attractive cases (c) and (d), higher `iBound` relaxations perform slightly worse. This is most likely because the smaller scope methods are able to build approximations with finer resolution for the same cost (meaning that they are able to represent model asymmetries more accurately). We see this effect, for example, in panel (c), where `iBound` =

5 performed 31 splits, `iBound` = 2 performs 60 splits, and `iBound` = 1 performs 73 splits by the final time-step. (The number of splits reported here are not represented on the plot.)

5.7 Conclusion

This chapter introduced the first lifted inference algorithm capable of controlling both the symmetry resolution and factor size of a lifted inference relaxation. We developed our algorithm within an anytime inference framework which adaptively chooses and performs split and join modifications to gradually provide more accurate results as more computational effort is applied. Furthermore, the entire algorithm operates in a FOL representation. This is a significant advantage over many other works on lifted inference, since this means that our method does not incur any significant hidden costs due to (approximate) symmetry detection procedures, which may touch the entire ground model.

Future work might look into improving the precision of the approximating structure, in particular, methods that coherently integrate factors with multiple resolutions of logical sets into a single inference structure. In practice, this may be useful to represent groups of people at a fine resolution within factors connecting strongly interacting groups, and groups of people at a coarse resolution within factors connecting weakly interacting groups. For example, for people known to be located in New York, we may need to reason about fine groups to determine their friendship relations, while we can reason about the friendships using coarse groups if people from one group are known to be in New York while people from one group are known to be in Kansas City (e.g., that most are unlikely to be friends).

Chapter 6

Adaptive Hierarchical State Space Partitions

High quality approximate inference in graphical models often requires reasoning over large sets of random variables. However, this reasoning incurs exponential cost when performed using the dense tabular factors typical of most inference algorithms. One approach to mitigate this cost is to introduce large factors with controlled state space symmetry. Although this restriction is straightforward in principle, many difficulties arise in practice which have forced many previous approaches to adopt unrealistic restrictions on the approximate inference structure, or to require an inference structure with exact symmetries encoded *a priori*.

In this chapter, we develop a more flexible and general solution with the following properties: we *recursively* construct complex state space symmetries needed to accurately capture the structure in real problems, we introduce groups of large factors with *aligned* symmetry such that they can exchange symmetric messages, and we select these structures *adaptively*, using preliminary inference estimates, to best capture the problem at hand. The resulting framework generalizes a number of previously proposed frameworks from disparate domains, shows promising initial results, and possesses a flexibility that invites many extensions.

6.1 Introduction

Computing inference quantities is a central problem in probabilistic graphical models. Since an exact inference computation is often intractable, relaxed inference algorithms that reason over factors with a controlled number of RVs are often necessary (as we discussed in Chapter 2). However, in many problems, to obtain accurate inference estimates, we must reason over large sets of RVs which incurs exponential cost for dense tabular factors required by most approximate inference algorithms.

This cost is mitigated if the factors – either model factors or factors introduced to tighten the inference relaxation – possess *state space symmetry*. In this case, large groups of states can be reasoned about with a single computation rather than individually, corresponding to a form of lifted inference described in Chapter 3.

In some applications, the model can be designed such that both the model and exact inference computations use a controlled number of symmetry groups. For example, Tarlow et al. [2012] models a hierarchical collection of counting factors over increasingly large groups of RVs. Each counting factor captures, for example, constraints on the fraction of pixels in a binary image patch that are `on`, e.g., for motion detection applications. Exact inference can be carried out in this model by viewing the hierarchy as a tree and passing messages structured as counting factors between neighboring resolutions [Tarlow et al., 2012].

However, models with exact inference symmetry are applicable only to a narrow range of problems. A wider class of models contains factors with state space symmetry but not inference symmetry. In these cases, the symmetry groups of model factors are *unaligned* and inference shatters symmetries into fine groups (as we discussed in Section 3.5.1). For example, consider a model with counting factors that constrain the number of `on` pixels within overlapping image regions. To first order, two pixels can be treated identically if and

only if they appear in the exact same set of overlapping regions. For a model with general pixel groupings, this results in no symmetries.

Even more generally, many graphical models are defined by dense tabular factors with no (or little) state space symmetry. Although it may seem counterintuitive at first, factors with state space symmetry can be very useful for approximating inference quantities in these models. We expect symmetric terms to be useful when the model (or a submodel) has one of following properties:

Nearly Identical Value: A set of states has nearly identical value and can be grouped and reasoned about together while incurring small error.

Low Value: A set of states has large relative disparity but low value and can be grouped into one "low-scoring" group. Thus, the approximation incurs error only in low probability mass regions of the distribution which have little or no effect on MAP inference or sum-inference queries, which depend mostly (or only) on the most probable states. Note that this case generalizes the popular idea of removing low-scoring states from the problem.

Symmetric Aggregation: Some MLN models contain symmetric factors laid out in a symmetric structure; the aggregation (or join) of this collection of factors is equivalent to a large factor with counting symmetry (see Section 3.8 for two examples with this property: a symmetric complete graph and a symmetric bipartite graph). For small perturbations of these models, we expect the aggregation (or join) of small factors to be approximately equal to a counting factor.

Several previous works for general PGMs have been developed to exploit the first two approximate symmetry properties. Gogate and Domingos [2013] performs lossy projected message passing on a region graph, where each message passing operation produces an unstructured

message factor which is then projected into a tractable symmetric form. This representation is useful when the message contains many nearly identical values or many small values. Sontag et al. [2009] tightens a standard inference relaxation by introducing high-order inference terms with a course resolution, defined by coarse groups of states of each RV (this is a special type of state space symmetry). In that work, each state of a RV with high belief (from preliminary inference estimates) is given its own group while the remaining states comprise a single low-scoring group. Thus, in the coarse-resolution high-order inference terms, each high scoring state is represented exactly and all of the low-scoring states are represented as one group. The utility of this approach is demonstrated on protein design problems whose RVs have large domain size (often one hundred or more states per variable).

Several works in the MLN literature exploit the symmetric aggregation rule. For example, Milch et al. [2008] performs a lifted bucket elimination algorithm; before factors in a bucket are joined (added), they check to see if the model contains a symmetric group of factors that can be joined to produces a large counting factor. If so, they efficiently form large-scope join with counting symmetry. Other works such as probabilistic theorem proving [Gogate and Domingos, 2012] also detect and exploit such rules.

A downside of these methods is that each is applicable to a narrow range of problems or is forced to adopt unrealistic restrictions. For example, the lossy message passing of Gogate and Domingos [2013] requires a fixed graph structure (i.e., it does not allow joins once inference has begun), the coarse factor tables of Sontag et al. [2009] are only applicable to problems whose RVs have large domain, and the rules used by Milch et al. [2008] and Gogate and Domingos [2012] require the user to identify an exact symmetric factor structure. The goal of this chapter is to introduce large-scope inference factors with controlled state space symmetry in a more intelligent and flexible way than previous work. The next subsection gives a brief overview of our approach, and the following subsection contrasts it with previous approaches.

6.1.1 Our approach

At a high level, the main components of our algorithm are as follows:

Hierarchical State Space Partitions: Our algorithm centers around a hierarchical state space partition that is constructed by alternating *coarsening operations* – that merge groups of states together to create a coarser group – with *joining operation*s – that form large-scope state space partitions from a set of smaller-scope state space partitions. This structure provides a natural way to incrementally construct large-scope symmetry groups that efficiently (implicitly) represent complex state space symmetries, which are necessary to capture the complex structure of real problems.

Aligned Factor Symmetry: These partitions in turn define a collection of inference factors with common *domain symmetry* which allow symmetric messages to be exchanged between them. All-in-all, we obtain a hierarchical collection of factors that represent inference information at varying sizes and resolutions which are coherently reasoned about via a standard message passing algorithm (with slight modifications).

Adaptive Inference: In general, it is difficult to select good symmetry groups *a priori* (before any inference). Instead, it is preferable to select, and subsequently refine, our symmetry groups using preliminary inference estimates. Our method admits flexible, anytime inference that allows us to adaptively add new levels and split symmetries on the fly. That is, we begin with small-scope factors with coarse symmetry, and gradually incorporate factors with larger scopes, while also splitting the symmetry of existing factors.

6.1.2 Comparison to Other Approaches

Although introducing terms with state space symmetry to represent inference quantities is straightforward in principle, several important practical difficulties arise, as we discussed above. Previously developed frameworks have addressed some of these difficulties, but each is feasible in a limited range of scenarios. In contrast, our framework offers a more complete and flexible solution. Relative to these other frameworks, our framework represents a generalization as follows:

Standard tabular join: Compared to methods which join dense tabular factors, our approach can obtain the same region graph and high-order inference terms. However, we use factors with controlled state space symmetry to mitigate the exponential cost of the higher-order terms.

Coarse clusters: From the point of view of Sontag et al. [2009], we build domain partitions with a much richer functional form. That is, we build a hierarchy of domain partitions over increasingly large groups of RVs, rather than a single partition of each univariate domain. From a technical perspective, this approach is closest to ours, as we discuss in more detail in Section 6.2.1.

Projected message passing: Compared to Gogate and Domingos [2013], we do not incur error via lossy message passing, and we adaptively form high order factors, which are difficult to incorporate into a lossy framework since the join incurs error that is difficult to analyze.

Recursive cardinality models: Our approach can be used to represent recursive cardinality models [Tarlow et al., 2012]; however, rather than a fixed, predefined counting partition, we utilize unstructured state space partitions which are more general and flexible, adaptively build high order factors rather than require them to already be

present in the model, and build a loopy structure that encompasses the hierarchical tree as a subgraph.

Counting aggregation: Comparing to Milch et al. [2008] and other methods that exploit the symmetric counting identity (see Section 3.8), our method builds state space partitions formed from arbitrary collections of factors, rather than being restricted to only incorporate counting factors corresponding to a collection of exactly symmetric factors.

In the following section we present a simple motivating example, which is essentially an instance of Sontag et al. [2009]. This example showcases the basic idea behind our algorithm and allows us to develop some technical notation and concepts. Using these concepts, we then, present an overview of our algorithm with slightly more technical precision. The remainder of the chapter then elaborates on each of these concepts.

6.2 Motivating Example

This section motivates the need for approximate inference structures that use state space symmetry in the context of a simple example; this example is (essentially) an instance of the approach taken by Sontag et al. [2009]. Following this example, we discuss shortcomings with this approach, namely that it uses univariate domain partitions, which can severely restrict our ability to represent certain useful inference patterns. We then present an overview of our approach to constructing large multivariate state space partitions that addresses these shortcomings. We conclude with an overview of the chapter which develops these concepts in detail.

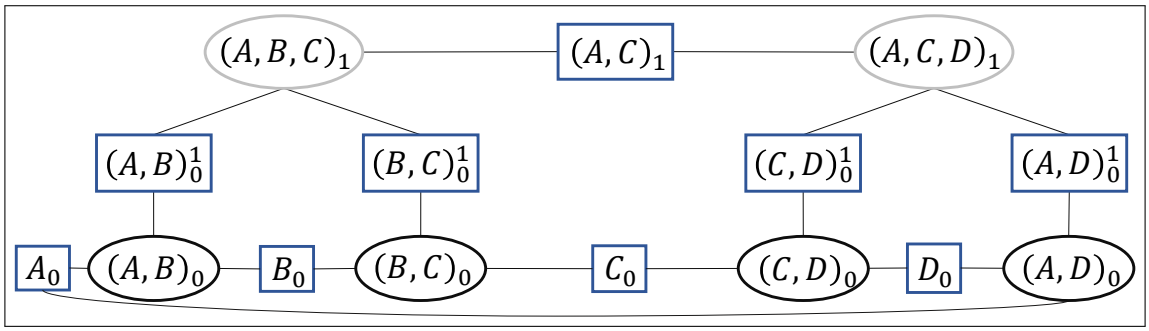


Figure 6.1: A two-level hierarchical region graph with univariate RV domain symmetry. The bottom level shows (small scope, fine resolution) subgraph G_0, the top level shows (large scope, coarse resolution) subgraph G_1, and in between levels are inter-level separators S_0^1 and edges E_0^1 that carry cost between levels. In this chapter, R_0 represents model factors (outlined in black) while R_1 represents terms introduced to tighten the relaxation (with factor value zero, outlined in gray).

6.2.1 Univariate Domain Partitions: Example

As a motivating example, we consider a graph consisting of a simple cycle, $A-B-C-D-A$, where each RV takes on k_0 possible states. A fully relaxed region graph $G_0 = (R_0, S_0, E_0)$ representing this model is shown in the bottom row of Figure 6.1. The top row corresponds to a higher order subgraph $G_1 = (R_1, S_1, E_1)$ introduced to tighten the relaxation where each region $r_1 \in R_1$ has factor value zero, i.e., $\theta_{r_1} = \mathbf{0}$, and, hence, does not change the distribution. A set of inter-level separators S_0^1 and edges E_0^1 shift cost between the low-order and high-order regions. These high-order regions incur cost k_0^3, which may be prohibitively large. For example, the protein design problems considered in Sontag et al. [2009] can have domain size $k_0 > 100$.

6.2.1.1 Exact Symmetry

This cost is reduced if inference operations produce high-order factors with state space symmetry with many fewer than k_0^3 groups. This occurs, for example, when each model factor $r_0 \in R_0$ has RV domain symmetry associated with RV domain partitions $\bar{\mathcal{D}}$ where each domain partition has $k_1 < k_0$ groups. Recall from Section 3.7.1 that model factors of

this form give rise to LSS-factors with a coarse tabular structure; moreover, factors of this form give rise to inference operations that do not shatter the LSS-factor symmetries. In the current example, model factors with these symmetries give rise to high-order LSS-factors that also take a coarse tabular form defined by domain partitions $\bar{\mathcal{D}}$. Here, each table has size $k_1 \times k_1 \times k_1$, yielding a computational cost of k_1^3.

As a concrete example, suppose that our variables have $k_0 = 3$ states, but the each factor has RV domain symmetry defined by domain partitions $\bar{\mathcal{D}}_v = \{\{0, 1\}, \{2\}\}$ with $k_1 = 2$ groups at each RV $v \in \mathcal{V}$, i.e., states 0 and 1 in each RV are indistinguishable. In this case, the model contains LSS-factors such as $\bar{\theta}_{AB_0} = \begin{bmatrix} e & f \\ g & h \end{bmatrix}$ of size $k_1 \times k_1$ (here, 2×2) which specifies a ground factor $\theta_{AB_0} = \left[\begin{array}{cc|c} e & e & f \\ e & e & f \\ \hline g & g & h \end{array}\right]$ of size $k_0 \times k_0$ (here, 3×3) with tabular state space symmetry.

An exact solution to the (ground) variational inference problem is obtained by setting each cost-shifting factor equal to the model factor, e.g., $\bar{\delta}_{AB_0^1} = \bar{\theta}_{AB_0}$. A cost-shifting operation leaves value zero in the low-order regions, i.e., $\bar{\theta}_{AB_0} - \bar{\delta}_{AB_0^1} = \mathbf{0}$, and produces high-order regions with RV domain symmetry that uses the same domain partitions $\bar{\mathcal{D}}$ as the low-order regions. In our running example, the high-order regions are LSS-factors with table size $k_1 \times k_1 \times k_1$ (here, $2 \times 2 \times 2$) which, as above, corresponds to treating states 0 and 1 identically at each RV. For example, at the region $(ABC)_1$, its neighboring sum $\bar{\delta}_{AB_0^1} + \bar{\delta}_{BC_0^1}$ is a LSS-factor with this form.

6.2.1.2 Exploiting Approximate Symmetry via Inter-Level Parameter Tying

In practice, models with exact symmetry are unlikely to arise; in fact, PGMs are often defined by dense factor tables with no state space symmetries. In this case, our goal is to exploit *approximate* state space symmetries. To do this, we seek to tighten the (asymmetric) low-order inference relaxation with high-order regions *restricted* to have state space symmetry.

This restriction allows high-order interactions to be represented efficiently, albeit at a coarser than optimal resolution.

We accomplish this by *restricting* the inter-level separators to have RV domain symmetry associated with RV domain partitions $\bar{\mathcal{D}}$. As in the case of exact symmetry above, this gives rise to high-order terms with RV domain symmetry over which symmetric inference operations can be performed. The inter-level separators mediate between regions with finer state space resolution but smaller cliques (lower level) and a coarser-resolution version of these factors with the larger cliques at the upper level.

We first discuss two different scenarios in which approximate symmetries are likely to be useful. We then describe the inference relaxation for two levels in mathematical detail.

Nearly Identical Value In our running example, suppose that the model factor is symmetric up to a small perturbation of its values, i.e., $\theta_{AB_0} = \left[\begin{array}{cc|c} e_1 & e_2 & f_1 \\ e_3 & e_4 & f_2 \\ \hline g_1 & g_2 & h_1 \end{array}\right]$ where values with the same base character are approximately identical, i.e., $e_j \approx e$, $f_j \approx f$, $g_j \approx g$, $h_j = h$ for all valid indices j. This is approximated by an inter-level cost-shifting factor $\delta_{AB_0^1} = \left[\begin{array}{cc|c} e & e & f \\ e & e & f \\ \hline g & g & h \end{array}\right]$. A cost-shifting operation leaves near-zero mass at the lower level, i.e., $\theta_{AB_0} - \delta_{AB_0^1} \approx \mathbf{0}$ (as opposed to leaving exactly zero mass at the lower level, as in the case of exact model symmetry). At the upper level, we have LSS-factors with coarse tabular state space symmetry (over $2 \times 2 \times 2$ tables in this example). These factors have the same dimensions as they did in the example with exact model symmetry in the last subsection.

