DECISION MAKING AND INFERENCE UNDER LIMITED INFORMATION AND HIGH DIMENSIONALITY

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Stefano Ermon

January 2015

DECISION MAKING AND INFERENCE UNDER LIMITED INFORMATION AND HIGH DIMENSIONALITY

Stefano Ermon, Ph.D.

Cornell University 2015

Statistical inference in high-dimensional probabilistic models is one of the central problems of statistical machine learning and stochastic decision making. To date, only a handful of distinct methods have been developed, most notably (Markov Chain Monte Carlo) sampling, decomposition, and variational methods. In this dissertation, we will introduce a fundamentally new approach based on random projections and combinatorial optimization. Our approach provides provable guarantees on accuracy, and outperforms traditional methods in a range of domains, in particular those involving combinations of probabilistic and causal dependencies (such as those coming from physical laws) among the variables. This allows for a tighter integration between inductive and deductive reasoning, and offers a range of new modeling opportunities. As an example, we will discuss an application in the emerging field of Computational Sustainability aimed at discovering new fuel-cell materials where we greatly improved the quality of the results by incorporating prior background knowledge of the physics of the system into the model.

BIOGRAPHICAL SKETCH

Stefano Ermon earned his Ph.D. degree in Computer Science at Cornell University in January 2015, working with Professor Carla Gomes and Professor Bart Selman. During his Ph.D. studies, he has co-authored nearly 20 publications, including two Best Paper Awards and one Runner-Up Prize. After graduation, he joined the Department of Computer Science at Stanford University as an Assistant Professor. His research is centered on techniques for scalable and accurate inference in graphical models, statistical modeling of data, large-scale combinatorial optimization, and robust decision-making under uncertainty, and is motivated by a series of applications, in particular ones in the emerging field of computational sustainability. In his research, he develops new foundational methods by combining rigorous theoretical analysis with a principled experimental component, drawing upon ideas from probabilistic reasoning, constraint optimization, information theory, statistical physics, and approximation algorithms. He further applies and evaluates the proposed techniques on concrete real-world applications.

To Anna Lisa and Vito.

ACKNOWLEDGEMENTS

I would like to extend my sincerest gratitude towards all the people who have supported me during my Ph.D. studies at Cornell. This, of course, includes the members of my thesis committee, Carla Gomes, Bart Selman and John Hopcroft. I wish to thank them for their encouragement and constant support, both on the academic and personal levels.

I also wish to thank Ashish Sabharwal, who was a Research Associate when I started my Ph.D studies, and later moved to IBM Research. He is a wonderful research collaborator, and I wish to thank him for the many hours spent discussing research ideas with me. The research presented in this thesis would not have been possible without him.

Finally, I would like to thank all my family for the constant support they gave me from the moment I decided to apply to graduate school and move to the United States.

TABLE OF CONTENTS

Biographical Sketch iii
Dedication iv
Acknowledgements v
Table of Contents vi
List of Tables ix
List of Figures x

1 Introduction **1**
1.1 Foundations of Probabilistic Inference and Decision Making 3
1.2 Probabilistic Inference and Decision Making for Sustainability 9

2 Preliminaries **13**
2.1 Undirected Graphical Models 13
2.1.1 Factor Graph Representation 14
2.1.2 Statistical Inference Problems 15
2.2 Applications of Graphical Models in AI 18
2.3 Universal Hashing 21

3 Partition Function Computation **24**
3.1 Problem Statement and Assumptions 26
3.1.1 Inference in Graphical Models 27
3.1.2 Quadratures of Integrals 28
3.2 The `WISH` Algorithm 29
3.3 Analysis 33
3.3.1 Estimating the Total Weight 34
3.3.2 Estimating the Tail Distribution 35
3.3.3 Improving the Approximation Factor 35
3.3.4 Further Approximations 36
3.4 Experimental Evaluation 37
3.4.1 Provably Accurate Approximations 38
3.4.2 Anytime Usage with Suboptimal Solutions 39
3.4.3 Hard Combinatorial Structures 41
3.4.4 Model Selection 43
3.5 Discussion 44

4 Sampling from Complex Probability Distributions **46**
4.1 Setup and Problem Definition 48
4.2 Sampling by Embed, Project, and Search 49
4.2.1 Weight Discretization 50
4.2.2 Embed: From Weighted to Uniform Sampling 52
4.2.3 Project and Search: Uniform Sampling with Universal Hashing and an NP-oracle 53

4.2.4 Main Results: Sampling with Accuracy Guarantees 56
4.3 Experiments 58
4.3.1 Ising Models 58
4.3.2 Test Case Generation for Software Testing 60
4.4 Discussion 63

5 Optimization with Parity Constraints: Hardness and Practical Approaches 64
5.1 Max-Likelihood Decoding and Worst Case Analysis 65
5.2 MAP Inference by Integer Linear Programming 67
5.2.1 Parity Constraints 69
5.2.2 Solving Integer Linear Programs 70
5.3 The Role of Sparsity 72
5.4 Inducing Sparsity 74
5.5 Message Passing Decoding 77
5.6 Discussion 79

6 Universal Hashing with Sparse Parity Constraints 81
6.1 Probabilistic Inference by Hashing 82
6.1.1 Concentration and Counting 83
6.2 Concentration and Hash Families 85
6.3 Average Universal Hashing 87
6.4 Hashing with Low-Density (Sparse) Parity Constraints 88
6.5 `WISH` With Sparse Parity Constraints 92
6.6 Lower Bounds Using Arbitrarily Sparse Parity Constraints 94
6.6.1 `WISH` with Uniform Hashing 95
6.7 Experimental Evaluation 98
6.7.1 Model Counting for SAT 99
6.8 Discussion 101

7 Probabilistic Inference with Complex Background Knowledge for the Discovery of New Materials 105
7.1 Combinatorial Materials Discovery 107
7.1.1 Phase Map Identification 108
7.2 Prior Work 110
7.3 SMT-Aided Phase Map Identification 111
7.3.1 Model Parameters 113
7.3.2 Variables 114
7.3.3 Constraints Encoding Prior Knowledge 115
7.4 Experimental Results 118
7.4.1 Sampling Phase Maps Using PAWS 125
7.5 Conclusions 127

8 Conclusions and Research Directions **128**
8.1 Summary of Contributions 128
8.2 Future Work 130

A Proofs **133**

LIST OF TABLES

6.1 Provable and empirical minimum length of random XOR constraints to be suitable for model counting for SAT on benchmark instances. 101

7.1 Runtime for analyzing X-Ray diffraction data with a Satisfiability Modulo Theory solver. P is the number of sampled points. K' is the number of unknown basis patterns. e is the number of peaks missing due to measurement errors. 121

7.2 Accuracy of the phase map (crystal structure) obtained with our automatic analysis. P is the number of sampled points. K' is the number of unknown basis patterns. e is the number of peaks missing due to measurement errors. 121

7.3 Comparison between the time needed to find a phase map using Z3 and sampling from the model using PAWS and Z3 as a blackbox. P is the number of sampled points. K' is the number of unknown basis patterns. 126

LIST OF FIGURES

1.1 Key challenges encountered in modeling real world systems: this thesis presents new computational models and algorithmic techniques to deal with these challenges. 2
1.2 Overview of our research in *Computational Sustainability*. The three challenges of high-dimensionality, uncertainty, and utility functions are also at the core of many of these domains. 4

2.1 Complexity of various probabilistic reasoning tasks. One of the main technical contribution of this thesis is a new approach approximate #-P complete probabilistic inference queries by reducing to a small number of NP-equivalent combinatorial optimization problems. 16

3.1 Visualization of the "thinning" effect of random parity constraints, after adding 0, 1, 2, and 3 parity constraints. Leftmost plot shows the original function to integrate. The optimal solution (subject to constraints) is shown in red. 29
3.2 Horizontal vs. vertical slices for integration. 30
3.3 Experimental evaluation on log-partition function estimation for clique-structured Ising models. 38
3.4 Estimation errors for the log-partition function on 10×10 randomly generated Ising Grids. 40
3.5 A partially completed Sudoku puzzle. 42
3.6 Model selection for hand-written digits: confabulations from RBM models trained with Contrastive Divergence CD-k for $k \in \{1, 10, 15\}$. 43

4.1 Estimated marginals vs. true marginals on 8×8 Ising Grid models. Closeness to the 45 degree line indicates accuracy. PAWS is run with $b \in \{1, 2\}, P = 4, \alpha = 1$, and $\ell = 25$ (mixed case) or $\ell = 40$ (attractive case). 58
4.2 Experiments on a software verification benchmark: sampling and marginal estimation. 62

5.1 Pictorial representation of the relationship between max-likelihood decoding and the queries generated by our scheme. While in coding schemes parity check nodes are used to define valid codewords, in our scheme they are used to implement a universal hash function which randomly selects a subset of the original high-dimensional space. Since we are starting from a more general graphical model, the optimization instances are at least as hard as maximum-likelihood decoding. 67

5.2 Upper bound found by solving a sequence of LP relaxations for MAP inference problems subject to parity constraints, as a function of runtime. Integer Linear Programming solvers obtain a sequence of upper bounds that improve over time and are guaranteed to eventually reach the global optimum of the original optimization problem. This solving strategy naturally provides a tradeoff between computational resources used and accuracy. . 72
5.3 Integrality gap at the root node of the search tree of an Integer Linear Programming solver when solving MAP inference queries subject to parity constraints, as a function of the density f (equivalently, average length) of the parity constraints used (random Ising model with parameters $w = 2.5$, $M = 10$). Empirically, problems subject to short parity constraints (small f) are easier to solve (smaller integrality gap on the y axis). 73
5.4 Upper and lower bounds with and without sparsification. Using our preprocessing techniques we can rewrite the parity constraints into an equivalent but much sparser form. This empirically leads to easier optimization problems for which we can obtain better bounds from Linear Programming relaxations. . . . 76
5.5 Optimization with short parity constraints: we compare combinatorial optimization techniques (branch and bound and integer linear programming) with message-passing techniques for the MAP inference problems subject to parity constraints (which generalize standard maximum-likelihood decoding) generated by WISH. Although message-passing techniques are very effective for standard decoding problems, they are empirically outperformed by combinatorial optimization techniques for these more complex probabilistic models. 80

6.1 Evaluation of the bound on the density of the parity constraints required to achieve sufficiently good statistical properties of the corresponding family of hash functions, as a function of the dimensionality n and the number of constraints m. Top: $m = n/2$, varying n. Bottom: $n = 100$, varying m. 103
6.2 Bounds on the partition function of Ising grids with mixed interactions. The bounds are obtained from LP relaxations using both dense and sparse parity constraints implementing respectively Strongly Universal and Average-Universal hash functions. Top: 10×10 grid. Bottom: 15×15 grid. 104

7.1 Left: Pictorial depiction of the phase map identification problem, showing a set of sampled points on a *thin film*. Each sample corresponds to a different composition, and has an associated measured X-ray diffraction pattern. Colors correspond to different combinations of the basis patterns $\alpha, \beta, \gamma, \delta$. On the right: Scaling (shifting) of the diffraction patterns as one moves from one point to a neighboring one. 110
7.2 Phase map for the synthetic Al-Li-Fe system with 45 sampled points. Each of the six colored areas represents one of the basis patterns $(\alpha, \beta, ..., \zeta)$ of the ground truth, while the colored dots correspond to the solution of our SMT model. The SMT results closely delimit each phase of the ground truth, which is quantitatively validated by the high precision and recall scores of our approach. 124
7.3 Median running time as a function of the bound on the total number of missing peaks T. 125

CHAPTER 1

INTRODUCTION

Recent technological advances are rendering the collection, storage, and communication of massive data sets increasingly economical. As a result, statistical models of data, systems, and users are becoming a central component of many scientific and engineering disciplines [14, 57]. In this context, statistical machine learning techniques have proliferated, and have been shown to be extremely successful at numerous predictive tasks, from speech recognition to cancer diagnosis [12]. Given the recent successes of these statistical approaches, there is a growing interest in the scientific community towards rich and complex models that can represent and extract deeper insights into the data and the environment that generated it. To capture rich statistical models, many different representation languages have been introduced, including graphical models [99], Markov Logic Networks [81], BLOG [72], and Church [54].

Making reliable and accurate inferences based on complex statistical models, e.g., to make predictions or to support analytic decision-making, is a key computational problem with a broad range of applications. Unfortunately, most interesting forms of probabilistic reasoning are extremely challenging from the computational point of view [64]. In this context, the long-standing goal of developing practical probabilistic reasoning technology that will allow computers to act intelligently and adaptively in increasingly complex and uncertain real-world environments is a key technical challenge. At a high level, we can identify three major sources of difficulties in achieving this goal:

1. **High dimensionality:** Real-world systems are typically very complex. Realistic models involve a large number of interconnected variables, can be

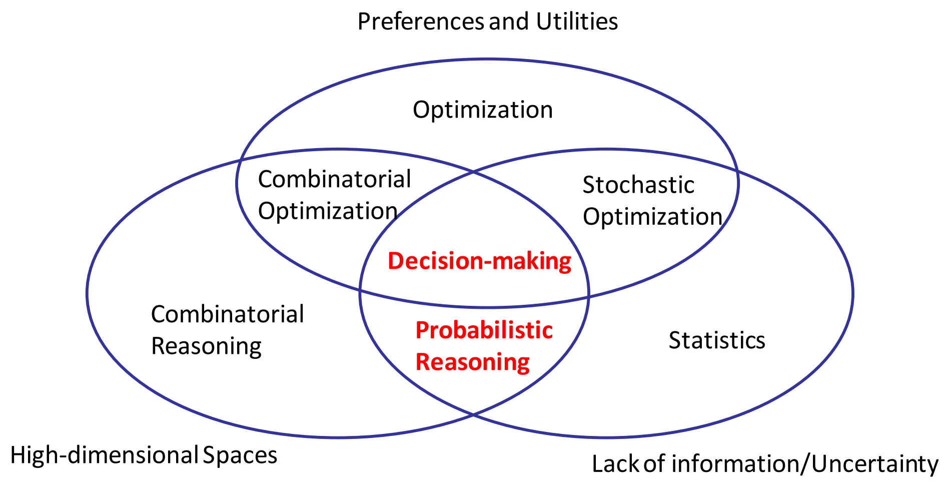


Figure 1.1: Key challenges encountered in modeling real world systems: this thesis presents new computational models and algorithmic techniques to deal with these challenges.

dynamic and spatio-temporal, and may involve multiple self-interested agents, each with a large number of potential actions with many possible outcomes. This naturally leads to very high-dimensional spaces, often combinatorial ones, that we need to consider if we want to build realistic models.

2. **Uncertainty:** We typically have limited information about the systems we are trying to model or control. For example, there could be uncertainty in the parameters, noisy measurements, or unobserved quantities that we cannot directly measure. In most applications it is necessary to consider *probabilistic models* to achieve a reasonable level of robustness.

3. **Preference or utility functions:** When we design a system, we are typically given performance metrics that we wish to optimize for. Other times, we build models to support decision making. In both cases, it is necessary to take into account *preference or utility functions* and *optimize* for them.

This thesis describes our research on developing computational models and algorithmic techniques to to deal with these three fundamental challenges (see Figure 1.1), specifically in terms of *probabilistic reasoning* (how can we make inferences about complex, high-dimensional statistical models) and *decision-making* (where we also consider an objective function to be optimized). We develop new foundational methods by combining rigorous theoretical analysis with a principled experimental component, drawing upon ideas from probabilistic reasoning, constraint optimization, information theory, statistical physics, and approximation algorithms. We further apply and evaluate the proposed techniques on concrete real-world applications, including ones in the new emerging field of *Computational Sustainability*. The first part of the thesis will highlight some of the foundational contributions we have made in terms of probabilistic inference and decision-making under uncertainty. In the second part of the thesis, we will discuss the applications of these ideas to challenging sustainability problems. An overview of the concrete sustainability problems we have considered is presented in Figure 1.2.

1.1 Foundations of Probabilistic Inference and Decision Making

Probabilistic modeling of data is the cornerstone of modern statistical machine learning and statistical applications in general. Probabilistic reasoning — making inferences about complex probabilistic models — is needed to make predictions and learn models from data. General probabilistic reasoning, however, is computationally intractable, and therefore heuristic strategies and approximations are often employed in practice. To date, only a handful of approaches have been proposed. Standard approaches are Markov Chain Monte

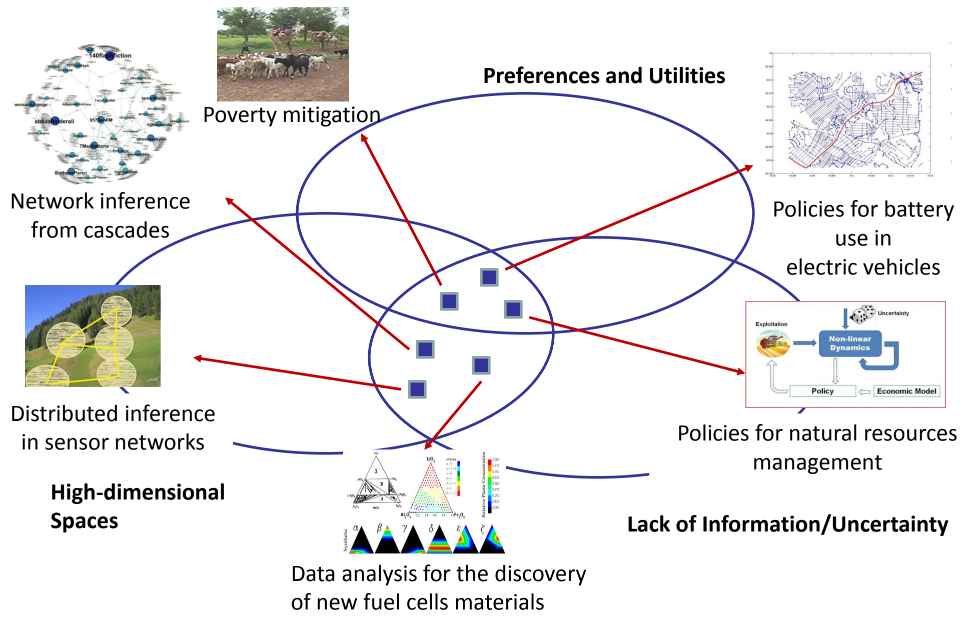

Poverty mitigation
Preferences and Utilities
Network inference from cascades
Policies for battery use in electric vehicles
Distributed inference in sensor networks
Policies for natural resources management
High-dimensional Spaces
Lack of Information/Uncertainty
Data analysis for the discovery of new fuel cells materials
Exploitation
Uncertainty
Non-linear Dynamics
Policy
Economic Model

Figure 1.2: Overview of our research in *Computational Sustainability*. The three challenges of high-dimensionality, uncertainty, and utility functions are also at the core of many of these domains.

Carlo (MCMC) methods (such as Gibbs sampling) [3, 62, 71], variational techniques (based on approximating complex probability distributions by more tractable ones) [63, 99], and decomposition methods (based on dynamic programming) [66]. MCMC techniques, invented many decades ago (1950s), are still the most widely used, and are the workhorse of statistical inference. Unfortunately, such methods typically do not provide tight guarantees on the accuracy of the results. In the first part of the thesis, we will introduce a fundamentally new paradigm for statistical inference which is a very different and promising alternative to these existing techniques and yields *provably accurate results* in a range of problem domains. The method works particularly well when the underlying statistical models combine probabilistic information with causal or deterministic constraints arising from domain knowledge (such as physical laws).

The key idea behind MCMC is that one can answer queries about complex statistical models by drawing a relatively small set of samples (typical scenarios) from the underlying probability distribution and calculate statistics of interest by averaging over the samples, which are representative of the entire (exponentially large) state space. The key difficulty of the approach is that to draw proper samples, one needs to set up a Markov Chain over the entire state space which has to reach an equilibrium distribution. For many statistical models of interest, reaching the equilibrium distribution will require exponential time. In practice, the approach will therefore only give approximate answers. Unfortunately, there is generally little or no information on the quality of the approximation. In fact, the Markov Chain may completely miss important parts of the state space because the chain gets trapped in less relevant areas of the state space.

In our new approach, we also compute statistics of interest by considering only a relatively small set of representative states (samples) from the statistical model. However, our samples are not drawn at random from the underlying probability distribution using a Markov Chain, rather *they are very particular states that can be discovered using state-of-the-art optimization tools and random projections*. More specifically, we obtain these special states by randomly projecting the original high-dimensional space to a lower-dimensional one (using universal hash functions) and then using optimization to look for "extreme states" (configurations or states that are the most likely) in the projected subspace. Quite surprisingly, we can show that a small collection of such extreme states is representative of the overall probability distribution and can be used to answer a range of queries about the original statistical model (e.g., compute the probability of an event according to the model) *with provable accuracy guarantees*. Because current optimization tools can handle large problems, often with a million or more variables, we can quickly ”hunt down” the special states and answer queries much more accurately than other methods.

From a computational complexity perspective, the class of probabilistic inference queries we consider is complete for the #P complexity class [94], a set of problems encapsulating the entire Polynomial Hierarchy and believed to be significantly harder than NP. For example, a canonical #P-complete problem is #-SAT (i.e., the problem of computing the number of satisfying assignments of a propositional formula), which is believed to be harder than SAT (i.e., the problem of determining whether a propositional formula has at least one satisfying assignment). The key idea behind our new method, called **W**eighted-**I**ntegrals-and-**S**ums-by-**H**ashing (WISH), is to reduce the #P-complete probabilistic inference problem to a small number (quasilinear in the dimensionality) of instances of a NP-equivalent combinatorial *optimization problem*. Each instance of the optimization problem is obtained by adding a set of randomly generated parity constraints to the original probabilistic model, thereby projecting to a lower dimensional subspace. For each instance, we find the most likely state using combinatorial optimization. We then aggregate the solutions to answer the original probabilistic inference query, obtaining a *provably accurate estimate with high probability*. The rationale behind this approach is that although combinatorial optimization is intractable in the worst case, it has witnessed great success in the past 50 years in fields such as Mixed Integer Programming (MIP) and propositional Satisfiability Testing (SAT). Finding the most likely state for a probabilistic model, although NP-hard, can in practice often be approximated or solved exactly fairly efficiently [21, 86]. In fact, modern solvers can exploit structure in real-world problems and prune large portions of the search space, often dramatically reducing the runtime. In contrast, in a #P counting problem such as computing a marginal probability, one needs to consider contributions from all possible states (variable assignments). Since we can incorporate any optimization tool, our approach can leverage current and potential future advances in combinatorial optimization, leading to a method that scales well on many real world problems and, at the same time, provides approximation guarantees. Further, since the optimization instances are independent of each other, our approach is highly parallelizable and can thus handle problems at a level of accuracy and scale well beyond the current state of the art.

We further uncover a surprising connection between the optimizations and random projections used in our WISH scheme and the maximum likelihood decoding problem in communication theory [10, 97]. Finding an extreme state for a randomly projected probabilistic model can be seen as a generalization of the problem of decoding a message transmitted over a noisy communication channel using an error-correcting code. Exploiting this connection, we show how to leverage ideas from popular decoding algorithms based on message passing and linear programming [38]. These techniques are known to be very successful in practice in decoding certain types of codes such as low density parity check (LDPC) codes [40]. By adapting problem encodings and developing novel preprocessing techniques, we are able to make the optimization problems generated by WISH significantly easier to solve in practice, drastically reducing the runtime. We are also able to obtain bounds on the value of probabilistic inference queries by considering tractable relaxations of the combinatorial optimization problems which can be solved in polynomial time. These bounds can be further improved using a new, more tractable class of random projections that we introduced and can be implemented using a construction that resembles that of low-density parity check codes [37]. These bounds are empirically much tighter (often, by several orders of magnitude) than those obtained using other techniques, and can be iteratively improved in an any-time fashion.

Random projections and optimization techniques can also be used to solve the problem of sampling from a probability distribution defined over a high-dimensional space, specified by a graphical model. We introduce a new sampling algorithm, called PAWS, based on an embedding into an even higher-dimensional space which is then randomly projected (using universal hash functions) to a lower-dimensional subspace and explored using combinatorial search methods [33]. Our scheme can again leverage fast combinatorial optimization tools, and the samples produced are guaranteed to be as close as desired to the target probability distribution. The use of powerful complete search methods allows PAWS to reason about a rich class of models that also include intricate dependencies among the variables, e.g., as specified by deterministic or causal relationships representing prior knowledge about the system. In contrast, these multi-modal distributions with extreme, near-zero probabilities are extremely difficult for MCMC and variational techniques. Our approach allows for a tighter integration between inductive and deductive reasoning, and is especially useful in scientific data analysis applications where there is often a large amount of background knowledge that can be incorporated into the model. We apply this idea to a data analysis problem in materials science where the goal is to facilitate and speed up the discovery of new solar and fuel cell materials. More specifically, we encode our background knowledge about the physical and chemical properties of materials using a Satisfiability Modulo Theory (SMT) reasoning framework. Using state-of-the-art SMT solvers and PAWS, we are able to automatically analyze X-ray diffraction measurements, generating interpretations that are physically meaningful and very accurate.

1.2 Probabilistic Inference and Decision Making for Sustainability

Moving towards a more sustainable society and economy is one of the grand challenges of the upcoming century. New strategies are urgently needed to address the effects of rapidly increasing urbanization, including soaring demands of natural resources such as energy, water, and land. Solving the complex environmental, economic, and societal issues of sustainability will require contributions from multiple disciplines, and it is becoming clear that *computational techniques have the potential to play a major role* [49]. Addressing sustainability issues in a quantitative way is however challenging for two reasons. First, we typically lack accurate models of the complex processes we want to control, and the models we have are riddled with uncertainties. Second, making inferences, predictions or devising optimal intervention strategies is difficult because of the complexity of the models, which typically involve uncertainty, complex spatio-temporal interactions, and multiple self-interested agents. We have investigated a number of concrete problems in this space (See Figure 1.2), ranging from finding policies for renewable resource management to the automatic analysis of X-ray diffraction data to aid materials scientists in the discovery of new fuel cell materials. Our research in this area is based on collaborations with researchers in other fields such as resource economics, ecology, and materials science. Below are some of the projects we have worked on at the *Institute for Computational Sustainability*; in the second part of the thesis, we will discuss in detail the materials discovery application.

Sequential decision-making under uncertainty Several problems in sustainability can be formulated as management or control problems for a stochastic dynamical system, e.g., modeling the dynamics of an ecosystem. Working with environmental scientists and resource economists, we introduced a new class of dynamic bio-economic models that capture many real-world renewable resource allocation problems, such as fisheries and forestry management [24, 26]. As a new approach to deal with uncertainty and environmental risks, we introduced a novel game theoretic framework, where the management problem is equivalent to a dynamic game. By exploiting provable structural properties of the optimal policy, we devised a fast policy optimization algorithm for this class of models, and we applied it to a model we constructed for the Northern Pacific Halibut marine fishery using biological and economic data. We obtained a policy that is structurally very different from the one currently employed, and significantly outperforms historical policies in terms of *minimizing the risk of an ecosystem collapse*. In follow-up work [30], we generalized this approach with a new framework to introduce worst-case guarantees in decision theoretic planning models, providing a fast dynamic programming algorithm to compute an optimal policy using an admissible heuristic to prune the search space. In a related project, we showed that we can significantly improve the efficiency of battery systems in electric vehicles using automatic planning and sequential decision-making methods [32, 36]. Specifically, we formalized the problem of optimizing real-time energy management of multi-battery systems as a stochastic planning problem, and we proposed a novel solution based on a combination of optimization, machine learning, and data-mining techniques. We evaluated the performance of our new energy management system on a large dataset of real commuter trips crowd-sourced in the United States, and we showed that our policy outperforms the leading algorithms that were previously proposed as part of an open algorithmic challenge, leading to up to 10% increased driving range with the same energy use.

Stochastic processes on networks As sensing and computing devices become ubiquitous, we face a growing need to develop scalable inference methods to analyze in real-time the large amounts of data they produce. In [25, 28], we demonstrated the use of message passing and graphical models as a framework to efficiently solve distributed inference tasks (e.g., tracking a global state from local information) and study the dynamics of multi-agent coordination. We developed a novel message-passing inference algorithm that outperforms current state of the art techniques, such as distributed Kalman filtering. We also revisited the problem of inferring latent network structure given observations from a diffusion process, such as the spread of trending topics in social media or animals through a fragmented landscape [100]. We defined a family of novel probabilistic models that can explain recurrent cascading behavior, and take into account not only the time differences between events but also a richer set of additional features. We provided a tractable inference algorithm that scales to large real-world networks and significantly improves the accuracy compared to previous approaches.

Combinatorial materials discovery Fuel cells are one of the most promising enabling technologies for electric vehicles and for renewable energy storage. However, their efficiency depends crucially on the catalysts used, and there is a quest among materials scientists to find more effective alternatives to the use of platinum as a catalyst. One of the most promising approaches is the so-called combinatorial method, where scientists search for new materials with desirable properties by obtaining measurements on hundreds of samples in a single high-throughput batch experiment. As manual data analysis is becoming more and more impractical, there is a growing need to develop new techniques to automatically analyze and interpret such data. Working closely with materials scientists at the Cornell High Energy Synchrotron, we introduced a novel approach to the problem of identifying the underlying crystal structure of sample materials using X-ray diffraction data. The main novelty is that we integrated domain-specific scientific background knowledge about the physical and chemical properties of the materials into a constraint-based model [31]. We then used our PAWS technique to analyze the data, using a Satisfiability Modulo Theory (SMT) solver to search for interpretations of the data that are consistent with our prior knowledge. By explicitly taking into account the physics of the system, our approach provides accurate interpretations of the data, in contrast with previous approaches that often do not provide physically meaningful results.

Developing successful computational tools to help address sustainability challenges requires a constant flow of ideas between foundational and applied research. Research in Computational Sustainability often leads to a whole new range of computational techniques and applications, benefiting not only the scientific community but also society at large.

CHAPTER 2
PRELIMINARIES

We begin by providing some background on graphical models [19, 64, 99]. Graphical models are a general framework to specify complex probability distributions in a compact way. The key idea is to use a graph structure to define a family of probability distributions that factorize according to the underlying graph. In this thesis, we will mainly focus on *discrete* and *undirected* graphical models.

2.1 Undirected Graphical Models

Let $G = (V, E)$ be a graph with vertex set V and edge set $E \subseteq V \times V$. We associate with each vertex $i \in V$ a discrete random variable x_i where $x_i \in \mathcal{X}_i$, and $\mathcal{X}_i$ is a discrete set. For any subset $\alpha \subseteq V$ of the vertex set V , we define the subvector $\{x\}_\alpha := \{x_i, i \in \alpha\}$. Similarly, we define $\otimes_{i\in\alpha}\mathcal{X}_i$ to be the Cartesian product of $\mathcal{X}_i$ for all $i \in \alpha$. The global random vector $x = \{x_s, s \in V\}$ takes value in the Cartesian product $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_N$. We consider a probability distribution over $x \in \mathcal{X}$ (also called *configurations*) that factorizes into functions defined on the cliques of the graph G. A clique C is a subset of vertices that are fully connected, that is for all $s, t \in C$, $(s, t) \in E$. Let $\psi_C : \otimes_{i\in C}\mathcal{X}_i \mapsto \mathbb{R}^+$ be a compatibility function associated with clique C. An undirected graphical model is a collection of distribution for the joint random vector that factorizes as follows

$$p(x) = p(x_1, \cdots, x_N) = \frac{1}{Z}\prod_{C\in\mathcal{C}} \psi_C(\{x\}_C) \tag{2.1}$$

where Z is a normalization constant and $\mathcal{C}$ is a set of (maximal) cliques. For compactness we also introduce a weight function $w : \mathcal{X} \to \mathbb{R}^+$ that assigns to each configuration $x \in \mathcal{X}$ its unnormalized probability, namely

$$w(x) = \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha) \tag{2.2}$$

With this notation, the normalization constant Z ensuring that the probabilities sum up to one, also known as the *partition function*, is defined as

$$Z = \sum_{x \in \mathcal{X}} w(x) = \sum_{x \in \mathcal{X}} \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha) \tag{2.3}$$

2.1.1 Factor Graph Representation

We consider an equivalent representation for graphical models in terms of a *factor graph*. Let $x_i, i \in V$ be a collection of $N = |V|$ discrete random variables, where $x_i \in \mathcal{X}_i$ and $\mathcal{X}_i$ is a discrete set. Let $p(x)$ be a joint probability distribution over $x \in \mathcal{X}$

$$p(x) = \frac{1}{Z} \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha) \tag{2.4}$$

that factors into potentials or factors $\psi_\alpha : \otimes_{i \in \alpha} \mathcal{X}_i \mapsto \mathbb{R}^+$, where $\mathcal{I}$ is an index set and $\{x\}_\alpha \subseteq V$ a subset of variables the factor ψ_α depends on.The corresponding *factor graph* is a bipartite graph with vertex set $V \cup \mathcal{I}$. In the factor graph, each variable node $i \in V$ is connected with all the factors $\alpha \in \mathcal{I}$ that depend on i. Similarly, each factor node $\alpha \in \mathcal{I}$ is connected with all the variable nodes $i \in \{x\}_\alpha$. An undirected graphical model can be represented as a factor graph by letting $\mathcal{I}$ index the collection $\mathcal{C}$ of cliques.