Low Value States Factors with state space symmetry can also provide useful approximations when the factor values have large relative disparity but low score. In this case, we can group the states with low value together and incur little error in marginal or MAP inference queries which inherently depend on high-scoring configurations. In our running example where states 0 and 1 are grouped together, this grouping incurs little error if the factor val-

ues for each configuration using states 0 or 1 are low (relative to the score for configuration 2). For example, a factor such as $\theta_{AB_0} = \left[\begin{array}{cc|c} -11 & -9 & -5 \\ -5 & -8 & -4 \\ \hline -20 & -4 & +0 \end{array}\right]$ has values with large disparity in each cell, however, all cells have much lower value than the one at the bottom right which corresponds to both RVs taking state 2.

It is important to contrast this approach to a simple but common approach in which we simply throw away low-scoring states to form RVs with smaller domains. Discarding variable states can remove joint configurations that have non-zero probability, making it difficult or impossible to produce an upper bound with such methods. Moreover, states that look unlikely at first may later come to be important; but once removed, it is difficult to re-include additional states later to improve the approximation accuracy. By merging states into a single group, we are able to continue to reason about the existence of those states while approximating their value, an approach which allows the approximation to be refined as more information becomes available.

6.2.1.3 Two Resolution Lifted Inference

Intra-level parameter tying gives rise to an inference approximation using factors at two resolutions; that is, the upper level contains coarse resolution LSS-factors while the lower level contains fine resolution (ground) factors. This can be seen by examining the form of the reparameterized factors at each level. For any region $r_1 \in R_1$ at the upper level, the ground reparameterization equation (Eq. (2.5)) has an equivalent lifted form as

$$\begin{aligned} \theta^{\delta}_{r_1}(x_{r_1}) &= \theta_{r_1}(x_{r_1}) + \sum_{s_1 \in N(r_1) \cap S_1} \delta_{s_1}(x_{s_1}) + \sum_{s^1_0 \in N(r_1) \cap S^1_0} \delta_{s^1_0}(x_{s^1_0}) \\ &= \bar{\theta}_{r_1}(y_{r_1}) + \sum_{s_1 \in N(r_1) \cap S_1} \bar{\delta}_{s_1}(y_{s_1}) + \sum_{s^1_0 \in N(r_1) \cap S^1_0} \bar{\delta}_{s^1_0}(y_{s^1_0}) \;=:\; \bar{\theta}^{\bar{\delta}}_{r_1}(y_{r_1}) \end{aligned}$$

for any ground state $x_{r_1} \in \text{gr}(y_{r_1})$ where the lifted auxiliary variables $y_{r_1} \in \bar{\mathcal{D}}_{r_1}$ in the second line range over the joint set of RV domain partitions; furthermore, the first summation (in each line) ranges over the *intra-level separators* incident to r_1, and the second summation (in each line) ranges over the *inter-level separators* incident to r_1. Note that in this chapter, the higher order potentials $\bar{\theta}_{r_1}(y_{r_1}) = 0$ for all y_{r_1}, since these regions are introduced during inference to tighten the relaxation.

For any region $r_0 \in R_0$ at the lower level, the ground reparameterization (Eq. (2.5)) is

$$\theta^{\delta}_{r_0}(x_{r_0}) \;=\; \theta_{r_0}(x_{r_0}) + \sum_{s_0 \in N(r_0) \cap S_0} \delta_{s_0}(x_{s_0}) + \sum_{s^1_0 \in N(r_0) \cap S^1_0} \delta_{s^1_0}(x_{s^1_0}) \tag{6.1}$$

for any ground state $x_{r_0} \in \mathcal{D}_{r_0}$.

We can then rewrite the costly $O(k_0^3)$ ground inference objective (first line) as a *LSS-inference-objective* (second line) with cheaper cost of only $O(k_1^3)$ as

$$\begin{aligned} L(\delta) \;&=\; \sum_{r_0 \in R_0} \max_{x_{r_0} \in \mathcal{D}_{r_0}} \theta^{\delta}_{r_0}(x_{r_0}) \;+\; \sum_{r_1 \in R_1} \max_{x_{r_1} \in \mathcal{D}_{r_1}} \theta^{\delta}_{r_1}(x_{r_1}) \\ &=\; \sum_{r_0 \in R_0} \max_{x_{r_0} \in \mathcal{D}_{r_0}} \theta^{\delta}_{r_0}(x_{r_0}) \;+\; \sum_{r_1 \in R_1} \max_{y_{r_1} \in \bar{\mathcal{D}}_{r_1}} \bar{\theta}^{\bar{\delta}}_{r_1}(y_{r_1}) \;=\; \bar{L}(\bar{\delta}). \end{aligned}$$

We emphasize again that the lifted auxiliary variables y are just used to index the efficient lifted objective and *not* additional RVs introduced into the model.

6.2.1.4 Coarse-to-Fine State Set Splitting

In general, it is not possible to accurately select the RV domain symmetry groups $\bar{\mathcal{D}}_v$ for each RV $v \in \mathcal{V}$ *a priori*. Instead, we must use preliminary inference estimates to select groups that we believe will best approximate desired inference quantities. For example, in the protein problems used in Sontag et al. [2009], relaxed inference on pairs of RVs results in identifying

many states that are nearly ruled out, despite not being ruled out in the original model factors. In essence, the relaxed inference discovers that these states are inconsistent with the high-scoring configurations of other variables. Then, states that have been identified as low-value can be grouped together.

Sontag et al. [2009] performs a symmetry grouping in one-shot; that is, they perform some number of relaxed inference iterations, then add the high-order inference terms and select the k_1 states set partitions for each RV. However, a more attractive framework is to select the domain partition gradually, in a coarse-to-fine manner where inference is interleaved with splitting of a domain partition $\bar{\mathcal{D}}_v^{(t)}$ at time-step t to a finer partition $\bar{\mathcal{D}}_v^{(t+1)}$ at time-step $t+1$ for each RV $v \in \mathcal{V}$. Ideally, this split should be selected using preliminary inference estimates. (Recall that we used a similar framework for models, such as MLNs, with factor symmetries in the previous two chapters.)

Each domain split has the effect of splitting the high-order factor tables. At iteration t, the high-order factor table has size $k_1^{(t)} \times k_1^{(t)} \times k_1^{(t)}$ where we assume that the domain of each RV is partitioned into $|\bar{\mathcal{D}}_v^{(t)}| = k_1^{(t)}$ sets. In our example above where each RV has $k_0 = 3$ states, at the first split, each domain is partitioned into two groups and each high-order factor table has size $2 \times 2 \times 2$. The next split produces three groups and each high-order factor table has size $3 \times 3 \times 3$. In this case, each group of states is a single state (since we have $k_0 = k_1 = 3$) and the factors are equivalent to those obtained by a dense join operation.

6.2.2 Univariate Hierarchical Region Graph

For the moment, let us again consider a static inference structure, i.e., we do not perform any anytime inference or splitting operations. To incorporate larger regions, we can apply the idea of domain partitioning recursively. That is, we generate a sequence of increasingly coarse univariate domain partitions. For example, for an RV with six states $\mathcal{D}_v = \{0, 1, 2, 3, 4, 5\}$,

at the first level we might form a partition $\{\{0,2\},\{4\},\{1,3,5\}\}$, and at the second level we might form and a coarser partition $\{\{0,2,4\},\{1,3,5\}\}$ (which treats group $\{4\}$ identically to group $\{0,2\}$). These increasingly coarse partitions can then be used to define increasingly large region factors with controlled cost. This forms a *hierarchical region graph* which represents information at multiple resolutions and sizes. The hierarchy of regions can be reasoned over coherently by passing messages within the same level and messages between adjacent levels, as in the two level example above.

Shortcomings However, such univariate domain partitions are often either infeasible or unhelpful in practice, for two reasons. First, for RVs with small domains, only a few levels of partitions can be applied – for binary RVs, no partitioning of a single variable's domain is possible. While we could potentially pre-groups sets of RVs to define a new model over "hyper-RVs" whose domains are joint configurations of several original variables, this relies on knowing *a priori* which variables would be useful to group together.

Second, the symmetry groups formed by univariate partitions are generally inefficient for large-scope regions. Imagine that we have coarsened several RVs in order to reason about them jointly. The resulting large-scope region will look like a function over many variables with few states each, and (just as with binary variables) we will be unable to exploit any symmetries that appear in this joint space. In practice, these larger regions often have low mass in many of the joint configurations, which should ideally be grouped together.

Another illustration of this effect is when groups of RVs strongly covary within the model – for example, in stereo vision problems, the factors depend on the difference in disparity, i.e., $\theta(x_i, x_j) = g(x_i - x_j)$. Then, coarsening x_i or x_j is not helpful, despite the fact that many joint configurations of (x_i, x_j) should be grouped together. Some methods take the approach of forcing certain variables to take on the same value; however, this provides only a lower bound (since it does not consider states where the linked variables are not equal),

and allows only a specific type of refinement (splitting the sets of linked variables).

6.2.3 Our Approach: Hierarchical State Space Partitions

Since univariate symmetry groups are insufficient in many settings, more flexible symmetry groups are required. However, recall from our discussion in Section 3.5.2 that unrestricted symmetry groups quickly become infeasible, since symmetry groups that are unaligned will shatter the symmetry of inference quantities. In this chapter, we develop a framework that satisfies the two competing desires of *flexibility* and *alignment*. The basic idea of our approach is to substitute a more flexible class of multivariate domain partitions for the univariate ones considered above. This flexible class of partitions is then used to define factors with aligned symmetry over which messages can be passed.

The centerpiece of our method is a tree-structured collection of state space partitions over increasingly large groups of RVs, rather than individual RVs, called a Coarsen-Join-Tree (CJ-Tree). These state space partitions represent complex groupings of joint states that are constructed recursively via an alternating sequence of *coarsening* and *joining* operations. A coarsening operation merges a collection of groups of states into fewer groups, forming a new state space partition in which the groups are treated identically; by controlling the number of state space symmetry groups, we can control the cost of representing the partition. A joining operation combines partitions over disjoint sets of RVs and allows us to construct a partition over the larger joint state space; this provides a natural way to incrementally build up large-scope state space partitions from small-scope partitions.

The CJ-Tree can be used to represent an interesting class of distributions in its own right. We allow the state space partition at a node to represent symmetry groups in a LSS-factor, so that the model consists of functions at multiple resolutions of state space symmetry. As a special case, we will see that we can recover the same structure of counting factors at

multiple resolutions used by Tarlow et al. [2012] and mentioned in the chapter introduction.

However, our goal is not to use the CJ-Tree directly as an inference structure, but rather to use it to guide the construction of state space symmetry for factors in region graph with a general loopy topology, which is required for realistic inference problems. A region graph of this form is known as a *hierarchical region graph* and its factors are composed of state sets from the CJ-Tree. From the point of view of region graphs defined by univariate RV domain symmetry used in the two-level example above, the hierarchical region graph can be viewed as a substitution of the univariate partitions for a much richer class of multivariate state space partitions.

Since the inference symmetries can not be determined *a priori*, we develop an adaptive inference procedure that selects symmetry groups using preliminary inference estimates. In our framework, the modifications take two forms: either splitting a symmetry group to obtain finer symmetries, or joining to add another level of factors of larger scope. Significantly, although we build complex state space symmetries and a complex model structure, we maintain flexibility of selecting each of them adaptively.

Chapter Organization The remainder of the chapter provides the details and is organized as follows. In Section 6.3, we develop the CJ-Tree and its associated coarsening and joining operations. We discuss the complex class of functions that the CJ-Tree can represent and show how inference operations on a CJ-Tree can be computed efficiently. In Section 6.4 we show how to construct a hierarchical region graph using symmetry groups defined on the CJ-Tree. This allows us to reason within a loopy graph composed of large-scope factors with aligned state space symmetry. In Section 6.5, we develop an anytime inference procedure that interleaves inference with *splitting* operations that form finer resolution state space partitions (on the CJ-Tree) and *joining* operations that form larger-scope regions, i.e., adds additional levels to the CJ-Tree. We present simple heuristics for selecting these operations;

future work will examine this selection step in more detail. Finally, Section 6.6 concludes with experimental results demonstrating high quality results in general inference problems.

6.3 Hierarchical State Space Partitions

In this section, we show how to construct complex multivariate state space partitions efficiently. These symmetry groups can be used define a tree-structured region graph whose factors are of increasingly large size but have controlled state space symmetry. Moreover, this structure provides the basis for our more general loopy inference procedure presented in Section 6.4.

The state space partitions are constructed via an alternating sequence of *coarsening* and *joining* operations. A coarsening operation merges groups of states into a single group of states; by controlling the number of groups, the coarsening operation can control the cost of the representation. A joining operation concatenates states from symmetry groups over (non-overlapping) subsets of RVs to form a symmetry group in a higher dimensional space; the joining operation provides a natural way to incrementally build up large-scope state space partitions.

The coarsening and joining operations are represented at a high level (as relationships between groups of RVs) on a graph called a Coarsen-Join-Tree (CJ-Tree). A CJ-Tree is structurally identical to a standard tree-structured region graph that is oriented *upward* where each leaf node has univariate scope and nonleaf nodes are of increasingly large scope size up the tree. Associated with the CJ-Tree is a Coarsen-Join-Lifted-State-Space-Tree (CJ-LSS-Tree) that represents state space symmetries within each factor and the relationships between them. The CJ-LSS-Tree is an instance of a LSS-Graph that we saw in Section 3.5.2 where the ground state space partitions are defined implicitly via the coarsening and joining

operations. Figure 6.2 shows an example that will be discussed in detail later; on the left is a CJ-Tree and on the right a corresponding CJ-LSS-Tree.

Section Overview In Section 6.3.1, we define the CJ-Tree and in Section 6.3.2, we define its associated CJ-LSS-Tree. In Section 6.3.3, we present the coarsening and joining operations. In Section 6.3.4, we show how lifted inference operations are represented on a CJ-LSS-Tree. Finally, in Section 6.3.5, we discuss how the CJ-Tree relates to previous work.

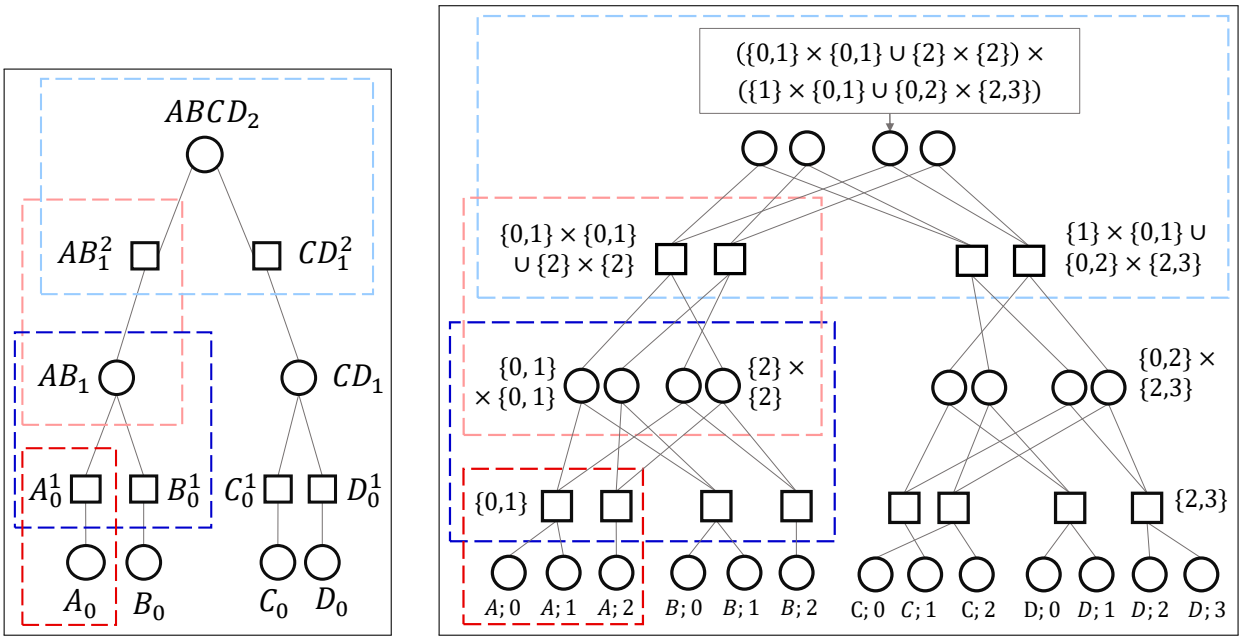
6.3.1 Region Graph

A *Coarsen-Join-Tree (CJ-Tree)* $T = (D^\circ, D^\square, \dot{E})$ is a tree-structured region graph consisting of region nodes, or *join-nodes*, D° (drawn as circles), separator nodes, or *coarsen-nodes*, $D^\square$ (drawn as squares), and edges $\dot{E}$ that each connect one coarsen-node and one join-node.*

Each model RV $v \in \mathcal{V}$ is associated with exactly one leaf join-node $j \in D^\circ$ with univariate scope $\text{sc}(j) = \{v\}$. The neighbors of any node $\eta \in D^\circ \cup D^\square$, denoted $N(\eta) = N_\downarrow(\eta) \cup N_\uparrow(\eta)$, are divided into *children* $N_\downarrow(\eta)$ which are closer to the leaves and one (or zero) *parent* $N_\uparrow(\eta)$ which is further from the leaves. The graph consists of (at least) one root which has zero parents; note also that the leaves have zero children.