2.1.2 Statistical Inference Problems

Given a probability distribution $p(x)$ defined by a graphical model, there are several statistical inference problems that are of particular interest in statistical machine learning applications:

1. **Marginalization:** compute the marginal distribution $p(x_A)$ for a subset $A \subseteq V$ of the variables
2. **Partition function:** compute the normalization constant or *partition function Z* defined as in equation (2.3)
3. **Sampling:** randomly output an element $x \in \mathcal{X}$, where each element x is selected with probability $p(x)$
4. **MAP Inference:** find a mode of $p(x)$, i.e. identify a configuration $\widehat{x} \in \mathcal{X}$ which is assigned the largest probability by the model, that is $\widehat{x} \in \arg\max p(x)$

Computational Complexity

All these computational problems are known to be intractable in the worst case [64]. Specifically, problems (1) and (2) are "counting problems" (they can be thought as the problem of counting the number of accepting paths of an appropriately chosen Turing Machine) and are known to be #-P complete [83, 94]. Sampling is essentially just as hard, since sampling and counting are known to be self reducible [62]. Intuitively, if one has access to samples from $p(x)$, statistics of interests such as the partition function Z or marginal probabilities can be computed via sample averaging (using a Monte Carlo approximation). On the

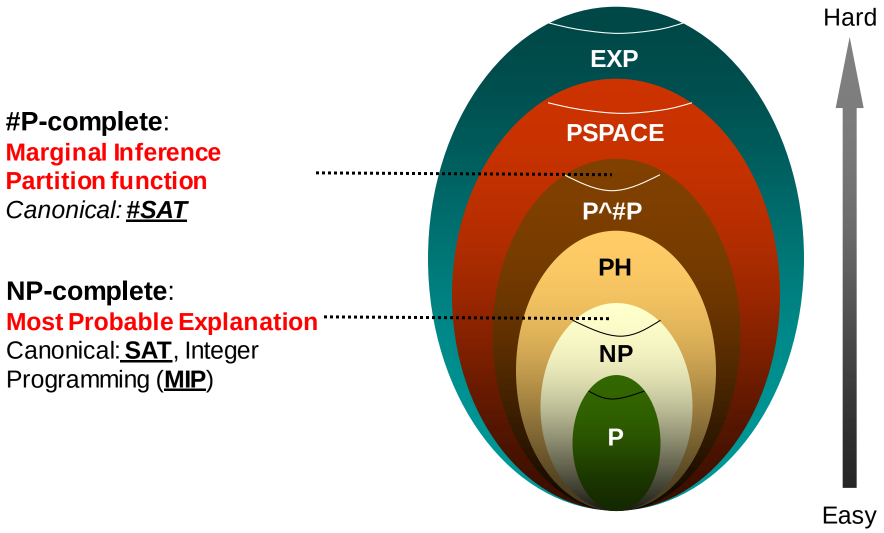


Figure 2.1: Complexity of various probabilistic reasoning tasks. One of the main technical contribution of this thesis is a new approach approximate #-P complete probabilistic inference queries by reducing to a small number of NP-equivalent combinatorial optimization problems.

other hand, it is easy to generate samples by sequentially sampling individual variables according to their marginal probabilities. These theoretical reductions however can be inefficient in practice, so there is interest in developing separate, specialized algorithms for solving problems (1),(2) and (3) directly. Problem (4) (MAP inference) is fundamentally different than problems (1),(2) and (3). In fact, problem (4) is an NP-equivalent *optimization* problem, a complexity class believed to be easier in the worst case than #-P, as can be seen in the widely believed hierarchy of complexity classes reported in Figure 2.1. A key result of this thesis is to highlight connections between these problems. Specifically, we introduce a new computational approach where problems (1),(2) and (3) can be approximated with provable guarantees if one has access to an oracle that solves problem (4).

Previous Approaches

There is a vast literature on probabilistic inference methods, and providing details is beyond the scope of the thesis. We briefly introduce the three main families of inference techniques and provide references for the interested reader:

1. Markov Chain Monte Carlo methods: the idea is to set up a Markov Chain which converges in the limit to the target distribution $p(x)$. The chain is designed so that simulating each step is computationally tractable. If one runs (simulates) the chain for a long enough time, the chain will converge to the target distribution and it directly provides samples. Unfortunately, these Markov Chains are guaranteed to converge quickly (rapidly mixing) only on a very restricted class of probability distributions, and in general they require exponential time for convergence (consistently with the hardness of sampling). We refer the reader to [62, 71] for an excellent review of the subject.
2. Variational Techniques: the basic idea is to approximate the intractable target probability distribution $p(x)$ with one that is more tractable. This is typically achieved by choosing a distribution from a family of distributions that are computationally easier to work with, and minimizing a measure of divergence. We refer the reader to [63, 99] for more details.
3. Decomposition-based: the idea is to leverage dynamic programming to *exactly* solve the inference problem when the distribution can be decomposed in a special way (low three-width condition). The method is exact but only applies to a restricted number of probability distributions [21, 66]. Variants of the method can be applied to both MAP, partition function and marginal inference problems.

2.2 Applications of Graphical Models in AI

The graphical model formalism has been used extensively to study a wide range of problems in AI. We illustrate two canonical examples in this section.

Constraint Satisfaction Problems

Graphical models are a very powerful modeling language and can be used to encode general constraint satisfaction problems such as Satisfiability Testing (SAT), the prototypical NP-complete decision problem.

Let V be a set of propositional (Boolean) variables, where $|V| = N$. A formula F is said to be in clausal normal form (CNF) form if it is a logical conjunction of a set of clauses $\mathcal{C}$. A clause C is a logical disjunction of a set of (possibly negated) variables, such as for example $(x_1 \vee \neg x_2 \vee x_3)$. A variable assignment $\sigma : V \to \{0, 1\}$ is a function that assigns a value in $\{0, 1\}$ to each variable in V . As usual, the value 0 is interpreted as FALSE and the value 1 as TRUE. We say that σ satises a clause C if at least one signed variable of C is TRUE. Satisfiability Testing is the problem of deciding if there exists a variable assignment that satisfies all the clauses $\mathcal{C}$ (equivalently, such that F, which is the conjunction of the clauses, evaluates to TRUE).

Let F be a formula in CNF over a set V of variables with $m = |\mathcal{C}|$ clauses. We can represent the formula F as a factor graph as follows. We have a collection of $N = |V|$ discrete binary random variables $x_i \in \{0, 1\}$ for each $i \in V$ (one random variable for each variable in the formula F). We define a probability distribution using $m = |\mathcal{C}|$ factors, one for each clause in the formula F. Each factor corresponds to a clause $C \in \mathcal{C}$, and is defined as $\psi_C : \otimes_{i \in C}\{0,1\} \mapsto \{0,1\}$, where $\psi_C(\{x\}_C) = 1$ if an only if the clause C is satisfied by $\{x\}_C$, and $\psi_C(\{x\}_C) = 0$ otherwise. For example, a clause $(x_1 \vee \neg x_2)$ would have a corresponding factor $\psi_{(x_1 \vee \neg x_2)} : \{0,1\}^2 \mapsto \{0,1\}$, where $\psi_{(x_1 \vee \neg x_2)}(0,0) = 1$, $\psi_{(x_1 \vee \neg x_2)}(0,1) = 0$, $\psi_{(x_1 \vee \neg x_2)}(1,0) = 1$, $\psi_{(x_1 \vee \neg x_2)}(1,1) = 1$.

It is easy to verify from equation (2.4) that a variable assignment $x \in \{0,1\}^N$ is assigned a probability 0 if it violates at least one clause $C \in \mathcal{C}$. On the other hand, a variable assignment $x \in \{0,1\}^N$ that satisfies all the clauses, also called a *satisfying assignment* is assigned a probability $1/Z$ where Z is equal to the number of distinct satisfying assignments. Therefore, the model can also be thought as defining a uniform probability distribution over the subset of the Boolean hypercube $\{0,1\}^N$ given by the satisfying assignments of F.

The problem of Satisfiability testing, namely deciding whether there exists at least one satisfying assignment, can be formulated as a MAP inference query for the graphical model we just defined. Computing the partition function is equivalent to computing the total number of satisfying assignments, a problem also known as $\#$-SAT. The marginal probabilities of a variable x_i are equal to the fraction of satisfying assignments having variable x_i set to TRUE and FALSE, respectively.

Error Correcting Codes

Error correcting codes are one of the key tools in communication theory to improve the reliability of communication networks, including computer, cellular and television ones. Intuitively, error correcting codes are used to add some level of redundancy to messages we wish to transmit over noisy communication channels, so that transmission errors can be detected and perhaps even corrected. As a simple example, suppose Alice wants to send a message $m \in \{0,1\}^N$ (represented as a vector of N bits) to Bob. The communication channel is noisy, so that each bit might be corrupted (for example, flipped) with probability λ during the transmission (independently). To improve the reliability of their communication, Alice could add a single redundant bit p_1 at the end of the message m, where p_1 is equal to the parity of the bits in m. It is easy to see that Bob can detect if a transmission error occurred on a single bit, because in that case the received parity bit will not match the parity of the received message. More advanced error correcting codes allow Alice and Bob to identity (and recover from) a larger number of transmission errors, while minimizing the overhead (redundancy) introduced by the code.

Important families of error correcting codes can be understood and described using the graphical model formalism. Suppose Alice and Bob have agreed on a valid codebook $\mathcal{C} \subseteq \{0,1\}^N$, which is a subset of all possible messages that are considered to be valid. As in the previous example, a common way to define the codebook is to use a set of h linear equations modulo 2, that is $\mathcal{C} = \{x \in \{0,1\}^N : Ax = b \bmod 2\}$, where $A \in \{0,1\}^{h \times n}$ and $b \in \{0,1\}^h$.

We can formalize this probabilistic formulation of the communication problem in the language of graphical models as follows. Let $y = (y_1, \cdots, y_N) \in \{0,1\}^N$ be the (noisy) message that was actually received by Bob. We construct a graphical model with N binary variables $x_1, \cdots, x_N$ representing the unknown message x that Alice has sent to Bob. We then have N factors

$$\psi_{x_i}(x_i) = \begin{cases} 1-\lambda & , x_i = y_i \\ \lambda & , x_i \neq y_i \end{cases}$$

Assuming $\lambda < 1/2$ (transmission over the channel succeeds more often than not), this is setting a bias towards values of x that are close (in Hamming distance) to the message $y$actually received by Bob. We also know that x must be a valid codeword, that is $x \in \mathcal{C}$. This is encoded using h factors, one for each of the equations modulo 2 defining $\mathcal{C}$. Let α_j denote the set of variables involved in the j-th equation, namely $\alpha_j = \{i : A_{ij} = 1\}$. Then for each $j = 0, \cdots, h-1$ we introduce a factor

$$\psi_{\alpha_j}(\{x\}_{\alpha_j}) = \begin{cases} 1 & , \text{ if } \bigoplus_{i \in \alpha_j} x_i = b_j \\ 0 & , \text{ otherwise} \end{cases}$$

Given the noisy message y actually received, Bob wants to guess which was the message m Alice had sent to him. Bob attempts to find what was the (valid, i.e. in the codebook) message Alice had sent to him by finding the most likely value of x according to the model, that is solving a MAP inference query for the graphical model described. This is a famous NP-hard optimization problem known as MAX-LIKELIHOOD-DECODING. In addition to their importance in communication and information theory, parity based codes and the maximum likelihood decoding problems will also play an important role in the new probabilistic inference techniques described in this thesis.

2.3 Universal Hashing

We start with definitions of standard universal hash functions [cf. 93, 48].

Definition 1. A family of functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is ϵ-SU (Strongly Universal) if the following two conditions hold when H is chosen uniformly at random from $\mathcal{H}$. 1) $\forall x \in \{0,1\}^n$, the random variable $H(x)$ is uniformly distributed in $\{0,1\}^m$. 2) $\forall x_1, x_2 \in \{0,1\}^n$ $x_1 \neq x_2$, $\forall y_1, y_2 \in \{0,1\}^m$, it holds that $P[H(x_1) = y_1, H(x_2) = y_2] \leq \epsilon/2^m$.

It can be verified that $\epsilon \geq 1/2^m$, and the case $\epsilon = 1/2^m$ corresponds to pairwise independent hash functions, defined as follows:

Definition 2. A family of functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is pairwise independent if the following two conditions hold when H is chosen uniformly at random from $\mathcal{H}$. 1) $\forall x \in \{0,1\}^n$, the random variable $H(x)$ is uniformly distributed in $\{0,1\}^m$. 2) $\forall x_1, x_2 \in \{0,1\}^n$ $x_1 \neq x_2$, the random variables $H(x_1)$ and $H(x_2)$ are independent.

Statistically optimal functions can be constructed by considering the family $\mathcal{H}$ of all possible functions from $\{0,1\}^n$ to $\{0,1\}^m$. It is easy to verify that this is a family of *fully* independent functions. However, functions from this family require $m2^n$ bits to be specified, making this construction not very useful for large n. On the other hand, *pairwise independent* hash functions can be specified compactly. They are generally based on modular arithmetic constraints of the form $Ax = b \mod 2$, referred to as parity or XOR constraints.

Proposition 1. *Let $A \in \{0,1\}^{m \times n}$, $b \in \{0,1\}^m$. The family $\mathcal{H} = \{h_{A,b}(x) : \{0,1\}^n \to \{0,1\}^m\}$ where $h_{A,b}(x) = Ax + b \mod 2$ is a family of pairwise independent hash functions.*

Pairwise independent hash functions can be constructed more efficiently by letting A be a random $i \times n$ Toeplitz matrix [79]. Specifically, the first column and row of A are filled with uniform i.i.d. Bernoulli variables in $\{0, 1\}$. The value of each entry is then copied into the corresponding descending top-left to bottom-right diagonal. This process requires $n + i - 1$ random bits rather than $ni = O(n^2)$. Let $\mathcal{T}(m, n) \subseteq \{0, 1\}^{m \times n}$ be the set of $m \times n$ Toeplitz matrices with $0, 1$ entries. Then:

Proposition 2 ([48, 90]). *Let $A \in \mathcal{T}(m, n)$, $b \in \{0, 1\}^m$. The family $\mathcal{H}_{\mathcal{T}}^{n,m} = \{h_{A,b}(x) : \{0, 1\}^n \rightarrow \{0, 1\}^m\}$ where $h_{A,b}(x) = Ax + b \mod 2$ is a family of pairwise independent hash functions.*

The space $\mathcal{C} = \{x : h_{A,b}(x) = p\}$ has a nice geometric interpretation as the translated nullspace of the random matrix A, which is a finite dimensional vector space, with operations defined on the field $\mathbb{F}(2)$ (arithmetic modulo 2). We will refer to constraints of the form $Ax = b \mod 2$ as parity constraints, as they can be rewritten in terms of logical XOR operations as $A_{i1}x_1 \oplus A_{i2}x_2 \oplus \cdots \oplus A_{in}x_n = b_i$.

CHAPTER 3

PARTITION FUNCTION COMPUTATION

Many probabilistic inference tasks involve the computation of high-dimensional integrals (see Chapter 2 for more details), where intuitively one has to look at all possible scenarios, and weight each one according to how likely it is according to the model. For example, this is needed to evaluate probabilities of events (e.g., to make predictions based on the model), and more generally to compute expectations. Computing integrals in very *high dimensional spaces* is a fundamental and largely unsolved problem of scientific computation [23, 84, 15], with numerous applications not only in machine learning and statistics but also in biology and physics. As the volume grows exponentially in the dimensionality, the problem quickly becomes computationally intractable, a phenomenon traditionally known as the *curse of dimensionality* [9].

In this chapter, we revisit the problem of approximately computing *discrete integrals*, namely weighted sums over (extremely large) sets of items. This problem encompasses several important probabilistic inference tasks, such as computing marginals or normalization constants (partition function) in discrete graphical models, which are in turn cornerstones for parameter and structure learning [99]. Although we focus on the discrete case, continuous cases can in principle also be addressed, as they can be approximated using *discrete integrals* via numerical integration.

We introduce a randomized scheme that computes with high probability ($1-\delta$ for any desired $\delta > 0$) an approximately correct estimate (within a factor of $1+\epsilon$ of the true value for any desired $\epsilon > 0$) for general weighted sums defined over exponentially large sets of items, such as the set of all possible variable assignments in a discrete probabilistic graphical model. From a computational complexity perspective, the counting problem we consider is complete for the #P complexity class [94], a set of problems encapsulating the entire Polynomial Hierarchy and believed to be significantly harder than NP.

The key idea is to reduce this #P problem to a small number (polynomial in the dimensionality) of instances of a NP-equivalent[1] combinatorial optimization problem defined on the same space and subject to randomly generated "parity" constraints. The rationale behind this approach is that although combinatorial optimization is intractable in the worst case, it has witnessed great success in the past 50 years in fields such as Mixed Integer Programming (MIP) and propositional Satisfiability Testing (SAT). Problems such as computing a Maximum a Posteriori (MAP) assignment, although NP-hard, can in practice often be approximated or solved exactly fairly efficiently [78, 80, 82, 86]. In fact, modern solvers can exploit structure in real-world problems and prune large portions of the search space, often dramatically reducing the runtime. In contrast, in a #P counting problem such as computing a marginal probability, one needs to consider contributions of an exponentially large number of items.

Our algorithm, called **W**eighted-**I**ntegrals-And-**S**ums-By-**H**ashing (`WISH`), relies on randomized hashing techniques to probabilistically "evenly cut" a high dimensional space. Such hashing was introduced by Valiant et al. [95] (see also [85, 91]) to study the relationship between the number of solutions and the hardness of a combinatorial search. These techniques were also applied by Gomes et al. Gomes et al. [50] and Chakraborty et al. Chakraborty et al. [17] to uniformly sample solutions for the SAT problem and to obtain bounds on their number [51]. Our work is more general in that it can handle general weighted

[1]As hard as the hardest problem in NP, but not harder.

sums, such as the ones arising in probabilistic inference for graphical models. Our work is also closely related to recent work by Hazan and Jaakkola [58], who obtain bounds on the partition function by taking suitable expectations of a combination of MAP queries over randomly perturbed models. We improve upon this in two crucial aspects, namely, our estimate is a constant factor approximation of the true partition function (while their bounds have no tightness guarantee), and we provide a concentration result showing that our bounds hold not just in expectation but with high probability with a polynomial number of MAP queries. Note that this is consistent with known complexity results regarding #P and $\mathrm{BPP}^{\mathrm{NP}}$; see Remark 1 below.

We demonstrate the practical efficacy of the `WISH` algorithm in the context of computing the partition function of random Clique-structured Ising models, Grid Ising models with known ground truth, and a challenging combinatorial application (Sudoku puzzle) completely out of reach of techniques such as Mean Field and Belief Propagation. We also consider the Model Selection problem in graphical models, specifically in the context of hand-written digit recognition. We show that our "anytime" and highly parallelizable algorithm can handle these problems at a level of accuracy and scale well beyond the state of the art.

3.1 Problem Statement and Assumptions

Let Σ be a (large) set of items. Let $w : \Sigma \to \mathbb{R}^+$ be a non-negative function that assigns a weight to each element of Σ. We wish to (approximately) compute the total weight of the set, defined as the following discrete integral or "partition function"

$$W = \sum_{\sigma \in \Sigma} w(\sigma) \tag{3.1}$$

We assume w is given as input and that it can be compactly represented, for instance in a factored form as the product of conditional probabilities tables. Note however that our results are more general and do not rely on a factored representation.

Assumption: We assume that we have access to an *optimization oracle* that can solve the following constrained optimization problem

$$\max_{\sigma \in \Sigma} \; w(\sigma) 1_{\{\mathcal{C}\}}(\sigma) \tag{3.2}$$

where $1_{\{\mathcal{C}\}} : \Sigma \to \{0, 1\}$ is an indicator function for a compactly represented subset $\mathcal{C} \subseteq \Sigma$, i.e., $1_{\{\mathcal{C}\}}(\sigma) = 1$ iff $\sigma \in \mathcal{C}$. For concreteness, we discuss our setup and assumptions in the context of probabilistic graphical models, which is our motivating application.

3.1.1 Inference in Graphical Models

Given a graphical model, we let $\Sigma = \mathcal{X}$ be the set of all possible configurations (variable assignments). Define a weight function $w : \mathcal{X} \to \mathbb{R}^+$ that assigns to each configuration a score proportional to its probability: $w(x) = \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha)$. Z may then be rewritten as

$$Z = \sum_{x \in \mathcal{X}} w(x) = \sum_{x \in \mathcal{X}} \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha) \tag{3.3}$$

Computing Z is typically intractable because it involves a sum over an exponential number of configurations, and is often the most challenging inference task for many families of graphical models. Computing Z is however needed for many inference and learning tasks, such as evaluating the likelihood of data for a given model, computing marginal probabilities, and parameter estimation [99].

In the context of graphical models inference (see 2.1.2), we assume to have access to an optimization oracle that can answer Maximum a Posteriori (MAP) queries, namely, solve the following constrained optimization problem

$$\arg\max_{x \in \mathcal{X}} \; p(x \mid \mathcal{C}) \tag{3.4}$$

that is, we can find the most likely state (and its weight) given some evidence $\mathcal{C}$. This is a strong assumption because MAP inference is known to be an NP-hard problem in general. Notice however that computing Z is a #P-complete problem, a complexity class believed to be even harder than NP.

3.1.2 Quadratures of Integrals

Suppose we are given a quadrature for a continuous (multidimensional) integral of a function $f : \mathbb{R}^n \to \mathbb{R}^+$ over a high dimensional set $S \subseteq \mathbb{R}^n$

$$\int_S f(\mathbf{x}) \mathrm{d}\mathbf{x} \approx \sum_{x \in \mathcal{X}} w(x) = W$$

where $\mathcal{X}$ is some discretization of S (e.g., grid based), and $w(x)$ approximates the integral of $f(\mathbf{x})$ over the corresponding element of volume. In this case, we require a compact representation for w and access to an oracle able to optimize the discretized function, subject to arbitrary constraints. See, e.g., Figure 3.1.

For simplicity, in the following we will restrict ourselves to the binary case, i.e., $\Sigma = \mathcal{X} = \{0, 1\}^n$. The general multinomial case where the sum is over

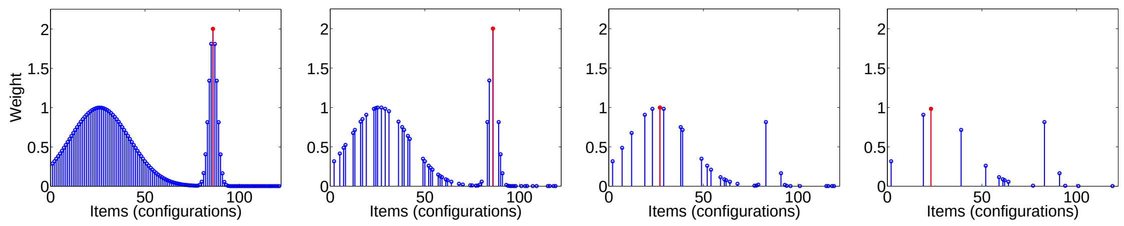


Figure 3.1: Visualization of the "thinning" effect of random parity constraints, after adding 0, 1, 2, and 3 parity constraints. Leftmost plot shows the original function to integrate. The optimal solution (subject to constraints) is shown in red.

$\mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_N$ can be transformed into the former case using a binary representation, requiring $\lceil \log_2 |\mathcal{X}_i| \rceil$ bits (binary variables) per dimension i.

3.2 The `WISH` Algorithm

We start with the intuition behind our algorithm to approximate the value of W called **W**eighted-**I**ntegrals-And-**S**ums-By-**H**ashing (`WISH`).

Computing W as defined in Equation (3.1) is challenging because the sum is defined over an exponentially large number of items, i.e., $|\Sigma| = 2^n$ when there are n binary variables. Let us define the *tail distribution* of weights as $G(u) \triangleq |\{\sigma \mid w(\sigma) \geq u\}|$. Note that G is a non-increasing step function, changing values at no more than 2^n points. Then W may be rewritten as $\int_{\mathbb{R}^+} G(u) \mathrm{d}u$, i.e., the total *area* A under the $G(u)$ vs. u curve. One way to approximate W is to (implicitly) divide this area A into either *horizontal* or *vertical* slices (see Figure 3.2), approximate the area in each slice, and sum up.

Suppose we had an efficient procedure to estimate $G(u)$ given any u. Then it is not hard to see that one could create enough slices by dividing up the x-axis, estimate $G(u)$ at these points, and estimate the area A using quadrature.

However, the natural way of doing this to any degree of accuracy would require a number of slices that grows at least logarithmically with the weight range on the x-axis, which is undesirable.

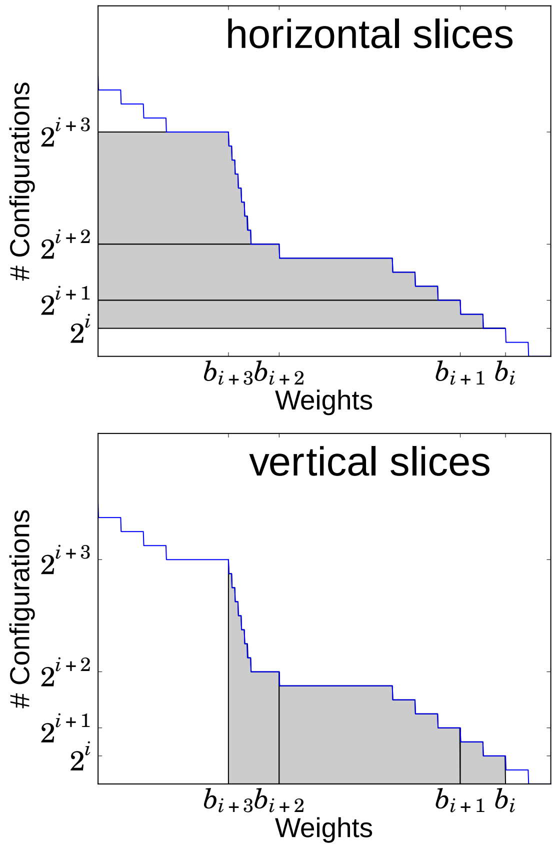


Figure 3.2: Horizontal vs. vertical slices for integration.

Alternatively, one could split the y-axis, i.e., the $G(u)$ value range $[0, 2^n]$, at geometrically growing values $1, 2, 4, \cdots, 2^n$, i.e., into bins of sizes $1, 1, 2, 4, \cdots, 2^{n-1}$. Let $b_0 \geq b_1 \geq \cdots \geq b_n$ be the weights of the configurations at the split points. In other words, b_i is the 2^i-th quantile of the weight distribution. Unfortunately, despite the monotonicity of $G(u)$, the area in the horizontal slice defined by each bin is difficult to bound, as b_i and b_{i+1} could be arbitrarily far from each other. However, the area in the *vertical* slice defined by b_i and b_{i+1} must be bounded between $2^i(b_i - b_{i+1})$ and $2^{i+1}(b_i - b_{i+1})$, i.e., within a factor of 2. Thus, summing over the lower bound for all such slices and the left-most slice, the total area A must be within a factor of 2 of $\sum_{i=0}^{n-1} 2^i(b_i - b_{i+1}) + 2^n b_n = b_0 + \sum_{i=1}^{n} 2^{i-1} b_i$. Of course, we don't know b_i. But if we could approximate each b_i within a factor of p, we would get a $2p$-approximation to the area A, i.e., to W.

`WISH` provides an efficient way to realize this strategy, using a combination of randomized hash functions and an optimization oracle to approximate the b_i values with high probability. Note that this method allows us to compute the partition function W (or the area A) by estimating weights b_i at $n+1$ carefully chosen points, which is "only" an optimization problem.

The key insight to compute the b_i values is as follows. Suppose we apply to configurations in Σ a randomly sampled pairwise independent hash function with 2^m buckets and use an optimization oracle to compute the weight w_m of a *heaviest* configuration in a fixed (arbitrary) bucket. If we repeat this process T times and consistently find that $w_m \geq w^*$, then we can infer by the properties of hashing that at least 2^m configurations (globally) are likely to have weight at least w^*. By the same token, if there were in fact at least 2^{m+c} configurations of a heavier weight $\hat{w} > w^*$ for some $c > 0$, there is a good chance that the optimization oracle will find $w_m \geq \hat{w}$ and we would not underestimate the weight of the 2^m-th heaviest configuration. As we will see shortly, this process, using pairwise independent hash functions to keep variance low, allows us to estimate b_i accurately with only $T = \mathrm{O}(\ln n)$ samples.

The pseudocode of `WISH` is shown as Algorithm 1. It is parameterized by the weight function w, the dimensionality n, a correctness parameter $\delta > 0$, and a constant $\alpha > 0$. Notice that the algorithm requires solving only $\Theta(n \ln n/\delta)$ op-

Algorithm 1 WISH $(w : \Sigma \rightarrow \mathbb{R}^+, n = \log_2 |\Sigma|, \delta, \alpha)$

```
$T \leftarrow \left\lceil \frac{\ln(n/\delta)}{\alpha} \right\rceil$
for $i = 0, \cdots, n$ do
    for $t = 1, \cdots, T$ do
        Sample hash function $h^i_{A,b} : \Sigma \rightarrow \{0,1\}^i$, i.e.
            sample uniformly $A \in \{0,1\}^{i \times n}$, $b \in \{0,1\}^i$
        $w^t_i \leftarrow \max_\sigma w(\sigma)$ subject to $A\sigma = b \mod 2$
    end for
    $M_i \leftarrow \text{Median}(w^1_i, \cdots, w^T_i)$
end for
Return $M_0 + \sum_{i=0}^{n-1} M_{i+1} 2^i$
```

timization instances (MAP inference) to compute a sum defined over 2^n items. In the following section, we formally prove that the output is a constant factor approximation of W with probability at least $1 - \delta$ (probability over the choice of hash functions). Figure 3.1 shows the working of the algorithm. As more and more random parity constraints are added in the outer loop of the algorithm ("levels" increasing from 1 to n), the configuration space is (pairwise-uniformly) thinned out and the optimization oracle selects the heaviest (in red) of the surviving configurations. The final output is a weighted sum over the median of T such modes obtained at each level.

Remark 1. The parity constraints $A\sigma = b \bmod 2$ do not change the worst-case complexity of the optimization problem, which remains NP-easy. Our result is thus consistent with the fact that #P can be approximated in BPP$^{\text{NP}}$, that is, one can approximately count the number of solutions with a randomized algorithm and a polynomial number of queries to an NP oracle [48].

Remark 2. Although the parity constraints we impose are simple linear equations over a field, they can make the optimization harder. For instance, finding a configuration with the smallest Hamming weight satisfying a set of parity constraints is known to be NP-hard, i.e. equivalent to computing the minimum distance of a parity code [10, 97]. On the other hand, most low density parity check codes can be solved extremely fast in practice using heuristic methods such as message passing. We will discuss these aspects in much more detail in Chapter 5.

Remark 3. Each of the optimization instances can be solved independently, allowing natural massive *parallelization*. We will also discuss how the algorithm can be used in an *anytime* fashion, and the implications of obtaining suboptimal solutions.

3.3 Analysis

Since many configurations can have identical weight, it will help for the purposes of the analysis to fix, w.l.o.g., a weight-based ordering of the configurations, and a natural partition of the $|\Sigma| = 2^n$ configurations into $n+1$ bins that the ordering induces.

Definition 3. Fix an ordering $\sigma_i, 1 \leq i \leq 2^n$, of the configurations in Σ such that for $1 \leq j < 2^n$, $w(\sigma_j) \geq w(\sigma_{j+1})$. For $i \in \{0, 1, \cdots, n\}$, define $b_i \triangleq w(\sigma_{2^i})$. Define a special *bin* $B \triangleq \{\sigma_1\}$ and, for $i \in \{0, 1, \cdots, n-1\}$, define *bin* $B_i \triangleq \{\sigma_{2^i+1}, \sigma_{2^i+2}, \cdots, \sigma_{2^{i+1}}\}$.

Note that bin B_i has precisely 2^i configurations. Further, for all $\sigma \in B_i$, it follows from the definition of the ordering that $w(\sigma) \in [b_{i+1}, b_i]$. This allows us to bound the sum of the weights of configurations in B_i (the "horizontal" slices) between $2^i b_{i+1}$ and $2^i b_i$.

3.3.1 Estimating the Total Weight

Our main theorem, whose proof relies on the two lemmas below, is that Algorithm 1 provides a constant factor approximation to the partition function. The complete proof of the theorem and all lemmas may be found in the Appendix.

Lemma 1. *Let $M_i = \mathrm{Median}(w_i^1, \cdots, w_i^T)$ be defined as in Algorithm 1 and b_i as in Definition 3. Then, for any $c \geq 2$, there exists $\alpha^*(c) > 0$ such that for $0 < \alpha \leq \alpha^*(c)$,*

$$\Pr\left[M_i \in [b_{\min\{i+c,n\}}, b_{\max\{i-c,0\}}]\right] \geq 1 - \exp(-\alpha T)$$

Lemma 2. *Let $L' \triangleq b_0 + \sum_{i=0}^{n-1} b_{\min\{i+c+1,n\}} 2^i$ and $U' \triangleq b_0 + \sum_{i=0}^{n-1} b_{\max\{i+1-c,0\}} 2^i$. Then $U' \leq 2^{2c} L'$.*

Theorem 1. *For any $\delta > 0$ and positive constant $\alpha \leq 0.0042$, Algorithm 1 makes $\Theta(n \ln n/\delta)$ MAP queries and, with probability at least $(1-\delta)$, outputs a 16-approximation of $W = \sum_{\sigma \in \Sigma} w(\sigma)$.*

Proof Sketch. It is clear from the pseudocode that it makes $\Theta(n \ln n/\delta)$ MAP queries. For accuracy analysis, we can write W as:

$$\begin{aligned} W &\triangleq \sum_{j=1}^{2^n} w(\sigma_j) = w(\sigma_1) + \sum_{i=0}^{n-1} \sum_{\sigma \in B_i} w(\sigma) \\ &\in \left[b_0 + \sum_{i=0}^{n-1} b_{i+1} 2^i, b_0 + \sum_{i=0}^{n-1} b_i 2^i \right] \triangleq [L, U] \end{aligned}$$

Note that $U \leq 2L$ because $2L = 2b_0 + \sum_{i=0}^{n-1} b_{i+1} 2^{i+1} = b_0 + \sum_{\ell=0}^{n} b_\ell 2^\ell \geq U$. Hence, if we had access to the true values of all b_i, we could obtain a 2-approximation to W. We do not know true b_i values, but Lemma 1 shows that the M_i values computed by Algorithm 1 are sufficiently close to b_i with high probability. Specifically, applying Lemma 1 with $T = \left\lceil \frac{\ln(n/\delta)}{\alpha} \right\rceil$, we can show that with probability at least $(1-\delta)$, the output of Algorithm 1 lies in $[L', U']$ as defined in

Lemma 2. Observing that $[L, U]$ is contained in $[L', U']$ and applying Lemma 2, we have a 2^{2c}-approximation of W. Fixing $c = 2$ and noting that $\alpha^*(2) \geq 0.0042$ finishes the proof. $\square$

3.3.2 Estimating the Tail Distribution

We can also estimate the entire tail distribution of the weights, defined as $G(u) \triangleq |\{\sigma \mid w(\sigma) \geq u\}|$.

Theorem 2. *Let M_i be defined as in Algorithm 1, $u \in \mathbb{R}^+$, and $q(u)$ be the maximum i such that $\forall j \in \{0, \cdots, i\}, M_j \geq u$. Then, for any $\delta > 0$, with probability $\geq (1 - \delta)$, $2^{q(u)}$ is an 8-approximation of $G(u)$ computed using $\mathrm{O}(n \ln n/\delta)$ MAP queries.*

While this is an interesting result in its own right, if the goal is to estimate the total weight W, then the scheme in Section 3.3.1, requiring a total of only $\Theta(n \ln n/\delta)$ MAP queries, is more efficient than first estimating the tail distribution for several values of u.

3.3.3 Improving the Approximation Factor

Given a κ-approximation algorithm such as Algorithm 1 and any $\epsilon > 0$, we can design a $(1 + \epsilon)$-approximation algorithm with the following construction. Let $\ell = \log_{1+\epsilon} \kappa$. Define a new set of configurations $\Sigma^\ell = \Sigma \times \Sigma \times \cdots \times \Sigma$, and a new weight function $w' : \Sigma^\ell \to \mathbb{R}$ as $w'(\sigma_1, \cdots, \sigma_\ell) = w(\sigma_1)w(\sigma_2) \cdots w(\sigma_\ell)$.

Proposition 3. *Let $\widehat{W}$ be a κ-approximation of $\sum_{\sigma' \in \Sigma^\ell} w'(\sigma')$. Then $\widehat{W}^{1/\ell}$ is a $\kappa^{1/\ell}$-approximation of $\sum_{\sigma \in \Sigma} w(\sigma)$.*

To see why this holds, observe that $W' = \sum_{\sigma' \in \Sigma^\ell} w'(\sigma') = \left(\sum_{\sigma \in \Sigma} w(\sigma)\right)^\ell = W^\ell$. Since $\frac{1}{\kappa} W' \leq \widehat{W} \leq \kappa W'$, we obtain that $\widehat{W}^{1/\ell}$ must be a $\kappa^{1/\ell} = 1 + \epsilon$ approximation of W.

Note that this construction requires running Algorithm 1 on an enlarged problem with ℓ times more variables. Although the number of optimization queries grows polynomially with ℓ, increasing the number of variables might significantly increase the runtime.

3.3.4 Further Approximations

When the instances defined in the inner loop are not solved to optimality, Algorithm 1 still provides approximate *lower bounds* on W with high probability. Similarly, if one has access to upper bounds to the values of the optimization instances, the output of the algorithm using these upper bounds is an *approximate upper bound* with high probability.

Theorem 3. *Let $\widetilde{w}_i^t$ be suboptimal solutions for the optimization problems in Algorithm 1, i.e., $\widetilde{w}_i^t \leq w_i^t$. Let $\widetilde{W}$ be the output of Algorithm 1 with these suboptimal solutions. Then, for any $\delta > 0$, with probability at least $1 - \delta$, $\frac{\widetilde{W}}{16} \leq W$. Similarly, let $\widehat{w}_i^t$ be upper bounds for the optimization problems in Algorithm 1, i.e., $\widehat{w}_i^t \geq w_i^t$. Let $\widehat{W}$ be the output of Algorithm 1 using these upper bounds. Then, for any $\delta > 0$, with probability at least $1 - \delta$, $\widehat{W} \geq \frac{W}{16}$. Further, if $\widetilde{w}_i^t \geq \frac{1}{L} w_i^t$ for some $L > 0$, then with probability at least $1 - \delta$, $\widetilde{W}$ is a $16L$-approximation to W.*

The output is always an approximate lower bound, even if the optimization is stopped early. The lower bound is monotonically non-decreasing over time, and is guaranteed to eventually reach within a constant factor of W. We thus have an *anytime* algorithm.

3.4 Experimental Evaluation

We implemented `WISH` using the open source solver ToulBar2 [1] to solve the MAP inference problem. ToulBar2 is a complete solver (i.e., given enough time, it will find an optimal solution and provide an optimality certificate), and it was one of the winning algorithms in the UAI-2010 inference competition. We augmented ToulBar2 with the IBM ILOG CPLEX CP Optimizer 12.3 based techniques borrowed from Gomes et al. [53] to efficiently handle the random parity constraints. Specifically, the set of equations $Ax = b \mod 2$ are linear equations over the field $\mathbb{F}(2)$ and thus allow for efficient propagation and domain filtering using Gaussian Elimination.

For our experiments, we run `WISH` in parallel using a compute cluster with 642 cores. We assign each optimization instance in the inner loop to one core, and finally process the results when all optimization instances have been solved or have reached a timeout.

For comparison, we consider Tree Reweighted Belief Propagation [98] which provides an upper bound on Z, Mean Field [99] which provides a lower bound, and Loopy Belief Propagation [74] which provides an estimate with no guarantees. We use the implementations available in the LibDAI library [73].

3.4.1 Provably Accurate Approximations

For our first experiment, we consider the problem of computing the partition function, Z (cf. Eqn. (3.3)), of random Clique-structured Ising models on n binary variables $x_i \in \{0, 1\}$ for $i \in \{1, \cdots, n\}$. The interaction between x_i and x_j is defined as $\psi_{ij}(x_i, x_j) = \exp(-w_{ij})$ when $x_i \neq x_j$, and 1 otherwise, where w_{ij} is uniformly sampled from $[0, w\sqrt{|i-j|}]$ and w is a parameter set to 0.2. We further inject some structure by introducing a closed chain of strong repulsive interactions uniformly sampled from $[-10w, 0]$. We consider models with n ranging from 10 to 60. These models have treewidth n and can be solved exactly (by brute force) only up to about $n = 25$ variables.

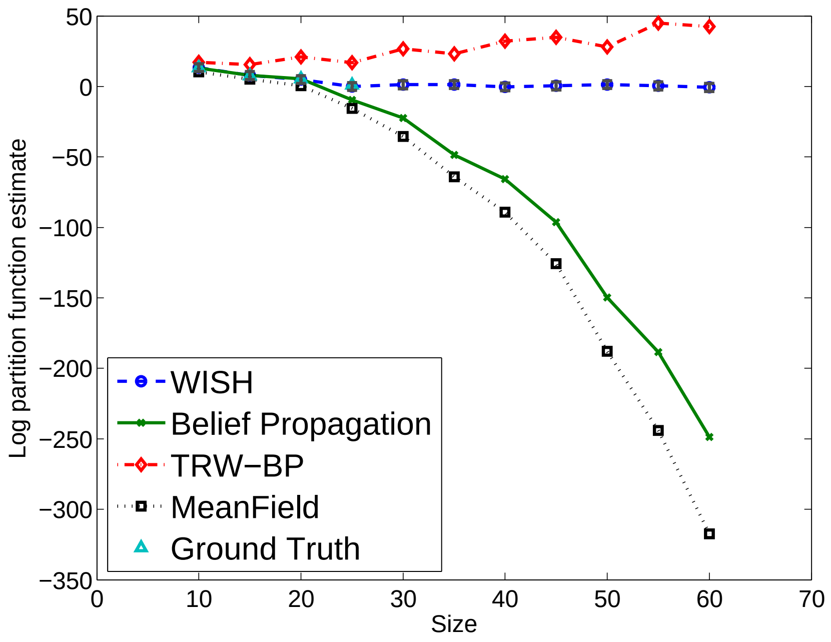


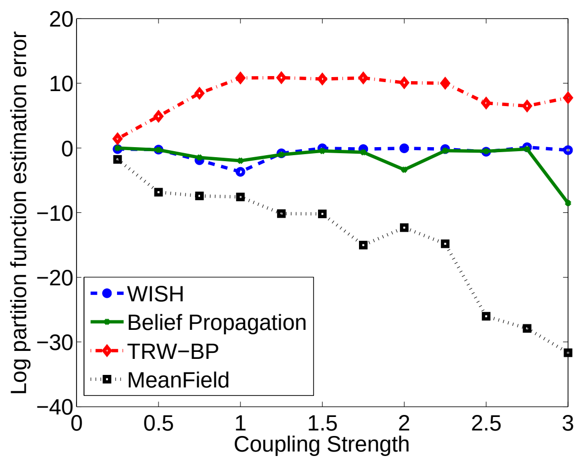
Figure 3.3: Experimental evaluation on log-partition function estimation for clique-structured Ising models.

Figure 3.3 shows the results using various methods for varying problem size. We also computed ground truth for $n \leq 25$ by brute force enumeration. While other methods start to diverge from the ground truth at around $n = 25$, our estimate, as predicted by Theorem 1, remains very accurate, visually overlapping in the plot. The actual estimation error is much smaller than the worst-case factor of 16 guaranteed by Theorem 1, as in practice over- and under-estimation errors tend to cancel out. For $n > 25$ we don't have ground truth, but other methods fall *well outside* the provable interval provided by `WISH`, reported as an error bar that is very small compared to the magnitude of errors made by the other methods.

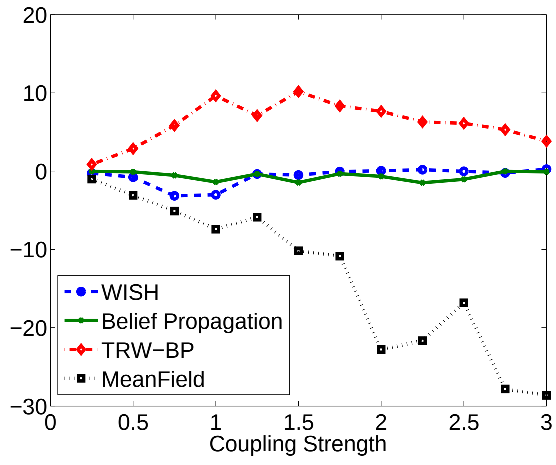
All optimization instances generated by `WISH` for $n \leq 60$ were solved (in parallel) to optimality within a timeout of 8 hours, resulting in high confidence tight approximations of the partition function. We are not aware of any other practical method that can provide such guarantees for counting problems of this size, i.e., a weighted sum defined over 2^{60} items.

3.4.2 Anytime Usage with Suboptimal Solutions

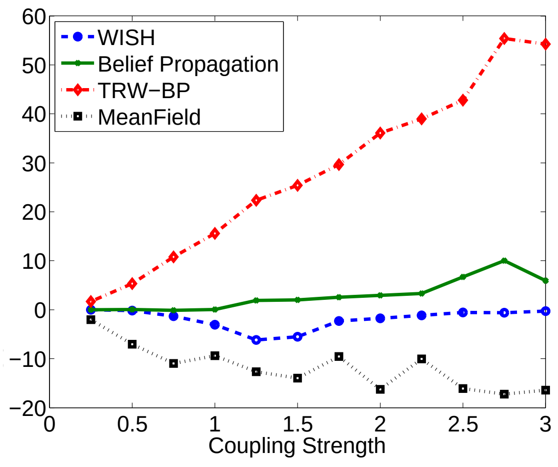
Next, we investigate the quality of our results when not all of the optimization instances can be solved to optimality because of timeouts, so that the strong theoretical guarantees of Theorem 1 do not apply (although Theorem 3 still applies). We consider 10×10 binary Grid Ising models, for which ground truth can be computed using the junction tree method [66]. We use the same experimental setup as Hazan and Jaakkola [58], who also use random MAP queries to derive bounds (without a tightness guarantee) on the partition function. Specifically, we have $n = 100$ binary variables $x_i \in \{-1, 1\}$ with interaction $\psi_{ij}(x_i, x_j) = \exp(w_{ij} x_i x_j)$. For the attractive case, we draw w_{ij} from $[0, w]$; for the mixed case, from $[-w, w]$. The "local field" is $\psi_i(x_i) = \exp(f_i x_i)$ where f_i is sam-



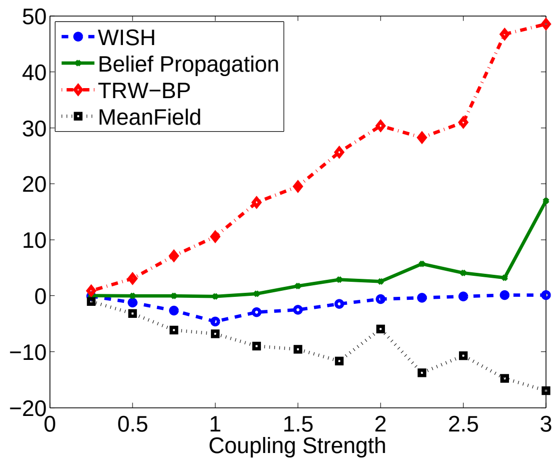
(a) Attractive. Field 0.1.



(b) Attractive. Field 1.0.



(c) Mixed. Field 0.1.



(d) Mixed. Field 1.0.

Figure 3.4: Estimation errors for the log-partition function on 10×10 randomly generated Ising Grids.

pled uniformly from $[-f, f]$, where f is a parameter with value 0.1 or 1.0.

Figure 3.4 reports the estimation *error* for the log-partition function, when using a timeout of 15 minutes. We see that `WISH` provides accurate estimates for a wide range of weights, often improving over all other methods. The slight performance drop of `WISH` for coupling strengths $w \approx 1$ appears to occur because in that weight range the terms corresponding to $i \approx n/2$ parity constraints are the most significant in the output sum $M_0 + \sum_{i=0}^{n-1} M_{i+1} 2^i$. Empirically, optimization instances with roughly $n/2$ parity constraints are often the hardest to solve, resulting in possibly a significant underestimation of the value of $W = Z$ when a timeout occurs. We do not directly compare with the work of Hazan and Jaakkola [58] as we did not have access to their code. However, a visual look at their plots suggests that `WISH` would provide an improvement in accuracy, although with longer runtime.

3.4.3 Hard Combinatorial Structures

An interesting and combinatorially challenging graphical model arises from Sudoku, which is a popular number-placement puzzle where the goal is to fill a 9×9 grid (see Figure 3.5) with digits from $\{1, \cdots, 9\}$ so that the entries in each row, column, and 3×3 block composing the grid, are all distinct. The puzzle can be encoded as a graphical model with 81 discrete variables with domain $\{1, \cdots, 9\}$, with potentials $\psi_\alpha(\{x\}_\alpha) = 1$ if and only if all variables in $\{x\}_\alpha$ are different, and $\alpha \in \mathcal{I}$ where $\mathcal{I}$ is an index set containing the subsets of variables in each row, column, and block. This defines a uniform probability distribution over all valid complete Sudoku grids (a non-valid grid has probability zero), and the normalization constant Z_s equals the total number of valid grids. It is known that $Z_s \approx 6.671 \times 10^{21}$. This number was computed exactly with a combination of computer enumeration and clever exploitation of symmetry properties [39]. Here, we attempt to approximately compute this number using the general-purpose scheme `WISH`.

First, following Felgenhauer and Jarvis [39], we simplify the problem by fixing the first block as in Figure 3.5, obtaining a new problem over 72 variables

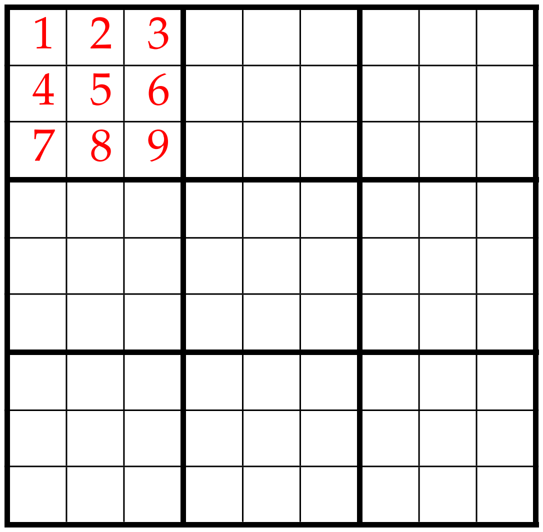


Figure 3.5: A partially completed Sudoku puzzle.

whose normalization constant is $Z' = Z_s/9! \approx 2^{54}$. Next, since we are dealing with a feasibility rather than optimization problem, we replace ToulBar2 with CryptoMiniSAT [88], a SAT solver designed for unweighted cryptographic problems and which natively supports parity constraints. We observed that `WISH` can consistently find solutions (60% of the times) after adding 52 random parity constraints, while for 53 constraints the success rate drops below 0.5, at 45%. Therefore $M_i = 1$ in Algorithm 1 for $i \leq 52$ and there should thus be at least $2^{52} \cdot 9! \approx 1.634 \times 10^{21}$ solutions to the Sudoku puzzle. Although Theorem 1 cannot be applied due to timeouts for larger values of i, this estimate is clearly very close to the known true count. In contrast, the simple "local reasoning" done by variational methods is not powerful enough to find even a single solution. Mean Field and Belief Propagation report an estimated solution count of $\exp(-237.921)$ and $\exp(-119.307)$, resp., on a relaxed problem where violating a constraint gives a penalty $\exp(-10)$ (similar results are obtained using a wide range of weights to model hard constraints). A sophisticated adapative MCMC approach tailored for (weighted) SAT instances [27] reports 5.6822×10^{21} solutions, with a runtime of about 45 minutes.

3.4.4 Model Selection

Many inference and learning tasks require computing the normalization constant of graphical models. For instance, it is needed to evaluate the likelihood of observed data for a given model. This is necessary for Model Selection, i.e., to rank candidate models, or to trigger early stopping during training when the likelihood of a validation set starts to decrease, in order to avoid overfitting [22].

We train Restricted Boltzmann Machines (RBM) [59] using Contrastive Divergence (CD) [102, 16] on MNIST hand-written digits dataset. In an RBM there is a layer of n_h hidden binary variables $h = h_1, \cdots, h_{n_h}$ and a layer of n_v binary visible units $v = v_1, \cdots, v_{n_v}$. The joint probability distribution is given by $P(h, v) = \frac{1}{Z}\exp(b'v + c'h + h'Wv)$. We use $n_h = 50$ hidden units and $n_v = 196$ visible units. We learn the parameters b, c, W using CD-k for $k \in \{1, 10, 15\}$, where k denotes the number of Gibbs sampling steps used in the inference phase, with 15 training epochs and minibatches of size 20.

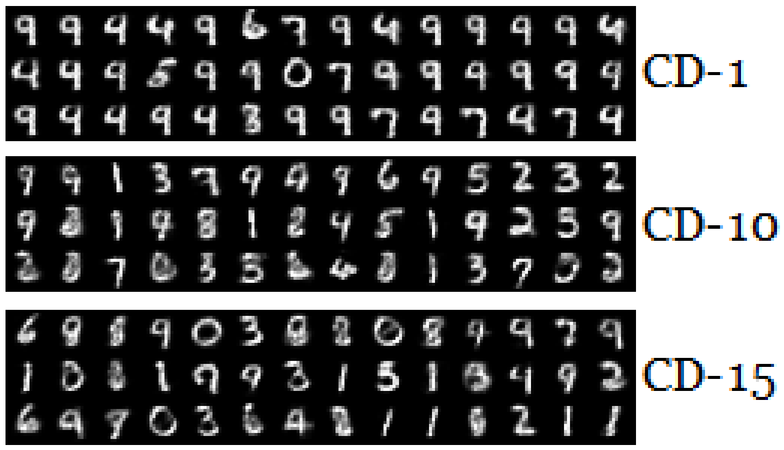


Figure 3.6: Model selection for hand-written digits: confabulations from RBM models trained with Contrastive Divergence CD-k for $k \in \{1, 10, 15\}$.

Figure 3.6 depicts confabulations (samples generated with Gibbs sampling) from the three learned models. To evaluate the loglikelihood of the data and determine which model is the best, one needs to compute Z. We use WISH to estimate this quantity, with a timeout of 10 minutes, and then rank the models according to the average loglikelihood of the data. The scores we obtain are $-41.70, -40.35, -40.01$ for $k = 1, 10, 15$, respectively (larger scores means higher likelihood). In this case ToulBar2 was not able to prove optimality for all instances, so only Theorem 3 applies to these results. Although we do not have ground truth, it can be seen that the ranking of the models is consistent with what visually appears closer to a large collection of hand-written digits in Figure 3.6. Note that $k = 1$ is clearly not a good representative, because of the highly uneven distribution of digit occurrences. The ranking of WISH is also consistent with the fact that using more Gibbs sampling steps in the inference phase should provide better gradient estimates and therefore a better learned model. In contrast, Mean Field results in scores $-35.47, -36.08, -36.84$, resp., and would thus rank the models in reverse order of what is visually the most representative order.

3.5 Discussion

We introduced WISH, a randomized algorithm that, with high probability, gives a constant-factor approximation of a general discrete integral defined over an exponentially large set. WISH reduces the intractable counting problem to a small number of instances of a combinatorial optimization problem subject to parity constraints used as a hash function. In the context of graphical models, we showed how to approximately compute the normalization constant, or partition function, using a small number of MAP queries. Using state-of-the-art combinatorial optimization tools, we are thus able to provide discrete integral or partition function estimates with approximation guarantees at a scale that could till now be handled only heuristically. One advantage of our method is that it is massively parallelizable, allowing it to easily benefit from the increasing availability of large compute clusters. Finally, it is an anytime algorithm which can also be stopped early to obtain empirically accurate estimates that provide lower bounds with a high probability.

CHAPTER 4

SAMPLING FROM COMPLEX PROBABILITY DISTRIBUTIONS

Sampling techniques are one of the most widely used approaches to approximate probabilistic reasoning for high-dimensional probability distributions where exact inference is intractable. In fact, many statistics of interest can be estimated from sample averages based on a sufficiently large number of samples. Since this can be used to approximate #P-complete inference problems, sampling is also believed to be computationally hard in the worst case [71, 62].

Sampling from a succinctly specified combinatorial space is believed to much harder than *searching* the space. Intuitively, not only do we need to be able to find areas of interest (e.g., modes of the underlying distribution) but also to balance their relative importance. Typically, this is achieved using Markov Chain Monte Carlo (MCMC) methods. MCMC techniques are a specialized form of *local search* that only allows moves that maintain detailed balance, thus guaranteeing the right occupation probability once the chain has *mixed*. However, in the context of hard combinatorial spaces with complex internal structure, mixing times are often exponential. An alternative is to use complete or systematic search techniques such as Branch and Bound for integer programming, DPLL for SATisfiability testing, and constraint and answer-set programming (CP & ASP), which are preferred in many application areas, and have witnessed a tremendous success in the past few decades. It is therefore a natural question whether one can construct sampling techniques based on these more powerful *complete search methods* rather than local search.

Prior work in cryptography by Bellare et al. [8] showed that it is possible to uniformly sample witnesses of an NP language leveraging universal hash functions and using only a small number of queries to an NP-oracle. This is significant because samples can be used to approximate #P-complete (counting) problems [62], a complexity class believed to be much harder than NP. Practical algorithms based on these ideas were later developed [29, 50, 17] to near-*uniformly* sample solutions of propositional SATisfiability instances, using a SAT solver as an NP-oracle. However, unlike SAT, most models used in Machine Learning, physics, and statistics are *weighted* (represented, e.g., as graphical models) and cannot be handled using these techniques.

We fill this gap by extending this approach, based on hashing-based projections and NP-oracle queries, to the weighted sampling case. Our algorithm, called PAWS, uses a form of approximation by quantization [47] and an embedding technique inspired by slice sampling [77], before applying projections. This parallels the work presented in Chapter 3 [35], which extended similar ideas for unweighted counting to the weighted counting world, addressing the problem of computing the partition function. Although in theory one could use that technique to produce samples by estimating ratios of discrete integrals [71, 62], the general sampling-by-counting reduction requires a large number of such estimates (proportional to the number of variables) for each sample. Further, the accuracy guarantees on the sampling probability quickly become loose when taking ratios of estimates. In contrast, PAWS is a more direct and practical sampling approach, providing better accuracy guarantees while requiring a much smaller number of NP-oracle queries per sample.

Answering NP-oracle queries, of course, requires exponential time in the worst case, in accordance with the hardness of sampling. We rely on the fact that combinatorial search tools, however, are often extremely fast in practice, and any complete solver can be used as a black box in our sampling scheme. Another key advantage is that when combinatorial search succeeds, our analysis provides a certificate that, with high probability, any samples produced will be distributed within an (arbitrarily small) constant factor of the desired probability distribution. In contrast, with MCMC methods it is generally hard to assess whether the chain has mixed. We empirically demonstrate that PAWS outperforms MCMC as well as variational methods on hard synthetic Ising Models and on a real-world test case generation problem for software verification.

4.1 Setup and Problem Definition

We are given a probability distribution p over a (high-dimensional) discrete set $\mathcal{X}$, where the probability of each item $x \in \mathcal{X}$ is proportional to a weight function $w : \mathcal{X} \to \mathbb{R}^+$, with $\mathbb{R}^+$ being the set of non-negative real numbers. Specifically, given $x \in \mathcal{X}$, its probability $p(x)$ is given by

$$p(x) = \frac{w(x)}{Z} \quad , \quad Z = \sum_{x \in \mathcal{X}} w(x)$$

where Z is a normalization constant known as the *partition function*. We assume w is specified compactly, e.g., as the product of factors or in a conjunctive normal form. As our driving example, we consider the case of undirected discrete graphical models [99] with $n = |V|$ random variables $\{x_i, i \in V\}$ where each x_i takes values in a finite set $\mathcal{X}_i$. We consider a factor graph representation for a joint probability distribution over elements (or *configurations*) $x \in \mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$:

$$p(x) = \frac{w(x)}{Z} = \frac{1}{Z} \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha). \tag{4.1}$$

This is a compact representation for $p(x)$ based on the weight function $w(x) = \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\{x\}_\alpha)$, defined as the product of potentials or factors $\psi_\alpha : \{x\}_\alpha \mapsto \mathbb{R}^+$, where $\mathcal{I}$ is an index set and $\{x\}_\alpha \subseteq V$ the subset of variables factor ψ_α depends on. For simplicity of exposition, without loss of generality, we will focus on the case of binary variables, where $\mathcal{X} = \{0, 1\}^n$.

We consider the fundamental problem of (approximately) *sampling* from $p(x)$, i.e., designing a randomized algorithm that takes w as input and outputs elements $x \in \mathcal{X}$ according to the probability distribution p. This is a hard computational problem in the worst case. In fact, it is more general than NP-complete decision problems (e.g., sampling solutions of a SATisfiability instance specified as a factor graph entails finding at least one solution, or deciding there is none). Further, samples can be used to approximate #P-complete problems [62], such as estimating a marginal probability.

4.2 Sampling by Embed, Project, and Search

Conceptually, our sampling strategy has three steps, described in Sections 4.2.1, 4.2.2, and 4.2.3, resp. (1) From the input distribution p we construct a new distribution p' that is "close" to p but more discrete. Specifically, p' is based on a new weight function w' that takes values only in a discrete set of geometrically increasing weights. (2) From p', we define a *uniform* probability distribution p'' over a carefully constructed higher-dimensional *embedding* of $\mathcal{X} = \{0, 1\}^n$. The previous discretization step allows us to specify p'' in a compact form, and sampling from p'' can be seen to be precisely equivalent to sampling from p'. (3) Finally, we indirectly sample from the desired distribution p by *sampling uniformly* from p'', by randomly projecting the embedding onto a lower-dimensional subspace using universal hash functions and then searching for feasible states.

The first and third steps involve a bounded loss of accuracy, which we can trade off with computational efficiency by setting hyper-parameters of the algorithm. A key advantage is that *our technique reduces the weighted sampling problem to that of solving one MAP query (i.e., finding the most likely state) and a polynomial number of feasibility queries (i.e., finding any state with non-zero probability)* for the original graphical model augmented (through an embedding) with additional variables and carefully constructed factors. In practice, we use a combinatorial optimization package, which requires exponential time in the worst case (consistent with the hardness of sampling) but is often fast in practice. Our analysis shows that whenever the underlying combinatorial search and optimization queries succeed, the samples produced are *guaranteed*, with high probability, to be coming from an approximately accurate distribution.

4.2.1 Weight Discretization

We use a geometric discretization of the weights into "buckets", i.e., a uniform discretization of the log-probability. As we will see, $\Theta(n)$ buckets are sufficient to preserve accuracy.

Definition 4. Let $M = \max_x w(x)$, $r > 1$, $\epsilon > 0$, and $\ell = \lceil \log_r(2^n/\epsilon) \rceil$. Partition the configurations into the following weight based disjoint *buckets*: $\mathcal{B}_i = \{x \mid w(x) \in \left(\frac{M}{r^{i+1}}, \frac{M}{r^i}\right]\}, i = 0, \ldots, \ell - 1$ and $\mathcal{B}_\ell = \{x \mid w(x) \in [0, \frac{M}{r^\ell}]\}$. The *discretized weight function* $w' : \{0,1\}^n \to \mathbb{R}^+$ is defined as follows: $w'(x) = \frac{M}{r^{i+1}}$ if $x \in \mathcal{B}_i$ for $i < \ell$ and $w'(x) = 0$ if $x \in \mathcal{B}_\ell$. The corresponding *discretized probability distribution*

$p'(x) = w'(x)/Z'$ where Z' is the normalization constant.

Lemma 3. *Let $\rho = r^2/(1-\epsilon)$. For all $x \in \cup_{i=0}^{l-1} \mathcal{B}_\ell$, $p(x)$ and $p'(x)$ are within a factor of ρ of each other. Furthermore, $\sum_{x \in \mathcal{B}_\ell} p(x) \leq \epsilon$.*

Proof. Since w maps to non-negative values, we have $Z \geq M$. Further,

$$\sum_{x \in \mathcal{B}_\ell} p(x) = \frac{1}{Z} \sum_{x \in \mathcal{B}_\ell} w(x) \leq \frac{1}{Z} |\mathcal{B}_\ell| \frac{M}{r^\ell} = \frac{|\mathcal{B}_\ell|}{2^n} \frac{\epsilon M}{Z} \leq \frac{\epsilon M}{Z} \leq \epsilon.$$

This proves the second part of the claim. For the first part, note that by construction, $Z' \leq Z$ and

$$Z' = \sum_{i=0}^{\ell} \sum_{x \in \mathcal{B}_i} w'(x) \geq \sum_{i=0}^{\ell-1} \sum_{x \in \mathcal{B}_i} \frac{1}{r} w(x) = \frac{1}{r} \left(Z - \sum_{x \in \mathcal{B}_\ell} w(x) \right) \geq (1-\epsilon) Z.$$

Thus Z and Z' are within a factor of $r/(1-\epsilon)$ of each other. For all x such that $w(x) \notin \mathcal{B}_n$, recalling that $r > 1 > 1-\epsilon$ and that $w(x)/r \leq w'(x) \leq r w(x)$, we have

$$\frac{1}{\rho} p(x) \leq \frac{w(x)}{rZ} \leq \frac{w(x)}{rZ'} \leq \frac{w'(x)}{Z'} = p'(x) = \frac{w'(x)}{Z'} \leq \frac{r w(x)}{Z'} \leq \frac{r^2}{1-\epsilon} \frac{w(x)}{Z} = \rho p(x).$$

This finishes the proof that $p(x)$ and $p'(x)$ are within a factor of ρ of each other.

□

Remark 4. If the weights w defined by the original graphical model are represented in finite precision (e.g., there are 2^{64} possible weights in double precision floating point), for every $b \geq 1$ there is a possibly large but *finite* value of ℓ (such that M/r^ℓ is smaller than the smallest representable weight) such that $\mathcal{B}_\ell$ is empty and the discretization error ϵ is effectively zero.

4.2.2 Embed: From Weighted to Uniform Sampling

We now show how to reduce the problem of sampling from the discrete distribution p' (weighted sampling) to the problem of uniformly sampling, without loss of accuracy, from a higher-dimensional discrete set into which $\mathcal{X} = \{0,1\}^n$ is embedded. This is inspired by slice sampling [77], and can be intuitively understood as its *discrete* counterpart where we uniformly sample points (x, y) from a discrete representation of the area under the (y vs. x) probability density function of p' .

Definition 5. Let $w : \mathcal{X} \to \mathbb{R}^+, M = \max_x w(x)$, and $r = 2^b/(2^b - 1)$. Then the embedding $\mathcal{S}(w, \ell, b)$ of $\mathcal{X}$ in $\mathcal{X} \times \{0,1\}^{(\ell-1)b}$ is defined as:

$$\mathcal{S}(w,\ell,b) = \left\{ \left(x, y_1^1, y_1^2, \ldots, y_{\ell-1}^{b-1}, y_{\ell-1}^b\right) \,\middle|\, w(x) \le \frac{M}{r^i} \Rightarrow \bigvee_{k=1}^{b} y_i^k, 1 \le i \le \ell - 1; w(x) > \frac{M}{r^\ell} \right\}.$$

where $\bigvee_{k=1}^b y_i^k$ may alternatively be thought of as the linear constraint $\sum_{k=1}^b y_i^k \ge 1$. Further, let p'' denote a uniform probability distribution over $\mathcal{S}(w, \ell, b)$ and $n' = n + (\ell - 1)b$.

Given a compact representation of w within a combinatorial search or optimization framework, the set $\mathcal{S}(w, \ell, b)$ can often be easily encoded using the disjunctive constraints on the y variables.

Lemma 4. *Let $(x, y) = (x, y_1^1, y_1^2, \cdots, y_1^b, y_2^1, \cdots, y_2^b, \cdots, y_{\ell-1}^1 \cdots, y_{\ell-1}^b)$ be a sample from p'', i.e., a uniformly sampled element from $\mathcal{S}(w, \ell, b)$. Then x is distributed according to p'.*

Informally, given $x \in \mathcal{B}_i$ and $x' \in \mathcal{B}_{i+1}$ with $i+1 \le l-1$, there are precisely $r = 2^b/(2^b - 1)$ times more valid configurations (x, y) than (x', y'). Thus x is sampled r times more often than x'. A formal proof may be found in the Appendix.