Nodes are of increasingly large scope size up the tree (away from leaves). Each coarsen-node $c \in D^\square$ has exactly one child, $N_\downarrow(c) = \{j_\downarrow\}$ (which is a join-node, $j_\downarrow \in D^\circ$), and it has the same scope as its child $\text{sc}(c) = \text{sc}(j_\downarrow)$. Each non-leaf join-node $j \in D^\circ$ has multiple children $N_\downarrow(j)$ (which are all coarsen-nodes, $N_\downarrow(j) \subset \bar{\mathcal{D}}^\square$) and has scope $\text{sc}(j) = \cup_{c_\downarrow \in N_\downarrow(j)} \text{sc}(c_\downarrow)$ equal to the union of its childrens' scopes.

*We reserve the traditional terminology and symbols $G = (R, S, E)$ for the loopy region graph in the next section.



(a) CJ-Tree T (region graph) (b) CJ-LSS-Tree $\bar{\mathcal{T}}$ (LSS-Graph)

Figure 6.2: (a) A CJ-Tree T represents alternating levels of coarsening and joining operations. Join-nodes are depicted as circles and coarsen-nodes are depicted as squares. Each node is labeled with its scope and a level index. (b) A CJ-LSS-Tree $\bar{\mathcal{T}}$ represents details of the coarsening and joining operations; this correspondence is highlighted with boxes of the same color in each subfigure. Each LSS-node represents a set of ground states and select nodes are labeled; non-leaf labels abbreviate a state by dropping its scope symbol, e.g., $\{0, 1\} \equiv \{(A; 0), (A; 1)\}$ in the dark red box.

Example An example CJ-Tree over four RVs $\mathcal{V} = \{A, B, C, D\}$ is shown in the left panel of Figure 6.2†. Each node is indexed with a concatenation of RVs in its scope plus integer sub- and super-scripts that represents a level index (which, for now, serves only to disambiguate between nodes). For an example coarsening operation boxed in light red, the parent coarsen-node has the same scope as its child $\text{sc}(AB_1^2) = \text{sc}(AB_1) = \{A, B\}$. For an example joining operation boxed in light blue, the parent join-node has scope equal to the union of its childrens' scopes, $\text{sc}(ABCD_2) = \{A, B, C, D\} = \{A, B\} \cup \{C, D\} = \text{sc}(AB_1^2) \cup \text{sc}(CD_1^2)$.

6.3.2 LSS-Graph

The CJ-Tree is associated with a CJ-LSS-Tree that represents the state space symmetry of its region and separator factors. A CJ-LSS-Tree is a special type of LSS-Graph where, as described Section 3.5.2, each node represents a group of states and each edge represents a relationship (alignment) between two groups of states. Our primary motivation for making this equivalence is it allows us to reuse the notation and graphical depiction (where each node is a set of states) of the LSS-Graph. However, it is also worth noting that the CJ-LSS-Tree is distinguished from a generic LSS-Graph in the way that it is constructed. In a CJ-LSS-Tree, groups of states are defined implicitly via an alternating sequence of *coarsening and joining operations* among groups of ground states. In contrast, in Section 3.5.2, the LSS-Graph was defined by detecting symmetries on a SS-Graph (representing dense tabular factors). The implicit representation of the CJ-LSS-Tree is critical as it avoids the exponential cost incurred by an explicit SS-Graph representation.

For clarity, this section reviews the LSS-Graph representation using the notation for a CJ-LSS-Tree. The next section discusses coarsening and joining operations that define state space symmetry groups.

6.3.2.1 Definition

A *Coarsen-Join-Lifted-State-Space-Tree (CJ-LSS-Tree)* $\bar{\mathcal{T}} = (\bar{\mathcal{D}}^{\circ}, \bar{\mathcal{D}}^{\square}, \bar{\dot{\mathcal{E}}})$ is a LSS-Graph associated with CJ-Tree $T = (D^{\circ}, D^{\square}, \dot{E})$ where $\bar{\mathcal{D}}^{\circ}$ is a set of LSS-regions (depicted as circles), $\bar{\mathcal{D}}^{\square}$ is a set of LSS-separators (depicted as squares), and $\bar{\dot{\mathcal{E}}}$ is a set of LSS-edges. Recall that each node in the LSS-Graph represents a group of ground states and that each edge in the LSS-Graph represents a relationship (alignment) between its incident groups of ground states.

[†]Note that the graph is defined by directed edges that point upward in this example. But for compactness, the arrowheads are not drawn; thus we abuse representation to depict this as an undirected graph.

Model LSS-Factor: Each join-node $j \in D^\circ$ is associated with a set of join-LSS-nodes $\bar{\mathcal{D}}_j^\circ \subset \bar{\mathcal{D}}^\circ$. Each LSS-node in the set is denoted as $y_j^\circ \in \bar{\mathcal{D}}_j^\circ$ and is associated with a set of ground states $\text{gr}(y_j^\circ) \subset \mathcal{D}_{\text{sc}(j)}$. Collectively, these sets represent the symmetry groups of a model ground factor $\theta_j : \mathcal{D}_j \to \mathbb{R}$ which is represented as a LSS-factor $\bar{\theta}_j : \bar{\mathcal{D}}_j^\circ \to \mathbb{R}$ where $\theta_j(x_j) = \bar{\theta}_j(y_j^\circ)$ for any $x_j \in \text{gr}(y_j^\circ)$ (note we also require that each state $x_j \in \mathcal{D}_j$ is in exactly one symmetry group).

Cost-shifting LSS-Factor: Similarly, each coarsen-node $c \in D^\square$ is associated with a set of coarsen-LSS-nodes $\bar{\mathcal{D}}_c^\square \subset \bar{\mathcal{D}}^\square$. Each LSS-node in the set is denoted as $y_c^\square \in \bar{\mathcal{D}}_c^\square$ and is associated with a set of ground states $\text{gr}(y_c^\square) \subset \mathcal{D}_{\text{sc}(c)}$. Collectively, these sets represent the symmetry groups of a cost-shifting ground factor $\delta_c : \mathcal{D}_c \to \mathbb{R}$ which is represented as a LSS-factor $\bar{\delta}_c : \bar{\mathcal{D}}_c^\square \to \mathbb{R}$ where $\delta_c(x_c) = \bar{\delta}_c(y_c^\square)$ for any $x_c \in \text{gr}(y_c^\square)$ (note we also require that each state $x_c \in \mathcal{D}_c$ is in exactly one symmetry group).

LSS-Reparameterization An edge $(j, c) \in \dot{E}$ between join-node $j \in D^\circ$ and coarsen-node $c \in D^\square$, is associated with LSS-edges of the form $(y_j^\circ, y_c^\square) \in \bar{\dot{\mathcal{E}}}$ where $y_j^\circ \in \bar{\mathcal{D}}_j^\circ$ and $y_c^\square \in \bar{\mathcal{D}}_c^\square$. The LSS-edge represents *alignment* between ground symmetry groups such that $[x_j \in \text{gr}(y_j^\circ)] \;\Rightarrow\; [x_c \in \text{gr}(y_c^\square)]$; that is, for any ground state $x_j \in \text{gr}(y_j^\circ)$ associated with the region-LSS-node y_j°, its substate $x_c \in \text{gr}(y_c^\square)$ is associated with the separator-LSS-node $y_c^\square$ (where x_c is the substate of x_j associated with RVs of c).

Since these alignments hold at each edge, a ground reparameterization operation does not shatter model symmetries and we can form a lifted reparameterization operation. That is, for any join-node $j \in D^\circ$ and associated LSS-node $y_j^\circ \in \bar{\mathcal{D}}_j^\circ$, the ground reparameterization equation (Eq. (2.5)) has an equivalent lifted form, represented by the neighborhood in the

LSS-Graph, as

$$\begin{aligned}\theta_j^{\delta}(x_j) \;&=\; \theta_j(x_j) \;+\; \sum_{c \in N(j)} \theta_c(x_c) \\ &=\; \bar{\theta}_j(y_j^{\circ}) \;+\; \sum_{\substack{c \in N(j) \\ \{y_c^{\square}\} = \bar{\mathcal{N}}(y_j^{\circ}) \cap \bar{\mathcal{D}}_c^{\square}}} \bar{\delta}_c(y_c^{\square}) \;=:\; \bar{\theta}_j^{\bar{\delta}}(y_j^{\circ}) \end{aligned} \tag{6.2}$$

for any $x_j \in \mathrm{gr}(y_j^{\circ})$. In the second line, the first row of the summation range indexes separator neighbors in the region graph and the second row selects the LSS-neighbor associated with that separator.

6.3.3 Coarsening and Joining Operations

Groups of states are defined *implicitly* via an alternating sequence of *coarsening and joining operations* among groups of ground states. At the bottom of the tree, each leaf join-node $j \in D^{\circ}$ is associated with an uncoarsened partition of the domain of a single RV (i.e., $\mathrm{gr}(\bar{\mathcal{D}}_j^{\circ}) = \{\mathrm{gr}(y_j^{\circ}) : y_j^{\circ} \in \bar{\mathcal{D}}_j^{\circ}\} = \{\{d\} : d \in \mathcal{D}_v\}$ where $\mathrm{sc}(j) = \{v\}$). At the next level, a coarsening operation merges distinct groups of states into a single group of states; thus, states that were treated distinctly at the child are treated identically at the parent. By controlling the number of such groups, we can control the cost of the representation. A joining operation then concatenates coarsened groups of states over disjoint sets of RVs to form a group of joint states in a higher dimensional space; this provides a natural way to incrementally build up large-scope state space partitions from small-scope state space partitions. These two operations are alternated up the tree to efficiently produce large complex state space symmetry groups.

We begin in this section by describing the coarsening operation; we then describe the joining operation. We then provide a detailed example to illustrate these operations. Finally, we discuss the types of functions that can be represented with coarsening and joining opera-

tions. These included structured symmetry classes used in previous works, such as recursive counting partitions.

6.3.3.1 Coarsening Operation

A coarsening operation is represented in the CJ-Tree by a parent coarsen-node $c_\uparrow \in D^\square$ and its child join-node $j \in D^\circ$, i.e., $N_\downarrow(c_\uparrow) = \{j\}$ (e.g., the subgraph boxed in dark red in Figure 6.2). The details of the coarsening operation are represented on the CJ-LSS-Tree by the set of parent coarsen-LSS-nodes $\bar{\mathcal{D}}^\square_{c_\uparrow}$ and children join-LSS-nodes $\bar{\mathcal{D}}^\circ_j$ as follows.

Each parent coarsen-LSS-node $y^\square_{c_\uparrow} \in \bar{\mathcal{D}}^\square_{c_\uparrow}$ is associated with the ground state set equal to the *union* of its children LSS-nodes' ground state sets

$$\mathrm{gr}(y^\square_{c_\uparrow}) \;=\; \cup_{y^\circ_j \in \bar{\mathcal{N}}_\downarrow(y^\square_{c_\uparrow})} \mathrm{gr}(y^\circ_j),$$

To ensure that each ground state is covered by exactly one parent LSS-node, each child LSS-node $y^\circ_j \in \bar{\mathcal{D}}^\circ_j$ must have exactly one parent LSS-node, i.e., $|\bar{\mathcal{N}}_\uparrow(y^\circ_j)| = 1$.

6.3.3.2 Joining Operation

A joining operation is represented in the CJ-Tree by a join-node $j \in D^\circ$ and its children $N_\downarrow(j) \subset \bar{\mathcal{D}}^\square$ coarsen-nodes (e.g., the nodes boxed in dark blue in Figure 6.2). The details of the joining operation are represented on the CJ-LSS-Tree by a set of parent join-LSS-nodes $\bar{\mathcal{D}}^\circ_j$ and children coarsen-LSS-node sets $\{\bar{\mathcal{D}}^\square_{c_\downarrow} : c_\downarrow \in N_\downarrow(j)\}$ as follows.

The ground state set of each parent LSS-node $y^\circ_j \in \bar{\mathcal{D}}^\circ_j$ is equal to the *Cartesian product* of its children LSS-nodes' ground state sets

$$\mathrm{gr}(y^\circ_j) \;=\; \times_{c_\downarrow \in N_\downarrow(j)} \mathrm{gr}(y^\square_{j;c_\downarrow}),$$

Collectively, the parent LSS-nodes represent a grid of hypercubes $\mathrm{gr}(\bar{\mathcal{D}}_j^\circ) = \times_{y_{c_\downarrow}^\square \in \bar{\mathcal{N}}_\downarrow(y_j^\circ)} \mathrm{gr}(\bar{\mathcal{D}}_{c_\downarrow}^\square)$ that partitions the joint state space $\mathcal{D}_{\mathrm{sc}(j)}$ associated with join-node j.

☐ 6.3.3.3 Example: Recursive State Set Definition

This section illustrates the recursive coarsening and joining operations with an example shown in Figure 6.2; the left pane (Figure 6.2a) shows a CJ-Tree and the right pane (Figure 6.2b) shows a CJ-LSS-Tree.

Select coarsening and joining operations are highlighted with colored boxes in the CJ-Tree; the details of how each operation relates groups of states is represented on the subgraph of the CJ-LSS-Tree boxed in the same color. For compactness, each block of join-LSS-nodes $\bar{\mathcal{D}}_j^\circ$ is laid out linearly, rather than as a $|N_\downarrow(j)|$-dimensional table (which would clearly illustrate that it represents all combinations of its children nodes).

Select nodes in the CJ-LSS-Tree are labeled with their associated ground state set. For compactness, we drop the scope symbol associated with non-leaf nodes of the CJ-LSS-Tree; for example, the label $\{0, 1\}$ of the coarsen-LSS-node in the dark-red box represents state set $\{(A; 0), (A; 1)\}$.

An alternating sequence of coarsening and joining operations in this example is represented as follows:

Coarsen at dark-red box: At the bottom level, the dark-red box represents a coarsening operation associated with coarsen-node A_0^1 and its child leaf node $N_\downarrow(A_0^1) = \{A_0\}$. In the right pane, the top row of coarsen-LSS-nodes represents a coarse partition $\mathrm{gr}(A_0^1) = \{\{0, 1\}, \{2\}\}$ of the ground domain $\mathcal{D}_A = \{(A; 0), (A; 1), (A; 2)\}$ associated with the leaf node A_0 (i.e., the coarsening operation treats states 0 and 1 identically).

Join at dark-blue box: At the next level, the dark-blue box represents a joining operation

associated with join-node AB_1 and its children $N_{\downarrow}(AB_1) = \{A_0^1,\ B_0^1\}$. In the right pane, the top row of join-LSS-nodes is associated with a partition $\text{gr}(AB_1) = \text{gr}(A_0^1) \times \text{gr}(B_0^1) = \{\{0,1\},\{2\}\} \times \{\{0,1\},\{2\}\}$ of the ground state set $\mathcal{D}_{AB} = \mathcal{D}_A \times \mathcal{D}_B$ associated with the parent's scope $\text{sc}(AB_1) = \{A, B\}$.

Coarsen at dark-red box: At the next level, the light-red box represents a coarsening operation associated with coarsen-node AB_1^2 and its child $N_{\downarrow}(AB_1^2) = \{AB_1\}$. In the right pane, the top row of coarsen-LSS-nodes is associated with a partition $\text{gr}(AB_1) = \{\{0,1\} \times \{0,1\} \cup \{2\} \times \{2\},\ \{0,1\} \times \{2\} \cup \{2\} \times \{0,1\}\}$ of the ground state set $\mathcal{D}_{AB} = \mathcal{D}_A \times \mathcal{D}_B$ associated with the parent's scope $\text{sc}(AB_1^2) = \{A, B\}$. Note the first state set in the top row merges state sets of the first and fourth LSS-nodes in the bottom row while the second state set merges state sets of the second and third LSS-nodes in the bottom row, i.e., their children in the CJ-LSS-Tree.

Join at light-blue box: At the next level, the light-blue box represents a joining operation associated with join-node AB_2 and its children $N_{\downarrow}(AB_2) = \{AB_1^2,\ CD_1^2\}$. In the right pane, the top row of join-LSS-nodes is associated with four state sets that partition the ground state set $\mathcal{D}_{ABCD} = \mathcal{D}_A \times \mathcal{D}_B \times \mathcal{D}_C \times \mathcal{D}_D$ associated with the parent's scope $\text{sc}(ABCD_2) = \{A, B, C, D\}$. For compactness, only one ground state set is labeled $\text{gr}(y^{\square}_{ABCD_2}) = \Big(\{0,1\} \times \{0,1\} \cup \{2\} \times \{2\}\Big) \times \Big(\{0,1\} \times \{2\} \cup \{2\} \times \{0,1\}\Big)$. Note that this represents a fairly complex grouping of states.

Note that this recursive structure efficiently represents a partition of an exponentially large number of states. For example, the top node represents a partition of the joint state space $\mathcal{D}_A \times \mathcal{D}_B \times \mathcal{D}_C \times \mathcal{D}_D$ into four groups. However, instead of explicitly enumerating the joint state space of size $3 \cdot 3 \cdot 3 \cdot 4 = 108$, the partition is represented recursively on the CJ-LSS-Tree. In this example, the CJ-LSS-Tree only uses 37 nodes. Clearly, the potential savings of the compact representation is much larger for larger problems.

6.3.4 Inference on the CJ-LSS-Tree

The reparameterization and inference operations on the CJ-LSS-Tree take an intuitive form which we discuss in this section.

6.3.4.1 Reparameterization

For a CJ-LSS-Tree, the reparameterization equation on a generic LSS-Graph (Eq. (6.2)) can be written in a more intuitive way by separating parent and child neighbors in the summation. That is, for a join-node $j \in D^\circ$ and associated join-LSS-node $y_j^\circ \in \bar{\mathcal{D}}_j^\circ$

$$\begin{aligned}\theta_j^\delta(x_j) \;&=\; \theta_j(x_j) \;+\; \sum_{c_\uparrow \in N_\uparrow(j)} \delta_{c_\uparrow}(x_{c_\uparrow}) \;+\; \sum_{c_\downarrow \in N_\downarrow(j)} \delta_{c_\downarrow}(x_{c_\downarrow}) \\ &=\; \bar{\theta}_j(y_j^\circ) \;+\; \sum_{c_\uparrow \in N_\uparrow(j)} \bar{\delta}_{c_\uparrow}(y_{c_\uparrow}^\square) \;+\; \sum_{c_\downarrow \in N_\downarrow(j)} \bar{\delta}_{c_\downarrow}(y_{c_\downarrow}^\square) \;=\; \bar{\theta}_j^{\bar{\delta}}(y_j^\circ) \end{aligned} \tag{6.3}$$

for any ground state $x_j \in \mathrm{gr}(y_j^\circ)$, where $\{y_{c_\uparrow}^\square\} = \bar{\mathcal{N}}_\uparrow(y_j^\circ)$ is the parent LSS-node (or empty if at root), and $\{y_{c_\downarrow}^\square\} = \bar{\mathcal{N}}_\downarrow(y_c^\square) \cap \bar{\mathcal{D}}_{c_\downarrow}^\square$ is the child LSS-node associated with child $c_\downarrow \in N_\downarrow(j)$ (or empty if leaf).

The first term of Eq. (6.3) is a symmetric model potential; the second term is a parent factor at a coarser resolution (which assigns identical value to many states at this (finer) level); the third summation over children factors aggregates one-dimensional factors associated with each dimension of the factor table.

Example As an example to illustrate the reparameterization equation (Eq. 6.3), consider the join-node AB_1, its parent $N_\uparrow(AB_1) = \{AB_1^2\}$, and its two children, $N_\downarrow(AB_1) = \{A_0^1,\ B_0^1\}$ in Figure 6.2. Let the model LSS-factor be $\bar{\theta}_{AB_1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, and let its parent cost-shifting LSS-factor be $\bar{\delta}_{AB_1^2} = [i,\ j]$, and its children cost-shifting LSS-factors be $\bar{\delta}_{A_0^1} = [e,\ f]$ and $\bar{\delta}_{B_0^1} = [g,\ h]$.

We then have the reparameterized LSS-factor from Eq. (6.3) as

$$\bar{\theta}^{\bar{\delta}}_{AB_1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} i & j \\ j & i \end{bmatrix} + \left(\begin{bmatrix} e & e \\ f & f \end{bmatrix} + \begin{bmatrix} g & h \\ g & h \end{bmatrix} \right)$$

where the parent and child LSS-factors are expanded to the larger LSS-factor associated with AB_1^2 based on the neighborhood in the LSS-Graph. Specifically,

Parent (Coarsen) term: The second term $\begin{bmatrix} i & j \\ j & i \end{bmatrix}$ results from spreading the values $\bar{\delta}_{AB_1^2} = [i,\ j]$ to indices associated with the children LSS-nodes as follows. The first and fourth LSS-nodes at AB_1 are children of the first LSS-node at AB_1^2 (associated with value i) and the second and third LSS-nodes at AB_1 are children of the second LSS-node at AB_1^2 (associated with value j).

Child (Join) term: The third term $\begin{bmatrix} e & e \\ f & f \end{bmatrix}$ results from spreading the values $\bar{\delta}_{A_0^1} = [e,\ f]$ across the second dimension (since the coarsen-LSS-nodes of A_0^1 are indexed on the first dimension). The fourth term $\begin{bmatrix} g & h \\ g & h \end{bmatrix}$ results from spreading the values $\bar{\delta}_{B_0^1} = [g,\ h]$ across the first dimension (since the coarsen-LSS-nodes of B_0^1 are indexed on the second dimension).

☐ 6.3.4.2 LSS-Inference Objective

The ground objective (first line) using reparameterized ground factors can now be written as a *LSS-objective* (second line) using reparameterized LSS-factors

$$\begin{aligned} L(\boldsymbol{\delta}) \ &= \ \sum_{j \in D^\circ} \max_{x_j \in \mathcal{D}_j} \ \theta_j^{\delta}(x_j) \\ &= \ \sum_{j \in D^\circ} \max_{y_j^\circ \in \bar{\mathcal{D}}_j^\circ} \ \bar{\theta}_j^{\bar{\delta}}(y_j) \ =: \ \bar{L}(\bar{\boldsymbol{\delta}}) \end{aligned}$$

where the LSS-objective is computed on the LSS-factors, and the parameters $\bar{\boldsymbol{\delta}}$ are associated with LSS-factors. In other words, this equation exploits the symmetry of the problem and hence admits an efficient computation.

6.3.5 Relation to Previous Work

The class of hierarchical state space models developed in this section bears an interesting relationship to two previous lines of work. First, as discussed in the next subsection, our hierarchical state space partitions can represent the recursive cardinality models used by Tarlow et al. [2012] as a special case. Another parallel in the literature is to Sum-Product Networks (SPNs) [Poon and Domingos, 2011], which represent a hierarchical mixture model defined via an alternating sequence of sum and product nodes. Loosely speaking, the sum nodes constitute an analogue to our coarsening nodes, and the product nodes an analogue to our joining nodes. However, although both result in a similar computational structure, the models they are used to define are quite different: our approach is used to represent a standard, exponential family graphical model, while the SPN framework can be used to represent a more general class of mixture models.

More significantly, however, our work differs from these two previous works in our final intention. In the two works, the model is learned under the restriction that it has a hierarchical tree-structure; thus, the computational structure needed for efficient inference dictates the model structure. In contrast, in our work, we make no assumptions on model structure and our CJ-LSS-Tree is used to define symmetry in the class of approximate inference algorithms (as the factors of a hierarchical region graph), and moreover, is built adaptively (see Section 6.5) to the inference problem at hand rather than being specified *a priori*.

6.3.5.1 Recursive Counting Factor

In this section, we show how to represent nested cardinality potentials (e.g., Tarlow et al. [2012]) with the coarsening and joining operations developed in the preceding subsections. For simplicity, we assume that the CJ-Tree forms a balanced binary partition of the ground RVs, i.e., each join node has two children.

Each coarsen-node $c \in D^{\square}$ represents an aggregate count over its scope, while each join-node $j \in D^{\circ}$ represents a *joint count* over its childrens' scopes. Collectively,

$$\begin{aligned}\bar{\mathcal{D}}_c^{\square} &= \{y_c^{\square\#k} : k \in [|\mathrm{sc}(c)|]\} \\ \bar{\mathcal{D}}_j^{\circ} &= \{y_j^{\circ\#ab} : a \in [|\mathrm{sc}(c)|],\ b \in [|\mathrm{sc}(c')|]\}\end{aligned}$$

where $N_{\downarrow}(j) = \{c, c'\}$ are the two children of join-node j. Each individual LSS-node corresponds to a ground set representing aggregate counts and joint counts, respectively, as

$$\begin{aligned}\mathrm{gr}(y_c^{\square\#k}) &= \{x_c \in \mathcal{D}_{\mathrm{sc}(c)} : \sum_{v \in \mathrm{sc}(c)} [x_v = 1] = k\} \\ \mathrm{gr}(y_j^{\circ\#ab}) &= \{x_j \in \mathcal{D}_{\mathrm{sc}(j)} : \sum_{v \in \mathrm{sc}(c)} [x_v = 1] = a,\ \sum_{v \in \mathrm{sc}(c')} [x_v = 1] = b\}\end{aligned}$$

where the first is the set of ground states with k `on` RVs, the second is the set of ground states with a `on` RVs in the set $\mathrm{sc}(c)$ and b `on` RVs in the set $\mathrm{sc}(c')$.

The coarsening and joining operations are represented as follows:

Coarsening: The coarsening operation merging sets corresponding to joint counts with the same aggregate count. That is, the coarsen-LSS-node $y_c^{\square\#k}$ has children

$$\bar{\mathcal{N}}_{\downarrow}(y_c^{\square\#k}) = \{y_j^{\circ\#ab} \in \bar{\mathcal{D}}_j^{\circ} : a + b = k\} \tag{6.4}$$

and hence corresponds to the aggregate-count ground set

$$\mathrm{gr}(y_c^{\square\#k}) \;=\; \bigcup_{\substack{a,b \\ a+b=k}} \mathrm{gr}(y_j^{\circ\#ab})$$

which says that the set of states with k `on` RVs in $\mathrm{sc}(j)$ can be formed as the union of all state sets with a `on` RVs in $\mathrm{sc}(c)$ and with $b = k - a$ `on` RVs in $\mathrm{sc}(c')$, for all a.

Joining: The joining operation combines aggregate counts to create a joint count. That is,

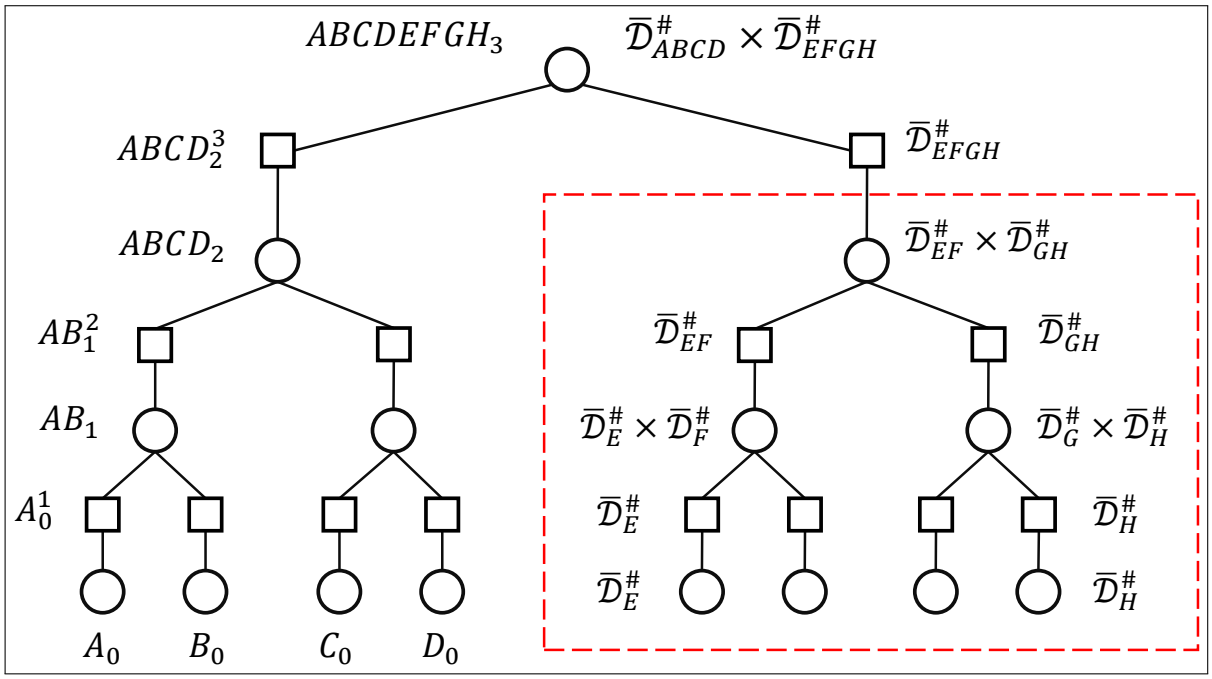
$$\bar{\mathcal{N}}_{\downarrow}(y_j^{\circ\#ab}) = \{y_c^{\square\#a},\; y_{c'}^{\square\#b}\}$$

which corresponds to ground sets

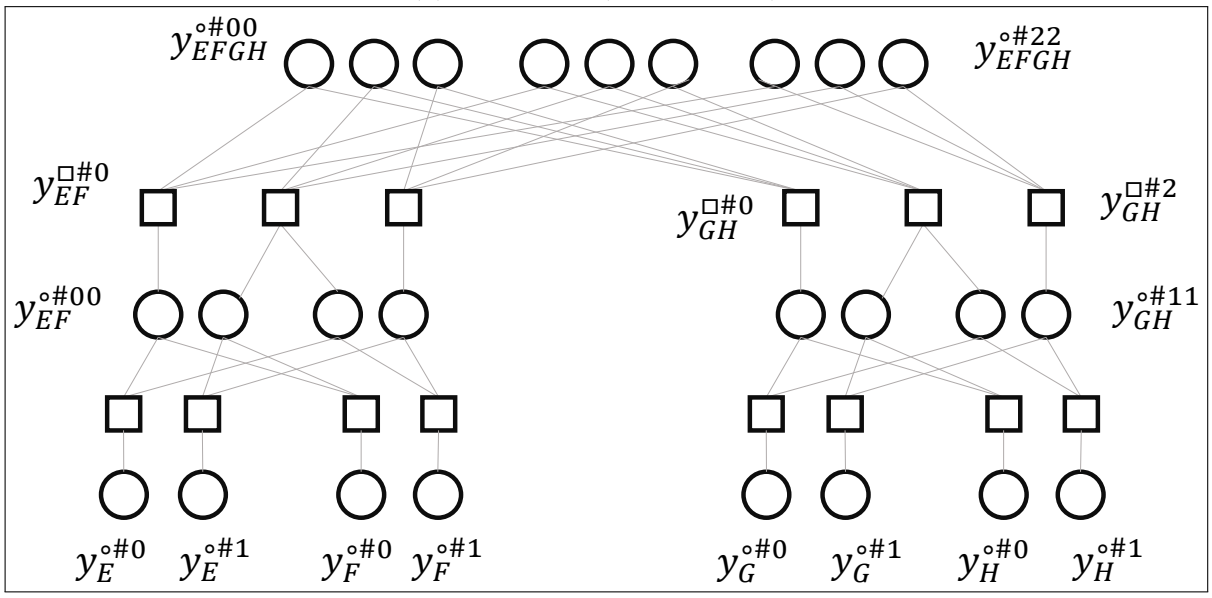
$$\mathrm{gr}(y_j^{\circ\#ab}) \;=\; \mathrm{gr}(y_c^{\square\#a}) \;\times\; \mathrm{gr}(y_{c'}^{\square\#b}).$$

Figure 6.3 shows an example of a CJ-Tree associated with counting partitions. The top pane (Figure 6.3a) shows a CJ-Tree on eight RVs $\{A, B, C, D, E, F, G, H\}$ where RVs are recursively grouped by alphabetic order. Select nodes in the CJ-Tree (top pane) on the left are labeled with a scope and level index; select nodes on the right are labeled with their associated counting state space partition. The bottom pane (Figure 6.3b) shows a subgraph of the CJ-LSS-Tree, corresponding to the subgraph of the CJ-Tree boxed in red. This pane illustrates the relationship between state sets of the counting partition. The neighborhood structure illustrates Eq. (6.4); that is, each coarsen-LSS-node is connected to join-LSS-nodes with equal aggregate count.