4.2.3 Project and Search: Uniform Sampling with Universal Hashing and an NP-oracle

In principle, using the technique of Bellare et al. [8] and n'-wise independent hash functions we can sample purely *uniformly* from $\mathcal{S}(w, \ell, b)$ using an NP oracle to answer feasibility queries. However, such hash functions involve constructions that are difficult to implement and reason about in existing combinatorial search methods. Instead, we use a more practical algorithm based on pairwise independent hash functions that can be implemented using *parity constraints* (modular arithmetic) and still provides accuracy guarantees. The approach is similar to [50], but we include an algorithmic way to estimate the number of parity constraints to be used. We also use the pivot technique from [17] but extend that work in two ways: we introduce a parameter α (similar to [50]) that allows us to trade off uniformity against runtime and also provide *upper bounds* on the sampling probabilities.

We refer to our algorithm as PArity-basedWeightedSampler (PAWS) and provide its pseudocode as Algorithm 1. The idea is to *project* by randomly constraining the configuration space using a family of universal hash functions, *search* for up to P "surviving" configurations, and then, if fewer than P survive, perform rejection sampling to choose one of them. The number k of constraints or factors (encoding a randomly chosen hash function) to add is determined first; this is where we depart from both Gomes et al. [50], who do not provide a way to compute k, and Chakraborty et al. [17], who do not fix k or provide upper bounds. Then we repeatedly add k such constraints, check whether fewer than P configurations survive, and if so output one configuration chosen using rejection sampling. Intuitively, we need the hashed space to contain no more than P solutions because that is a base case where we know how to produce uniform samples via enumeration. k is a guess (accurate with high probability) of the number of constraints that is likely to reduce (by hashing) the original problem to a situation where enumeration is feasible. If too many or too few configurations survive, the algorithm fails and is run again. The small failure probability, accounting for a potentially poor choice of random hash functions, can be bounded irrespective of the underlying graphical model.

A combinatorial *optimization* procedure is used once in order to determine the maximum weight M through MAP inference. M is used in the discretization step. Subsequently, several *feasibility queries* are issued to the underlying combinatorial search procedure in order to, e.g., count the number of surviving configurations and produce one as a sample.

Lemma 5 (see Appendix for a proof) shows that the subroutine COMPUTEK in Algorithm 2 outputs with high probability a value close to $\log(|\mathcal{S}(w,\ell,b)|/P)$. The idea is similar to an unweighted version of the WISH algorithm [35] but with tighter guarantees and using more feasibility queries.

Lemma 5. *Let $\mathcal{S} = \mathcal{S}(w,\ell,b) \subseteq \{0,1\}^{n'}$, $\delta > 0$, and $\gamma > 0$. Further, let $P \geq \min\{2, 2^{\gamma+2}/(2^\gamma - 1)^2\}$, $Z = |\mathcal{S}|$, $k^*_P = \log(Z/P)$, and k be the output of procedure* COMPUTEK$(n', \delta, P, \mathcal{S})$*. Then, $\mathbb{P}[k^*_P - \gamma \leq k \leq k^*_P + 1 + \gamma] \geq 1 - \delta$ and* COMPUTEK *uses $O(n' \ln(n'/\delta))$ feasibility queries.*

Lemma 6. *Let $\mathcal{S} = \mathcal{S}(w,\ell,b) \subseteq \{0,1\}^{n'}$, $\delta > 0$, $P \geq 2$, and $\gamma = \log\left((P + 2\sqrt{P+1} + 2)/P\right)$. For any $\alpha \in \mathbb{Z}$, $\alpha > \gamma$, let $c(\alpha, P) = 1 - 2^{\gamma-\alpha}/(1 - \frac{1}{P} - 2^{\gamma-\alpha})^2$. Then with probability at least $1-\delta$ the following holds:* PAWS$(w, \ell, b, \delta, P, \alpha)$ *outputs a sample with probability at least $c(\alpha, P)2^{-(\gamma+\alpha+1)}\frac{P}{P-1}$ and, conditioned on outputting a sample, every element $(x, y) \in \mathcal{S}(w, \ell, b)$ is selected (Line 27)*

Algorithm 2 Algorithm PAWS for sampling configurations σ according to w

```
 1: procedure ComputeK($n'$, $\delta$, $P$, $\mathcal{S}$)
 2:     $T \leftarrow 24 \lceil \ln(n'/\delta) \rceil$ ;   $k \leftarrow -1$ ;   $count \leftarrow 0$
 3:     repeat
 4:         $k \leftarrow k + 1$ ;   $count \leftarrow 0$
 5:         for $t = 1, \cdots, T$ do
 6:             Sample hash function $h^k_{A,c} : \{0,1\}^{n'} \to \{0,1\}^k$
 7:             Let $\mathcal{S}^{k,t} \triangleq \{(x,y) \in \mathcal{S}, h^k_{A,c}(x,y) = 0\}$
 8:             if $|\mathcal{S}^{k,t}| < P$ then   /* search for $\geq P$ different elements */
 9:                 $count \leftarrow count + 1$
10:         end for
11:     until $count \geq \lceil T/2 \rceil$ or $k = n'$
12:     return $k$
13: end procedure

14: procedure PAWS($w : \{0,1\}^n \to \mathbb{R}^+$, $\ell$, $b$, $\delta$, $P$, $\alpha$)
15:     $M \leftarrow \max_x w(x)$   /* compute with one MAP inference query on $w$ */
16:     $\mathcal{S} \leftarrow \mathcal{S}(w, \ell, b)$;   $n' \leftarrow n + b(\ell - 1)$        /* as in Definition 5 */
17:     $i \leftarrow$ ComputeK($n'$, $\delta$, $\gamma$, $P$, $\mathcal{S}$) $+ \alpha$
18:     Sample hash fn. $h^i_{A,c} : \{0,1\}^{n'} \to \{0,1\}^i$, i.e., uniformly choose $A \in \{0,1\}^{i \times n'}$, $c \in \{0,1\}^i$
19:     Let $\mathcal{S}^i \triangleq \{(x,y) \in \mathcal{S}, h^i_{A,c}(x,y) = 0\}$
20:     Check if $|\mathcal{S}^i| \geq P$ by searching for at least $P$ different elements
21:     if $|\mathcal{S}^i| \geq P$ or $|\mathcal{S}^i| = 0$ then
22:         return $\perp$                                   /* failure */
23:     else
24:         Fix an arbitrary ordering of $\mathcal{S}^i$       /* for rejection sampling */
25:         Uniformly sample $p$ from $\{0, 1, \ldots, P - 1\}$
26:         if $p \leq |\mathcal{S}^i|$ then
27:             Select $p$-th element $(x,y)$ of $\mathcal{S}^i$ ;   return $x$
28:         else
29:             return $\perp$                               /* failure */
30: end procedure
```

with probability $p'_s(x,y)$ within a constant factor $c(\alpha, P)$ of the uniform probability $p''(x,y) = 1/|\mathcal{S}|$.

Proof Sketch. For lack of space, we defer details to the Appendix. Briefly, the probability $\mathbb{P}[\sigma \in \mathcal{S}^i]$ that $\sigma = (x,y)$ survives is 2^{-i} by the properties of the hash functions in Definition 2, and the probability of being selected by rejection sampling is $1/(P-1)$. Conditioned on σ surviving, the mean and variance of the size of the surviving set $|\mathcal{S}^i|$ are independent of σ because of 3-wise independence. When $k_P^* - \gamma \leq k \leq k_P^* + 1 + \gamma$ and $i = k + \alpha$, $\alpha > \gamma$, on average $|\mathcal{S}^i| < P$ and the size is concentrated around the mean. Using Chebychev's inequality, one can upper bound by $1 - c(\alpha, P)$ the probability $\mathbb{P}[S_i \geq P \mid \sigma \in \mathcal{S}^i]$ that the algorithm fails because $|S_i|$ is too large. Note that the bound is independent of σ and lets us bound the probability $p_s(\sigma)$ that σ is output:

$$c(\alpha, P)\frac{2^{-i}}{P-1} = \left(1 - \frac{2^{\gamma-\alpha}}{(1 - \frac{1}{P} - 2^{\gamma-\alpha})^2}\right)\frac{2^{-i}}{P-1} \leq p_s(\sigma) \leq \frac{2^{-i}}{P-1}. \tag{4.2}$$

From $i = k + \alpha \leq k_P^* + 1 + \gamma + \alpha$ and summing the lower bound of $p_s(\sigma)$ over all σ, we obtain the desired lower bound on the success probability. Note that given σ, σ', $p_s(\sigma)$ and $p_s(\sigma')$ are within a constant factor $c(\alpha, P)$ of each other from (4.2). Therefore, the probabilities $p_s'(\sigma)$ (for various σ) that σ is output conditioned on outputting a sample are also within a constant factor of each other. From the normalization $\sum_\sigma p_s'(\sigma) = 1$, one gets the desired result that $p_s'(x, y)$ is within a constant factor $c(\alpha, P)$ of the uniform probability $p''(x, y) = 1/|\mathcal{S}|$. $\square$

4.2.4 Main Results: Sampling with Accuracy Guarantees

Combining pieces from the previous three sections, we have the following main result:

Theorem 4. *Let $w : \{0,1\}^n \to \mathbb{R}^+$, $\epsilon > 0$, $b \geq 1$, $\delta > 0$, and $P \geq 2$. Fix $\alpha \in \mathbb{Z}$ as in Lemma 6, $r = 2^b/(2^b - 1)$, $\ell = \lceil \log_r(2^n/\epsilon) \rceil$, $\rho = r^2/(1-\epsilon)$, bucket $\mathcal{B}_\ell$ as in Definition 4, and $\kappa = 1/c(\alpha, P)$. Then $\sum_{x \in \mathcal{B}_\ell} p(x) \leq \epsilon$ and with probability at least $(1-\delta)c(\alpha, P)2^{-(\gamma+\alpha+1)}\frac{P}{P-1}$, PAWS$(w, \ell, b, \delta, P, \alpha)$ succeeds and outputs a sample σ from $\{0,1\}^n \setminus \mathcal{B}_\ell$. Upon success, each $\sigma \in \{0,1\}^n \setminus \mathcal{B}_\ell$ is output with probability $p_s'(\sigma)$*

within a constant factor $\rho\kappa$ of the desired probability $p(\sigma) \propto w(\sigma)$.

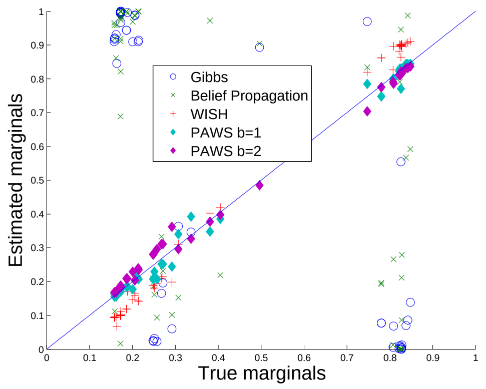
Proof. Success probability follows from Lemma 6. For $x \in \{0,1\}^n \backslash \mathcal{B}_\ell$, combining Lemmas 3, 4, 6 we obtain

$$\begin{aligned}\frac{1}{\rho\kappa}p(x) \leq \frac{1}{\kappa}p'(x) = \sum_{y:(x,y)\in\mathcal{S}(w,\ell,b)} \frac{1}{\kappa}p''(x,y) \leq \sum_{y|(x,y)\in\mathcal{S}(w,\ell,b)} & p'_s(x,y) \;=\; p'_s(x) \\ \leq \sum_{y:(x,y)\in\mathcal{S}(w,\ell,b)} & \kappa p''(x,y) = \kappa p'(x) \leq \rho\kappa p(x)\end{aligned}$$

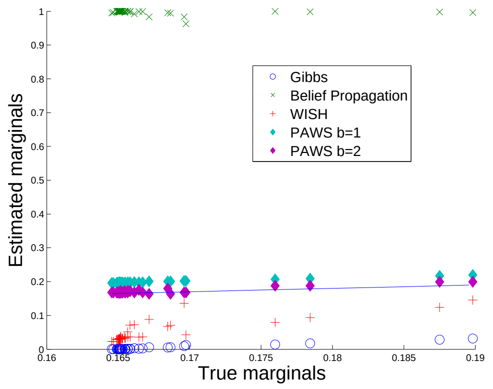
where the first inequality accounts for discretization error from $p(x)$ to $p'(x)$ (Lemma 3), equality follows from Lemma 4, and the sampling error between p'' and p'_s is bounded by Lemma 6. The rest is proved in Lemmas 3, 4. $\square$

Remark 5. By appropriately setting the hyper-parameters b and ℓ we can make the discretization errors ρ and ϵ arbitrarily small. Although this does not change the number of required feasibility queries, it can significantly increase the runtime of combinatorial search because of the increased search space size $|\mathcal{S}(w,\ell,b)|$. Practically, one should set these parameters as large as possible, while ensuring combinatorial searches can be completed within the available time budget. Increasing parameter P improves the accuracy as well, but also increases the number of feasibility queries issued, which is proportional to P (but does not affect the structure of the search space). Similarly, by increasing α we can make κ arbitrarily small. However, the probability of success of the algorithm decreases exponentially as α is increased. We will demonstrate in the next section that a practical tradeoff between computational complexity and accuracy can be achieved for reasonably sized problems of interest.

Corollary 1. *Let $w, b, \epsilon, \ell, \delta, P, \alpha$, and $\mathcal{B}_\ell$ be as in Theorem 4, and $p'_s(\sigma)$ be the output distribution of* PAWS(w, ℓ, b, δ, P, α). *Let $\phi : \{0,1\}^n \to \mathbb{R}$ and $\eta_\phi =$*



(a) Mixed ($w = 4.0$,$f = 0.6$)



(b) Attractive ($w = 3.0$,$f = 0.45$)

Figure 4.1: Estimated marginals vs. true marginals on 8×8 Ising Grid models. Closeness to the 45 degree line indicates accuracy. PAWS is run with $b \in \{1, 2\}, P = 4, \alpha = 1$, and $\ell = 25$ (mixed case) or $\ell = 40$ (attractive case).

$\max_{x \in \mathcal{B}_\ell} |\phi(x)| \leq \|\phi\|_\infty$. *Then,*

$$\frac{1}{\rho\kappa}\mathbb{E}_{p'_s}[\phi] - \epsilon\eta_\phi \quad \leq \quad \mathbb{E}_p[\phi] \quad \leq \quad \rho\kappa\mathbb{E}_{p'_s}[\phi] + \epsilon\eta_\phi$$

where $\mathbb{E}_{p'_s}[\phi]$ *can be approximated with a sample average using samples produced by PAWS.*

4.3 Experiments

We evaluate PAWS on synthetic Ising Models and on a real-world test case generation problem for software verification. All experiments used Intel Xeon 5670 3GHz machines with 48GB RAM.

4.3.1 Ising Models

We first consider the *marginal computation* task for synthetic grid-structured Ising models with random interactions (attractive and mixed). Specifically, the corresponding graphical model has n binary variables $x_i, i = 1, \cdots, n$, with single node potentials $\psi_i(x_i) = \exp(f_i x_i)$ and pairwise interactions $\psi_{ij}(x_i, x_j) = \exp(w_{ij} x_i x_j)$, where $f_i \in_R [-f, f]$ and $w_{ij} \in_R [-w, w]$ in the *mixed* case, while $w_{ij} \in_R [0, w]$ in the *attractive* case.

Our implementation of PAWS uses the open source solver ToulBar2 [1], one of the winners in the 2010 UAI Approximate Inference Challenge, to compute $M = \max_x w(x)$ and as an oracle to check the existence of at least P different solutions. We augmented ToulBar2 with the IBM ILOG CPLEX CP Optimizer 12.3 [60] based on techniques borrowed from [53] to efficiently reason about parity constraints (the hash functions) using Gauss-Jordan elimination. We run the subroutine COMPUTEK in Algorithm 1 only once at the beginning, and then generate all the samples with the same value of i (Line 17). The comparison is with Gibbs sampling, Belief Propagation, and the recent WISH algorithm [35]. Ground truth is obtained using the Junction Tree method [66].

In Figure 4.1, we show a scatter plot of the estimated vs. true marginal probabilities for two Ising grids with mixed and attractive interactions, respectively, representative of the general behavior in the large-weights regime. Each sampling method is run for 10 minutes. Marginals computed with Gibbs sampling are clearly very inaccurate (far from the 45 degree line), an indication that the Markov Chain had not mixed as an effect of the relatively large weights that tend to create barriers between modes which are hard to traverse. In contrast, samples from PAWS provide much more accurate marginals, in part because it does not rely on local search and hence is not directly affected by the energy landscape (with respect to the Hamming metric). Further, we see that we can improve the accuracy by increasing the hyper-parameter b. These results highlight the practical value of having accuracy guarantees on the quality of the samples after finite amounts of time vs. MCMC-style guarantees that hold only after a potentially exponential mixing time.

Belief Propagation can be seen from Figure 4.1 to be quite inaccurate in this large-weights regime. Finally, we also computed marginals using the recent WISH algorithm [35] which uses similar hash-based techniques to estimate the partition function of graphical models. Since estimating each marginal as the ratio of two partition functions (with and without a variable clamped) would be too expensive (requiring $n + 1$ calls to WISH) we heuristically run it once and use the solutions of the optimization instances it solves in the inner loop as samples. We see in Figure 4.1 that while samples produced by WISH can sometimes produce fairly accurate marginal estimates, these estimates can also be far from the true value because of an inherent bias introduced by the $\arg\max$ operator.

4.3.2 Test Case Generation for Software Testing

Hardware and software verification tools are becoming increasingly important in industrial system design. For example, IBM estimates \$100 million savings over the past 10 years from hardware verification tools alone [76]. Given that complete formal verification is often infeasible, the paradigm of choice has become that of randomly generating “interesting” test cases to stress the code or chip with the hope of uncovering bugs. Typically, a model based on *hard constraints* is used to specify consistent input/output pairs, or valid program execution traces. In addition, in some systems, domain knowledge can be specified by experts in the form of *soft constraints*, for instance to introduce a preference for test cases where operands are zero and bugs are more likely [76].

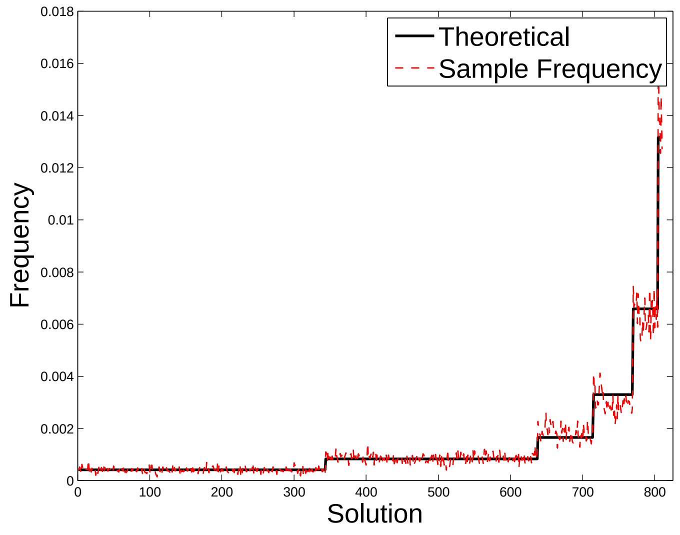
For our experiments, we focus on software (SW) verification, using an industrial benchmark [6] produced by Microsoft's SAGE system [43, 44]. Each instance defines a uniform probability distribution over certain valid traces of a computer program. We modify this benchmark by introducing soft constraints defining a weighted distribution over valid traces, indicating traces that meet certain criteria should be sampled more often. Specifically, following Naveh et al. [76] we introduce a preference towards traces where certain registers are zero. The weight is chosen to be a power of two, so that there is no loss of accuracy due to discretization using the previous construction with $b = 1$.

These instances are very difficult for MCMC methods because of the presence of very large regions of zero probability that cannot be traversed and thus can break the ergodicity assumption. Indeed we observed that Gibbs sampling often fails to find a non-zero probability state, and when it finds one it gets stuck there, because there might not be a non-zero probability path from one feasible state to another. In contrast, our sampling strategy is not affected and does not require any ergodicity assumption. Table 4.3(a) summarizes the results obtained using the propositional satisfiability (SAT) solver CryptoMiniSAT [88] as the feasibility query oracle for PAWS. CryptoMiniSAT has built-in support for parity constraints $Ax = c \mod 2$. We report the time to collect 1000 samples and the Mean Squared Error (MSE) of the marginals estimated using these samples. We report results only on the subset of instances where we could enumerate all feasible states using the exact model counter Relsat [7] in order to obtain ground truth marginals for MSE computation. We see that PAWS scales

Figure 4.2: Experiments on a software verification benchmark: sampling and marginal estimation.

Instance	Vars	Factors	Time (s)	MSE ($\times 10^{-5}$)
bench1039	330	785	1710	5.76
bench431	173	410	34.97	4.35
bench115	189	458	52.75	20.74
bench97	170	401	67.03	45.57
bench590	244	527	593.71	8.11
bench105	243	524	842.35	8.56

(a) Marginals: runtime and mean squared error



(b) True vs. observed sampling frequencies.

to fairly large instances with hundreds of variables and gives accurate estimates of the marginals. Figure 4.3(b) shows the theoretical vs. observed sampling frequencies (based on 50000 samples) for a small instance with 810 feasible states (execution traces), where we see that the output distribution p'_s is indeed very close to the target distribution p.

4.4 Discussion

We introduced a new approach, called PAWS, to the fundamental problem of sampling from a discrete probability distribution specified, up to a normalization constant, by a weight function, e.g., by a discrete graphical model. While traditional sampling methods are based on the MCMC paradigm and hence on some form of local search, PAWS can leverage more advanced combinatorial search and optimization tools as a black box. A significant advantage over MCMC methods is that PAWS comes with a strong accuracy guarantee: whenever combinatorial search succeeds, our analysis provides a certificate that, with high probability, the samples are produced from an approximately correct distribution. In contrast, accuracy guarantees for MCMC methods hold only in the limit, with unknown and potentially exponential mixing times. Further, the hyper-parameters of PAWS can be tuned to trade off runtime with accuracy. Our experiments demonstrate that PAWS outperforms competing sampling methods on challenging domains for MCMC.

CHAPTER 5
OPTIMIZATION WITH PARITY CONSTRAINTS: HARDNESS AND PRACTICAL APPROACHES

In this chapter, we investigate the class of combinatorial search and optimization problems subject to random parity constraints arising from the WISH and PAWS schemes described in Chapters 3 and 4. These optimization problems turn out to be intimately connected with the fundamental problem of maximum likelihood decoding of an error correcting code in information theory [97, 10]. We leverage this connection to show that the inference queries generated by WISH are NP-hard to solve and to approximate. Although generally hard in the worst case, message passing and related linear programming techniques [38] are known to be very successful in practice in decoding certain types of codes such as low density parity check (LDPC) codes [40]. Inspired by the success of these methods, we formulate the MAP inference queries generated by WISH as Integer Linear Programs (ILP). Our ILP formulation empirically provides very good lower bounds on the optimization problems, while at the same time providing also upper bounds based on solving a sequence of LP relaxations. These bounds can in turn be used to obtain a new family of approximate upper and lower bounds on the value of the partition function, which are much tighter than those obtained by tree decomposition and convexity [98]. This is a significant advance, because other state-of-the-art sampling based algorithms [51, 101, 46, 45] can usually provide probabilistic guarantees on lower bounds, but are not able to reason at all about upper bounds.

Finally, we empirically investigate how the length of the parity constraints (equivalently, their *sparsity*) affects the empirical hardness of the optimization problems corresponding to finding the mode for a graphical model augmented with random parity constraints (MAP inference). Empirically, we found that *sparse parity constraints*, such as the ones used in low density parity check (LDPC) codes from Gallager [40], tend to be much easier to solve. Motivated by this observation, we propose a technique to construct equivalent but sparser (and empirically easier to solve) parity constraints. This connection with low-density parity check codes and the use of sparse parity constraints in our probabilistic inference schemes and will be studied in depth in the next chapter.

5.1 Max-Likelihood Decoding and Worst Case Analysis

For a problem with n binary variables, WISH requires solving $\Theta(n \log n)$ optimization instances. If these optimizations could be approximated (within a constant factor of the true optimal value) in polynomial time, this would give rise to a polynomial time algorithm that gives, with high probability, a constant factor approximation for the original counting problem (see Theorem 3). Note that this is a reasonable assumption, because perhaps the most interesting #-P complete counting problems are those whose corresponding decision problem are easy, e.g. counting weighted matchings in a graph (computing the permanent). A natural question arises: *are there interesting counting problems for which we can approximate* $\max_\sigma w(\sigma)$ *subject to* $A\sigma = b \mod 2$ *in polynomial time?*

To shed some light on this question, we show a connection with a decision problem arising in coding theory:

Definition 6 (MAXIMUM-LIKELIHOOD DECODING)**.** Given a binary $m \times n$ matrix A, a vector $b \in \{0,1\}^m$, and an integer $w > 0$, is there a vector $z \in \{0,1\}^n$ of Hamming weight $\leq w$, such that $Az = b \mod 2$?

As noted by Vardy [97], Berlekamp et al. [10] showed that this problem is NP-complete with a reduction from 3-DIMENSIONAL MATCHING. Further, Stern [89] and Arora et al. [4] proved that even approximating within any constant factor the solution to this problem is NP-hard.

These hardness results restrict the kind of problems we can hope to solve in our setting, which is more general. In fact, we can define a graphical model with single variable factors $\psi_i(x_i) = \exp(-x_i)$ for $x_i \in \{0, 1\}$. Let $\mathcal{S} = \{x \in \{0,1\}^n : Ax = b \bmod 2\}$. Then

$$\begin{aligned}\max_{x\in\mathcal{S}} w(x) = \max_{x\in\mathcal{S}} \prod_{i=1}^{n} \psi_i(x_i) &= \exp\left(\max_{x\in\mathcal{S}} \sum_{i=1}^{n} \log \psi_i(x_i)\right)\\ &= \exp\left(\max_{x\in\mathcal{S}} -H(x)\right) = \exp\left(-\min_{x\in\mathcal{S}} H(x)\right)\end{aligned}$$

where $H(x)$ is the Hamming weight of x. Thus, MAXIMUM-LIKELIHOOD DECODING of a binary code is a special case of MAP inference subject to parity constraints, but on a simple (disconnected) factor graph with factors acting only on single variable nodes. Intuitively, in the context of coding theory, there is a variable for each transmitted bit, and factors capture the probability of a transmission error on each bit. Thus there are no interactions between the variables, except for the ones introduced by the parity constraints $Ax = b \mod 2$, while in our context we allow for more complex probabilistic dependencies between variables specified as in Equation (4.1). See Figure 5.1 for a pictorial representation. We therefore have the following theorem:

Theorem 5. *Given a binary $m \times n$ matrix A, a vector $b \in \{0, 1\}^m$, and $w(x)$ as in Equation (2.2), the following optimization problem*

$$\max_{x\in\{0,1\}^n} \log w(x) \text{ subject to } Ax = b \mod 2$$

is NP-hard to solve and to approximate within any constant factor.

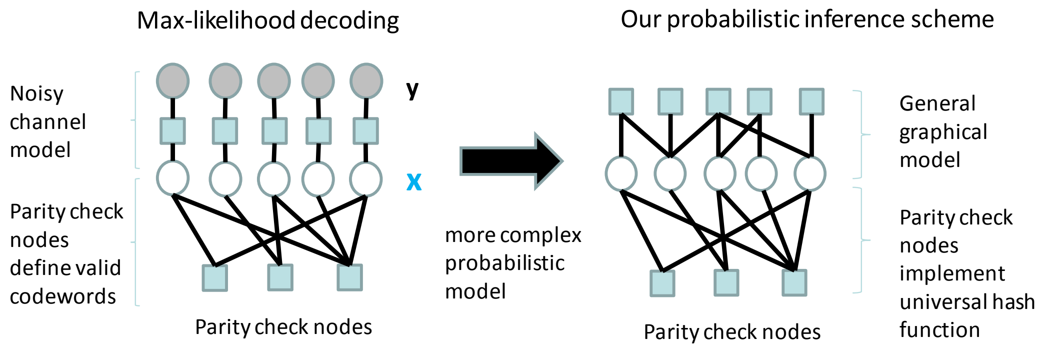

Max-likelihood decoding
Our probabilistic inference scheme
Noisy channel model
y
x
Parity check nodes define valid codewords
Parity check nodes
more complex probabilistic model
General graphical model
Parity check nodes implement universal hash function
Parity check nodes

Figure 5.1: Pictorial representation of the relationship between max-likelihood decoding and the queries generated by our scheme. While in coding schemes parity check nodes are used to define valid codewords, in our scheme they are used to implement a universal hash function which randomly selects a subset of the original high-dimensional space. Since we are starting from a more general graphical model, the optimization instances are at least as hard as maximum-likelihood decoding.

Connections with coding theory is even deeper, and is not just an artifact of the particular hash function construction used. In fact, there is an intimate connection and a correspondence between universal hash functions and (binary) codes, where one can construct hash functions from binary codes and vice versa [90]. We will show how to exploit this connection in Chapter 6, where we will introduce a new class of hash functions with useful statistical properties leveraging ideas and constructions that were previously applied to low-density parity check codes.

5.2 MAP Inference by Integer Linear Programming

The NP-hard combinatorial optimization problem $\max_\sigma w(\sigma)$ subject to $A\sigma = b \mod 2$ can be formulated as an Integer Program [11]. This is a promising approach because Integer Linear Programs and related Linear programming (LP) relaxations have been shown to be a very effective at decoding binary codes by Feldman et al. [38]. Further, the empirically successful iterative message-passing decoding algorithms are closely related to LP relaxations of certain Integer Programs, either because they are directly trying to solve an LP or its dual like the MPLP and TRWBP [42, 87, 98], or attempting to approximately solve a variational problem over the same polytope like Loopy Belief Propagation [99].

For simplicity, we consider the case of binary factors (pairwise interactions between variables), where equation (2.2) simplifies to $w(x) = \prod_{i\in V} \psi_i(x_i) \prod_{(i,j)\in E} \psi_{ij}(x_i, x_j)$ for some edge set E. Rewriting in terms of the logarithms, the unconstrained MAP inference problem can be stated as $\max_{x\in\{0,1\}^n} \sum_{i\in V} \theta_i(x_i) + \sum_{(i,j)\in E} \theta_{ij}(x_i, x_j)$ which can be written as an Integer Linear Program using binary indicator variables $\{\mu_i, i \in V\}$ and $\{\mu_{ij}(x_i, x_j), (i,j) \in E, x_i \in \{0,1\}, x_j \in \{0,1\}\}$ as follows [99]:

$$\max_{\mu_i, \mu_{ij}(x_i,x_j)} \sum_{i\in V} \theta_i(1)\mu_i + \theta_i(0)(1-\mu_i) + \sum_{(i,j)\in E} \sum_{x_i,x_j} \theta_{ij}(x_i, x_j)\mu_{i,j}(x_i, x_j)$$

subject to

$$\forall i \in V, (i,j) \in E, \qquad \sum_{x_j\in\{0,1\}} \mu_{i,j}(0, x_j) = 1 - \mu_i$$

$$\forall i \in V, (i,j) \in E, \qquad \sum_{x_j\in\{0,1\}} \mu_{i,j}(1, x_j) = \mu_i$$

$$\forall i \in V, (i,j) \in E, \qquad \sum_{x_i\in\{0,1\}} \mu_{i,j}(x_i, 0) = 1 - \mu_j$$

$$\forall i \in V, (i,j) \in E, \qquad \sum_{x_i\in\{0,1\}} \mu_{i,j}(x_i, 1) = \mu_j$$

5.2.1 Parity Constraints

There are several possible encodings for the parity constraints $A\sigma = b \mod 2$, defining the so called *parity polytope* over $\sigma \in \mathbb{R}^n$. We summarize them next. Let $\mathcal{J}$ be the set of parity constraints (one entry per row of A). Let $\mathcal{N}(j)$ be the set of variables the j-th parity constraint depends on, namely the indexes of the non-zero columns of the j-th row of A[1]. We'll refer to $|\mathcal{N}(j)|$ as the length of the j-th XOR.

Exponential Polytope Representation

The simplest encoding is due to Jeroslow [61]. It requires that for all $j \in \mathcal{J}$, $S \subseteq \mathcal{N}(j)$, and $|S|$ odd, the following should hold

$$\sum_{i \in S} \mu_i + \sum_{i \in (\mathcal{N}(j) \setminus S)} (1 - \mu_i) \leq |\mathcal{N}(j)| - 1$$

Clearly, this requires a number of constraints that is exponential in the length of the XOR.

Another representation exponential in the length of the parity constraint is due to Feldman et al. [38]. For each S in the set $E_j = \{S \subseteq \mathcal{N}(j) : |S| \text{ even}\}$ there is an extra binary variable $w_{j,S} \in \{0,1\}$. It requires $\forall j \in \mathcal{J}, \sum_{S \in E_j} w_{j,S} = 1$ and $\forall j \in \mathcal{J}, \forall i \in \mathcal{N}(j), \mu_i = \sum_{S \in E_j : i \in S} w_{j,S}$.

Compact Polytope Representation

Yannakakis [103] introduced the following compact representation which re-

[1]To represent the desired parity of the j-th constraint imposed by b_j we use a dummy variable $d = 1$, and include d in $\mathcal{N}(j)$ whenever $b_j = 1$.

quires only $O(n^3)$ variables and constraints, where n is the number of variables. For each constraint j, define $T_j = \{0, 2, \cdots, 2\lfloor|\mathcal{N}(j)|/2\rfloor\}$ as the set of even numbers between 0 and $|\mathcal{N}(j)|$.