Reparameterized LSS-factor The reparameterization operation combines counting functions at multiple resolutions. That is, for any join-node $j \in D^{\circ}$ with children $N_{\downarrow}(j) = \{c_{\downarrow}, c'_{\downarrow}\}$ in the CJ-Tree, and any $y_j^{\circ\#ab} \in \bar{\mathcal{D}}_j^{\circ}$ which represents ground states with a `on` RVs in $c_{\downarrow}$ and



(a) CJ-Tree T (region graph)



(b) CJ-LSS-Tree $\bar{\mathcal{T}}$ (LSS-Graph)

Figure 6.3: (a) A CJ-Tree T associated with recursive cardinality state space partitions. Select nodes (on the left) are labeled with their scope and level index. Select nodes (on the right) are labeled with their associated state space partition; each coarsen-node (square) is a counting partition over its scope each join-node (circle) is a joint counting partition over its childrens' scopes. (b) A subgraph of an associated CJ-LSS-Tree $\bar{\mathcal{T}}$ that shows state groups, corresponding to subgraph boxed in red in the top pane.

b on RVs in $c'_{\downarrow}$, the reparameterization (Eq. (6.3)) is

$$\bar{\theta}_j^{\bar{\delta}}(y_j^{\circ \# ab}) \;=\; \bar{\theta}_j(y_j^{\circ \# ab}) \;+\; \bar{\delta}_{c_\uparrow}(y_{c_\uparrow}^{\square \#(a+b)}) \;+\; \left(\bar{\delta}_{c_\downarrow}(y_{c_\downarrow}^{\square \# a}) \;+\; \bar{\delta}_{c'_\downarrow}(y_{c'_\downarrow}^{\square \# b})\right)$$

where the first term is a model factor, the second term represents a factor with aggregate count $a+b$ on RVs over the set $\mathrm{sc}(c_\downarrow) \cup \mathrm{sc}(c'_\downarrow)$, while the third and fourth represent univariate counts with a on RVs in $c_\downarrow$ and b on RVs in $c'_\downarrow$, respectively.

6.4 Loopy Hierarchical Region Graph

In this section, we develop a hierarchical region graph that coherently reasons about factors of varying scope size and state space resolution that are connected in a general (loopy) topology. By controlling the state space resolution we can reason about factors of large scope with controlled cost; by allowing a loopy topology, we are able to represent realistic inference problems. A hierarchical region graph can be viewed as a generalization of two separate inference structures that we have seen so far:

Hierarchical tree model: From the point of view of CJ-Tree models discussed in Section 6.3, we augment the CJ-Tree graph with factors that relate terms from different branches of the tree. For example, we can handle a structure where the tree in Figure 6.2a is augmented with a factor $(A, D)_0$ linked to nodes A_0 and D_0 from different branches of the tree; this forms a loop and breaks the tree structure.

Univariate domain symmetry: Recall that our motivating example in Section 6.2.1, used univariate domain symmetry and two levels of resolution. A hierarchical region graph substitutes the univariate domain partitions for the more expressive multivariate partitions defined on a CJ-Tree via recursive coarsening and joining operations. Furthermore, a hierarchical region graph incorporates more than two levels of coarsening.

The main difficulty in constructing the hierarchical region graph is ensuring that inference operations do not shatter factor symmetry. This is facilitated by forcing each factor in the graph to be composed of symmetry groups defined by a subset of nodes, called its *CJ-scope*, in a CJ-Tree associated with the hierarchical region graph. Factors defined in this way can exchange symmetric messages with factors at the same *resolution* and can exchange symmetric messages with factors at adjacent *resolutions* (the notion of *resolution* is defined in the next section).

6.4.1 Hierarchical Region Graph

A *hierarchical region graph* $G = (R, S, E)$ is a region graph whose separators $S = S_{\updownarrow} \cup S_{\leftrightarrow}$ and edges $E = E_{\updownarrow} \cup E_{\leftrightarrow}$ are divided into *intra-resolution* sets, $S_{\leftrightarrow}$ and $E_{\leftrightarrow}$, that shift cost between regions at the same resolution, and *inter-resolution* sets, $S_{\updownarrow}$ and $E_{\updownarrow}$, that shift cost between regions at adjacent resolutions. The region graph is associated with a CJ-Tree $T = (D^{\circ}, D^{\square}, \dot{E})$ that is used to define the "resolution" of each node in the region graph.

CJ-scope The "resolution" of a region or separator node is defined via a *Coarsen-Join-scope (CJ-scope)* $\gamma \subset D^{\circ} \cup D^{\square}$ associated with non-overlapping sets of RVs (i.e., for all pairs $\{a, a'\} \subset \gamma$, $\mathrm{sc}(a) \cap \mathrm{sc}(a') = \emptyset$; or, equivalently, a is not a descendant or ancestor of a' in the CJ-Tree). The region or separator node's scope is defined indirectly as the union of scopes of nodes in its CJ-scope, i.e., $\mathrm{sc}(\eta) = \cup_{a \in \gamma} \mathrm{sc}(a)$.

The CJ-scope of each region or separator node consists of all join-nodes or all coarsen-nodes as follows: each region node $r \in R$ is associated with a CJ-scope $\mathrm{sc}^{\circ}(r) \subset D^{\circ}$ consisting of join-nodes; each inter-resolution separator $s_{\updownarrow} \in S_{\updownarrow}$ is associated with a CJ-scope $\mathrm{sc}^{\square}(s_{\updownarrow}) \subset D^{\square}$ consisting of coarsen-nodes; each intra-resolution separator $s_{\leftrightarrow} \in S_{\leftrightarrow}$ is associated with a CJ-scope $\mathrm{sc}^{\square}(s_{\updownarrow}) \subset D^{\circ}$ consisting of join-nodes.

6.4.1.1 Neighborhood Relationship

Each region node is connected to factors at the same resolution and at adjacent (coarser and finer) resolutions. The region node is related to its neighbors by a set of coarsening and joining operations applied to each dimension of the factor table.

The neighbors of each region node $r \in R$ are divided as $N(r) = N_{\leftrightarrow}(r) \cup \big(N_{\uparrow}(r) \cup N_{\downarrow}(r)\big)$ where $N_{\leftrightarrow}(r) \subset S_{\leftrightarrow}$ are intra-resolution separators, $N_{\uparrow}(r) \subset S_{\updownarrow}$ is a *parent* intra-resolution separator, and $N_{\downarrow}(r) \subset S_{\updownarrow}$ are *child* intra-resolution separators. The region node is related to separator nodes of each of these sets as follows:

Coarsening (parent): A parent intra-resolution separator $s_{\uparrow}$ has exactly one child $N_{\downarrow}(s_{\uparrow}) = \{r\}$. The parent separator's CJ-scope is equal to the union of (one) parent of each element of the child separator's CJ-scope, $\mathrm{sc}^{\square}(s_{\uparrow}) = \cup_{j \in \mathrm{sc}^{\circ}(r)} N_{\uparrow}(j)$. This means that each dimension of the parent separator factor is coarser than each dimension of the child region factor.

Joining (children): A parent region node $r \in R$ has multiple children $N_{\downarrow}(r)$. The parent's CJ-scope is $\mathrm{sc}^{\circ}(r) = \cup_{s_{\downarrow} \in N_{\downarrow}(r)} \cup_{c_{\downarrow} \in \mathrm{sc}^{\square}(s_{\downarrow})} N_{\uparrow}(c_{\downarrow})$; the first union ranges over all child separators and the second union ranges over the CJ-scope of the child (consisting of coarsen nodes). As we will see, this has the effect of adding coarse factors associated with the children together, to produce a parent with scope large enough to contain the sum.

Lower-dimension (intra-resolution): An intra-resolution separator $s_{\leftrightarrow} \in S_{\leftrightarrow}$ links region node $r \in R$ and region node $r' \in R$ and has CJ-scope equal to the intersection of its incident regions' CJ-scopes, i.e., $\mathrm{sc}^{\circ}(s) = \mathrm{sc}^{\circ}(r) \cap \mathrm{sc}^{\circ}(r')$. The region nodes shift cost over this lower dimensional space.

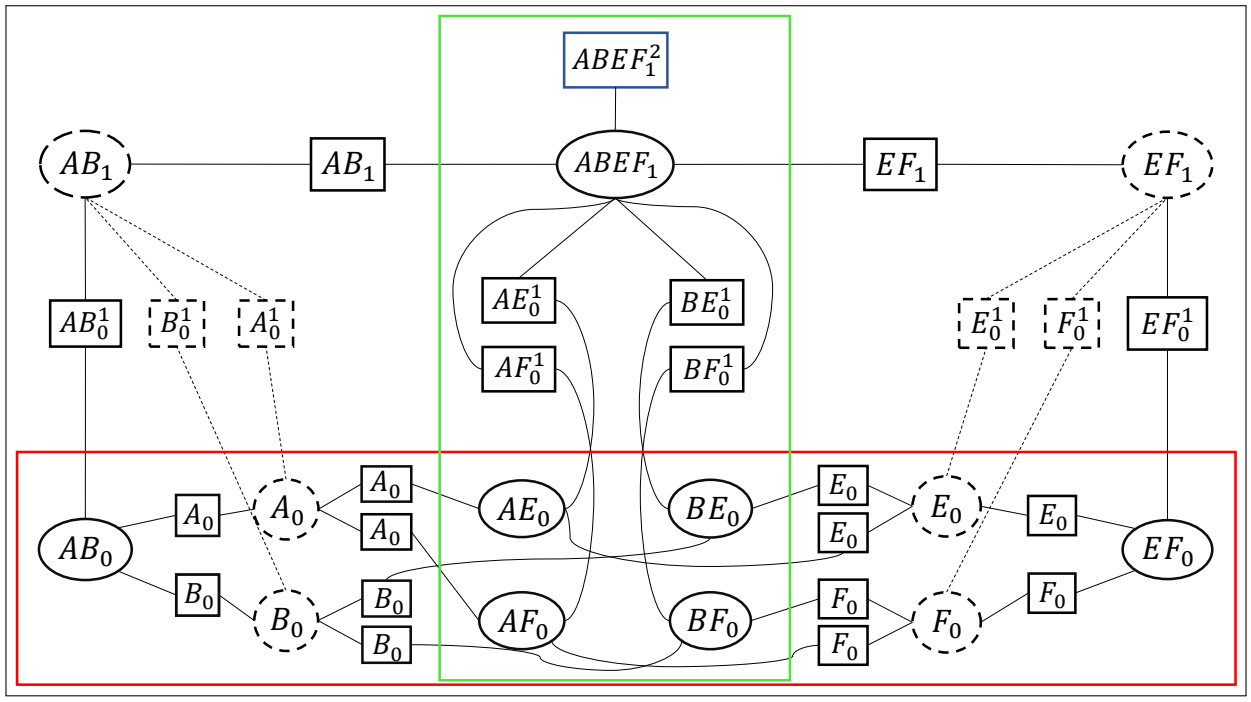


Figure 6.4: A subgraph of of hierarchical region graph G corresponding to an original model that is a complete PGM. Nodes are labeled with their CJ-scope and a level index. Several important subgraphs of the graph are marked: (i) The subgraph corresponding to the CJ-Tree T is shown with dashed nodes and edges. (ii) Boxed in green is a subgraph consisting of regions and intra-resolution separators; this corresponds to coarsening and joining operations on each dimension of the MD-LSS-factor table (e.g., the parent of factor $AE_0 := (A_0, E_0)$ is coarsened on each dimension yielding $AE_0^1 := (A_0^1, E_0^1)$). (iii) Boxed in red, in the bottom row, is an intra-resolution subgraph of regions that share common CJ-scope; a pair of regions shift cost over the intersecting elements of their CJ-scopes.

Example We now give an example to illustrate a hierarchical region graph and the terms defined above for a graphical model whose factors are all pairs of RVs where the set of RVs is $\mathcal{V} = \{A, B, C, D, E, F, G, H\}$. The hierarchical region graph is associated with a CJ-Tree where RVs are recursively grouped by alphabetic order, i.e., the form as Figure 6.3a, but, in this example, we don't require that its nodes are associated with counting factors.

Figure 6.4 shows a subgraph of the hierarchical region graph consisting of all pairwise factors on RVs $\{A, B, E, F\}$. The CJ-Tree is a subgraph of the hierarchical region graph, depicted by the dashed nodes and edges (the nodes corresponding to EF_1 and its children, and the nodes corresponding to AB_1 and its children). At the bottom of the graph, intra-level terms

associated with model factors are boxed in red. Shown in the green box is a set of inter-resolution separators connecting regions from each level. The CJ-scope of each region or separator consists of nodes in the CJ-Tree at the same level (at the same vertical height in the figure). For example, the CJ-scope of the node in the top level labeled $(ABEF)_1$ is (AB_1, EF_1) and each of these nodes is a dashed node in the top level, the CJ-scope of the node in between levels labeled $(AE)_0^1$ is (A_0^1, E_0^1) and each of these nodes is a dashed node in between levels.

6.4.2 State Space Symmetry in Factor Tables

In this section, we define a *multidimensional LSS-factor (MD-LSS-factor)* composed from symmetry groups represented on a CJ-LSS-Tree $\bar{\mathcal{T}} = (\bar{\mathcal{D}}^\circ, \bar{\mathcal{D}}^\square, \bar{\dot{\mathcal{E}}})$ (corresponding to the CJ-Tree T used to define factors' CJ-scopes).

6.4.2.1 MD-LSS-factor

A *multidimensional LSS-factor (MD-LSS-factor)* $\bar{\theta}_\gamma : \bar{\mathcal{D}}_\gamma^\circ \to \mathbb{R}$ with *CJ-scope* $\text{sc}^\circ(\bar{\theta}_\gamma) = \gamma$ assigns has domain equal to all combinations of LSS-nodes associated with its CJ-scope $\bar{\mathcal{D}}_\gamma^\circ := \times_{j\in\gamma}\bar{\mathcal{D}}_j^\circ$. An element of its domain is denoted $(y_j^\circ)_{j\in\gamma} =: y_\gamma^\circ \in \bar{\mathcal{D}}_\gamma^\circ$ and is associated with all ground states whose substates are in the ground state set of each individual index, $\text{gr}(y_\gamma^\circ) = \times_{j\in\gamma}\text{gr}(y_j^\circ)$. The MD-LSS-factor defines a ground factor with these symmetric groups of states, i.e., $\bar{\theta}_\gamma(y_\gamma^\circ) = \theta_\gamma(x_\gamma)$ for all $x_\gamma \in \text{gr}(y_\gamma^\circ)$ and each $y_\gamma^\circ \in \bar{\mathcal{D}}_\gamma^\circ$.

A MD-LSS-factor for a CJ-scope using coarsen-nodes $\sigma \subset D^\square$ is similarly defined (details are not shown for compactness).

Region Graph Terms Each region or separator node is defined as a MD-LSS-factor associated with its CJ-scope as follows: each region $r \in R$ is associated with a model MD-LSS-factor $\bar{\theta}_r : \bar{\mathcal{D}}_r^\circ \to \mathbb{R}$ and CJ-scope $\text{sc}^\circ(r) := \text{sc}^\circ(\bar{\theta}_r) \subset D^\circ$, each intra-resolution separator

$s_{\leftrightarrow} \in S_{\leftrightarrow}$ is associated with a cost-shifting MD-LSS-factor $\bar{\delta}_{s_{\leftrightarrow}} : \bar{\mathcal{D}}^{\circ}_{s_{\leftrightarrow}} \to \mathbb{R}$, and each inter-resolution separator $s_{\updownarrow} \in S_{\updownarrow}$ is associated with a cost-shifting MD-LSS-factor $\bar{\delta}_{s_{\updownarrow}} : \bar{\mathcal{D}}^{\square}_{s_{\updownarrow}} \to \mathbb{R}$.

Note that we abused notation to allow graph nodes to act as domain subscripts (i.e., $\bar{\mathcal{D}}^{\square}_{r} := \bar{\mathcal{D}}^{\circ}_{\mathrm{sc}^{\circ}(r)}$, $\bar{\mathcal{D}}^{\square}_{s_{\leftrightarrow}} := \bar{\mathcal{D}}^{\square}_{\mathrm{sc}^{\square}(s_{\leftrightarrow})}$, and $\bar{\mathcal{D}}^{\square}_{s_{\updownarrow}} := \bar{\mathcal{D}}^{\square}_{\mathrm{sc}^{\square}(s_{\updownarrow})}$) and to act as index subscripts (i.e., $y^{\circ}_{r} = y^{\circ}_{\mathrm{sc}^{\circ}(r)}$, $y^{\circ}_{s_{\leftrightarrow}} = y^{\circ}_{\mathrm{sc}^{\circ}(s_{\leftrightarrow})}$, and $y^{\square}_{s_{\updownarrow}} = y^{\square}_{\mathrm{sc}^{\square}(s_{\updownarrow})}$).

6.4.3 Lifted Reparameterization

The alignment of state space symmetry groups in the CJ-LSS-Tree implies alignment of state space symmetry groups comprising each dimension of the MD-LSS-factors; that is, cost-shifting operations do not shatter the state space symmetry of region factors.

For any region node $r \in R$, the ground reparameterization equation (Eq. (2.5)) has an equivalent lifted form as

$$\begin{aligned}\theta^{\delta}_{r}(x_r) &= \theta_r(x_r) + \sum_{s_{\uparrow} \in N_{\uparrow}(r)} \delta_{s_{\uparrow}}(x_{s_{\uparrow}}) + \sum_{s_{\downarrow} \in N_{\downarrow}(r)} \delta_{s_{\downarrow}}(x_{s_{\downarrow}}) + \sum_{s_{\leftrightarrow} \in N_{\leftrightarrow}(r)} \delta_{s_{\leftrightarrow}}(x_{s_{\leftrightarrow}}) \\ &= \bar{\theta}_r(y^{\circ}_{r}) + \sum_{s_{\uparrow} \in N_{\uparrow}(r)} \bar{\delta}_{s_{\uparrow}}(y^{\square}_{s_{\uparrow}}) + \sum_{s_{\downarrow} \in N_{\downarrow}(r)} \bar{\delta}_{s_{\downarrow}}(y^{\square}_{s_{\downarrow}}) + \sum_{s_{\leftrightarrow} \in N_{\leftrightarrow}(r)} \bar{\delta}_{s_{\leftrightarrow}}(y^{\circ}_{s_{\leftrightarrow}}) \;=:\; \bar{\theta}^{\bar{\delta}}_{r}(y^{\circ}_{r}) \end{aligned} \tag{6.5}$$

for a joint index $y^{\circ}_{r} \in \bar{\mathcal{D}}^{\circ}_{r}$ in the region LSS-factor domain, and any associated ground state $x_r \in \mathrm{gr}(y^{\circ}_{r})$. The indexing of the terms is defined as follows:

Model: The first term is the model factor indexed with y°_{r}.

Coarsening (parent): Recall that a parent intra-resolution separator $s_{\uparrow}$, (i.e., $N_{\uparrow}(r) = \{s_{\uparrow}\}$) has coarser resolution CJ-scope, i.e., $\mathrm{sc}^{\square}(s_{\uparrow}) = \cup_{j \in \mathrm{sc}^{\circ}(r)} N_{\uparrow}(j)$.

The index $y^{\square}_{s_{\uparrow}}$ is a vector of parents of *each* join-LSS-node element of y°_{r}, i.e., $y^{\square}_{s_{\uparrow}} = (y^{\square}_{s_{\uparrow};j})_{j \in \mathrm{sc}^{\circ}(r)}$ where $\{y^{\square}_{s_{\uparrow};j}\} = \bar{\mathcal{N}}_{\uparrow}(y^{\circ}_{j})$ is the (one) parent LSS-node for the join-node (acting as a dimension) j, for each $j \in \mathrm{sc}^{\circ}(r)$ in the CJ-scope of r.

Joining (children): Recall that each child $s_\downarrow \in N_\downarrow(r)$ of a parent region node $r \in R$ has same coarsening resolution, but smaller dimension.

The index $y^\square_{s_\downarrow}$ is a vector of child coarsen-LSS-nodes associated with $\text{sc}^\square(s_\downarrow)$, i.e., $y^\square_{s_\downarrow} = (y^\square_{s_\downarrow;c_\downarrow})_{c_\downarrow \in \text{sc}^\square(s_\downarrow)}$ where $\{y^\square_{s_\downarrow;c_\downarrow}\} = \bar{\mathcal{N}}_\downarrow(y^\circ_r) \cap \bar{\mathcal{D}}^\square_{c_\downarrow}$ is the child LSS-node associated with the coarsen-node (acting as a dimension) $c_\downarrow$, for each $c_\downarrow \in \text{sc}^\square(s_\downarrow)$ in the CJ-scope of $s_\downarrow$.

Lower-dimension (intra-resolution): Recall that the CJ-scope of an intra-resolution separator $s_\leftrightarrow \in S_\leftrightarrow$ is a subset of the CJ-scope of the region node r, i.e., $\text{sc}^\circ(s_\leftrightarrow) \subseteq \text{sc}^\circ(r)$. The index $y^\circ_{s_\leftrightarrow} = (y^\circ_j)_{j \in \text{sc}^\circ(s_\leftrightarrow)}$ indexes a lower dimensional subspace.

Example Consider the graph shown in Figure 6.4 where MD-LSS-factors are defined by a basis of recursive cardinality potentials (the CJ-Tree representing this basis corresponds to the CJ-Tree used in our example in Figure 6.3a). This CJ-Tree corresponds to a subgraph (depicted with dashed nodes and edges) of Figure 6.4. The lifted reparameterization (Eq. 6.5) around the region node labeled $ABEF_1$ is

$$\begin{aligned}
\bar{\theta}^{\bar{\delta}}_{AB,EF}(y^{\circ\#ab}_{AB},\, y^{\circ\#ef}_{EF}) \;=\;& \bar{\theta}_{AB,EF}(y^{\circ\#ab}_{AB},\, y^{\circ\#ef}_{EF}) && \text{(6.6a)}\\
&+ \Big[\bar{\delta}_{AB,EF}(y^{\square\#a+b}_{AB},\, y^{\square\#e+f}_{EF})\Big] && \text{(6.6b)}\\
&+ \Big[\bar{\delta}_{A,E}(y^{\square\#a}_{A},\, y^{\square\#e}_{E}) \;+\; \bar{\delta}_{B,E}(y^{\square\#b}_{B},\, y^{\square\#e}_{E})\\
&\quad \bar{\delta}_{A,F}(y^{\square\#a}_{A},\, y^{\square\#f}_{F}) \;+\; \bar{\delta}_{B,F}(y^{\square\#b}_{B},\, y^{\square\#f}_{F})\Big] && \text{(6.6c)}\\
&+ \Big[\bar{\delta}_{AB}(y^{\circ\#ab}_{AB}) \;+\; \bar{\delta}_{EF}(y^{\circ\#ef}_{EF})\Big] && \text{(6.6d)}
\end{aligned}$$

where the RHS of the first line (Eq. (6.6a)) is the model factor, the second line (Eq. (6.6b)) is the parent term that is indexed by aggregate counts (for each of two groups of RVs) the third and fourth lines (Eq. (6.6c)) are children terms indexed by aggregate counts over subsets of RVs, and the fifth line (Eq. (6.6d)) are the intra-resolution terms that are each indexed by one joint count.

6.5 Adaptive Inference: Structure Selection

Up until now we have assumed that the inference structure was fixed. Of course, the inference quality depends on how well the inference structure represents the relevant inference quantities. However, in general, this structure will not be known *a priori* and it must be selected adaptively using preliminary inference information.

In our framework, the inference structure has two things to select. The first is the CJ-Tree that specifies the hierarchical grouping of RVs. The second is the associated CJ-LSS-Tree that specifies the state space partitions. In theory, we also have flexibility in selecting the regions that are used and how they are connected via separators to other regions. However, as we will see, the structure of the CJ-Tree and CJ-LSS-Tree in conjunction with the model factors can be used to fully determine the hierarchical region graph structure; we use this simplifying assumption in this chapter.

We begin in Section 6.5.1 by discussing an `AddLvl` operation that incrementally builds up the hierarchical grouping of RVs; this is represented by joining root nodes in the CJ-Tree. Then, in Section 6.5.2 we describe a `SplitSS` operation that incrementally splits a group of states into a finer group of states; this is representing by splitting a coarsen-LSS-node in the CJ-LSS-Tree. Lastly, in Section 6.5.3, we present an anytime inference algorithm that interleaves inference with either `AddLvl` or `SplitSS` modifications to the inference structure.

6.5.1 `AddLvl` – Joining RV groups

Initially, the CJ-Tree is a forest of disconnected leaf nodes. The CJ-Tree is built up by incrementally joining together root join-nodes (recall a join-node $j \in D^\circ$ is a root if it has no parents, i.e., $N_\uparrow(j) = \emptyset$). In the simplest case we select two root join-nodes and join them. That is, for two root join-nodes j' and j'', we add parent coarsen-nodes c' and c'' connected

to the two roots, i.e., $N_{\downarrow}(c') = \{j'\}$ and $N_{\downarrow}(c'') = \{j''\}$, and a new root parent join-node $j_{\uparrow}$ whose children are the coarsen-nodes i.e., $N_{\downarrow}(j_{\uparrow}) = \{c', c''\}$.

In this chapter, instead of performing one join at a time, we perform joins of all root nodes in a batch. That is, we partition the root nodes into groups of size two and each group is joined as above (if there is an odd number of roots, one of the groups has group size one). To represent this structure, each node in the graph is associated with a *level*. The leaf nodes of the CJ-Tree (initial disconnected roots) are at level 0. The joining join-nodes at level l results in coarsen-nodes that are said to be *between levels* l *and* $l+1$, and join-nodes at level $l+1$. In our depictions, each join-node is labeled with its level as a subscript, and each coarsen-node is labeled with its lower level l as a subscript and upper level $l+1$ as a superscript.

In our running example in Figure 6.2, leaf nodes are labeled with level 0. An initial join of join-nodes at level 0 in groups $\{A_0, B_0\}$ and $\{C_0, D_0\}$ produces coarsen-nodes A_0^1, B_0^1, C_0^1, and D_0^1 between levels 0 and 1, and join-nodes AB_1 and CD_1 at level 1.

Organizing nodes in levels like this is advantageous for two reasons. First, it results in a simple region graph structure as we discuss in the next section. Second, it results in a (roughly) balanced CJ-Tree, i.e., two CJ-nodes at the same level correspond (roughly) to the same number of underlying RVs. This balancing is desirable since, as discussed in other sections of this thesis, it is often difficult to decide how to allocate disparate work throughout the inference structure.

6.5.1.1 Modifying the Region Graph

The CJ-Tree and model factors can be used to specify the region graph structure as we describe in this section.

Each factor in the region graph has a *level*. A region or intra-resolution separator is said to be at level l if its CJ-scope uses only join-nodes at level l in the CJ-Tree; an inter-resolution separator is said to be between levels l and $l+1$ if its CJ-scope uses only coarsen-nodes between levels l and $l+1$ in the CJ-Tree.

Initially, the CJ-Tree consists of a set of disconnected leaf nodes (at level 0) and the region graph consists of model factors connected in a factor graph structure, i.e., each model factor is connected to a CJ-Tree region node via a separator. Joining roots at level l of the CJ-Tree to produce join-nodes at level $l+1$ in the CJ-Tree modifies the hierarchical region graph by creating inter-resolution separators between levels l and $l+1$, which are joined to produce regions at level $l+1$; finally, separators between regions at level $l+1$ are added. These terms are detailed as follows:

Inter-Resolution Separators: For each region r at level l, its parent separator's CJ-scope is equal to the union of parents of each element of the child separator's CJ-scope, $\text{sc}^{\square}(s_{\uparrow}) = \cup_{j \in \text{sc}^{\circ}(r)} N_{\uparrow}(j)$. This means that each dimension of the parent separator factor is coarser than each dimension of the child region factor.

Regions: The regions at level $l+1$ are formed by joining the intra-resolution separators along with the join-nodes at level $l+1$ in the CJ-Tree. The region scopes are determined as follows. First, define the operator $\texttt{pushUp}(s) = \cup_{c \in \text{sc}^{\square}(s)} N_{\uparrow}(c)$ that expands each dimension of the scope to higher levels in the CJ-Tree and is applied to intra-resolution separators. The region scopes at level $l+1$ are the set of unique scopes $\text{sc}^{\circ}(R_{l+1}) = \{\texttt{pushUp}(s) : s \in S_l^{l+1}\}$, and the children of each region are equal to the set of intra-resolution separators whose up-scope matches it $N_{\downarrow}(r_{l+1}) = \{s \in S_l^{l+1} : \texttt{pushUp}(s) = \text{sc}^{\circ}(r_{l+1})\}$.

For example, in the Figure 6.4, the separators CJ-scope $\text{sc}^{\square}(s) = (A_0^1, E_0^1)$, $\text{sc}^{\square}(s') = (A_0^1, F_0^1)$, $\text{sc}^{\square}(s'') = (B_0^1, E_0^1)$, and $\text{sc}^{\square}(s'') = (B_0^1, F_0^1)$ have $\texttt{pushUp}(s) = \texttt{pushUp}(s') =$

$\texttt{pushUp}(s'') = \texttt{pushUp}(s''') = (AB_1, EF_1)$ which is the CJ-scope of their common parent.

Intra-Resolution Separator: There is an intra-resolution separator between any region and a uni-CJ-variate region. That is, for each region r_{l+1} at level $l+1$, and each uni-CJ-variate region $j \in \text{sc}^\circ(r_{l+1})$, there is a separator s with CJ-scope $\text{sc}^\circ(s)$ connecting r_{l+1} and j.

Global Structure The region graph at level $l+1$ can be seen as a lower resolution region graph than the graph at level l. That is, it has the same structure as the level l graph but over larger hyper-RVs.

For example, consider G_0 as the region graph of a pairwise complete graph. At level l, G_l consists of all regions with CJ-scope size two connected via separators to all regions with CJ-scope size one Figure 6.4 shows a subgraph of the region graph G; at the bottom of the figure (boxed in red) is a subgraph of the level l subgraph G_l; at the top of the figure is a subgraph of the level $l+1$ subgraph G_{l+1}.

Another example is a where G_0 is a $n \times n$ grid where each RV has domain size k. If we form groups of RVs as 2×2 blocks, then G_1 is a $n/2 \times n/2$ grid where each hyper-RV represents k^4 ground states.

6.5.1.2 Selecting the Join

The quality of the bound depends heavily on the grouping of RVs at each level. This problem is related to the choice of regions in a standard region graph relaxation which is a difficult and still largely open problem. One option to tackling this problem is to compare joins based on their (local) decrease in the variational objective. In our framework using factors with state space symmetry, this would also require selecting the initial state space partitions for the joined factors. These choices present us with an interesting but difficult problem that

we defer until future work. In the experiments in this chapter, we test on graphs where a reasonable grouping can be selected from the graph structure. For example, in complete graphs, no structural information can be used to distinguish among groupings; in grid graphs, we use a linear ordering that forms groups of neighboring nodes.

6.5.2 `SplitSS` – Splitting a State Space Partition

A `SplitSS` operation keeps the structure of the CJ-Tree fixed and selects a group of states in the CJ-LSS-Tree to split into two finer groups. This allows us to represent functions that treat the two finer groups distinctly (whereas before the split they were treated identically). We first describe the modification to the inference structure that is induced by the split of a single state space partition into two groups; this modifies the CJ-LSS-Tree and associated MD-LSS-factors. Then we discuss how to select the partition to split and how to split it.

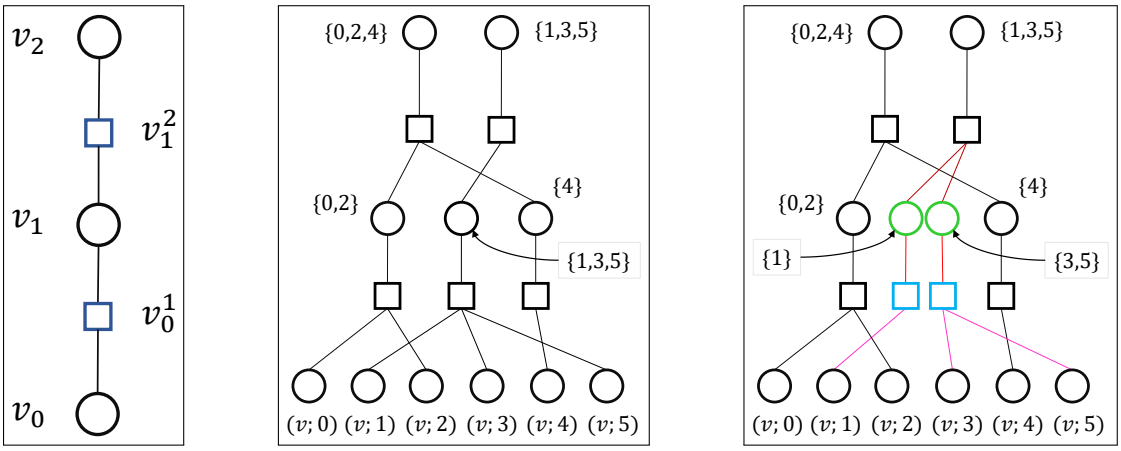
6.5.2.1 Structure Modification

A `SplitSS` operation splits the ground state set associated with a coarsen-LSS-node $y_c^{\square} \in \bar{\mathcal{D}}_c^{\square}$ into two sets represented by new coarsen-LSS-nodes $y_c^{\square\prime}$ and $y_c^{\square\prime\prime}$. This operation does not affect the ground state sets of any other coarsen-LSS-nodes.

Figure 6.5 will guide our discussion; for simplicity, it represents a univariate coarsening whose CJ-Tree T is a simple chain. Figure 6.5b shows a CJ-LSS-Tree and Figure 6.5c shows the CJ-LSS-Tree after splitting the middle coarsen-LSS-node $y_c^{\square}$ associated with ground state sets $\mathrm{gr}(y_c^{\square}) = \{1, 3, 5\}$ into two coarsen-LSS-nodes associated with ground state sets $\mathrm{gr}(y_c^{\square\prime}) = \{1\}$ and $\mathrm{gr}(y_c^{\square\prime\prime}) = \{3, 5\}$.