- for all $j \in \mathcal{J}$ and for all $k \in T_j$ we have a binary variable $\alpha_{j,k} \in \{0, 1\}$
- for all $j \in \mathcal{J}$ and for all $k \in T_j$ and for all $i \in \mathcal{N}(j)$ we have a binary variable $z_{i,j,k} \in \{0, 1\}$, $0 \leq z_{i,j,k} \leq \alpha_{j,k}$

Then the following constraints are enforced:

$$\begin{aligned} \forall i \in V, j \in \mathcal{N}(i), \qquad & \mu_i = \sum_{k \in T_j} z_{i,j,k} \\ \forall j \in \mathcal{J}, \qquad & \sum_{k \in T_j} \alpha_{j,k} = 1 \\ \forall j \in \mathcal{J}, \forall k \in T_j, \qquad & \sum_{i \in \mathcal{N}(j)} z_{i,j,k} = k\alpha_{j,k} \end{aligned}$$

For any set of parity constraints, these 3 encodings are equivalent, in the sense that the subset of $\{\mu_i\}$ satisfying the constraints is the same [38]. Thus, the corresponding MAP inference problems are also equivalent, as the objective function, by expressing each $\mu_{i,j}$ in terms of the $\{\mu_i\}$ variables, can be re-written as a (possibly non-linear) function of only $\{\mu_i\}$.

5.2.2 Solving Integer Linear Programs

Solving ILPs typically relies on solving a sequence of Linear Programming (LP) relaxations obtained by relaxing the integrality constraints. The solution to the relaxation provides an upper bound to the original integer maximization problem. Since LP can be solved in polynomial time, using Theorem 1 and Theorem

3 we have a *polynomial time* method to obtain approximate upper bounds on the partition function which hold with high probability, although without tightness guarantees. Notice that upper bounds of this form could also be obtained using message passing techniques such as MPLP or TRWBP [42, 87, 98], which can also provide upper bounds to the values of the MAP inference queries in the inner loop of WISH.

IP solvers such as IBM ILOG CPLEX Optimization Studio [60] solve a sequence of LP relaxations based on branching on the problem's variables and possibly adding cutting planes, iteratively improving the upper bound and keeping track of the best integer solution found, until lower and upper bounds match. Thus, one advantage of using an IP solver over standard Message Passing techniques is that the upper and lower bounds improve over time, and it is guaranteed to eventually provide an optimal solution for the original integer problem. In Figure 5.2 we plot the upper bound reported by CPLEX as a function of runtime for a random 10×10 Ising model with mixed interactions. It's clear that there is quickly a dramatic improvement over the value of the basic LP relaxation, which is the value reported by CPLEX around time zero, and that the upper bound keeps improving although at a slower rate. We note that other techniques such as by Sontag et al. [87] could also be used to iteratively tighten the LP relaxation, and might lead to better scaling behavior on certain classes of very large problems [104].

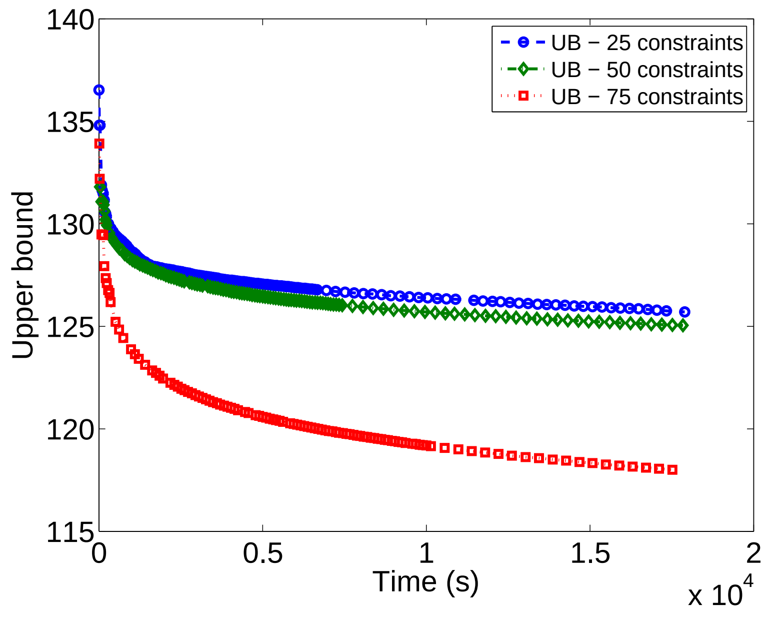


Figure 5.2: Upper bound found by solving a sequence of LP relaxations for MAP inference problems subject to parity constraints, as a function of runtime. Integer Linear Programming solvers obtain a sequence of upper bounds that improve over time and are guaranteed to eventually reach the global optimum of the original optimization problem. This solving strategy naturally provides a tradeoff between computational resources used and accuracy.

5.3 The Role of Sparsity

As we have shown, solving MAP inference queries subject to parity constraints is hard in general. However, *adding parity constraints can sometimes make the optimization easier*. For example, when A is the identity matrix, enforcing $A\sigma = b \mod 2$ corresponds to fixing the values of all variables and leads to a trivial optimization problem. Empirically, **sparse** constraints, such as the ones used in low density parity check (LDPC) codes from Gallager [40], tend to be much easier to solve. For example, in Figure 5.3 we show the median integrality gap[2] (over 500 runs) for the ILP formulation of the MAP inference instances, for $i \in \{10, 15, 20, 25, 30\}$ random parity constraints generated at various den-

[2]At the root node of the ILP solver search tree.

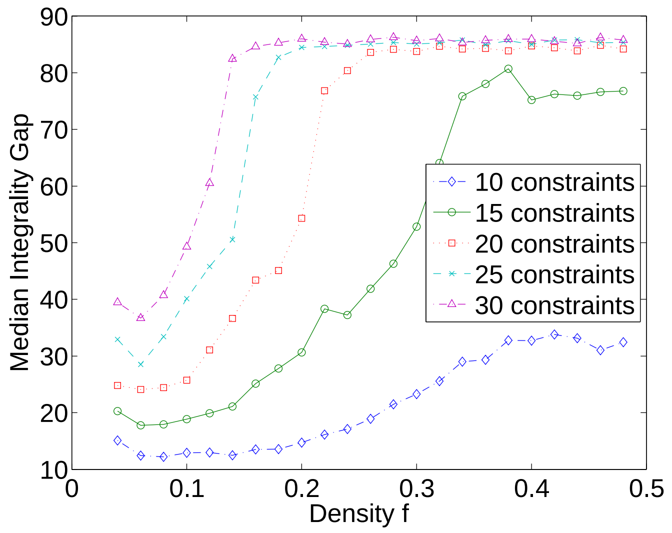


Figure 5.3: Integrality gap at the root node of the search tree of an Integer Linear Programming solver when solving MAP inference queries subject to parity constraints, as a function of the density f (equivalently, average length) of the parity constraints used (random Ising model with parameters $w = 2.5$, $M = 10$). Empirically, problems subject to short parity constraints (small f) are easier to solve (smaller integrality gap on the y axis).

sity levels. Specifically, we consider Ising grids as in section 3.4.2 and construct parity constraints by adding each variable independently with probability $0 \leq f \leq 1/2$ ($f = 1/2$ corresponds to the standard constructions in Chapter 2 for strongly universal hash functions). We see that problems with short XORs (generated with small f) typically have smaller integrality gaps, which confirms the fact that short XORs are easier to reason about. This is not surprising, because the optimizations involved are analogous to max likelihood decoding problems, and sparse codes are known to be easier to decode empirically [70].

Long parity constraints are difficult to reason about not only for Integer Programming solvers but also for SAT solvers [51]. This is because in the context of backtrack search, short parity constraint naturally lead to a more effective pruning of the search space. In fact, before any conclusion can be drawn based on a parity constraint, all the variables but one need to be set (then the value of the last one can be set to match the desired parity). When using shorter parity constraints, it is sufficient to guess a smaller number of variables before an inference can be made and some propagation can be triggered. Of course, the earlier in the search these inferences can be made, the faster the search will be because this will reduce unnecessary branching.

Unfortunately, constructions in both Propositions 1 and 2 to create pairwise independent hash functions (see Chapter 2) add each variable with probability $f = 1/2$ and therefore require parity constraints that are of average length $n/2$ for a problem with n variables, i.e., the corresponding matrix A is not sparse. Motivated by the empirical results in this section, in the next section we consider several techniques based on linear algebra to sparsify the constraints, i.e. produce a new set of constraints that are mathematically equivalent but sparser (shorter).

5.4 Inducing Sparsity

A set of parity constraints specified through matrices A, b defines a set of solutions $\mathcal{S} = \{x \in \{0,1\}^n : Ax = b \mod 2\}$, which is the translated nullspace of the matrix A. The nullspace is a vector space, defined with operations over the finite field $\mathbb{F}(2)$, i.e. modular arithmetic. Exploiting basic linear algebraic properties, it can be shown that applying *elementary row operations* (namely row switching, row multiplication and row addition) [92] to $[A|b]$ does not change the solution set $\mathcal{S}$ and thus the optimization problem. On the other hand, the parity polytope we described in section 5.2.1 is *not* a function of the solution set

$\mathcal{S}$ but *depends explicitly* on the form of the matrices A and b. This fact was also noted by Feldman et al. [38], who showed that a new matrix $[A'|b']$ constructed from $[A|b]$ by adding new rows that are linear combinations of the rows of $[A|b]$ can lead to a tighter LP relaxation, although $Ax = b$ and $A'x = b'$ define the same solution set $\mathcal{S}$ (because the constraints added are all implied).

We propose to exploit these facts and rewrite the constraints in a form that is equivalent, i.e., defines the same set of solutions, but is easier to solve. Specifically, given a a set of parity constraints specified through matrices A, b we look for matrices A', b' that define the same set of solutions, namely $\{x \in \{0,1\}^n : Ax = b\} = \{x \in \{0,1\}^n : A'x = b'\}$ but are much sparser, namely $||[A'|b']||_1 \ll ||[A|b]||_1$. Unfortunately, even finding a sparse linear combination of the rows is computationally intractable, as it can be seen as an instance of MAXIMUM-LIKELIHOOD DECODING, where the code is given in terms of the generators (the rows of A) rather than the check matrix. We therefore propose to use two approaches:

- Perform Gauss-Jordan elimination on $[A|b]$ to convert $[A|b]$ to reduced row echelon form;
- Try all combinations of up to C rows $r_1, \cdots, r_C$ of $[A|b]$, and if their sum $r_1 \oplus \cdots \oplus r_C$ is sparser than any of the r_i, substitute r_i with $r_1 \oplus \cdots \oplus r_C$.

Both techniques are based on elementary row operations and therefore are guaranteed to maintain the solution set $\mathcal{S}$ and to improve sparsity.

In Figure 5.4 we show the median upper and lower bounds found by CPLEX for several randomly generated constraints on a random 10×10 Ising grid model with mixed interactions. Starting with a matrix A generated using the Toeplitz

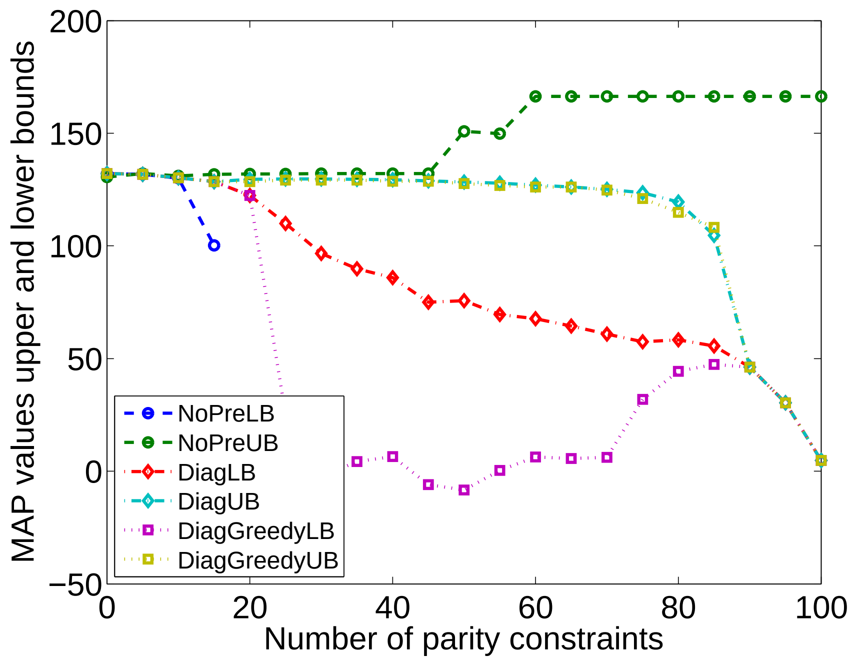


Figure 5.4: Upper and lower bounds with and without sparsification. Using our preprocessing techniques we can rewrite the parity constraints into an equivalent but much sparser form. This empirically leads to easier optimization problems for which we can obtain better bounds from Linear Programming relaxations.

matrix construction in Proposition 2, we run CPLEX for 10 minutes with and without sparsification, reporting the best upper and lower bounds found. We see that without any preprocessing (NoPre) CPLEX fails at finding any integer solution when there are more than 15 parity constraints. Performing Gauss-Jordan elimination (Diag) significantly improves both the upper bound and the lower bound. The effect is particularly significant for a large number of constraints, when the reduced row echelon form of A is close to the identity matrix. Adding the additional greedy substitution step (DiagGreedy, looking at all combinations of up to $C = 4$ rows) slightly improves the quality of the upper bound, but the lower bound significantly degrades.

5.5 Message Passing Decoding

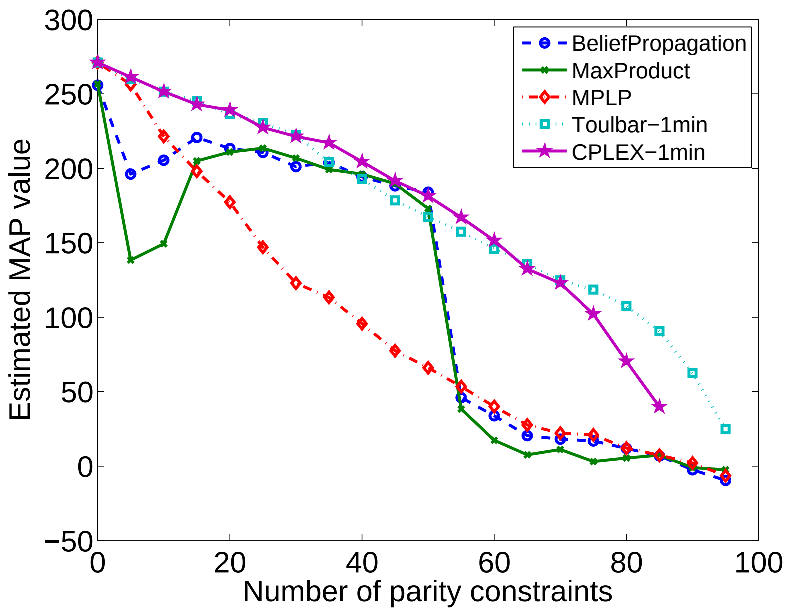
Iterative Message Passing (MP) methods are among the most widely used techniques for decoding error correcting codes. Although the decoding problem is computationally intractable, they usually have very good performance in practice [65, 38]. Since we can represent parity constraints as additional factors in our original factor graph model, Message Passing techniques can also be applied to the more general MAP inference queries with parity constraints generated by WISH. While these techniques have been previously applied to graphical models structured as the ones in the left panel of Figure 5.1 (max-likelihood decoding), we can attempt to use them for the similar but more complex graphical models in the right panel of Figure 5.1 (optimization queries generated by WISH). Specifically, although a parity constraint over k variables would require a conditional probability table (CPT) of size 2^k to be specified, efficient Dynamic-Programming-based updates for parity constraints are known [65]. These updates have complexity which is linear in k.

As briefly alluded to earlier, we will use *short XORs* (involving up to k variables) in order to make MAP inference more efficient in practice and put ourselves in the regime where Message Passing techniques are known to work well for decoding tasks. Empirically, smaller values of k lead to faster execution of the combinatorial optimization problems subject to parity constraints we need to solve. We will discuss the validity of using shorter parity constraints within our probabilistic inference schemes in depth in Chapter 6, both in terms of the accuracy and statistical guarantees that can be achieved.

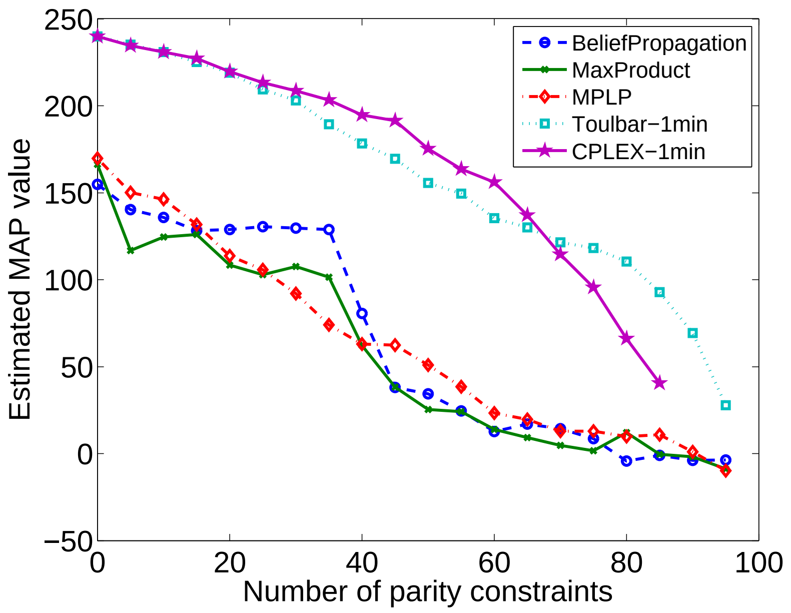
Figure 5.5 compares several approaches to solve the MAP inference problems constrained by random parity constraints of length $k = 4$ for a 10×10 Ising grid model with attractive and mixed interactions (external field $f = 1.0$ and weight $w = 3.0$; see section 3.4.2 for a formal description of the probabilistic model). We compare three message passing approaches, namely Belief Propagation (BP), Max-Product (MP), and MPLP [42], and two combinatorial optimization solvers, namely Toulbar [1] and CPLEX 12.3 [60] based on our ILP formulation, both with a 1 minute time limit. We show the median value of the solution found over 50 realizations, for each number of parity constraints added. We run the Message Passing methods until they find a feasible solution satisfying the parity constraints or up to 10000 iterations. If no feasible solution is found, we round the final beliefs to an integer solution and project it on the feasible set by solving the linear equations with Gaussian Elimination, thus changing the value of some of the variables. For this problem, using "long" parity constraints of length 50, Message Passing methods can only find feasible solutions for up to 10 constraints (consistent with CPLEX performance in Figure 5.4). In contrast, as shown in Figure 5.5, using short XORs of length 4 (typical values encountered for low density parity check codes), Message Passing methods can find feasible solutions for up to about $40 - 50$ constraints, at which point there is a significant performance drop caused by the need for a projection step. We see that for the attractive case, Message Passing methods are competitive with combinatorial optimization approaches but only for a moderate number of constraints. In the more challenging mixed interactions case, CPLEX and Toulbar appear to be clearly superior. We think the the unsatisfactory performance of message passing techniques (compared to when used for LDPC decoding) is caused by the more complicated probabilistic dependencies imposed by the Ising model, which is much more intricate than a typical transmission error model.

5.6 Discussion

Using a connection with max-likelihood decoding problems, we showed that the MAP inference queries generated by WISH and PAWS are in general not polynomial time solvable or even approximable. On the positive side, this led to the use of an ILP formulation for the problem, inspired by iterative message passing decoding. An advantage of the ILP based approach is that it provides upper bounds (in polynomial time from LP relaxations) on the value of the optimization instances generated by WISH, and these can be used to obtain upper bounds on the partition function using Theorem 3. This approach of leveraging Theorem 3 in conjunction with standard relaxations for the combinatorial optimization problems generated by WISH provides an entirely new strategy for obtaining provable bounds on probabilistic inference queries. For example, semi-definite programming based relaxations could potentially be used to obtain another family of upper bounds on the partition function. To increase the practicality of the ILP approach, we introduced sparsification techniques based on linear algebra to obtain sparser parity constraints while preserving their desirable properties. The next chapter will explore the use of other families of universal hash functions that can be implemented directly using sparse, short constraints, which can then be further sparsified using the algebraic techniques described in this chapter. The theory developed in the next chapter will again highlight interesting and useful connections with ideas from the coding and information theory literature.



(a) 10×10 Ising grid with attractive interactions. Length 4 Xors



(b) 10×10 Ising grid with mixed interactions. Length 4 Xors

Figure 5.5: Optimization with short parity constraints: we compare combinatorial optimization techniques (branch and bound and integer linear programming) with message-passing techniques for the MAP inference problems subject to parity constraints (which generalize standard maximum-likelihood decoding) generated by WISH. Although message-passing techniques are very effective for standard decoding problems, they are empirically outperformed by combinatorial optimization techniques for these more complex probabilistic models.

CHAPTER 6
UNIVERSAL HASHING WITH SPARSE PARITY CONSTRAINTS

In the previous chapter, we have seen how the MAP inference queries generated by the WISH and PAWS algorithms (Chapters 3 and 4) are intimately connected to the max-likelihood decoding problem from information theory. As we have seen in Chapter 5, MAP queries generated by WISH and PAWS are empirically harder than traditional decoding problems because they involve more complex probabilistic models (see Figure 5.1), and because known constructions for universal hash functions (see Chapter 2) naturally give rise to "dense" parity constraints (affecting half of the variables on average) as opposed to typical sparse, low-density codes used in information theory. To address this issue, in this chapter we introduce a more general version of WISH that relies on sparse parity constraints, thus giving rise to easier to solve MAP queries. We show that *arbitrarily sparse* parity constraints can be used and still provide one-sided guarantees on the approximation error for the partition function. Leveraging ideas from low-density parity check codes, we also show that if the constraints used are *not too sparse*, then they still provide the same theoretical guarantees on the approximation error as "dense" parity constraints implementing strongly universal hash functions. This approach leverages a new class of hash functions, termed Average Universal, that are statistically weaker than strongly universal ones, but strong enough to be used within the WISH scheme, and they can be implemented with sparser parity constraints. Specifically, we provide an analytic condition that relates the average length of the constraint with the statistical efficiency of the corresponding hash function. This condition can be used to generate constraints that are as sparse as possible, while still implementing hash functions that are statistically good enough to be used in the WISH scheme.

6.1 Probabilistic Inference by Hashing

A range of recent probabilistic inference methods, such as the WISH and PAWS algorithms presented in Chapters 3 and 4 and other related techniques including ApproxMC [18], MBound and Hybrid-MBound [51], and XORSample [50] rely heavily on theoretical properties of universal hash functions.

The key idea is that one can reliably estimate properties of a very large, high-dimensional set S (such as the integral of a function, or a partition function) by randomly dividing it into cells using a hash function h sampled at random from a family $\mathcal{H}$ and looking at properties of a randomly chosen, lower-dimensional cell $h^{-1}(y) \cap S$ (i.e., the subset of points that are hashed to the same, randomly chosen value y). Remarkably, although $h^{-1}(y)$ is in general exponentially large, using certain classes of hash functions it is possible to specify and represent this set in a compact way, without having to enumerate each individual element. For example, we have seen in the previous Chapters that it is possible to apply a pairwise independent hash function to all the exponentially many variable assignments of a discrete graphical model by augmenting it with a set of randomly generated parity constraints (i.e., extra factors which can be specified in a compact way).

To work with high probability, this family of inference algorithms relies on good statistical properties of the hash functions used. Intuitively, the division into cells needs to be "sufficiently random". Specifically, the hash functions used need to behave in a uniform and concentrated way, so that it is possible to predict $|h^{-1}(y) \cap S|$ (as a function of unknown $|S|$) with high probability, no matter what the structure of S is. Full independence (i.e., a function h that acts on each element of S independently at random) would be ideal, as it would make the structure of S irrelevant. However, fully independent hash functions are computational intractable. If there is high correlation between $\{h(x)\}_{x \in S}$, things might not work. In the extreme case , if the hash function h does not act in a sufficiently random way, it is possible that all the random variables $h(x), x \in S$ are identical, i.e., the division into cells does not break up S in a nice way and S is contained in a single cell. Fortunately, as we have seen in Chapters 3 and 4, pairwise independence often suffices and is also computationally tractable.

6.1.1 Concentration and Counting

Formally, let $S \subseteq \{0,1\}^n$ and $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ be a family of hash functions. Let h be a hash function chosen uniformly from $\mathcal{H}$. Then, for $y \in \{0,1\}^m$, define the following random variable:

$$X(h, S, y) = |h^{-1}(y) \cap S| \tag{6.1}$$

Let $\mu(h, S, y) = \mathbb{E}[X(h, S, y)]$ denote its expected value and $\sigma(h, X, y)$ the standard deviation. Note that if $\mathcal{H}$ is uniform (property 1 in Definitions 1 and 2 in Section 2.3), then $\mu(h, S, y) = |S|/2^m$. For brevity, we will sometimes use X and μ when h, S, and y are implicit in the context. Our main interest is in understanding how concentrated X is around μ, under various choices of $\mathcal{H}$. We introduce a general notion of concentration that will come handy:

Definition 7. Let $k \geq 0$ and $\delta > 2$. Let X be a random variable with $\mu = \mathbb{E}[X]$. Then X is *strongly* (k, δ)*-concentrated* if $\Pr[|X - \mu| \geq \sqrt{k}] \leq 1/\delta$ and *weakly* (k, δ)*-concentrated* if both $\Pr[X \leq \mu - \sqrt{k}] \leq 1/\delta \text{ and } \Pr[X \geq \mu + \sqrt{k}] \leq 1/\delta$.

For a given δ, smaller k corresponds to higher concentration. Clearly, strong (k, δ)-concentration implies weak (k, δ)-concentration and, for $k' > k$, (k, δ)-concentration implies (k', δ)-concentration. Further, by union bound, weak (k, δ)-concentration implies strong $(k, \delta/2)$-concentration.

As an illustrative example of how (k, δ)-concentration can be used for estimating sizes of high-dimensional sets, consider the following simple randomized algorithm $\mathcal{A}$ for approximately computing $|S|$ with high probability (for example, in a model counting application S could be the set of solutions of a SAT problem, see Section 2.2). Let $\mathcal{H}^i = \{h : \{0,1\}^n \to \{0,1\}^i\}$ for $i = 1, 2, \ldots, n$ be families of pairwise independent hash functions. For i increasing from 1 to m, compute "*is the set $h_t^{-1}(0) \cap S$ empty*" T times, each time with a different hash function h_t chosen uniformly from $\mathcal{H}^i$. If the answer is "yes" for a majority of the T times, stop increasing i and return 2^{i-1} as the estimate of $|S|$.

Proposition 4. *Let $S \subseteq \{0,1\}^n$. If $\mathcal{H}^i = \{h : \{0,1\}^n \to \{0,1\}^i\}$, $i \in \{1, 2, \ldots, n\}$ are universal families of hash functions such that $X(h, S, y)$ is weakly $(\mu^2, 4)$-concentrated, then $\mathcal{A}$ using $\mathcal{H}^i$ and $T \geq 8\ln(8n)$ correctly computes $|S|$ within a factor of 4 with probability at least $3/4$.*

The proof can be found in the Appendix, and is similar to the one of Theorem 1 which proves the main properties of the WISH algorithm. Intuitively, it follows form the concentration properties that $|h_t^{-1}(0) \cap S|$ will be close to $|S|/2^i$ with high probability, and therefore the set $h_t^{-1}(0) \cap S$ will become empty for $i \approx \log(|S|)$.

6.2 Concentration and Hash Families

We discuss how the statistical properties of various hash families $\mathcal{H}$ influence the strength of (k, δ)-concentration of $X = |h^{-1}(y) \cap S|$. Proofs may be found in the appendix. Without any assumptions on the nature of $\mathcal{H}$, Chebychev's inequality and Cantelli's one-sided inequalities yield the following general observation for strong and weak concentration, respectively:

Proposition 5. *Let $\delta > 2$ and $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ be a family of hash functions. For any $S \subseteq \{0,1\}^n$ and $y \in \{0,1\}^m$, and for $h \in_R \mathcal{H}$, $X(h, S, y)$ is strongly $(\delta\sigma^2, \delta)$-concentrated and weakly $((\delta - 1)\sigma^2, \delta)$-concentrated.*

Ideally, one would like to choose a family of fully independent hash functions, which results in very strong concentration guarantees from Chernoff's bounds:

Proposition 6. *Let $\delta > 2$, $c > 3$ and $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ be a family of fully independent hash functions. For any $S \subseteq \{0,1\}^n$ and $y \in \{0,1\}^m$, and for $h \in_R \mathcal{H}$, $X(h, S, y)$ is strongly $((c \ln \delta)\mu, \delta)$-concentrated and weakly $((3 \ln \delta)\mu, \delta)$-concentrated.*

However, as discussed earlier, it is often impossible to construct such a family. One commonly uses a family of only pairwise independent hash functions, which have compact constructions involving objects such as parity or XOR constraints (see constructions in Section 2.3). The concentration guarantees we get are much weaker but still very powerful:

Proposition 7. *Let $\delta > 2$ and $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ be a family of pairwise independent hash functions. For any $S \subseteq \{0,1\}^n$ and $y \in \{0,1\}^m$, and for $h \in_R \mathcal{H}$, $X(h, S, y)$ is strongly $(\delta\mu, \delta)$-concentrated and weakly $((\delta - 1)\mu, \delta)$-concentrated.*

Proof. This follows from observing that pairwise independence implies $\sigma^2 = |S|/2^m(1-1/2^m) < \mu$ and then applying Proposition 5. $\square$

Although one can compactly represent pairwise independent hash functions $\mathcal{H} = \{h : \{0,1\}^n \rightarrow \{0,1\}^m\}$ using m parity constraints (cf. Proposition 1), these constraints have average length $n/2$. As we have seen in Section 5.3, such long parity constraints are often particularly hard to reason about using standard inference methods. To start addressing this issue, we observe that in many applications, $O(\mu)$-concentration is unnecessary and it suffices to have only $O(\mu^2)$-concentration. An example of this is Proposition 4. We next discuss how one can exploit ϵ-SU hash functions (see definition 1 in Section 2.3) to explore this wide spectrum of possible concentrations by varying ϵ.

Theorem 6. *Let $\delta > 2$, $\epsilon \geq 1/2^m$, and $\mathcal{H} = \{h : \{0,1\}^n \rightarrow \{0,1\}^m\}$ be a family of ϵ-SU hash functions. For any $S \subseteq \{0,1\}^n$ and $y \in \{0,1\}^m$, and for $h \in_R \mathcal{H}$, $X(h,S,y)$ is strongly $(\delta\mu(1+\epsilon(|S|-1)-\mu), \delta)$-concentrated and weakly $((\delta-1)\mu(1+\epsilon(|S|-1)-\mu), \delta)$-concentrated.*

Proof. The variance of X can be computed as follows.

$$\begin{aligned}\mathbb{E}[X(h,S,y)^2] &= \sum_{s,s' \in S} \mathbb{E}[1_{h(s)=y,h(s')=y}] = \sum_{s \in S} \mathbb{E}[1_{h(s)=y}] + \sum_{s \neq s'} \mathbb{E}[1_{h(s)=y,h(s')=y}] \\ &\leq \mu + |S|(|S|-1)\epsilon/2^m \quad \text{(from SU)} \\ &= \mu(1+\epsilon(|S|-1))\end{aligned}$$

Therefore, $\sigma^2 = \mathbb{E}[X^2] - \mu^2 \leq \mu(1+\epsilon(|S|-1)-\mu)$. The result now follows from Proposition 5. $\square$

Corollary 2. *Let $\delta > 2$, $\epsilon \geq 1/2^m$, and $\mathcal{H} = \{h : \{0,1\}^n \rightarrow \{0,1\}^m\}$ be a family of ϵ-SU hash functions. For any $S \subseteq \{0,1\}^n$ and $y \in \{0,1\}^m$, and for $h \in_R \mathcal{H}$,*

$X(h,S,y)$ is strongly (μ^2,δ)-concentrated whenever $\epsilon \leq (\frac{\mu}{\delta} + \mu - 1)/(|S|-1)$ and weakly (μ^2,δ)-concentrated whenever $\epsilon \leq (\frac{\mu}{\delta-1} + \mu - 1)/(|S|-1)$.

Thus, by increasing ϵ, we can achieve lower (but still acceptable) levels of concentration of X. In practice, however, it is not easy to construct ϵ-SU families that allow efficient inference. Simply using sparser parity constraints (i.e., with fewer than $n/2$ variables on average), for instance, does not lead to SU hash functions because if $s, s' \in S$ are close in Hamming distance, sparser parity-based hash functions will act on them in a very correlated way. The next section provides a way around this.

6.3 Average Universal Hashing

We now define a new family of hash functions that have the same statistical concentration properties as ϵ-SU (namely, the guarantees in Theorem 6) but are computationally much more tractable for inference methods.

Definition 8. A family of functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is (ϵ, i)-AU (Average Universal) if the following two conditions hold when H is a function chosen uniformly at random from $\mathcal{H}$.

- $\forall x \in \{0,1\}^n$, the random variable $H(x)$ is uniformly distributed in $\{0,1\}^m$.
- $\forall S \subseteq \{0,1\}^n$, $|S| = i$, $\forall y_1, y_2 \in \{0,1\}^m$, the following property holds $\sum_{x_1,x_2 \in S; x_1 \neq x_2} \Pr[H(x_1) = y_1, H(x_2) = y_2] \leq |S|(|S|-1)\epsilon/2^m$.