A `SplitSS` operation requires the following modifications to the CJ-LSS-Tree:

Children: We split the children set of $y_c^{\square}$ into two groups, such that $\{\bar{\mathcal{N}}_{\downarrow}(y_c^{\square\prime}),\ \bar{\mathcal{N}}_{\downarrow}(y_c^{\square\prime\prime})\}$



(a) CJ-Tree T

(b) CJ-LSS-Tree $\bar{\mathcal{T}}^t$ at time-step t

(c) CJ-LSS-Tree $\bar{\mathcal{T}}^{t+1}$ at time-step $t+1$

Figure 6.5: Example showing modifications to the CJ-LSS-Tree resulting from splitting a single ground state set into two. (a) Univariate CJ-Tree T is a simple chain; squares represent coarsening, circles represent joins (the node has no siblings, so the join is simply an identity operation), (b) CJ-LSS-Tree $\bar{\mathcal{T}}^t$ at time-step t, (c) CJ-LSS-Tree $\bar{\mathcal{T}}^{t+1}$ at time-step $t+1$ after coarsen-LSS-node (square) associated with ground set $\{1,3,5\}$ is split into $\{1\}$ and $\{3,5\}$. The modified nodes and edges are highlighted while the unmodified are shown in original colors. Note that each top nodes represents the same ground state set before and after the split (both have the same labels). This illustrates that splitting only affects local symmetry (and does not propagate up the tree).

partitions $\bar{\mathcal{N}}_{\downarrow}(y_c^{\square})$; this is represented by blue square nodes and magenta edges in the figure. Note that the finer ground state sets $\{\mathrm{gr}(y_c^{\square\prime}),\ \mathrm{gr}(y_c^{\square\prime\prime})\}$ partition the original ground state set $\mathrm{gr}(y_c^{\square})$ since the ground state set associated with each LSS-node is the union of its childrens' ground state sets.

Parent Join Node: Splitting the coarsen-LSS-node $y_c^{\square}$ into two causes each of its parent join-LSS-nodes $y_j^{\circ} \in \bar{\mathcal{N}}_{\uparrow}(y_c^{\square})$ to split into two join-LSS-nodes $y_j^{\circ\prime}$ and $y_j^{\circ\prime\prime}$ (this is represented by the green nodes and the light red edges in the figure). Each of these new nodes has the same LSS-children as y_j° except with $y_c^{\square}$ replaced by $y_c^{\square\prime}$ and $y_c^{\square\prime\prime}$,

respectively. That is,

$$\bar{\mathcal{N}}_{\downarrow}(y_j^{\circ\prime}) \;=\; (\bar{\mathcal{N}}_{\downarrow}(y_j^{\circ}) \setminus y_c^{\square}) \;\cup\; y_c^{\square\prime}$$
$$\bar{\mathcal{N}}_{\downarrow}(y_j^{\circ\prime\prime}) \;=\; (\bar{\mathcal{N}}_{\downarrow}(y_j^{\circ}) \setminus y_c^{\square}) \;\cup\; y_c^{\square\prime\prime}$$

Parent Edges: Each parent coarsen-LSS-nodes $y_{c_\uparrow}^{\square} \in \bar{\mathcal{D}}_{c_\uparrow}^{\square}$ must have identical ground state set before and after the split. This is accomplished as follows: if a node was in the child set, then both of its split nodes are in the child set (this is represented by the dark red edges in the figure). That is,

$$\bar{\mathcal{N}}_{\downarrow}(y_{c_\uparrow}^{\square}) \;= (\bar{\mathcal{N}}_{\downarrow}(y_{c_\uparrow}^{\square}) \setminus \bar{\mathcal{N}}_{\uparrow}(y_c^{\square})) \cup$$
$$\cup_{y_j^{\circ} \in \bar{\mathcal{N}}_{\downarrow}(y_{c_\uparrow}^{\square}) \cap \bar{\mathcal{N}}_{\uparrow}(y_c^{\square})} \{y_j^{\circ\prime}, y_j^{\circ\prime\prime}\}$$

where the set $\bar{\mathcal{N}}_{\downarrow}(y_{c_\uparrow}^{\square}) \cap \bar{\mathcal{N}}_{\uparrow}(y_c^{\square}) \subset \bar{\mathcal{D}}_{j_\uparrow}^{\circ}$ is the set of join-LSS-nodes that are children of $y_{c_\uparrow}^{\square}$ and that split.

Splitting the Region Graph Factors Each table element in the region graph that is associated with a LSS-node that split, splits into two. Each new node is assigned the same value in its factor table as the node that split. Thus, the factor after the split can represent a finer resolution function, but it is initialized with the old coarser resolution function.

For example, for an initial MD-LSS-factor table, $\begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$ splitting the second LSS-node, as in Figure 6.5, into two results in a MD-LSS-factor table whose second row is split into two rows with identical value $\left[\begin{array}{ccc} a & d & g \\ \hline b & e & h \\ b & e & h \\ \hline c & f & i \end{array}\right]$.

6.5.2.2 Selecting the Split

The previous section showed how the inference structure changes when a coarsen-LSS-node $y_c^{\square} \in \bar{\mathcal{D}}_c^{\square}$ (associated with coarsen-node $c \in D^{\square}$) is split into two finer groups $y_c^{\square\prime}$ and $y_c^{\square\prime\prime}$. This section discusses how to select (i) the state set $y_c^{\square} \in \bar{\mathcal{D}}_c^{\square}$ to split, and (ii) the finer groups $y_c^{\square\prime}$ and $y_c^{\square\prime\prime}$.

First, we select the state set attaining the local MAP configuration $y_c^{\square *} = \arg\max_{y_c^{\square} \in \bar{\mathcal{D}}_c^{\square}} \bar{\theta}_c^{\bar{\delta}}(y_c^{\square})$ where $\bar{\theta}_c^{\bar{\delta}}(y_c^{\square})$ are the preliminary estimates of the max-marginals. Second, we split the fine resolution children $N_{\downarrow}(y_c^{\square *})$ into two groups of (roughly) equal size based on their max-marginals: those with the highest scoring max-marginals in one group and those with the lowest scoring max-marginals in the other group. Intuitively, we expect the high scoring group to represent more precise information at the upper level, thus allowing a higher quality solution to be found; we expect the lower scoring group to not attain the MAP configuration (that is, after inference information propagates). The equal size restriction is a heuristic that prevents us from having to choose an unequal allocation of group size (which is difficult to do based on local information).

Note that the split selection presented here is similar to the selection used by Sontag et al. [2009]. In that work, they created a group for each of the top-scoring (by max-marginal) k_1 states and an "other" group for the lower scoring states. Our approach of splitting into larger groups allows for more gradual splits, thus allowing inference information to propagate before choosing future splits.

Balanced Cost The previous section showed how to select a coarsen-LSS-node associated with coarsen-node $c \in D^{\square}$ to split and how to split it. Another issue is selecting the coarsen-node $c \in D^{\square}$. In theory we can allocate work disparately throughout the graph; that is, we can introduce finer resolution state sets where they have the biggest impact on inference quality. However, making this selection is difficult.

To avoid the selection problem, we assume that work is balanced throughout the graph. That is, each coarsen-node $c \in D^{\square}$ corresponds to the same number $\bar{k}$ of state space partitions, i.e., $|\bar{\mathcal{D}}_c^{\square}| = \bar{k}$. Our anytime inference algorithm in Section 6.5.3 incrementally increases $\bar{k}$, resulting in finer and finer inference approximations.

Other Ways to Select the Split Other ways to select the split are possible. One alternative would measure the (local) decrease in the inference bound. That is, it would examine the effect of the split on the high-order interactions. This would potentially lead to more accurate selection, but seems to be a fairly difficult problem for two reasons. First, the split affects the factor tables at (many) incident regions. Second, this would require a search over an exponential number of splits.

In contrast, our simple approach above provides a very cheap way to select the split. Nevertheless, future work should be directed toward a more systematic analysis of the split selection problem, including appropriate metrics for the selection, and trading off computational cost spent in selecting the split with inference cost.

6.5.3 Anytime Inference

This section presents an anytime inference algorithm, shown in Algorithm 11, that interleaves inference with `SplitSS` and `AddLvl` modifications. The outer loop executes until all computational resources have been exhausted. It performs inference on the current graph structure (in `Step` 1) then, if memory resources are not exhausted, chooses a transition to a higher cost and higher accuracy graph via `SplitSS` operation or `AddLvl` operation (in `Step` 2) We describe some aspects of the algorithm that require further detail below.

Algorithm 11 Anytime Inference for Hierarchical Region Graphs

Input: An initial region graph G^0, a number of inference sweeps on each modified graph I, the available memory and computational cost τ_{mem} and τ_{cpu}.

Output: Inference summary as pairs of objective evaluation and iteration cost $\{(L_{t;i},\ c_{t;i}) : t \in [m'],\ i \in [I]^+\}$, transition types between graphs $\{\texttt{marker}_t^{t+1} : t \in [m'-1]\}$.

Initialize: $t \leftarrow 0, \quad \texttt{marker}_{-1}^{0} \leftarrow \texttt{Init}, \quad \tau_{SS}^{-1:0} \leftarrow \mathcal{C}(G_0), \quad \bar{k}_0 \leftarrow 1;$

while $\Big(\sum_{t' \in [t-1]} \sum_{i \in [I]^+} c_{t';i} \leq \tau_{CPU} \Big)$ **do**

`Step 1` (Inference): Inference hot-started at the old parameterization:

$$(\delta^t,\ L_{t;\cdot}, c_{t;\cdot}) \ = \ \texttt{Infer}_{\texttt{DD}}(G^{t-1}, \delta^{t-1};\ I)$$

`Step 2` (Transition): Transition to a graph with higher cost and accuracy:

(i) $\tau_{SS}^{t;t+1} \leftarrow \min(\tau_{SS}^{t-1;t} \cdot 4,\ \tau_{mem}), \qquad \bar{k}_{t+1} = \bar{k}_t \cdot 2;$

(ii) $(G^{t+1}_{\texttt{SplitSS}},\ \texttt{marker\_SplitSS}) = \texttt{SplitSS}(G^t,\ \tau_{SS}^{t;t+1},\ \bar{k}_{t+1})$

(iii) $(G^{t+1}_{\texttt{AddLvl}},\ \texttt{marker\_AddLvl}) = \texttt{AddLvl'}(G^t,\ \tau_{SS}^{t;t+1}, \bar{k}_{t+1})$

(iv) $\texttt{token} \ = \ \texttt{SelectSplitOrJoin}(G_{\texttt{splitSS}},\ G_{\texttt{addLvl}})$

(v) $G^{t+1} = G^{t+1}_{(\texttt{token})}, \quad \texttt{marker}_t^{t+1} \leftarrow \texttt{flag\_}(\texttt{token});$

3. $t \leftarrow t+1;$

end while

6.5.3.1 Balancing Inference Cost with Structure Modification

One aspect that we have not discussed so far is balancing the inference cost with selection of the modification (`SplitSS` or `AddLvl`). In theory, this could be selected adaptively by predicting the benefit of another iteration of inference versus any of the modifications. But, in practice, this selection is difficult. Instead, we use a balanced allocation that performs I global inference iterations (where I is a meta-parameter) then transitions to a graph with an inference cost that is approximately $\tau_{SS}^{t;t+1}$.

The inference cost is controlled by the parameter $\bar{k}_t$ that controls the number of state space partitions at each node. This is multiplied by a constant factor at each iteration. Thus, the `SplitSS` function in this algorithm iteratively performs individual splits (as described in

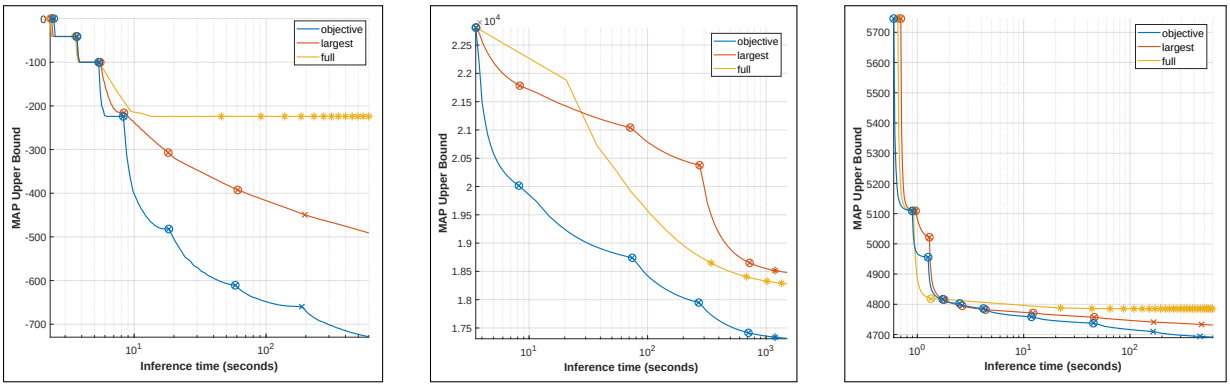
Section 6.5.2.2) until each coarsen-node $c \in D^{\square}$ has $\bar{k}_t$ partitions, i.e., $|\bar{\mathcal{D}}_c^{\square}| = \bar{k}_t$.

The function `AddLvl`' performs an `AddLvl` operation then `SplitSS` operation throughout the graph. The reason for this is that at higher levels, the `AddLvl` operation adds only a small amount of work to the graph, i.e., it does not attain the desired doubling of inference cost. Splitting throughout the graph ensures that the modified graph has double the cost of the original graph.

6.5.3.2 Choosing Between `SplitSS` and `AddLvl`

In our initial experiments, it seemed beneficial to add levels as soon as possible. This allows very large scope regions with coarse state space symmetry to be added early and refined later. Thus, we did not develop a way to select between `SplitSS` and `AddLvl`' (which, additionally, would be a difficult problem); instead, we chose `AddLvl`' until all the RVs have been joined (i.e., the inference structure contains a large clique over all RVs); at this point, only the `SplitSS` operation is available. The move that was taken is represented by the "marker" tag which will be denoted in the plots in the experiments section.

Note that the property that the CJ-LSS-Tree can be split at an intermediate level – and is not prevented from doing do by the presence of higher levels – provides a crucial source of flexibility in our anytime framework. In particular, it allows us to quickly add high-order regions before splitting the lower levels to their final resolutions. Thus, choosing to add a level at "the wrong time" is not as detrimental as it would be if intermediate splits in the CJ-LSS-Tree were prohibited.



(a) Perturbation of symmetric complete graph

(b) UAI Object detection instance

(c) UAI 20x20 wrapped grid

Figure 6.6: Comparison of three different state space splitting procedures (colored curves) on three different model instances (subfigures). The y-axis measures the decomposed MAP objective. Each method was allowed a maximum of 10^8 table elements in the hierarchical region graph (to bound memory cost). Methods were run for 600 seconds. The UAI object detection instance (panel (b)) did not visually converge in that time and was run for longer to better distinguish among methods. The main text contains further discussion of these results.

6.6 Experiments

This section empirically validates the main claims in this chapter. Namely, (a) that we can efficiently incorporate larger-scope factors than possible with conventional methods (dense join), (b) that we can adaptively discover a good hierarchical coarsening using inference information, and (c) that this procedure can be applied without model assumptions such as large RV domain size (e.g., required by Sontag et al. [2009]) or exact symmetry in factors and structure (e.g., required by [Milch et al., 2008]). We demonstrate good performance on generic graphical models. These initial results are promising. Furthermore, future work that selects the inference structure more accurately should produce increasing gains.

6.6.1 Setup

The main goal of our work is to demonstrate the effectiveness of using hierarchically defined state space partitions for inference. We compare between the standard method of full-join

(yellow curve) with no coarsening and two methods for selecting the state space partitions (i.e., the CJ-LSS-Tree): splitting using inference information as described in Section 6.5.2.2 (blue curve), and splitting the largest group at each node such that state space partitions represent (roughly) the same number of ground states (red curve).

To facilitate this comparison, we ensure that each method uses the same CJ-Tree (RV hierarchy); this ensures that performance differences between methods result from the choice of state space partitions. In these experiments, we test on models where a reasonable grouping can be selected from the graph structure *a priori*; in complete graphs, all groupings are structurally indistinguishable; in grid graphs, we use a linear ordering that forms groups of neighboring nodes.

Inference Inference is performed using block-coordinate descent with star neighborhood blocks centered at each region. For ground inference, this optimization algorithm was described in Section 2.4. For coarse univariate factors (which we discussed in Section 6.2), Sontag et al. [2009] presents a similar block coordinate descent algorithm where closed-form updates use a coarse max-projection operation. In our hierarchical model, we essentially use the same updates but over larger clusters of RVs.

Initialization and Plotting Recall that our goal is to evaluate the effectiveness of different structures tightening the "standard" dual decomposition relaxation whereby each model factor is connected to a univariate factor associated with each RV in its scope (the "factor-graph" topology). Consequently, inference on this initial structure is first run for 100 iterations for each method. The top left of the plot begins after its completion.

After this, we interleave a fixed number, $I = 20$, of global inference iterations with structure modification. Each modification either chooses and refines a set of coarsenings throughout the current graph (marked with an "x" on the plot) or adds another level of RVs before

refinement throughout the graph (marked with an "o" around an "x"). At each modification, the method quadruples its work (or gets as close as possible). If only a larger step is possible, e.g., the full-join method, the method is permitted to take it as long as it does not exceed a memory bound. When the memory bound is reached, it continues inference at its current structure (marked with a "*") until it hits a time bound.

In our preliminary experiments, always choosing to add a level (until none are left, then splitting) often performed best, hence we use it in our reported experiments. Thus, curves have circled x's until all the RVs are joined, then they have regular x's. Note that we can read off the number of RVs in the scope of the largest join-node from the graph (indicating the size of regions being reasoned over). The number of RVs in the scope of the largest join-node doubles with each level and consequently is equal to 2^k where k is the number of crossed o's before it.