In other words, we allow pairs of random variables $H(x_1), H(x_2)$ to be potentially heavily correlated, for instance $\Pr[H(x_1) = y_1, H(x_2) = y_2]$ could be much larger than $\epsilon/2^m$. However, it needs to balance out so that the average correlation among configuration pairs on (large enough sets) S is smaller than $\epsilon/2^m$. This is a strict generalization of strongly independent hash functions:

Proposition 8. *Let $\mathcal{H}$ be a family of hash functions. (a) If $\mathcal{H}$ is $(\epsilon, 2)$-AU, then $\mathcal{H}$ is also ϵ-SU. (b) If $\mathcal{H}$ is ϵ-SU, then $\mathcal{H}$ is also (ϵ, i)-AU for all $i \geq 2$. (c) If $\mathcal{H}$ is (ϵ, i)-AU, then $\mathcal{H}$ is also $(\epsilon, i+1)$-AU.*

Theorem 7. *Let $\delta > 2$ and $\mathcal{H}$ be a family of (ϵ, i)-AU hash functions. For any $S \subseteq \{0,1\}^n$, $|S| \geq i$ and $y \in \{0,1\}^m$, and for $h \in_R \mathcal{H}$, $X(h, S, y)$ is strongly $(\delta\mu(1+\epsilon(|S|-1)-\mu), \delta)$-concentrated and weakly $((\delta-1)\mu(1+\epsilon(|S|-1)-\mu), \delta)$-concentrated.*

Proof. The proof is close to that of Theorem 6. The key observation is that whenever $|S| \geq i$, under an (ϵ, i)-AU hash family we obtain the same bound on the variance, σ^2, as in the ϵ-SU hash family case. □

Corollary 3. *Let $\delta > 2$ and $\mathcal{H}$ be a family of (ϵ, i)-AU hash functions. For any $S \subseteq \{0,1\}^n$, $|S| \geq i$ and $y \in \{0,1\}^m$, and for $h \in_R \mathcal{H}$, $X(h, S, y)$ is strongly (μ^2, δ)-concentrated whenever $\epsilon \leq (\frac{\mu}{\delta} + \mu - 1)/(|S|-1)$ and weakly (μ^2, δ)-concentrated whenever $\epsilon \leq (\frac{\mu}{\delta-1} + \mu - 1)/(|S|-1)$.*

6.4 Hashing with Low-Density (Sparse) Parity Constraints

Our main technical contribution is the following construction for AU hash functions based on sparse parity constraints:

Theorem 8. *Let $A \in \{0,1\}^{m\times n}$ be a random matrix whose entries are Bernoulli i.i.d. random variables of parameter $f \leq 1/2$, i.e., $\Pr[A_{ij} = 1] = f$. Let $b \in \{0,1\}^m$ be chosen uniformly at random, independently from A. Let $w^* = \min\left\{w \mid \sum_{j=1}^{w} \binom{n}{j} \geq q\right\}$*

and

$$\epsilon(n, m, q, f) = \frac{1}{|S| - 1} \sum_{w=1}^{w^*} \binom{n}{w} \left(\frac{1}{2} + \frac{1}{2} (1 - 2f)^w \right)^m$$

Then the family $\mathcal{H}^f = \{h_{A,b}(x) : \{0,1\}^n \to \{0,1\}^m\}$*, where* $h_{A,b}(x) = Ax + b \mod 2$ *and* $H \in \mathcal{H}^f$ *is chosen randomly according to this process, is a family of* $(\epsilon(n, m, q, f), q)$*-AU hash functions.*

Proof. Let $S \subseteq \{0,1\}^n$ and $y_1, y_2 \in \{0,1\}^m$. Then,

$$\begin{aligned}
&\sum_{\substack{x_1, x_2 \in S \\ x_1 \neq x_2}} \Pr[H(x_1) = y_1, H(x_2) = y_2] \\
&= \sum_{\substack{x_1, x_2 \in S \\ x_1 \neq x_2}} \sum_{v \in \{0,1\}^m} \Pr\left[Ax_1 + v = y_1, Ax_2 + v = y_2\right] \Pr[b = v] \\
&= 2^{-m} \sum_{\substack{x_1, x_2 \in S \\ x_1 \neq x_2}} \sum_{v \in \{0,1\}^m} \Pr\left[Ax_1 + v = y_1, Ax_2 + v = y_2\right] \\
&= 2^{-m} \sum_{\substack{x_1, x_2 \in S \\ x_1 \neq x_2}} \Pr\left[A(x_1 - x_2) = y_1 - y_2\right]
\end{aligned}$$

For brevity, let $\Delta = y_1 - y_2 \mod 2$. The probability $\Pr\left[A(x_1 - x_2) = \Delta\right]$ depends on the Hamming weight w of $x_1 - x_2$, and is precisely the probability that the w columns of the (sparse) matrix A corresponding to the bits in which x_1 and x_2 differ sum to Δ (mod 2).

In order to compute this probability, we use an analysis similar to MacKay [70], based on treating the m random entries in each of the w columns of A as defining w steps of a *biased random walk* in each of the m dimensions of the m-dimensional Boolean hypercube. The probability that $A(x_1 - x_2)$ equals Δ, when viewed this way, is nothing but the probability that starting from the origin and taking these w steps brings us to Δ. Note that this is a function of only w, f, and

Δ; the exact columns in which x_1 and x_2 differ do not matter. Let us denote this probability $r^{(w,f)}(0,\Delta)$.

Unlike MacKay [70], each row of our matrix A is sampled independently. So we can model the random walk with m independent Markov Chains (one for each of the m dimensions) with two states $\{0,1\}$ and with transition probabilities

$$p_{0\to 0} = 1-\alpha,\ p_{0\to 1} = \alpha,\ p_{1\to 1} = 1-\beta,\ p_{1\to 0} = \beta.$$

Let X_t denote the state of the Markov Chain after t steps. Observing that the eigenvalues for the transition matrix are 1 and $1-\alpha-\beta$, it is easy to verify that

$$\Pr[X_t = 0 \mid X_0 = 0] = \frac{\beta}{\alpha+\beta} + \frac{\alpha}{\alpha+\beta}(1-\alpha-\beta)^t \tag{6.2}$$

and $\Pr[X_t = 1 \mid X_0 = 0] = 1 - \Pr[X_t = 0 \mid X_0 = 0]$. Setting $\alpha = \beta = f$ and $\Delta = y_1 - y_2$ we get

$$\begin{aligned} r^{(w,f)}(0,\delta) &= \prod_{j=1}^{m} \left(\frac{1}{2} + (1-2\Delta_j)\frac{1}{2}\left(1-2f\right)^w \right) \\ &\leq \left(\frac{1}{2} + \frac{1}{2}\left(1-2f\right)^w \right)^m = r^{(w,f)}(0,0) \end{aligned}$$

because $f \leq 1/2$. Plugging into the previous expression we get

$$\begin{aligned} \sum_{\substack{x_1,x_2\in S \\ x_1\neq x_2}} \Pr[H(x_1)=y_1, H(x_2)=y_2] &= 2^{-m} \sum_{\substack{x_1,x_2\in S \\ x_1\neq x_2}} \Pr\left[A(x_1-x_2) = y_1-y_2\right] \\ &\leq 2^{-m} \sum_{x_1\in S}\sum_{w=1}^{n} h(w|x_1) r^{(w,f)}(0,0) \end{aligned}$$

where $h(w|x)$ is defined as the number of vectors in S that are at Hamming distance w from x. Clearly, $h(w|x) \leq \binom{n}{w}$. Since $r^{(w,f)}(0,0)$ is monotonically decreasing in w, we can derive a worst case bound on the above expression by assuming all $\binom{n}{w}$ vectors at distance w from x are actually present in S for small w. Recalling the definition of w^* from the statement of the theorem, for all $|S| \leq q$ this gives:

$$\sum_{\substack{x_1,x_2 \in S \\ x_1 \neq x_2}} \Pr[H(x_1) = y_1, H(x_2) = y_2] \leq 2^{-m} \sum_{x_1 \in S} \sum_{w=1}^{w^*} \binom{n}{w} r^{(w,f)}(0,0)$$

$$= 2^{-m}|S| \sum_{w=1}^{w^*} \binom{n}{w} r^{(w,f)}(0,0) = |S|(|S|-1)\frac{\epsilon(n,m,q,f)}{2^m}$$

which proves that $\mathcal{H}$ is $(\epsilon(n,m,q,f),q)$-AU. $\square$

The significance of this result is that given $n, m,$ and q, we have a family of hash functions parameterized by f for which we can control the "average correlation" across all pairs of points in *any* set of size at least q. These range from rather dense but fully pairwise independent families when $f = 0.5$, to statistically useful but much sparser and hence much more tractable families when $f < 0.5$. Algorithms for probabilistic inference that use universal hashing often do not need full or even pairwise independence but only (μ^2, δ)-concentration for success with high probability. The results in Section 6.3 (Theorem 7 and Corollary 3) thus prescribes a value of $\epsilon > \frac{1}{2^m}$ that suffices to achieve the desired concentration. Using the above theorem, we can therefore look for the smallest value of f such that the resulting $\epsilon(n,m,q,f)$ is compatible with the concentration requirement. For example, using Theorem 8 and Corollary 3, we can look for the smallest value f^* such that the resulting family $\mathcal{H}^f = \{h_{A,b}(x) : \{0,1\}^n \to \{0,1\}^m\}$ guarantees (μ^2, δ)-concentration for sets S of size at least 2^{m+2} and for $\delta = 9/4$.

In the top panel of Figure 6.1 we plot the corresponding $f^*(n)$ as a function of the number of variables n, for $m = n/2$ constraints. In the bottom panel, we plot $f^*(m)$ as a function of m, for $n = 100$. We see that for $m = n/2$ constraints we can obtain hash functions with provable concentration guarantees using constraints with average length scaling empirically as $\log(n)$ as opposed to $n/2$ for the pairwise independent construction. We also plot the smallest value of f that guarantees concentration when the set S in Definition 8 is not an arbitrary set of size $q = 2^{m+2}$ but is instead restricted to be an $(m + 2)$-dimensional hypercube, for which we have an exact expression for $h(w|x)$. This is clearly a lower-bound for f^* (which is guaranteed to work for *any* set, hypercube included). The hypercube is intuitively close to a worst-case distribution for $h(w|x)$ because points have small average distance and hence high correlation. The comparison with the hypercube case highlights that our bounds are fairly tight.

6.5 `WISH` With Sparse Parity Constraints

Our technique and analysis based on Average Universal hash functions applies to a range of probabilistic inference and counting techniques such as WISH [35, 34], ApproxMC [18], and MBound [51]. While preserving all their theoretical properties in terms of approximation guarantees, by substituting pairwise independent hash functions with *sparse* Average Universal ones we obtain significant improvements in terms of runtime. For concreteness and brevity, we discuss its application to the WISH algorithm introduced in Chapter 3.

The original WISH [35] is based on pairwise independent hash functions, constructed using random parity constraints of average length $n/2$ for a problem with n binary variables. Our previous analysis allows us replace them with sparser constraints as in Theorem 8, obtaining an extension that we call SPARSE-WISH of which we provide the pseudocode as Algorithm 3.

Algorithm 3 SPARSE-WISH $(w, n = \log_2 |\mathcal{X}|, \Delta, \alpha)$

```
$T \leftarrow \left\lceil \frac{\ln(1/\Delta)}{\alpha} \ln n \right\rceil$
for $i = 0, \cdots, n$ do
    for $t = 1, \cdots, T$ do
        $f^* = \min\{f | \epsilon(n, i, 2^{i+2}, f) < \frac{31}{5(2^{i+2}-1)}\}$
        Sample hash function $h^i_{A,b}$ from $\mathcal{H}^{f*}$
        i.e. sample sparse $A \in \{0,1\}^{i \times n}$, $b \in \{0,1\}^i$
        $w^t_i \leftarrow \max_\sigma w(\sigma)$ subject to $h^i_{A,b}(\sigma) = \mathbf{0}$
    end for
    $M_i \leftarrow \text{Median}(w^1_i, \cdots, w^T_i)$
end for
Return $M_0 + \sum_{i=0}^{n-1} M_{i+1} 2^i$
```

The key property is that the key Lemma used to prove the correctness of the WISH scheme (Lemma 1 from Chapter 3) still holds:

Lemma 7. *Fix an ordering $\sigma_i, 1 \leq i \leq 2^n$, of the configurations in $\{0,1\}^n$ such that for $1 \leq j < 2^n$, $w(\sigma_j) \geq w(\sigma_{j+1})$. For $i \in \{0, 1, \cdots, n\}$, define $b_i \triangleq w(\sigma_{2^i})$. Let $M_i = \text{Median}(w^1_i, \cdots, w^T_i)$ be defined as in Algorithm 3. Then, for $0 < \alpha \leq 0.0042$,*

$$\Pr\left[M_i \in [b_{\min\{i+2,n\}}, b_{\max\{i-2,0\}}]\right] \geq 1 - \exp(-\alpha T)$$

The proof is based on a variance argument. Intuitively, the hash functions used at iteration i are by Theorem 8 $(\epsilon, 2^{i+2})$-AU with ϵ chosen such that by Corollary 2 and Theorem 7 they guarantee weak $(\mu^2, 9/4)$-concentration for sets of size at least 2^{i+2}. In particular, this guarantees that the hash functions will behave nicely on the set formed by the 2^{i+2} heaviest configurations (no matter what is the structure of the set), and M_i will not underestimate b_i by too much. See Appendix for a formal proof. We then have the following result analogous to Theorem 1 in Section 3.3:

Theorem 9. *For any $\Delta > 0$, positive constant $\alpha \leq 0.0042$, and the hash families $\mathcal{H}^f$ given by Proposition 8, SPARSE-WISH makes $\Theta(n \ln n \ln 1/\delta)$ MAP queries and, with* *probability at least* $(1 - \Delta)$*, outputs a 16-approximation of* $Z = \sum_{\sigma \in \{0,1\}^n} w(\sigma)$.

This means that if we carefully choose the density f^* as in the pseudocode (which is a function of the number of constraints i, and in general much smaller than 0.5; see the bottom panel of Figure 6.1), we maintain the same accuracy guarantees but using much sparser constraints.

6.6 Lower Bounds Using Arbitrarily Sparse Parity Constraints

As we have shown in the previous section, we can use Average Universal (AU) hash functions in place of pairwise independent ones within the SPARSE-WISH probabilistic inference algorithm while preserving all its nice theoretical properties (Theorem 9). The main advantage of AU hash functions is that they can be implemented with parity constraints that are sparser than the ones used to implement fully pairwise independent hash functions. In particular, we provided provable conditions on their average length (controlled by the parameter f, which is the probability that each variable is added to a randomly generated parity constraint) such that if $f \geq f^*$ (the constraints are not too sparse), then the random parity constraints provide enough concentration and they can be safely used within the SPARSE-WISH scheme. A natural question that remains open is what happens if we try to use even sparser constraints. Does WISH compute something useful if we use an *arbitrarily small* $f < f^*$? We provide an answer to this question in the following section. We show that these (potentially extremely sparse) constraints implement a family of *uniform hash functions*, which are generally weaker than AU. When *uniform hash functions* are used in the WISH scheme in place of *pairwise independent* ones, it is not guaranteed

Algorithm 4 LB-WISH $(w, n = \log_2 |\mathcal{X}|, \Delta, \alpha, f)$

```
$T \leftarrow \left\lceil \frac{\ln(1/\Delta)}{\alpha} \ln n \right\rceil$
for $i = 0, \cdots, n$ do
    for $t = 1, \cdots, T$ do
        Sample hash function $h^i_{A,b}$ from $\mathcal{H}^f$
        i.e. sample sparse $A \in \{0,1\}^{i \times n}$, $b \in \{0,1\}^i$
        $w^t_i \leftarrow \max_\sigma w(\sigma)$ subject to $h^i_{A,b}(\sigma) = \mathbf{0}$
    end for
    $M_i \leftarrow \text{Median}(w^1_i, \cdots, w^T_i)$
end for
Return $M_0 + \sum_{i=0}^{n-1} M_{i+1} 2^i$
```

to provide a constant factor approximation of the partition function anymore. However, we will show that *uniform hash functions* can still be used to provide *approximate lower bounds with high probability*.

6.6.1 `WISH` with Uniform Hashing

Let us consider a variation of the WISH algorithm, which takes an additional parameter $f \in (0, 1/2]$ controlling the sparsity of the parity constraints used. We call this algorithm LB-WISH and for completeness we report the pseudocode as Algorithm 4. Notice that LB-WISH $(w, n = \log_2 |\mathcal{X}|, \Delta, \alpha, 1/2)$, where each variable is added with probability $1/2$, is identical to the original WISH algorithm. As we have seen before, for $f < 1/2$ the family of hash functions $\mathcal{H}^f$ defined as in Theorem 8 is not pairwise independent. However we have the following result:

Proposition 9. *Let $A \in \{0,1\}^{m \times n}$ be a random matrix whose entries are Bernoulli i.i.d. random variables of parameter $f \leq 1/2$, i.e., $\Pr[A_{ij} = 1] = f$. Let $b \in \{0,1\}^m$ be chosen uniformly at random, independently from A. Then for any value of $f \in (0, 1/2]$, the family $\mathcal{H}^f = \{h_{A,b}(x) : \{0,1\}^n \to \{0,1\}^m\}$, where $h_{A,b}(x) = Ax + b \mod 2$*

and $H \in \mathcal{H}^f$ is chosen randomly according to this process, is a family of uniform hash functions.

Proof. Let $x \in \{0,1\}^n$, $A \in_R \mathcal{A}^{m\times n}$, $b \in_R \{0,1\}^m$, and a_i denote the i-th row of A. Then, for all i:

$$\begin{aligned}\Pr[a_i x \oplus b_i = 0] &= \sum_{v\in\{0,1\}^n} \Pr[a_i = v] \Pr[vx \oplus b_i = 0] \\ &= \sum_{v\in\{0,1\}^n} \Pr[a_i = v] \Pr[b_i = vx \bmod 2] \\ &= \sum_{v\in\{0,1\}^n} \Pr[a_i = v] \ \frac{1}{2} \ = \ \frac{1}{2}\end{aligned}$$

Hence, $\Pr[Ax+b = 0 \bmod 2] = \prod_i \Pr[a_i x + b_i = 0 \bmod 2] = \frac{1}{2^m}$ for any x, proving uniformity. □

With such a family of hash functions, Theorem 1 as such does not hold, but the following weaker, one-directional version still does:

Theorem 10. *For any $\delta > 0$, positive constant $\alpha \leq 0.0042$ with probability at least $(1-\delta)$, LB-WISH $(w, n = \log_2 |\mathcal{X}|, \Delta, \alpha, f)$ outputs an estimate no larger than $16Z = 16\sum_{\sigma\in\mathcal{X}} w(\sigma)$.*

In other words, even without pairwise independence, the output divided by 16 is a lower bound with high probability. To prove this result, we employ a proof strategy similar to the one used to prove Theorem 1.

Specifically, we prove a new bound on M_i that holds *regardless of pairwise independence*:

Lemma 8. *Suppose $h^i_{A,b}$ is chosen from a family $\mathcal{H}^{n,i}$ of universal (but not necessarily pairwise independent) hash functions. Let $M_i = \mathrm{Median}(w_i^1, \cdots, w_i^T)$ be defined as in*

Algorithm 4 and b_i as in Definition 3. Then, for all $c \geq 2$, there exists an $\alpha^*(c) > 0$ such that for $0 < \alpha \leq \alpha^*(c)$,

$$\Pr\left[M_i \leq b_{\max\{i-c,0\}}\right] \geq 1 - \exp(-\alpha T)$$

Proof. The statement trivially holds when $i - c \leq 0$. Otherwise, let us define the set of the 2^j heaviest configurations as in Definition 3, $\mathcal{X}_j = \{\sigma_1, \sigma_2, \cdots, \sigma_{2^j}\}$. Define the following random variable $S_j(h^i_{A,b}) \triangleq \sum_{\sigma \in \mathcal{X}_j} 1_{\{A\sigma = b \bmod 2\}}$ which gives the number of elements of $\mathcal{X}_j$ satisfying the random parity constraints $A\sigma = b \bmod 2$. The randomness is over the choice of A and b when $h^i_{A,b}$ is sampled from $\mathcal{H}^{n,i}$. Since $\mathcal{H}^{n,i}$ is a family of uniform hash functions, by definition for any σ the random variable $1_{\{A\sigma = b \bmod 2\}}$ is Bernoulli distributed with probability $1/2^i$. Then it follows that $\mathbb{E}[S_j(h^i_{A,b})] = \sum_{\sigma \in \mathcal{X}_j} 1/2^i = \frac{|\mathcal{X}_j|}{2^i} = 2^{j-i}$.

The random variable w_i is defined as $w_i = \max_\sigma w(\sigma)$ subject to $A\sigma = b \bmod 2$. Then we have:

$$\Pr[w_i \leq b_j] = \Pr[w_i \leq w(\sigma_{2^j})] \geq \Pr[S_j(h^i_{A,b}) < 1]$$

which is the probability that no configuration from $\mathcal{X}_j$ satisfies i randomly chosen parity constraints. Notice that $S_j(h^i_{A,b})$ is non-negative, hence from Markov's Inequality, $\Pr[S_j(h^i_{A,b}) \geq 1] \leq \mathbb{E}[S_j(h^i_{A,b})] = 2^{j-i}$. Thus for $j = i - c$ and $c \geq 2$, we have:

$$\Pr[w_i \leq b_{i-c}] \geq \Pr[S_{i-c}(h^i_{A,b}) < 1] \geq 1 - 2^{-c} \geq 3/4$$

Finally, since $w_i^1, \cdots, w_i^T$ are i.i.d. realizations of w_i, we can apply Chernoff's Inequality to the corresponding indicator variables $I_t = I(w_i^t \leq b_{i-c})$ each with mean $\geq 3/4$ and obtain:

$$\Pr\left[M_i \leq b_{i-c}\right] = \Pr\left[\sum_t I_t \geq T/2\right] \geq 1 - \exp(-\alpha^*(c)T)$$

where $\alpha^*(2) = 2(3/4 - 1/2)^2 = 1/8$. □

With this new lemma, we have all we need to prove Theorem 10. The proof is identical to the one of Theorem 1.

6.7 Experimental Evaluation

We evaluate SPARSE-WISH using the Integer Linear Programming (ILP) formulation from Section 5.2 to solve the MAP inference instances in the inner loop of the algorithm. We use the Integer Programming solver CPLEX with a timeout of 10 minutes on Intel Xeon 5670 3GHz machines with 48GB RAM, obtaining at the end a lower bound and, by solving a sequence of LP relaxations, an upper bound on the optimization instances. These translate into bounds for the generally intractable partition function Z [1]. We evaluate these bounds on $M \times M$ grid Ising models for $M \in \{10, 15\}$. In an Ising model, there are M^2 binary variables, with unary potentials $\psi_i(x_i) = \exp(f_i x_i)$ and (mixed) binary interactions $\psi_{ij}(x_i, x_j) = \exp(w_{ij} x_i x_j)$, where $w_{ij} \in_R [-w, w]$ and $f_i \in_R [-F, F]$. The external field is $F \in \{0.1, 1.0\}$.

We compare SPARSE-WISH with WISH (based on the same ILP formulation, but with denser $f = 0.5$ constraints which on average involve half of the variables), with Loopy BP [74] which estimates Z without providing any accuracy guarantee, Tree Reweighted BP [98] which gives a provable upper bound, and Mean Field [99] which gives a provable lower bound. We use the implementations in the LibDAI library [73], allowing 1000 random restarts for Mean Field. Figure 6.2 shows the error in the resulting estimates, where ground truth is from Junction Trees [66].

[1]When all the ILPs are solved to optimality, upper and lower bounds match and the value is guaranteed to be a constant factor approximation for Z.

We see that SPARSE-WISH provides significantly better bounds compared to the original WISH algorithm. Since they are both run for the same amount of time using the same combinatorial optimization suite, the improvement is to be attributed entirely to the sparser constraints employed by SPARSE-WISH, which are easier to reason about. Intuitively, as seen in Figure 5.3 the LP relaxation obtained using shorter XORs is much tighter, hence improving the quality of the bounds, and yielding overall the best provable upper and lower bounds among all algorithms we considered. Notice the improvement in terms of lower bound is smaller because in both cases the bounds are quite tight (with an error close to 0). Remarkably, SPARSE-WISH is the only method that does not deteriorate as the coupling strength is increased. We emphasize that the improvement over WISH comes at no cost, because thanks to our carefully chosen density thresholds f^*, we maintain the same theoretical properties without trading off accuracy for speed.

6.7.1 Model Counting for SAT

Long parity constraints are difficult to reason about not only for Integer Programming solvers but also for SAT solvers. In fact, SAT solvers can be substantially faster on sparser parity constraints than those of length $n/2$ [51].

The use of short parity constraints for model counting (i.e., count the number of solutions of a SAT instance) was investigated by Gomes et al. [52], where it was empirically shown that short XORs perform well on a wide variety of instances a number of problem domains. Our analysis provides the first theoretical basis for this observed empirical phenomenon, while also providing a principled way to estimate a priori a suitable length of parity constraints to use.

Table 6.1 reports the bounds obtained with our analysis on the benchmark used by Gomes et al. [52]. The best previously known theoretical bound on the length was $n/2$, based on the pairwise independent construction (Proposition 1). The best previously known empirical bound was computed by finding the smallest XOR length such that the variance of the resulting model count estimate is the same as what one would obtain with pairwise independent functions, which can be easily computed analytically. The new provable bound is computed by looking for the shortest XOR length that gives (μ^2, δ)-concentration and therefore provides a "correct" answer more than half the time (as in Proposition 4). Specifically, by Theorem 7, we look for the minimum XOR length satisfying the weak $(\mu^2, 9/4)$-concentration condition given by Corollary 2, where the number of variables (n), the log of the set size ($\log_2 |S|$), and the number of XORs (m) are taken from Gomes et al. [52] and reported in the first three columns of Table 6.1. The new empirical bound is also based on the shortest XOR length yielding weak $(\mu^2, 9/4)$-concentration, but using Proposition 5 for general hash families and taking the sample variance as a proxy for the true variance, σ^2.

On this diverse benchmark spanning a variety of domains (Latin square completion, logistic planning, hardware verification, random, and synthetic), the new theoretical bound on the minimum XOR length is significantly smaller than $n/2$. The empirical bound we achieve (last column, often in single digits and thus extremely efficient for SAT solvers) is also much smaller than the previously reported empirical bound. The gap between our new provable and empirical bounds is due to the intricate structure of the set S of solutions. If the

Instance				Provable Bounds		Empirical Bounds	
	Num	$\log_2$	Num				
Name	Vars	Solns	XORs	Old	New	Old	New
ls7R34med	119	10	7	59	46	6	3
ls7R35med	136	12	9	68	53	7	3
ls7R36med	149	14	11	74	56	7	3
log.c.red	352	19	10	176	112	98	28
2bitmax_6	252	97	87	126	26	–	8
wff-3-100-330	100	32	25	50	21	17	7
wff-3-100-380	100	22	15	50	27	26	8
wff-3-100-396	100	18	11	50	29	38	10
string-50-30	50	30	20	25	8	11	4
string-50-40	50	40	30	25	5	>10	4
string-50-49	50	49	39	25	3	6	3
blk-50-3-10-20	50	23	13	25	10	>15	5
blk-50-6-5-20	50	26	16	25	9	15	4
blk-50-10-3-20	50	30	20	25	8	15	3

Table 6.1: Provable and empirical minimum length of random XOR constraints to be suitable for model counting for SAT on benchmark instances.

solutions in S are far away on average (hence less correlated when using sparse parity constraints), we obtain an empirical variance that is much tighter than our provable worst-case bound. Overall, our new bounds significantly improve upon the previous best known bounds in all cases considered.

6.8 Discussion

We explored the use of sparse low density parity check constraints inside the recent family of probabilistic inference techniques based on universal hash functions, providing a new set of guarantees on the results obtained. We showed how we can utilize arbitrarily sparse constraints (corresponding to uniform hash functions, without any other guarantee on the independence of the action), while still providing probabilistic lower bound guarantees for the problem of computing the partition function. We introduced a new class of hash functions, called Average Universal, that can be constructed using low-density (but sufficiently dense, based on a provable analytic condition) parity constraints. While statistically weaker than traditional strongly universal hash functions, these are still powerful enough to be used in hashing-based randomized probabilistic inference schemes such as WISH. Sparse parity constraints are empirically much easier to do inference with, a well-known fact in the context of low-density parity check codes. By substituting dense parity constraints with sparser ones, we obtain variations of inference and counting techniques that have the same provable guarantees but are empirically much more tractable. Finally, we showed empirically that by solving a sequence of LP relaxations we can obtain not only very accurate lower bounds but also upper bounds that are much tighter than the ones provided by TRWBP, which is based on tree decomposition and convexity. We also showed that using average universal hash function in place of strongly universal ones (and hence sparser parity constraints) leads to significant improvements in the bounds obtained by the recent WISH algorithm and in model counting applications for SAT.

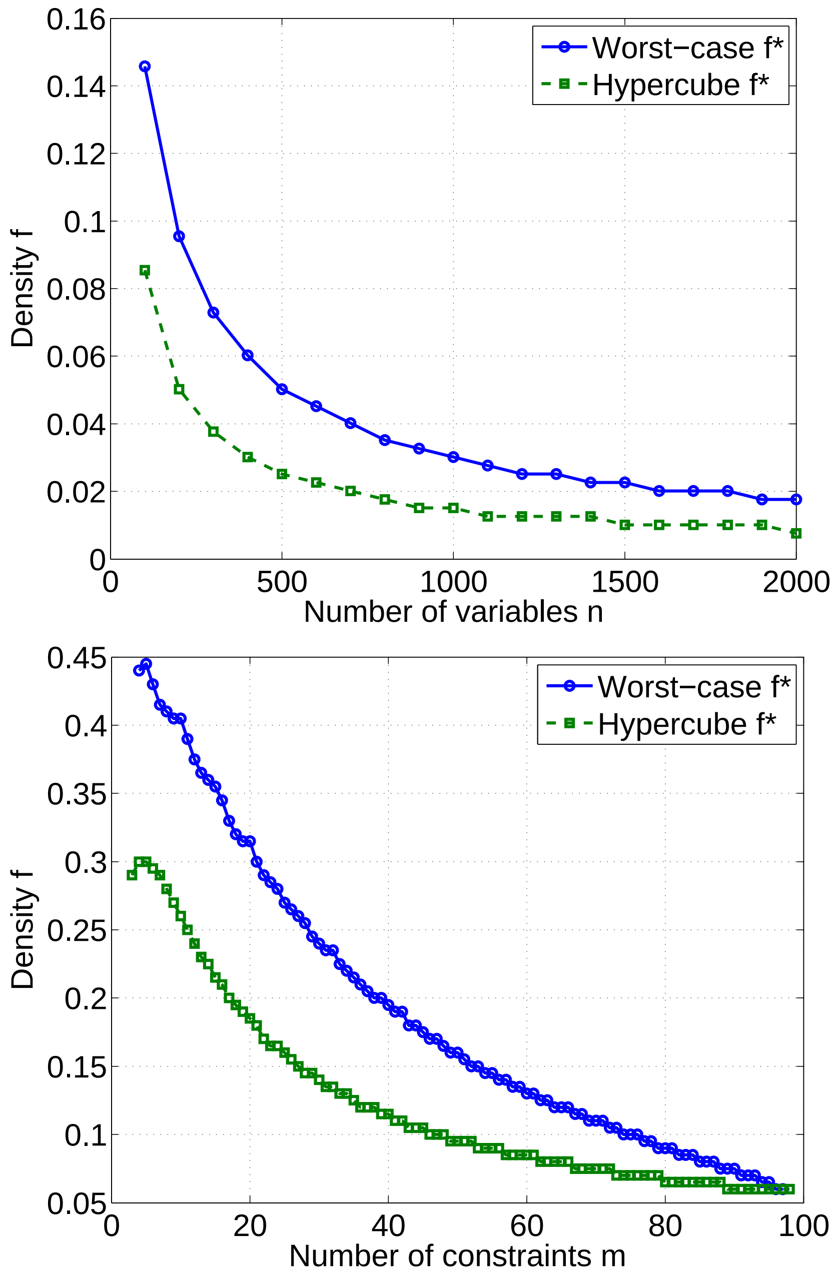
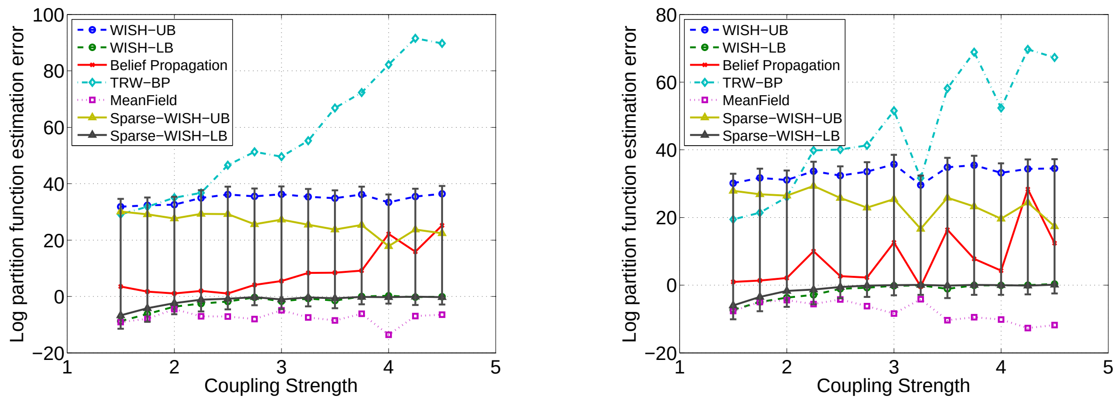
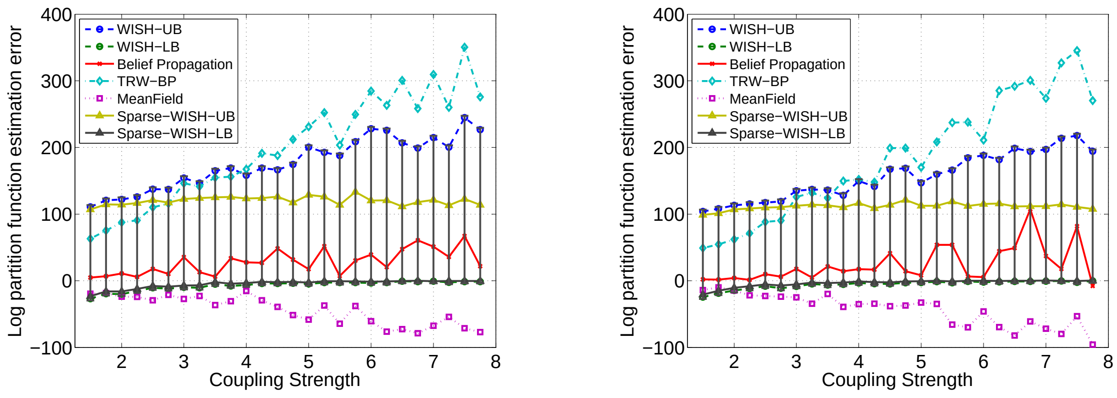


Figure 6.1: Evaluation of the bound on the density of the parity constraints required to achieve sufficiently good statistical properties of the corresponding family of hash functions, as a function of the dimensionality n and the number of constraints m. Top: $m = n/2$, varying n. Bottom: $n = 100$, varying m.



(a) Mixed 10×10. Field 1.0.



(b) Mixed 15×15. Field 1.0.

Figure 6.2: Bounds on the partition function of Ising grids with mixed interactions. The bounds are obtained from LP relaxations using both dense and sparse parity constraints implementing respectively Strongly Universal and Average-Universal hash functions. Top: 10×10 grid. Bottom: 15×15 grid.

CHAPTER 7

PROBABILISTIC INFERENCE WITH COMPLEX BACKGROUND KNOWLEDGE FOR THE DISCOVERY OF NEW MATERIALS

In recent years, we have witnessed an unprecedented growth in data generation rates in many fields of science [57]. For instance, in combinatorial materials discovery, one searches for materials with new desirable properties by obtaining measurements on hundreds of samples in a single batch experiment [41, 75]. These are referred to as 'high-throughput' experiments, and are common to many other fields such as molecular biology or astronomy, where there is a need to optimize the data throughput of high-cost equipment [2]. As manual data analysis is becoming more and more impractical, there is a growing need to develop new techniques to automatically analyze and interpret such vast amount of data for important trends and results. Modern statistical machine learning and data-mining approaches have been quite effective in extracting relevant information from the ever increasing streams of raw digital data. However, in scientific data analysis, there is a large amount of rather complex domain-specific background knowledge that needs to be taken into account, such as the physical and chemical properties of the materials in the combinatorial materials discovery domain.

In this Chapter, we describe a novel approach to the phase map identification problem, a key step towards understanding the properties of new materials created and examined using the combinatorial materials discovery method. The process of identifying a phase map has been traditionally carried out manually by domain-experts, but a completely automatic solution for the phase map identification problem would open the way for even more automation in the combinatorial approach pipeline. Further, a scalable and reliable automatic data interpretation procedure would allow us to analyze larger datasets that go beyond the capabilities of human experts.

In our approach, we integrate domain-specific scientific background knowledge about the physical and chemical properties of the materials into an SMT reasoning framework based on linear arithmetic. The problem has a hybrid nature, with continuous measurement data, discrete decision variables and combinatorial constraints at the same time. We show that using our novel encoding, state-of-the-art SMT solvers can automatically analyze large synthetic datasets, and generate interpretations that are physically meaningful and very accurate, even in the presence of artificially added noise. Moreover, our approach scales to realistic-sized problem instances, outperforming a previous approach based on Constraint Programming and a set-variables encoding [67]. Finally, we can leverage the reasoning power of SMT solvers to jointly reason about the data and the background knowledge in a fully *probabilistic way* using the PAWS and WISH schemes presented in this thesis. For example, using the PAWS scheme presented in Chapter 4, we can sample interpretations of the data which are consistent with the background knowledge using a state-of-the-art SMT solver as a black box. We think this approach will open a range of new possibilities for combining deductive reasoning based on prior knowledge (e.g., the physics of the system) and inductive reasoning based on experimental data, providing an elegant way to leverage the reasoning power of a state of the art constraint optimizer like an SMT solver in a machine learning setting where we want to extract knowledge from data. We see this work as a first step towards using automated reasoning technology to aid the scientific discovery process. While several aspects of our method are specific to the materials discovery application, the approach we take to scientific data analysis is general. Given the flexibility and reasoning power of modern day SMT solvers, we expect to see more applications of this technology to other fields of science.

7.1 Combinatorial Materials Discovery

The combinatorial method is a general experimentation setting where many simultaneous experiments are performed and analyzed in batch at each step. This experimental methodology is intended to speed up the scientific discovery process, and is becoming common in a number of areas, including catalyst discovery, drug discovery, polymer optimization, and chemical synthesis. For example, new catalysts have been discovered 10 to 30 times faster using the combinatorial approach rather than conventional methodology [41, 75]. This is an important research direction in the field of Computational Sustainability, for instance because new materials with improved catalytic activity can be used for fuel cell applications [55].

In this paper, we consider a combinatorial materials discovery approach called *composition-spread*, that has been recently applied with success to speed up the discovery of new catalysts [96]. In the composition spread approach, three metals (or oxides) are sputtered onto a silicon wafer using guns pointed at three distinct locations, resulting in a so-called *thin film*. Different locations on the silicon wafer correspond to different concentrations of the sputtered materials, depending on their distance from the gunpoints. During experimentation, a number of locations (samples) on the thin film are examined using an x-ray diffraction technique, obtaining a diffraction pattern for each sampled point that gives the intensity of the electromagnetic waves as a function of the scattering angle of radiation. The observed diffraction pattern is closely related to the underlying crystal structure, which provides important insights into chemical and physical properties of the corresponding composite material.

A key step towards understanding the chemical and physical properties of the composite materials on a *thin film* is to obtain a so-called *phase map*, that is used to identify regions of the silicon wafer that share the same underlying crystal structure (see Figure 7.2 for an example). Intuitively, the idea is that the different diffraction patterns observed across the *thin film* can all be explained as combinations of a small number (typically, less than 6) of diffraction patterns called *basis patterns* or *phases*. Finding the phase map corresponds to identifying these *basis patterns* and their location on the silicon wafer. This is a challenging task because we only observe combinations of the *basis patterns*, and the measurements are affected by noise. Furthermore, due to a fairly complicated physical process dealing with the expansion of crystals on the lattice, *basis patterns* can appear scaled (contracted to a smaller or larger frequency range), and they must satisfy a number of physical constraints (for instance, basis patterns must appear in contiguous locations on the *thin film* and there is a maximum number of *basis patterns* that can appear in each sample diffraction pattern).

7.1.1 Phase Map Identification

Formally, we are given P diffraction patterns $\mathbf{D}_0, \cdots, \mathbf{D}_{P-1}$, one for each of the P points sampled on the *thin film*, where each vector $\mathbf{D}_i = (d_{0,i}, \cdots, d_{B-1,i}) \in (\mathbb{R}_{\geq 0})^B$ represents the intensity of the electromagnetic waves for a fixed set of B scattering angles of radiation. The sample points are embedded into a graph $\mathcal{G}$, such that there is a vertex for every point and edges connect points that are close on the *thin film* (for instance, based on Delaunay triangulation). Given a norm $||\cdot||$ (for instance, an L_∞ norm), we want to find K basis patterns $\mathbf{B}_0, \cdots, \mathbf{B}_{K-1}$ where $\mathbf{B}_i \in (\mathbb{R}_{\geq 0})^B$, coefficients $a_{i,j} \in \mathbb{R}$ and scaling factors $s_{i,j} \in \mathbb{R}$ for $i = 0, \cdots, P-1, j = 0, \cdots, K-1$ that minimize

$$\sum_{i=0}^{P-1} ||\mathbf{D}_i - \sum a_{i,j} S\left(\mathbf{B}_j, s_{i,j}\right)|| \tag{7.1}$$

where $S(\cdot)$ is an operator modeling the scaling phenomena (see below), and the coefficients $a_{i,j}$ must satisfy

$$\begin{aligned} a_{i,j} \geq 0 \qquad & i = 0, \cdots, P-1, j = 0, \cdots, K-1 \\ |\{j | a_{i,j} > 0\}| \leq M \qquad & i = 0, \cdots, P-1 \end{aligned}$$

that is, they are non-negative and no more than M basis patterns can be used to explain a point i. Furthermore, the subgraph induced by $\{i | a_{i,j} > 0\}$ must be connected for $j = 0, \cdots, K-1$ (so that the basis patterns appear in contiguous locations on the *thin film*). The scaling operator $S(\cdot)$ models the potential expansion of the crystals on the lattice. Specifically, a peak appearing at scattering angle a in the k-th basis pattern might appear respectively at scattering angles $s_{p,k} \cdot a$ and $s_{p',k} \cdot a$ at points p, p' of the silicon wafer because of the scaling effect. For each basis pattern k, the corresponding scaling coefficients $s_{i,k}$ must be continuous and monotonic as a function of the corresponding location i on the *thin film*. Further, the presence of 3 or more basis patterns in the same point prevents any significant expansion of the crystals, and therefore scalings do not occur.

Notice that this formulation is closely related to a principal component analysis (PCA) of the data, but includes additional constraints needed to ensure that the solution is physically meaningful, such as the non-negativity of eigenvectors, connectivity, and phase usage limitations.

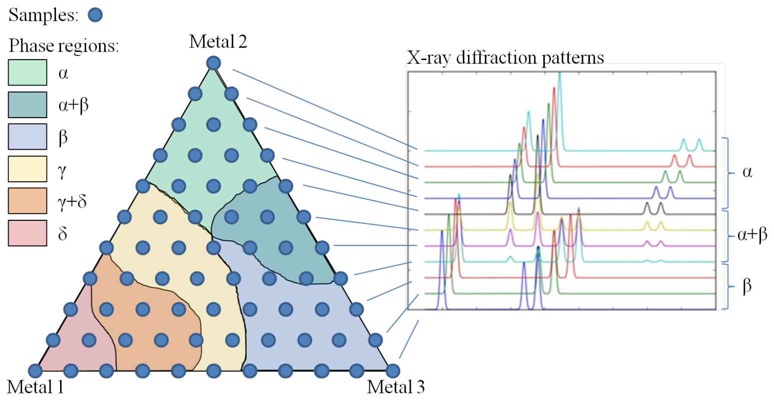


Figure 7.1: Left: Pictorial depiction of the phase map identification problem, showing a set of sampled points on a *thin film*. Each sample corresponds to a different composition, and has an associated measured X-ray diffraction pattern. Colors correspond to different combinations of the basis patterns $\alpha, \beta, \gamma, \delta$. On the right: Scaling (shifting) of the diffraction patterns as one moves from one point to a neighboring one.

7.2 Prior Work

There have been several attempts to automate the *phase map* identification process. Most of the solutions in the literature are based on unsupervised machine learning techniques, such as clustering and non-negative matrix factorization [68, 69]. While these approaches are quite effective at extracting information from large amounts of noisy data, their major limitation is that it is hard to enforce the physical constraints of the problem at the same time. As a result, the interpretations obtained with these techniques are often not physically meaningful, for instance because regions corresponding to some basis patterns are not connected [67].

To address these limitations, in [67] they used a Constraint Programming approach to enforce the constraints on the phase maps, defining a new problem called *Pattern Decomposition with Scaling*. They propose an encoding based on set variables, but the main limitation of their work is that current state-of-the-art CP solvers cannot scale to realistic size instances (e.g., with at least 40 sample points). To overcome this limitation, the authors used a heuristic preprocessing step based on clustering to fix the value of certain variables before attempting to solve the problem. While the solutions they found are empirically shown to be accurate, their strategy cannot provide any guarantee because it only explores part of the search space.

Our approach is similar to the one proposed in [67], but in this work we introduce a novel SMT encoding based on arithmetic to formulate the phase map identification problem. The SMT formalism nicely captures the hybrid nature of the problem, which involves discrete decision variables and continuous measurement data at the same time. Furthermore, we show that the ability to reason at the level of arithmetic operations of SMT solvers allows our approach to scale to instances of realistic size without need for Machine Learning-based heuristics.

7.3 SMT-Aided Phase Map Identification

In our first attempt to model the phase map identification problem, we constructed an SMT-based model where we described the entire spectrum of all the unknown basis patterns $\mathbf{B}_0, \cdots, \mathbf{B}_{K-1}$. However, this approach requires too many variables to obtain a sufficiently fine-grained description of the diffraction patterns, and ultimately leads to instances that cannot be solved using current state-of-the art solvers. We therefore use the same approach presented in [67], and we preprocess the diffraction patterns $\mathbf{D}_0, \cdots, \mathbf{D}_{P-1}$ using a peak detection algorithm, extracting the locations of the *peaks* $\mathcal{Q}(p)$ in the x-ray diffraction pattern of each point p (see Figure 7.1). This is justified by the nature of the diffraction patterns, as constructive interference of the scattered x-rays occurs at specific angles (thus creating peaks of intensities) that characterize the underlying crystal. Furthermore, matching the locations of the peaks is what human experts do when they try to manually solve these problems.

Given the sets of observed peaks $\{\mathcal{Q}(p)\}_{p=0}^{P-1}$ extracted from the measured diffraction patterns $\mathbf{D}_0, \cdots, \mathbf{D}_{P-1}$, our goal is to find a set of peaks $\{\mathcal{E}_k\}_{k=0}^{K-1}$ for the K basis patterns that can explain the observed sets of peaks $\{\mathcal{Q}(p)\}_{p=0}^{P-1}$. The new variables $\{\mathcal{E}_k\}_{k=0}^{K-1}$ therefore replace the original variables $\mathbf{B}_0, \cdots, \mathbf{B}_{K-1}$ in the problem described earlier in Section 2. For each peak $c \in \mathcal{Q}(p)$ we want to have at least one peak $e \in \mathcal{E}_k$ that can explain it, i.e.

$$\forall c \in \mathcal{Q}(p) \exists e \in \mathcal{E}_k \ s.t.\ (a_{p,k} > 0 \wedge \ |c - s_{p,k} \cdot e| \leq \epsilon)$$

where ϵ is a parameter that depends on how accurate the peak-detection algorithm is. Notice that we match the location of the peak, which can be measured accurately, but not its intensity, which can be very noisy. At the same time, we want to limit the number of missing peaks, i.e. peaks that should appear because they belong to some basis pattern but have not actually been measured. Therefore, instead of optimizing the objective in equation (7.1), we consider an approximation given by

$$\sum_{p=0}^{P-1} \sum_{k=0}^{K-1} \mathbb{1}_{\left[a_{p,k}>0\right]} \sum_{e \in \mathcal{E}_k} \mathbb{1}_{\left[\forall c \in \mathcal{Q}(p), |c - s_{p,k} \cdot e| > \epsilon\right]}$$

that gives the total number of missing peaks. All the other constraints of the problem previously introduced are not affected and still need to be satisfied. Note that we can avoid the use of expensive non-linear arithmetic by using a logarithmic scale for the x-ray data, so that multiplicative scalings become linear operations. We refer to these effects (corresponding to the scalings in the original problem formulation) as *shifts*. For each point, we therefore define a set $\mathcal{A}(p) = \{\log q, q \in \mathcal{Q}(p)\}$ of peak positions in log-scale and similarly we represent the positions of the peaks of the basis patterns using the same logarithmic scale.

After a preliminary investigation where we evaluated the performance of real-valued arithmetic, we decided to discretize the problem and use Integer variables to represent peak locations (with a user-defined discretization step). Since the diffraction data is measured using digital sensors, there is no actual loss of information if we use a small enough discretization step, and it significantly improves the efficiency of the solvers. In the resulting SMT model we therefore use a *quantifier-free linear integer arithmetic theory*.

7.3.1 Model Parameters

Let P be the number of sampled points on the *thin film*. We define L as the maximum number of peaks per point, i.e. $L = \max_p |\mathcal{A}_p|$. Based on the observed patterns, we precompute an upper and lower bound e_{max} and e_{min} for the positions of the peaks: $e_{max} = \max_p \max_{a \in \mathcal{A}(p)} a$, $e_{min} = \min_p \min_{a \in \mathcal{A}(p)} a$. There are also a number of user-defined parameters. K is the total maximum number of basis patterns used to explain the observed diffraction patterns, while M is the maximum number of basis patterns that can appear in any point p. ϵ is a tolerance level such that two peaks within an interval of size 2ϵ are considered to be overlapping. ϵ_S is a bound on the maximum allowed difference in the shifts of neighboring locations on the thin film, while S_{max} is a bound on the maximum possible shift. Furthermore, the user specifies a parameter T which gives a bound on the total number of peaks that should appear because they belong to some basis pattern but have not actually been measured (we will refer to them as *missing peaks*).

7.3.2 Variables

We use a set of Boolean variables

$$r_{p,k}, \qquad p = 0, \cdots, P-1, k = 0, \cdots, K-1$$

where $r_{p,k} = \mathit{TRUE}$ means that phase (basis pattern) k appears in point p (i.e., $a_{p,k} > 0$). We also have the following *Integer* variables:

$$\begin{aligned} e_{k,\ell}, &\qquad k = 0, \cdots, K-1, \ell = 0, \cdots, L-1 \\ S_{p,k}, &\qquad p = 0, \cdots, P-1, k = 0, \cdots, K-1 \\ I_{p,k}, &\qquad p = 0, \cdots, P-1, k = 0, \cdots, K-1 \\ t_p, &\qquad p = 0, \cdots, P-1 \end{aligned}$$

where $e_{k,\ell}$ represents the position of the ℓ-th peak of the k-th basis pattern. $S_{p,k}$ represents the shift of the k-th basis pattern at point p. The variables $I_{p,k}$ are redundant and used to count the number of phases used at point p. The variables t_p represent the number of unexplained peaks at point p, i.e. the number of missing peaks at point p. These are peaks that should appear according to the values of $\{r_{p,k}\}_{k=0}^{K-1}$, $\{e_{k,\ell}\}_{\ell=0}^{L-1}$, and $\{S_{p,k}\}_{k=0}^{K-1}$, but are not present, i.e. they do not belong to $\mathcal{Q}(p)$.

7.3.3 Constraints Encoding Prior Knowledge

The variables $I_{p,k}$ are Integer indicators for the Boolean variables $r_{p,k}$ that must satisfy

$$0 \leq I_{p,k} \leq 1 \; k = 0, \cdots, K-1, p = 0, \cdots, P-1$$

$$r_{p,k} \Leftrightarrow (I_{p,k} = 1) \; k = 0, \cdots, K-1, p = 0, \cdots, P-1$$

Peak locations $e_{k,\ell}$ in the basis patterns are bounded by what we observe in the x-ray diffraction pattern:

$$e_{min} \leq e_{k,\ell} \leq e_{max}, \qquad k = 0, \cdots, K-1, \ell = 0, \cdots, L-1$$

Shifts are bounded by the maximum allowed shift, and can be assumed to be non-negative without loss of generality:

$$0 \leq S_{p,k} \leq S_{max}, \; k = 0, \cdots, K-1, p = 0, \cdots, P-1$$

Every peak $a \in \mathcal{A}(p)$ appearing at point p must be explained by at least one peak belonging to one phase k, which can appear shifted by $S_{p,k}$:

$$\bigvee_{k=0}^{K-1} \bigvee_{\ell=0}^{L-1} \big(r_{p,k} \wedge (|e_{k,\ell} + S_{p,k} - a| \leq \epsilon) \big) \forall p, \forall a \in \mathcal{A}(p)$$

Inequalities involving the absolute value of an expression of the form $|e| < c$ where c is a positive constant are encoded as $(e < c) \wedge (e > -c)$.

If a phase k is chosen for point p (i.e., $r_{p,k} = \mathit{TRUE}$), then most of the peaks $e_{k,0}, \cdots, e_{k,L-1}$ should belong to $\mathcal{Q}(p)$. We count the number of missing peaks as follows:

$$t_p = \sum_{k=0}^{K-1} \sum_{\ell=0}^{L-1} ITE(r_{p,k} \wedge \neg \left(\bigvee_{a \in \mathcal{A}(p)} (|e_{k,\ell} + S_{p,k} - a| \leq \epsilon) \right), 1, 0), \forall p$$

where ITE is an if-then-else expression. Here we assume that each phase contains at least one peak, but since peaks can be overlapping (e.g., $e_{k,\ell} = e_{k,\ell+1}$) a basis pattern is allowed to contain less than L distinct peaks.

Missing Peaks Bound

We limit the number of total missing peaks (across all points p) with the user-defined parameter T

$$\sum_{p=0}^{P-1} t_p \leq T$$

Intuitively, the smaller T is, the better an interpretation of the data.

Phase Usage

There is a bound M on the total number of phases that can be used to explain the peaks observed at any location p:

$$\sum_{k=0}^{K-1} I_{p,k} \leq M, p = 0, \cdots, P-1$$

For instance, when three metals or oxides are used to obtain the thin film, we have a *ternary system*, where no more than three phases can appear in each point p, that is $M = 3$.

Shift Continuity

Phase shifting is a continuous process over the *thin film*. We therefore have the following constraint:

$$|S_{p,k} - S_{p',k}| < \epsilon_S, \forall p, \forall p' \in \mathcal{N}(p)$$

where $\mathcal{N}(p)$ is the set of neighbors of p according to the connectivity graph $\mathcal{G}$ (i.e., points that lie close to p on the *thin film*).

Shift Monotonicity

Let $\mathcal{D} = (d_0, \cdots, d_t)$ where $d_i \in \{0, \cdots, P-1\}$ be a sequence of points that lie in a straight line on the thin film. Shifting is a monotonic process, i.e. it must satisfy the following constraint

$$\left(\bigwedge_{i=0}^{t-1} \left(S_{d_i,k} \geq S_{d_{i+1},k}\right)\right) \vee \left(\bigwedge_{i=0}^{t-1} \left(S_{d_i,k} \leq S_{d_{i+1},k}\right)\right), k = 0, \cdots, K-1$$

Since points are usually collected on a grid lattice on the silicon wafer, we enforce ?hift monotonicity?on the lines forming the grid.

Ternary Phases Shift

Ternary phases (where 3 basis patterns are used) are not affected by shifting:

$$\left(\left(\sum_{k=0}^{K-1} I_{p,k} = 3\right) \wedge \bigwedge_{k=0}^{K-1} \left(r_{p,k} \Leftrightarrow r_{p',k}\right)\right) \Rightarrow \left(S_{p,k} = S_{p',k}\right), \forall p, \forall p' \in \mathcal{N}(p)$$

where $\mathcal{N}(p)$ is the set of neighbors of p.

Connectivity Constraint

Each of the basis patterns must be connected. Formally, for every pair of points p, p' such that $r_{p,k} \wedge r_{p',k}$, there must exist a path $\mathbb{P}$ from p to p' such that $r_{j,k} =$ *TRUE* for all $j \in \mathbb{P}$. Since it would require too many constraints, we use a lazy approach to enforce connectivity. If we find a solution where a basis pattern k

is not connected, i.e. there exists p, p' such that $r_{p,k} \wedge r_{p',k}$ but there is no path $\mathbb{P}$ with p, p' as endpoints such that $r_{j,k} = \mathit{TRUE}$ for all $j \in \mathbb{P}$, then we consider the smallest cut C between p and p' such that $r_{j,k} = FALSE$ for all $j \in C$ and we add a new constraint

$$(r_{p,k} \wedge r_{p',k}) \Rightarrow \bigvee_{c \in C} r_{c,k}$$

Symmetry Breaking

Without loss of generality, we can impose an ordering on the peak locations within every phase k:

$$e_{k,\ell} \leq e_{k,\ell+1}, \ell = 0, \cdots, L-2, k = 0, \cdots, K-1$$

Furthermore, notice that the problem is symmetric with respect to permutations of the phase indexes $k = 0, \cdots, K-1$. We therefore enforce an ordering on the way phases are assigned to points

$$\textstyle\bigwedge_{k=1}^{K-1} (r_{0,k} \Rightarrow r_{0,k-1})$$

$$\cdots$$

$$\textstyle\bigwedge_{j=1}^{K-1} \left(\left(\bigwedge_{i=0}^{Y} \neg r_{i,j} \right) \Rightarrow \bigwedge_{k=j}^{K-1} (r_{Y+1,k} \Rightarrow r_{Y+1,k-1}) \right)$$

where we set $Y = 4$.

7.4 Experimental Results

We evaluate the performance of our approach on a benchmark set of synthetic instances for which the ground truth is known (namely, what the true basis patterns are and how they are combined to form the observed diffraction patterns).

All the systems we consider are ternary, where three metals are combined, so that M is set to 3 in the entire experimental section. For all experiments, two peaks are considered to be overlapping if they are within 1% of each other, and the maximum allowed shift is 15%.

We compare our SMT-based approach with the Constraint Programming based solution presented in [67]. Since their CP-based formulation does not scale to realistic-sized instances, they integrate a Machine Learning based component to simplify the problem that the CP solver needs to solve to improve scalability. Note that by doing this they lose the completeness of the search, because they only explore a subtree (suggested by the ML part) of the original search space. In contrast, our approach scales to instances of realistic size (with over 40 points) without need for the ML component. Note however that if desired, the ML heuristic component could be easily integrated with our method.

Synthetic Data

We consider the known Al-Li-Fe system previously used in [67], represented with a ternary diagram in Figure 7.2. A ternary diagram is a simplex where each point corresponds to a different concentration of the three constituent elements, in this case Al, Li, and Fe. The composition of a point depends on its distance from the corners. For a fixed value of the parameter P, synthetic instances are generated by sampling P points in the ternary diagram, each corresponding to different concentrations of the three constituent elements. For each point, synthetic x-ray diffraction patters are generated starting from known diffraction patterns of the constituent phases, that are combined according to the concentrations of the elements in that point. A peak detection algorithm is then used to generate a discrete set of peaks.

We first consider a set of instances without any noise, for which we have the exact location of all the peaks for every sample (the maximum number of peaks per sample is $L = 12$), without any outlier or missing peak. Starting from the diffraction patterns and the corresponding peaks, we generate the corresponding instance using the formulation described in the previous section, encoded in the SMTLibV2 language. In this case, we set $K = 6$, the true number of underlying unknown basis patterns, and we try to recover a solution with $T = 0$ missing peaks. We also consider a set of simplified instances, where we fix some of the six unknown basis patterns to their true values. We solved these instances on a 3 Ghz Intel Core2Duo machine running Windows, using the SMT solvers Z3 [20] and MathSAT5 [56]. However, MathSAT is significantly slower (for instance, it takes over 50 minutes to solve a small instance with $P = 10$ points that Z3 solves in about 15 seconds) and it does not scale to larger problems. We therefore report only times obtained with Z3.

Running time

We compare our method with previous CP-based approach presented in [67] on the same set of benchmark instances. The runtime for the CP solver are taken from [67], and were obtained on a comparable 3.8 GHz Intel Xeon machine. In Table 7.1 we show runtime as a function of the instance size P and the number of basis patterns left unknown K' (e.g., $K' = 3$ when the instance has been simplified by fixing three out of the six unknown basis patterns).

As we can see from the runtimes reported in Table 7.1, our approach based on SMT and Z3 is always considerably faster, except for the smallest simplified problems where the difference is in the order of a few seconds. More importantly, our SMT-based approach shows a significantly improved scaling behavior, and can solve problems of realistic size with 6 unknown phases and over 40 points within an hour. In contrast, the previous CP-based approach can only solve simplified problems and cannot solve any problem with 6 unknown basis patterns [67].

Dataset		**Z3 (s)**	**ILOG Solver (s)**
P=10	K′=3	8	**0.5**
	K′=6	**12**	timeout at 1200
P=15	K′=3	13	**0.5**
	K′=6	**20**	timeout at 1200
P=18	K′=3	29	384.8
	K′=6	**125**	timeout at 1200
P=29	K′=3	**78**	276
	K′=6	**186**	timeout at 1200
P=45	K′=6	1110	timeout at 1200

Table 7.1: Runtime for analyzing X-Ray diffraction data with a Satisfiability Modulo Theory solver. P is the number of sampled points. K' is the number of unknown basis patterns. e is the number of peaks missing due to measurement errors.

Dataset	**Precision** (%)	**Recall** (%)
P=10, e=0	95.8	100
P=15, e=0	96.6	100
P=18, e=0	97.2	96.6
P=28, e=0	96.1	92.8
P=45, e=0	95.8	91.6
P=15, e=1	96.1	99.6
P=15, e=2	96.3	99.3
P=15, e=3	96.7	99.5
P=15, e=4	95.3	98.9
P=15, e=4	94.8	99.7

Table 7.2: Accuracy of the phase map (crystal structure) obtained with our automatic analysis. P is the number of sampled points. K' is the number of unknown basis patterns. e is the number of peaks missing due to measurement errors.

Solving Strategy

In order to understand whether the improvement comes from the new problem encoding (based on integer arithmetic and not on set variables as the one in [67]) or from the SMT solving strategy, we translated our arithmetic-based encoding as a Constraint Satisfaction Problem and as a Mixed Integer Program. As our SMT model combines logical constraints and linear inequalities exclusively, a Mixed Integer Programming (MIP) approach is particularly appealing. Indeed, one can fairly naturally translate the logical constraints of our model, namely 'Or', 'And', 'Not', 'IfThenElse', into a system of linear inequalities by using additional binary variables, and be left with a MIP formulation. The ability of the MIP to handle continuous variables for both the peak locations and the shifts, as well as to reason in terms of an objective function (e.g., the total number of missing peaks) makes it an attractive option. Nevertheless, the translation of the logical constraints yields a high number of binary variables (e.g., over 23K binary variables for a synthetic instance with $P = 10$), which contrasts with a low total number of continuous variables (about 120 for the same instance) and thus, weakens the potential of the MIP formulation. Empirically, none of the instances could be solved by the MIP formulation within the time limit of one hour. Similarly, we were not able to solve any of the instances (not even when simplified) obtained from translating our SMT formulation (symmetry breaking constraints included) to a CSP using the state-of-the-art IBM ILOG Cplex Solver within one hour. This suggests that the improvement over CP based solutions is not achieved thanks to the different problem encoding, but is due to the SMT solving procedure itself, which is stronger in the reasoning part and can handle well the intricate combinatorial constraints of the problem.

Accuracy

We evaluate the accuracy of our method by comparing the solutions we find (i.e., the phase map given by the values of $r_{p,k}$ for $p = 0, \cdots, P-1, k = 0, \cdots, K-1$) with the ground truth in terms of precision/recall scores, reported in Table 7.2. Precision is defined as the fraction of the number of points correctly identified as belonging to phase k (true positives), over the total number of points identified as belonging to phase k (true positives + false positives). Recall is defined as the fraction of points correctly identified as belonging to phase k (true positives) over the true number of points belonging to phase k (true positives + false negatives). These values are obtained by comparing with ground truth all $K!$ permutations of the phases we obtain, and taking the one with the smallest number of errors (recall that the problem is symmetric with respect to permutations of the phase indexes k). Further, the values in Table 7.2 are the precision/recall scores obtained for each single phase k averaged over the $K = 6$ phases. The results show that the phase maps we identify are always very accurate, with precision and recall values always larger than 90%.

Robustness

To evaluate the robustness of our method to experimental noise, we also consider another dataset from [67] where peaks are removed from the observed diffraction patterns with probability proportional to the square of the inverse peak height, in order to simulate the fact that low-intensity peaks might not be detected or they can be discarded by the peak detection algorithm. This situation is common for real-world instances, where measurements are affected by noise. We consider instances generated by removing exactly e peaks from the

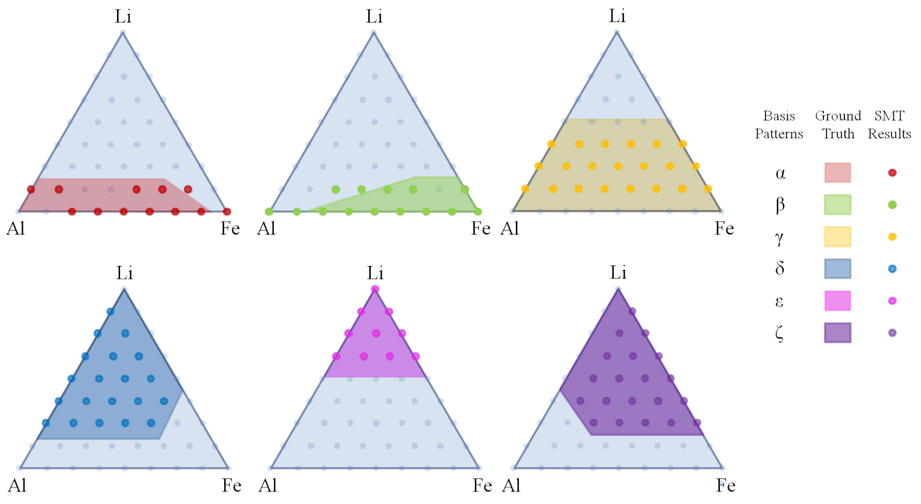


Figure 7.2: Phase map for the synthetic Al-Li-Fe system with 45 sampled points. Each of the six colored areas represents one of the basis patterns ($\alpha, \beta, ..., \zeta$) of the ground truth, while the colored dots correspond to the solution of our SMT model. The SMT results closely delimit each phase of the ground truth, which is quantitatively validated by the high precision and recall scores of our approach.

observed diffraction patterns, and we solve them by setting the upper bound T on the number of missing peaks equal to e. In figure 7.3 we see the median running time as a function of the number of missing peaks T. This is averaged over 10 instances with $P = 15$ points, and 20 runs per instance, with a timeout set at 1 hour. As shown in figure 7.3, the problem becomes significantly harder as we introduce missing peaks, because the constraint on the total number of missing peaks allowed T becomes less and less effective at pruning the search space as T grows. However, the median running time appears to increase linearly, and we are still able to recover a phase map efficiently even for instances affected by noise.

In table 7.2 we show precision recall values for these instances affected by noise. We see that the phase maps we identify are still very accurate even in presence of noise, with precision/recall scores over 95%.

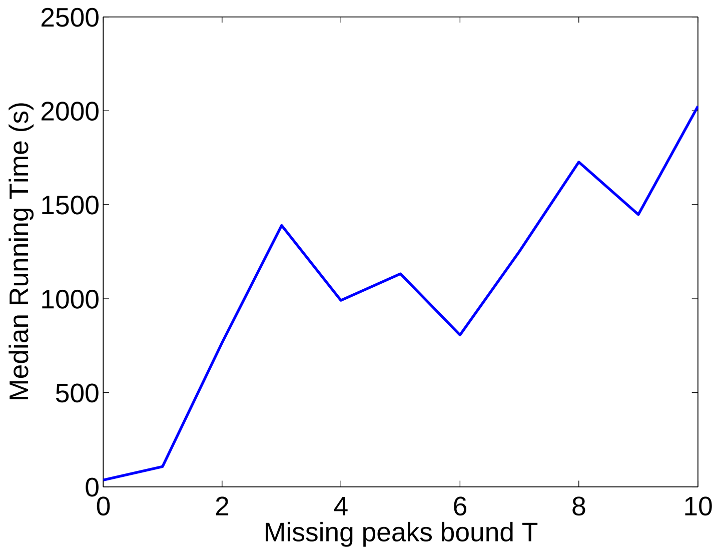


Figure 7.3: Median running time as a function of the bound on the total number of missing peaks T.