6.6.2 Models and Results

We now discuss the test models and results.

Perturbed Symmetric Complete Graph This is a simulated model. It is a complete symmetric graph perturbed by a deletion of 50% of its edges at random. Recall from this chapter's introduction (and detailed in Section 3.8) that a symmetric complete graph gives rise to a counting factor over all RVs; we expect a similar set of state space partitions to work well in the perturbation. In this model, we expect methods that reason using factors with state space partitions to provide a huge gain over the standard method that reasons with dense table (since the counting factor is represented with a small number of state space partitions).

The pairwise terms in the model have potential $\begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix}$. Since the low-order relaxation is accurate for attractive ($a > 0$) models, we test on a repulsive model with $a = -1$ where

high-order regions are crucial for accurate approximate inference.

Methods that look for exact symmetry, like Milch et al. [2008], or methods that perform coarsening on RVs with large domain size, like Sontag et al. [2009], cannot be applied to a model of this form. Our results show that our hierarchical methods are able to build large-scope terms and moreover that these result in a significant improvement over full join which only incorporates moderate sized groups. The utility of objective-based selection (blue) over none (red) is also demonstrated.

UAI Object Detection This model comes from the UAI challenge benchmark Elidan et al. [2012]. Models of this class have moderately-size RV domains (10-20), often dense connectivity. Our test model (Giraffe_rescaled_5025.K20.F100) is a complete graph on 60 with RV domain size 21. This test demonstrates the effectiveness of our method on problems where the model factor tables are unstructured.

The full join (yellow curve) performs one RV grouping (yielding CJ-Tree representing pairs of nodes, and a region graph whose factors use four RVs) before exceeding the memory bound. In the plot, a single join is performed then all markers are stars ("*") meaning that memory has been exhausted and batches of inference with I iterations are continued on this structure. In contrast, methods that use state space partitions can tractably perform many joins, producing large scope factors, eventually over the entire model. We see that the objective-based split selection (blue curve) is superior to the non-objective-based split selection (red curve).

Wrapped Grid This model is a 20x20 wrapped grid (grid20x20.f15.wrap) from the UAI benchmark Elidan et al. [2012]. This is a sparsely connected model in contrast to the previous two complete graphs. Additionally, it has binary RVS so the univaraite coarsening method of Sontag et al. [2009] would be inapplicable. While the full join (yellow curve) reaches

its memory bound quickly, the other methods are able to continue. Again, objective-based splitting (blue curve) is superior to non-objective-based splitting (red curve).

6.7 Conclusions and Future Work

In this chapter, we introduced an inference framework that reasons about factors with state space symmetry. In contrast to standard inference methods that use dense tablular factors which are restricted restricted to reasoning over small groups of RVs, we are able to construct inference terms over large groups of RVs at controlled cost. We showed that our framework generalizes many previous attempts to incorporate factors with state space symmetry. These include projected message passing algorithms [Gogate and Domingos, 2013] (which require a fixed region graph), univariate domain coarsening [Sontag et al., 2009] (which requires RVs with large domain size), recursive cardinality models [Tarlow et al., 2012] (which is specific to counting factors and requires a fixed graph structure), and symmetric counting models [Milch et al., 2008] (which use a specific rule to identify symmetric group of factors that aggregate to a counting factor).

Our framework centers around a hierarchical state space partition over increasingly large groups of RVs. The state space partitions were defined recursively via coarsening and joining operations; this allows large and complex state space partitions to be built up efficiently. Furthermore, our structure can be interleaved with inference and selected adaptively. We demonstrated good results on a wide variety of problems; our experiments showcased our ability to select the inference structure adaptively using preliminary inference information. Some directions for future work are detailed below.

The inference quality depends heavily on the choice of inference structure; in our framework, this is the choice of RV groups (via `AddLvl` operation) and state space partitions (via `SplitSS`

operation). Future work should look into more intelligent ways to select the grouping of RVs and state space partitions. This should include a systematic investigation of different metrics for judging the quality of each transition.

The central contribution of this chapter was a method for intelligently selecting a symmetric inference structure. This chapter focused on the MAP inference problem, but our method should be applicable to other inference and learning tasks where we use the same structure but a different objective function. Future work should look into other inference problems and objectives such as, decoding the MAP solution to produce a MAP configuration, performing approximate sum-inference using the generalized dual decomposition (or weighted mini-bucket elimination), and performing approximate maximum likelihood learning.

Chapter 7

Conclusions and Future Directions

The central goal of this thesis was to develop more intelligent ways to represent and reason using (approximate) symmetries in PGMs for two distinct classes of problems. The first half our of novel work (in Chapter 4 and Chapter 5) operates on models with symmetric factors perturbed by asymmetric evidence. The second half of our novel work (in Chapter 6) operates on classic PGMs (with no symmetry) and tightens the inference relaxation by adding factors with state space symmetry using a sophisticated grouping of states. Although these two problem classes are very different, the methods we develop for exploiting and representing symmetry share deep similarities. The following section discusses our contributions in more detail.

7.1 Our Contributions

In Chapter 4 and Chapter 5, we addressed the problem of performing approximate inference in large symmetric models perturbed by evidence (asymmetries). While a large body of literature had previously existed for lifted inference in the presence of perfect symmetry, few works address the significant practical problem posed by asymmetric perturbations.

In our *Lifted GDD Algorithm* in Chapter 4, we introduced coarse-to-fine lifted inference

procedures which maintained both (i) strong control over the computational cost of the approximation (in contrast to approaches which exploit exact symmetry), and (ii) a concrete connection to the ground problem (in contrast to approaches which form a heuristic over-symmetric model approximation with unquantifiable bias). Empirically, we demonstrated a superiority to ground inference approaches which grows larger with model size. We also demonstrated the utility of adaptively selecting the coarse-to-fine refinement using inference information.

Our *Lifted WMB Algorithm* in Chapter 5 extended our *Lifted GDD Algorithm* in Chapter 4 by introducing large scope factors into the inference relaxation for MLN models. This provided two axes of approximation: small to large scope factor templates and coarse-to-fine symmetry groups. To our knowledge, this was the first algorithm with this flexibility.

In Chapter 6, we addressed the problem of incorporating state space symmetry into inference relaxations for classic PGM models with asymmetric factors. The centerpiece of our algorithm was a novel hierarchical state space partition which recursively built complex state space symmetry over large groups of RVs from state space symmetry over small groups of RVs. A collection of factors using these state space partitions were constructed; these factors had aligned symmetry by construction and could thus pass messages among each other without breaking symmetry. We demonstrated that this complex inference structure could be constructed dynamically, where the state space symmetry is constructed in a coarse-to-fine manner. We showed that our framework encompasses properties of several previously proposed frameworks as special cases, including the standard method that joins dense factor tables, aggregating symmetric groups of factors to a counting factor, and performing inference with a nested hierarchy of counting factors. We demonstrated promising empirical results on a number of UAI benchmark problems, and discussed how the flexibility of our method invites many avenues of future work.

7.2 Future Directions

Our work in this thesis opens up many new research directions. Below are a few suggestions.

Fusing state space and factor symmetry: We developed algorithms for state space symmetry and factor symmetry separately. However, throughout the thesis we highlighted the deep similarities between algorithms for each class of symmetry. These similarities suggest a hybrid approach that operates on models with state space symmetry and factor symmetry should be possible within the frameworks that we developed.

Fusing top-down and bottom-up information: Our work on *Adaptive Hierarchical State Space Partitions* in Chapter 6 builds large-scope factors from small-scope factors in a bottom-up fashion. In contrast, our work on *Lifted GDD Algorithm* in Chapter 4 and *Lifted WMB Algorithm* in Chapter 5 work in a top-down fashion: a large symmetry group (over many RVs) is split into finer groups, each over a smaller number of RVs. Ideally, we would like to be able to introduce incremental complexity into the approximate inference structure using a hybid of bottom-up and top-down approaches: bottom-up approach would correspond to grouping evidence terms, while top-down splits coarse symmetry grouping in model factors.

Symmetry group selection: Each framework in this thesis relies on adaptively selecting the symmetry partition. This choice has a huge impact on the quality of the inference approximation, but in this thesis we presented very simple selection metrics. Future work should look at more principled methods of selecting the symmetry and conduct a rigorous empirical comparison of competing methods.

Inference Objective: The primary contributions in this thesis dealt with organizing the structure of inference computations to control shattering (symmetry breaking). Consequently, the central ideas regarding the inference structure should be applicable to

problems with different choices of the inference objective. In particular, we should be able to extend these methods to maximum-likelihood based learning (which requires evaluation of the partition function), which is a fundamental problem in PGMs.

Bibliography

U. Apsel. Lifted message passing for the generalized belief propagation. *arXiv preprint arXiv:1610.01525*, 2016.

D. Batra, S. Nowozin, and P. Kohli. Tighter relaxations for map-mrf inference: A local primal-dual gap based separation algorithm. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 146–154, 2011.

C. Berkholz, P. Bonsma, and M. Grohe. Tight lower and upper bounds for the complexity of canonical colour refinement. In *European Symposium on Algorithms*, pages 145–156. Springer, 2013.

S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

R. D. S. Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In *Proceedings of the 19th international joint conference on Artificial intelligence*, pages 1319–1325, 2005.

H. H. Bui, T. N. Huynh, and S. Riedel. Automorphism groups of graphical models and lifted variational inference. *arXiv preprint arXiv:1207.4814*, 2012.

H. H. Bui, T. N. Huynh, and D. Sontag. Lifted tree-reweighted variational inference. *arXiv preprint arXiv:1406.4200*, 2014.

L. De Raedt, A. Kimmig, and H. Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, pages 2462–2467. Hyderabad, 2007.

R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2):41–85, 1999.

R. Dechter and I. Rish. A scheme for approximating probabilistic inference. In *Proc. Uncertainty in Artificial Intelligence*, pages 132–141. Morgan Kaufmann Publishers Inc., 1997.

R. Dechter et al. *Constraint processing*. Morgan Kaufmann, 2003.

G. Elidan, A. Globerson, and U. Heinemann. Pascal 2011 probabilistic inference challenge, 2012.

S. Forouzan and A. T. Ihler. Incremental region selection for mini-bucket elimination bounds. In *UAI*, pages 268–277, 2015.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

N. Gallo and A. Ihler. Lifted generalized dual decomposition. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018a.

N. Gallo and A. T. Ihler. Lifted weighted mini-bucket. In *Advances in Neural Information Processing Systems*, pages 10329–10337, 2018b.

N. Gallo and A. T. Ihler. Adaptive hierarchical state space partitions for probabilistic inference. In *In submission*, 2020.

M. R. Genesereth and N. J. Nilsson. *Logical foundations of artificial intelligence*. Morgan Kaufmann, 2012.

V. Gogate and P. Domingos. Probabilistic theorem proving. *arXiv preprint arXiv:1202.3724*, 2012.

V. Gogate and P. Domingos. Structured message passing. *arXiv preprint arXiv:1309.6832*, 2013.

H. Habeeb, A. Anand, P. Singla, et al. Coarse-to-fine lifted map inference in computer vision. *arXiv preprint arXiv:1707.07165*, 2017.

G. Hardy and J. Littlewood. andg. polya, inequalities, 1952.

A. T. Ihler, A. S. Willsky, et al. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6(May):905–936, 2005.

A. T. Ihler, N. Flerova, R. Dechter, and L. Otten. Join-graph based cost-shifting schemes. *arXiv preprint arXiv:1210.4878*, 2012.

M. Isard, J. MacCormick, and K. Achan. Continuously-adaptive discretization for message-passing algorithms. In *Advances in Neural Information Processing Systems*, pages 737–744, 2009.

A. Jha, V. Gogate, A. Meliou, and D. Suciu. Lifted inference seen from the other side: The tractable features. In *Advances in Neural Information Processing Systems*, pages 973–981, 2010.

M. Kazemi, A. Kimmig, G. Van den Broeck, and D. Poole. New liftable classes for first-order probabilistic inference. 10 2016.

K. Kersting, Y. El Massaoudi, F. Hadiji, and B. Ahmadi. Informed lifting for message-passing. In *AAAI*, 2010.

A. Kimmig, S. Bach, M. Broecheler, B. Huang, and L. Getoor. A short introduction to probabilistic soft logic. In *Proceedings of the NIPS Workshop on Probabilistic Programming: Foundations and Applications*, pages 1–4, 2012.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Q. Liu and A. T. Ihler. Bounding the partition function using holder's inequality. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 849–856, 2011.

J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

B. M. McCoy and T. T. Wu. *The two-dimensional Ising model*. Courier Corporation, 2014.

B. Milch, B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov. 1 blog: Probabilistic models with unknown objects. *Statistical relational learning*, page 373, 2007.

B. Milch, L. S. Zettlemoyer, K. Kersting, M. Haimes, and L. P. Kaelbling. Lifted probabilistic inference with counting formulas. In *Aaai*, volume 8, pages 1062–1068, 2008.

T. P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369, 2001.

M. Mladenov and K. Kersting. Equitable partitions of concave free energies. In *UAI*, pages 602–611, 2015.

M. Mladenov, B. Ahmadi, and K. Kersting. Lifted linear programming. In *AISTATS*, pages 788–797, 2012.

M. Mladenov, A. Globerson, and K. Kersting. Lifted message passing as reparametrization of graphical models. In *UAI*, pages 603–612, 2014a.

M. Mladenov, K. Kersting, and A. Globerson. Efficient lifting of map lp relaxations using k-locality. In *AISTATS*, pages 623–632, 2014b.

A. Pfeffer. Ibal: A probabilistic rational programming language. In *IJCAI*, pages 733–740. Citeseer, 2001.

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

D. Poole. First-order probabilistic inference. In *IJCAI*, volume 3, pages 985–991, 2003.

H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690. IEEE, 2011.

N. Ramakrishnan, E. Ertin, and R. L. Moses. Assumed density filtering for learning gaussian process models. In *2011 IEEE Statistical Signal Processing Workshop (SSP)*, pages 257–260. IEEE, 2011.

M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62(1):107–136, 2006.

S. Russell and P. Norvig. Artificial intelligence: a modern approach. 2002.

S. Sarkhel, P. Singla, and V. G. Gogate. Fast lifted map inference via partitioning. In *Advances in Neural Information Processing Systems*, pages 3240–3248, 2015.

M. Schmidt. minfunc: unconstrained differentiable multivariate optimization in matlab. 2005.

P. Sen, A. Deshpande, and L. Getoor. Bisimulation-based approximate lifted inference. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 496–505. AUAI Press, 2009.

P. Singla and P. M. Domingos. Lifted first-order belief propagation. In *AAAI*, volume 8, pages 1094–1099, 2008.

P. Singla, A. Nath, and P. M. Domingos. Approximate lifting techniques for belief propagation. In *AAAI*, pages 2497–2504, 2014.

D. Smith, P. Singla, and V. Gogate. Lifted region-based belief propagation. *arXiv preprint arXiv:1606.09637*, 2016.

D. Sontag and T. Jaakkola. Tree block coordinate descent for map in graphical models. In *Artificial Intelligence and Statistics*, pages 544–551, 2009.

D. Sontag, A. Globerson, and T. S. Jaakkola. Clusters and coarse partitions in lp relaxations. In *Advances in Neural Information Processing Systems*, pages 1537–1544, 2009.

D. Sontag, T. Meltzer, A. Globerson, T. S. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. *arXiv preprint arXiv:1206.3288*, 2012.

E. B. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky. Distributed occlusion reasoning for tracking with nonparametric belief propagation. In *Advances in neural information processing systems*, pages 1369–1376, 2005.

N. Taghipour, D. Fierens, J. Davis, and H. Blockeel. Lifted variable elimination with arbitrary constraints. In *International Conference on Artificial Intelligence and Statistics*, pages 1194–1202, 2012.

D. Tarlow, K. Swersky, R. S. Zemel, R. P. Adams, and B. J. Frey. Fast exact inference for recursive cardinality models. *arXiv preprint arXiv:1210.4899*, 2012.

S. Tourani, A. Shekhovtsov, C. Rother, and B. Savchynskyy. Taxonomy of dual block-coordinate ascent methods for discrete energy minimization. *arXiv preprint arXiv:2004.07715*, 2020.

G. Van den Broeck and A. Darwiche. On the complexity and approximation of binary evidence in lifted inference. In *Advances in Neural Information Processing Systems*, pages 2868–2876, 2013.

G. Van den Broeck and M. Niepert. Lifted probabilistic inference for asymmetric graphical models. *arXiv preprint arXiv:1412.0315*, 2014.

G. Van den Broeck, A. Choi, and A. Darwiche. Lifted relax, compensate and then recover: From approximate to exact lifted probabilistic inference. *arXiv preprint arXiv:1210.4840*, 2012.

D. Venugopal and V. Gogate. On lifting the gibbs sampling algorithm. In *Advances in Neural Information Processing Systems*, pages 1655–1663, 2012.

D. Venugopal and V. Gogate. Evidence-based clustering for scalable inference in markov logic. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 258–273. Springer, 2014a.

D. Venugopal and V. G. Gogate. Scaling-up importance sampling for markov logic networks. In *Advances in Neural Information Processing Systems*, pages 2978–2986, 2014b.