7.4.1 Sampling Phase Maps Using PAWS

In general there is not a unique solution (i.e., an assignment to all the variables, corresponding to a set of basis patterns and a phase map) to the constraint program formulated in the previous section. Given some measurement data, there are generally several competing interpretations, and we can score each candidate phase map based on the number of "missing peaks" (that is, peaks that are measured but not accounted for by the chosen basis patterns). So far we looked at the problem of finding a single phase map that achieves the the smallest number of missing peaks. However, to get a more precise idea of what is the unknown, underlying phase map, scientists would like to know if there are other competitive interpretations of the data and how they look like. In order to achieve this goal, we use the PAWS sampling algorithm described in Chapter 4 to uniformly sample from the set of phase maps achieving the the smallest number of missing peaks. This approach is very appealing because using PAWS we can leverage a state-of-the-art SMT solver such as Z3, without having to introduce any modification. At a high level, we simply need to use Z3 to solve the constraint program augmented with randomly generated parity constraints.

In Table 7.3 we report the runtime required to find an optimal solution using Z3 (an interpretation with the smallest number of missing peaks) versus the time required to sample one (approximately) uniformly at random using the PAWS scheme. The samples obtained have the theoretical guarantees proved in Section 4. We see that although sampling is computationally harder than finding a single solution, the overhead introduced by the PAWS scheme is very limited. Qualitatively, the samples we obtain for the Al-Li-Fe system are consistent with each other and they all reflect an underlying phase map close to ground truth. The differences between the samples are at the borders of the phase regions, where phases overlap and intuitively there is usually the highest uncertainty because peaks get smaller and smaller and they overlap with each other. By using a set of samples instead of a single solution we can therefore get a precise sense of where the uncertainty lies in the phase map.

Dataset		**Finding a solution (s)**	**Sampling with PAWS (s)**
P=10	K′=6	8	9
P=15	K′=6	17	19
P=18	K′=6	40	99
P=29	K′=6	184	578
P=45	K′=6	1265	2587

Table 7.3: Comparison between the time needed to find a phase map using Z3 and sampling from the model using PAWS and Z3 as a black-box. P is the number of sampled points. K' is the number of unknown basis patterns.

7.5 Conclusions

We described a novel approach to the phase map identification problem, a key step towards automatically understanding the properties of new materials created and examined using the combinatorial method. In our approach, we integrate domain-specific scientific background knowledge about the physical and chemical properties of the materials into an SMT reasoning framework. Using state-of-the-art SMT solvers, we are able to automatically analyze large synthetic datasets, generating interpretations that are physically meaningful and very accurate, even in the presence of measurement noise. Our experiments show a novel application area for SMT technology, where we can exploit its reasoning power in a hybrid setting with continuous measurement data and rather intricate combinatorial constraints. Further, we demonstrated that by using our PAWS sampling scheme we can directly leverage SMT technology to reason *probabilistically* about complex models which incorporate structured a priori background knowledge.

As there is an ever-growing amount of data in many fields of science, a key challenge is how to provide efficient methods for interpreting such data, a process that often requires integration with domain-specific scientific background knowledge. As a first step towards this goal, we demonstrated the use of automated reasoning technology to support the scientific data analysis process in materials discovery. While several aspects of our method are specific to the phase map identification problem, the data analysis approach is quite general. Given the flexibility and ever-growing reasoning power of modern day SMT solvers, we expect to see more applications to other areas of scientific exploration that require sophisticated reasoning to interpret experimental data.

CHAPTER 8

CONCLUSIONS AND RESEARCH DIRECTIONS

8.1 Summary of Contributions

Developing automated reasoning technology that will allow computer systems to interact in an intelligent way with the real world is one of the long-standing goals of AI. In this thesis we investigated computational aspects of dealing with three of the major challenges that arise, namely *high-dimensionality*, *uncertainty*, and the need to consider *utility functions*. Our research, which is partially included in this thesis, makes contributions at two levels: 1) developing foundational methods and algorithmic techniques to address these challenges and 2) demonstrating the potential for applying probabilistic reasoning and decision making technology to help solve some of the most challenging sustainability problems of our times.

The first part of the thesis focused on developing new algorithmic techniques for making inferences about complex, high-dimensional statistical models. This is a fundamental reasoning problem in AI and it is at the core of statistical machine learning methods and statistical applications in general. We introduced a new approach based on hashing and combinatorial optimization to tackle fundamental probabilistic inference problems, like computing marginal probabilities or sampling from complex, high-dimensional statistical models. Our techniques complement previous approaches such as MCMC and variational techniques, as it provides strong accuracy guarantees on the quality of the results. Further, it allows us to leverage directly fast, off-the-shelf combinatorial optimization techniques. We believe this strategy is very promising because it allows us to directly leverage and take full advantage of the latest developments in combinatorial optimization.

We investigated the hardness of the combinatorial optimization problems subject to randomly generated parity constraints that arise in our scheme, both from a theoretical and empirical perspective. We showed some deep connections with the max-likelihood decoding problem in information theory, and demonstrated how to leverage techniques and ideas originally developed in that context to make our combinatorial optimization problems more tractable in practice. Exploiting a connection with information and coding theory, we also introduced techniques to obtain provable bounds in polynomial time by considering tractable relaxations. These bounds can be further improved using a new, more tractable class of random hash functions that we introduced and can be implemented using low-density parity check codes (LDPC). These bounds are empirically much tighter (often, by several orders of magnitude) than those obtained using competing techniques.

We demonstrated the effectiveness of the new method on a variety of challenging application domains. Among several others, we looked at a scientific data analysis problem arising in materials science, where we constructed a complex statistical model to automatically analyze high-energy X-ray diffraction data and determine the underlying crystal structure. The model is particularly complex because it has to include information about the laws of physics to achieve physically meaningful interpretations of the data and high levels of accuracy. We think that our new probabilistic reasoning framework is particularly well suited to deal with models that incorporate complex, causal relationships among the variables (such as the ones imposed by the laws of physics, which rule out certain possibilities by assigning them a zero probability) because it can leverage the reasoning power of state-of-the-art constraint optimization technology such as Satisfiability Modulo Theory solvers.

8.2 Future Work

Our long-term plan is to continue developing the formal foundations of probabilistic reasoning, inference, and decision-making under uncertainty. Addressing these challenges will require a constant flow of ideas between foundational and applied research, and successful solutions will likely lead to a whole new range of computational techniques and applications. In particular, we will continue to explore applications in computational sustainability. Given the importance and relevance of sustainability issues, these new insights and technological developments will benefit not only the scientific community but also society at large. Some of the research directions we plan to explore are the following:

Probabilistic programming Probabilistic programs provide a very general and powerful probabilistic modeling framework. Unfortunately, because of the complex dependencies created by control-flow statements, probabilistic programs are also very difficult to reason about. These intricate deterministic dependencies are particularly difficult to analyze using standard MCMC and variational methods. On the other hand, combinatorial reasoning tools such as Satisfiability Modulo Theory (SAT/SMT) solvers are quite effective at reasoning about deterministic programs, and are used widely in industry to analyze code in software verification. Since we have already demonstrated the use of SMT tools in the PAWS sampling scheme for the materials discovery application, the idea of using SMT within our new probabilistic reasoning framework to make inferences about probabilistic programs appears quite promising.

Combining deductive and inductive reasoning with applications to language We believe that building the next generation of AI systems capable of dealing with complex real-world processes will require a tighter integration between *deductive* reasoning (where a conclusion is reached by reasoning from general statements) and *inductive* reasoning (where a conclusion is reached from large number of examples, as in machine learning). Specifically, we are interested in pushing forward the idea of including structured prior human knowledge into statistical models of data. Our physics-guided data-mining application in combinatorial materials discovery is a first step in this direction, but the same ideas can be applied in numerous other fields. We are particularly excited about text analysis, specifically *extracting knowledge from text*, where combining structured knowledge with large amounts of noisy data presents great opportunities and challenges [5, 13].

Combinatorial materials science Applying computer science and machine learning techniques to materials science is a largely uncharted territory with lots of opportunities, just as biology was in the early days of computational biology. We plan to continue our collaboration with materials scientists, developing more sophisticated models and better data analysis tools. For example, our current analysis of diffraction data is much faster than human interpretation (hours vs. weeks or months). However, it's not fast enough to provide on-the-fly interpretations of the data. Faster solution techniques would allow us to use active/online learning approaches to guide the exploration towards the most promising samples.

In conclusion, we believe that this long-term research program focused on *developing techniques for robust and effective data-rich reasoning and decision-making under uncertainty poses great opportunities for long-lasting contributions to the field as well as for having an impact on society at large by finding new solutions to alleviate key environmental and sustainability challenges facing our planet today*.

APPENDIX A
PROOFS

Lemma 9 (pairwise independent hash functions construction). *Let $a \in \{0,1\}^n$, $b \in \{0,1\}$. Then the family $\mathcal{H} = \{h_{a,b}(x) : \{0,1\}^n \to \{0,1\}\}$ where $h_{a,b}(x) = a \cdot x + b \mod 2$ is a family of pairwise independent hash functions. The function $h_{a,b}(x)$ can be alternatively rewritten in terms of XORs operations $\oplus$, i.e. $h_{a,b}(x) = a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n \oplus b$.*

Proof. Uniformity is clear because it is the sum of uniform Bernoulli random variables over the field $\mathbb{F}(2)$ (arithmetic modulo 2). For pairwise independence, given any two configurations $x_1, x_2 \in \{0,1\}^n$, consider the sets of indexes $S_1 = \{i : x_1(i) = 1\}$, $S_2 = \{i : x_2(i) = 1\}$. Then

$$\begin{aligned} H(x_1) = & \sum_{i \in S_1 \cap S_2} a_i \oplus \sum_{i \in S_1 \setminus S_2} a_i \oplus b \\ = & R(S_1 \cap S_2) \oplus R(S_1 \setminus S_2) \oplus b \\ H(x_2) = & R(S_1 \cap S_2) \oplus R(S_2 \setminus S_1) \oplus b \end{aligned}$$

where $R(S) \triangleq \sum_{i \in S} a_i$. Note that $R(S_1 \cap S_2)$, $R(S_1 \setminus S_2)$, $R(S_2 \setminus S_1)$ and b are independent as they depend on disjoint subsets of independent variables. When $x_1 \neq x_2$, this implies that $(H(x_1), H(x_2))$ takes each value in $\{0,1\}^2$ with probability $1/4$. □

As pairwise independent random variables are fundamental tools for derandomization of algorithms, more complicated constructions based larger finite fields generated by a prime power $\mathbb{F}(q^k)$ where q is a prime number are known [93]. These constructions require a smaller number of random bits as input, and would therefore reduce the variance of our algorithm (which is deterministic except for the randomized hash function use).

Proof of Proposition 1. Follows immediately from Lemma 9. $\square$

Proof of Lemma 1. The cases where $i + c > n$ or $i - c < 0$ are obvious. For the other cases, let's define the set of the 2^j heaviest configurations as in Definition 3:

$$\mathcal{X}_j = \{\sigma_1, \sigma_2, \cdots, \sigma_{2^j}\}$$

Define the following random variable

$$S_j(h^i_{A,b}) \triangleq \sum_{\sigma \in \mathcal{X}_j} 1_{\{A\sigma = b \bmod 2\}}$$

which gives the number of elements of $\mathcal{X}_j$ satisfying i random parity constraints. The randomness is over the choice of A and b, which are uniformly sampled in $\{0,1\}^{i \times n}$ and $\{0,1\}^i$ respectively. By Proposition 1, $h^i_{A,b} : \Sigma \to \{0,1\}^i$ is sampled from a family of pairwise independent hash functions. Therefore, from the uniformity property in Definition 2, for any σ the random variable $1_{\{A\sigma = b \bmod 2\}}$ is Bernoulli with probability $1/2^i$. By linearity of expectation,

$$E[S_j(h^i_{A,b})] = \frac{|\mathcal{X}_j|}{2^i} = \frac{2^j}{2^i}$$

Further, from the pairwise independence property in Definition 2,

$$\begin{aligned} Var[S_j(h^i_{A,b})] &= \sum_{\sigma \in \mathcal{X}_j} Var\left[1_{\{A\sigma = b \bmod 2\}}\right] \\ &= \frac{2^j}{2^i}\left(1 - \frac{1}{2^i}\right) \end{aligned}$$

Applying Chebychev Inequality, we get that for any $k > 0$,

$$\Pr\left[\left|S_j(h^i_{A,b}) - \frac{2^j}{2^i}\right| > k\sqrt{\frac{2^j}{2^i}\left(1 - \frac{1}{2^i}\right)}\right] \leq \frac{1}{k^2}$$

Recall the definition of the random variable $w_i = \max_\sigma w(\sigma)$ subject to $A\sigma = b \mod 2$ (the randomness is over the choice of A and b). Then

$$\Pr[w_i \geq b_j] = \Pr[w_i \geq w(\sigma_{2^j})] \geq \Pr[S_j(h^i_{A,b}) \geq 1]$$

which is the probability that at least one configuration from $\mathcal{X}_j$ "survives" after adding i parity constraints.

To ensure that the probability bound $1/k^2$ provided by Chebychev Inequality is smaller than a $1/2$, we need $k > \sqrt{2}$. We use $k = 3/2$ for the rest of this proof, exploiting the following simple observations which hold for $k = 3/2$ and any $c \geq 2$:

$$k\sqrt{2^c} \leq 2^c - 1$$
$$k\sqrt{2^{-c}} \leq 1 - 2^{-c}$$

For $j = i + c$ and k and c as above, we have that

$$\begin{aligned}
\Pr[w_i \geq b_{i+c}] \geq \Pr[S_{i+c}(h^i_{A,b}) \geq 1] \geq \\
\Pr\left[|S_{i+c}(h^i) - 2^c| \leq 2^c - 1\right] \geq \\
\Pr\left[|S_{i+c}(h^i) - 2^c| \leq k\sqrt{2^c}\right] \geq \\
\Pr\left[\left|S_{i+c}(h^i_{A,b}) - 2^c\right| \leq k\sqrt{2^c\left(1 - \frac{1}{2^i}\right)}\right] \geq \\
1 - \frac{1}{k^2} = 5/9 > 1/2
\end{aligned}$$

Similarly, for $j = i - c$ and k and c as above, we have $\Pr[w_i \leq b_{i-c}] \geq 5/9 > 1/2$.

Finally, using Chernoff inequality (since $w_i^1, \cdots, w_i^T$ are i.i.d. realizations of

w_i)

$$\Pr\left[M_i \leq b_{i-c}\right] \geq 1 - \exp(-\alpha'(c)T) \tag{A.1}$$

$$\Pr\left[M_i \geq b_{i+c}\right] \geq 1 - \exp(-\alpha'(c)T) \tag{A.2}$$

where $\alpha'(2) = 2(5/9 - 1/2)^2$, which gives the desired result

$$\begin{aligned}\Pr\left[b_{i+c} \leq M_i \leq b_{i-c}\right] &\geq 1 - 2\exp(\alpha'(c)T) \\ &= 1 - \exp(-\alpha^*(c)T)\end{aligned}$$

where $\alpha^*(2) = \ln 2\alpha'(2) = 2(5/9 - 1/2)^2 \ln 2 > 0.0042$ □

Proof of Lemma 2. Observe that we may rewrite L' as follows:

$$L' = b_0 + \sum_{i=n-c-1}^{n-1} b_n 2^i + \sum_{i=0}^{n-c-2} b_{i+c+1} 2^i =$$
$$b_0 + \sum_{i=n-c-1}^{n-1} b_n 2^i + \sum_{j=c+1}^{n-1} b_j 2^{j-c-1}$$

Similarly,

$$U' = b_0 + \sum_{i=0}^{c-1} b_0 2^i + \sum_{i=c}^{n-1} b_{i+1-c} 2^i =$$
$$b_0 + \sum_{i=0}^{c-1} b_0 2^i + \sum_{j=1}^{n-c} b_j 2^{j+c-1} = 2^c b_0 + 2^c \sum_{j=1}^{n-c} b_j 2^{j-1} =$$
$$2^c b_0 + 2^c \left(\sum_{j=1}^{c} b_j 2^{j-1} + \sum_{j=c+1}^{n-c} b_j 2^{j-1} \right) \leq$$
$$2^c b_0 + 2^c \left(\sum_{j=1}^{c} b_0 2^{j-1} + \sum_{j=c+1}^{n-c} b_j 2^{j-1} \right) =$$
$$2^{2c} b_0 + 2^{2c} \sum_{j=c+1}^{n-c} b_j 2^{j-1-c} \leq$$
$$2^{2c} \left(b_0 + \sum_{i=n-c-1}^{n-1} b_n 2^i + \sum_{j=c+1}^{n-1} b_j 2^{j-c-1} \right) = 2^{2c} L'$$

This finishes the proof. □

Proof of Theorem 1. It is clear from the pseudocode of Algorithm 1 that it makes $\Theta(n \ln n/\delta)$ MAP queries. For accuracy analysis, we can write W as:

$$\begin{aligned} W &\triangleq \sum_{j=1}^{2^n} w(\sigma_j) = w(\sigma_1) + \sum_{i=0}^{n-1} \sum_{\sigma \in B_i} w(\sigma) \\ &\in \left[b_0 + \sum_{i=0}^{n-1} b_{i+1} 2^i, b_0 + \sum_{i=0}^{n-1} b_i 2^i \right] \triangleq [L, U] \end{aligned}$$

Note that $U \leq 2L$ because $2L = 2b_0 + \sum_{i=0}^{n-1} b_{i+1} 2^{i+1} = 2b_0 + \sum_{\ell=1}^{n} b_\ell 2^\ell = b_0 + \sum_{\ell=0}^{n} b_\ell 2^\ell \geq U$. Hence, if we had access to the true values of all b_i, we could obtain a 2-approximation to W.

We do not know true b_i values, but Lemma 1 shows that the M_i values computed by Algorithm 1 are sufficiently close to b_i with high probability. Recall that M_i is the median of MAP values computed by adding i random parity constraints and repeating the process T times. Specifically, for $c \geq 2$, it follows from Lemma 1 that for $0 < \alpha \leq \alpha^*(c)$,

$$\begin{aligned} \Pr\left[\bigcap_{i=0}^{n} \left(M_i \in [b_{\min\{i+c,n\}}, b_{\max\{i-c,0\}}] \right) \right] \\ \geq 1 - n \exp(-\alpha T) \geq (1 - \delta) \end{aligned}$$

for $T \geq \frac{\ln(n/\delta)}{\alpha}$, and $M_0 = b_0$. Thus, with probability at least $(1-\delta)$ the output of Algorithm 1, $M_0 + \sum_{i=0}^{n-1} M_{i+1} 2^i$, lies in the range:

$$\left[b_0 + \sum_{i=0}^{n-1} b_{\min\{i+c+1,n\}} 2^i, b_0 + \sum_{i=0}^{n-1} b_{\max\{i+1-c,0\}} 2^i \right]$$

Let us denote this range $[L', U']$. By monotonicity of b_i, $L' \leq L \leq U \leq U'$. Hence, $W \in [L', U']$.

Applying Lemma 2, we have $U' \leq 2^{2c} L'$, which implies that with probability at least $1 - \delta$ the output of Algorithm 1 is a 2^{2c} approximation of W. For

$c = 2$, observing that $\alpha^*(2) \geq 0.0042$ (see proof of Lemma 1), we obtain a 16-approximation for $0 < \alpha \leq 0.0042$. $\square$

Proof of Theorem 2. As in the proof of Lemma 1, define the random variable

$$S_u(h^i_{A,b}) \triangleq \sum_{\sigma \in \{\sigma | w(\sigma) \geq u\}} 1_{\{A\sigma = b \bmod 2\}}$$

that gives the number of configurations with weight at least u satisfying i random parity constraints. Then for $i \leq \lfloor \log G(u) \rfloor - c \leq \log G(u) - c$ using Chebychev and Chernoff inequalities as in Lemma 1

$$\Pr\left[M_i \geq u\right] \geq 1 - \exp(-\alpha' T)$$

For $i \geq \lceil \log G(u) \rceil + c \geq \log G(u) + c$, using Chebychev and Chernoff inequalities as in Lemma 1

$$\Pr[M_i < u] \geq 1 - \exp(-\alpha' T)$$

Therefore,

$$\Pr\left[\frac{1}{2^{c+1}} 2^{q(u)} \leq G(u) \leq 2^{c+1} 2^{q(u)}\right] \geq$$

$$\Pr\left[\bigcap_{i=0}^{\lfloor \log_2 G(u) \rfloor - c} (M_i \geq u) \bigcap \left(M_{\lceil \log_2 G(u) \rceil + c} < u\right)\right] \geq$$

$$1 - n \exp(-\alpha' T) \geq 1 - \delta$$

This finishes the proof. $\square$

Proof of Theorem 3. If $\widetilde{w}^t_i \leq w^t_i$, from Theorem 1 with probability at least $1 - \delta$ we have $\widetilde{W} \leq M_0 + \sum_{i=0}^{n-1} M_{i+1} 2^i \leq UB'$. Since $\frac{UB'}{2^{2c}} \leq LB' \leq W \leq UB'$, it follows that with probability at least $1 - \delta$, $\frac{\widetilde{W}}{2^{2c}} \leq W$.

If $w^t_i \geq \widetilde{w}^t_i \geq \frac{1}{L} w^t_i$, then from Theorem 1 with probability at least $1 - \delta$ the output is $\frac{1}{L} LB' \leq \widetilde{W} \leq UB'$, and $LB' \leq W \leq UB'$. $\square$

Proof of Lemma 5. Let $T \leftarrow 24 \lceil \ln (n'/\delta) \rceil$ as in Algorithm 1. For $t \in \{1, \dots, T\}$, let $S_i^t = |\{(x,y) \in \mathcal{S} : h_{A,c}^i(x,y) = 0\}|$ be the number of elements of $\mathcal{S}$ that satisfy $h_{A,c}^i(x,y) = 0$, i.e., "survive" after adding i random parity constraints. The output of COMPUTEK is nothing but

$$k = \min\left\{\min\left\{i \mid \text{Median}(S_i^1, \cdots, S_i^T) < P\right\},\ n'\right\}$$

where default value n' is taken if the inner "min" is over an empty set. It follows from from pairwise independence of the chosen hash functions that:

$$\mu_i \triangleq E[S_i^t] = \frac{Z}{2^i}\ , \quad \sigma_i^2 \triangleq Var[S_i^t] = \frac{Z}{2^i}\left(1 - \frac{1}{2^i}\right)$$

For $i \leq k_P^*$, Chebychev inequality yields:

$$\mathbb{P}[S_i^t < P] \leq \mathbb{P}[|S_i^t - \mu_i| > (\mu_i - P)] \leq \frac{\sigma_i^2}{(\mu_i - P)^2} \leq \frac{\frac{Z}{2^i}}{(\frac{Z}{2^i} - P)^2}$$

The RHS is an increasing function of i, so for $i \leq k_P^* - \gamma$, which implies $Z/2^i \leq P2^\gamma$, we have $\mathbb{P}[S_i^t < P] \leq 2^\gamma/((2^\gamma - 1)^2 P) \triangleq 1 - q$. For $P \geq 2^{\gamma+2}/(2^\gamma - 1)^2$, we thus have $\mathbb{P}[S_i^t < P] \leq 1/4$ and $q \geq 3/4$. In other words, more than half the S_i^t are expected to be at least P. Using Chernoff inequality,

$$\begin{aligned} \mathbb{P}\left[\text{Median}(S_i^1, \cdots, S_i^T) \geq P\right] = &\quad 1 - \mathbb{P}\left[|\{t \mid S_i^t < P\}| < T/2\right] \\ \geq &\quad 1 - \exp\left(-\frac{1}{2q} T \left(q - \frac{1}{2}\right)^2\right) \end{aligned}$$

Similarly, for $i \geq k_P^* + \gamma$, we have $\mu_i < P$ and from Chebychev Inequality

$$\mathbb{P}[S_i^t \geq P] \leq \mathbb{P}[|S_i^t - \mu_i| \geq (P - \mu_i)] \leq \frac{\sigma_i^2}{(\mu_i - P)^2} \leq \frac{\frac{Z}{2^i}}{(\frac{Z}{2^i} - P)^2} \leq 2^\gamma/((2^\gamma - 1)^2 P) \leq \frac{1}{4}$$

Using Chernoff inequality for $i \geq k_P^* + \gamma$,

$$\mathbb{P}\left[\text{Median}(S_i^1, \cdots, S_i^T) < P\right] \geq 1 - \exp\left(-\frac{1}{2q} T \left(q - \frac{1}{2}\right)^2\right)$$

Combining we get that

$$\mathbb{P}\left[k_P^* - \gamma \leq \min\left\{i \mid \text{Median}(S_i^1, \cdots, S_i^T) < P\right\} \leq \lceil k_P^* + \gamma \rceil\right] \geq$$
$$\mathbb{P}\left[\bigcap_{i=1}^{\lfloor k_P^* - \gamma \rfloor} \left(\text{Median}(S_i^1, \cdots, S_i^T) \geq P\right) \bigcap \left(\text{Median}(S^1_{\lceil k_P^* + \gamma \rceil}, \cdots, S^T_{\lceil k_P^* + \gamma \rceil}) < P\right)\right] \geq$$
$$1 - n' \exp\left(-\frac{4}{6} T \left(\frac{3}{4} - \frac{1}{2}\right)^2\right) = 1 - n' \exp\left(-\beta T\right) \geq 1 - \delta$$

for $T \geq \frac{1}{\beta} \ln\left(n'/\delta\right)$ where $\beta = \frac{1}{24}$. It holds trivially that

$$k_P^* = \log Z - \log P \leq n' - \log P$$

so from $\lceil k_P^* + \gamma \rceil \leq 1 + k_P^* + \gamma$ we also get

$$\mathbb{P}\left[k_P^* - \gamma \leq k \leq 1 + k_P^* + \gamma\right] \geq 1 - \delta$$

This finishes the proof. $\square$

Proof of Lemma 6. It can be verified that $\gamma = \log\left((P + 2\sqrt{P+1} + 2)/P\right)$ is the be the unique positive solution to $P = 2^{\gamma+2}/(2^\gamma - 1)^2$. Therefore, γ, P satisfy the conditions of Lemma 5. Let k be the output of procedure COMPUTEK$(n', \delta, P, \mathcal{S})$. Then from Lemma 5, we have that $\mathbb{P}[k_P^* - \gamma \leq k \leq k_P^* + 1 + \gamma] \geq 1 - \delta$. All probabilities below are implicitly conditioned on this event. Let

$$S_i = |\{(x, y) \in \mathcal{S}(w, \ell, b), h^i_{A,c}(x, y) = 0\}| = |\mathcal{S}(w, \ell, b)^i| = |\mathcal{S}^i|$$

be the number of solutions surviving after adding i random parity constraints. It follows from from pairwise independence of the hash functions (Definition 2) that

$$\mu_i \triangleq E[S_i] = \frac{Z}{2^i} \ , \ \sigma_i^2 \triangleq Var[S_i] = \frac{Z}{2^i}\left(1 - \frac{1}{2^i}\right)$$

Let $\alpha \geq \gamma$ and $i = k + \alpha$. Then

$$\mu_{k+\alpha} = \frac{Z}{2^{k+\alpha}} \leq \frac{P}{2^{\alpha-\gamma}}$$

that is, on average we are left with less than P elements after adding i random parity constraints. Let $\sigma = (x, y) \in \mathcal{S}(w, \ell, b)$ be an element of the set we want to sample from. The probability $p_s(\sigma)$ that σ is output is

$$\begin{aligned} p_s(\sigma) &\triangleq \mathbb{P}\left[S_i < P, \sigma \in \mathcal{S}(w, \ell, b)^i\right] \frac{1}{P-1} \\ &= \mathbb{P}\left[S_i < P \mid \sigma \in \mathcal{S}(w, \ell, b)^i\right] \mathbb{P}\left[\sigma \in \mathcal{S}(w, \ell, b)^i\right] \frac{1}{P-1} \end{aligned}$$

where for any σ, $\mathbb{P}[\sigma \in \mathcal{S}(w, \ell, b)^i] = 2^{-i}$. Thus we have

$$p_s(\sigma) = \mathbb{P}[S_i < P \mid \sigma \in \mathcal{S}(w, \ell, b)^i] \frac{2^{-i}}{P-1} \tag{A.3}$$

Now the expected value of the size of the set (and its variance) conditioned on $\sigma \in \mathcal{S}(w, \ell, b)^i$ are independent of σ because of three-wise independence [50]. So we have

$$\begin{aligned} \mathbb{E}[S_i \mid \sigma \in \mathcal{S}(w, \ell, b)^i] &= 1 + \frac{(Z-1)}{2^i} = \mu_i(\sigma) \\ Var[S_i \mid \sigma \in \mathcal{S}(w, \ell, b)^i] &= \frac{(Z-1)}{2^i}\left(1 - \frac{1}{2^i}\right) < \mathbb{E}[S_i \mid \sigma \in \mathcal{S}(w, \ell, b)^i] \end{aligned}$$

We first note that $(Z-1)/2^i < Z/2^i = Z/2^{k+\alpha} \leq Z/2^{k_P^* - \gamma + \alpha} = P2^{\gamma - \alpha}$. Using Chebychev's inequality

$$\begin{aligned} &\mathbb{P}[S_i \geq P \mid \sigma \in \mathcal{S}(w, \ell, b)^i] \leq \mathbb{P}[|S_i - \mu_i(\sigma)| \geq (P - \mu_i(\sigma)) \mid \sigma \in \mathcal{S}(w, \ell, b)^i] \\ &\leq \frac{\frac{(Z-1)}{2^i}\left(1 - \frac{1}{2^i}\right)}{(P - (1 + \frac{(Z-1)}{2^i}))^2} \leq \frac{P2^{\gamma-\alpha}\left(1 - \frac{1}{2^i}\right)}{(P - 1 - P2^{\gamma-\alpha})^2} \leq \frac{2^{\gamma-\alpha}}{(1 - \frac{1}{P} - 2^{\gamma-\alpha})^2} \triangleq 1 - c(\alpha, P) \end{aligned}$$

Plugging into (A.3) we get

$$c(\alpha, P)\frac{2^{-i}}{P-1} = \left(1 - \frac{2^{\gamma-\alpha}}{(1 - \frac{1}{P} - 2^{\gamma-\alpha})^2}\right)\frac{2^{-i}}{P-1} \leq p_s(\sigma) \leq \frac{2^{-i}}{P-1} \tag{A.4}$$

where $c(\alpha, P) \to 1$ as $\alpha \to \infty$. This shows that the sampling probabilities $p_s(\sigma), p_s(\sigma')$ of σ, σ' must be within a constant factor $c(\alpha, P)$ of each other.

From $k \leq k_P^* + 1 + \gamma$ it follows that

$$p_s(\sigma) \geq c(\alpha, P) \frac{2^{-(1+\gamma+\alpha)}}{Z} \frac{P}{P-1}$$

This shows that the probability that the algorithm does not output $\perp$ is at least

$$\mathbb{P}[\text{output} \neq \perp] = Q = \sum_{\sigma \in \mathcal{S}(w,\ell,b)} p_s(\sigma) \geq c(\alpha, P) 2^{-(1+\gamma+\alpha)} \frac{P}{P-1}$$

The probability $p_s'(\sigma)$ that σ is sampled given that the algorithm does not output $\perp$ is

$$\begin{aligned} \frac{\mathbb{P}\left[S_i < P, \sigma \in \mathcal{S}(w,\ell,b)^i, \text{output} \neq \perp\right]}{Q} &= \frac{\mathbb{P}\left[S_i < P, \sigma \in \mathcal{S}(w,\ell,b)^i\right]}{Q} \\ &= \frac{p_s(\sigma)}{Q} = p_s'(\sigma) \end{aligned}$$

Plugging in (A.4)

$$c(\alpha, P) \frac{2^{-i}}{P-1} \frac{1}{Q} \leq p_s'(\sigma) \leq \frac{2^{-i}}{P-1} \frac{1}{Q}$$

From $\sum_\sigma p_s'(\sigma) = 1$ we get

$$c(\alpha, P) \frac{2^{-i}}{P-1} \frac{1}{Q} Z \leq 1 \leq \frac{2^{-i}}{P-1} \frac{1}{Q} Z$$

which implies

$$c(\alpha, P) \frac{1}{Z} \leq c(\alpha, P) \frac{2^{-i}}{P-1} \frac{1}{Q} \leq p_s'(\sigma) \leq \frac{2^{-i}}{P-1} \frac{1}{Q} \leq \frac{1}{c(\alpha, P)} \frac{1}{Z}$$

This finishes the proof. □

Proof of Corollary 1. Suppose we want to compute an expectation of $\phi : \{0,1\}^n \to \mathbb{R}$

$$\mathbb{E}_p[\phi] = \sum_{x \in \{0,1\}^n} p(x)\phi(x) = \sum_{x \in \{0,1\}^n \setminus \mathcal{B}_\ell} p(x)\phi(x) + \sum_{x \in \mathcal{B}_\ell} p(x)\phi(x)$$

$$\sum_{x \in \{0,1\}^n \setminus \mathcal{B}_\ell} p(x)\phi(x) - \epsilon\eta_\phi \leq \mathbb{E}_p[\phi] \leq \sum_{x \in \{0,1\}^n \setminus \mathcal{B}_\ell} p(x)\phi(x) + \epsilon\eta_\phi$$

From Theorem 4

$$\sum_{x\in\{0,1\}^n\setminus\mathcal{B}_\ell} \frac{1}{\rho\kappa} p(x)\phi(x) \le \mathbb{E}_{p'_s}[\phi] = \sum_{x\in\{0,1\}^n\setminus\mathcal{B}_\ell} p'_s(x)\phi(x) \le \sum_{x\in\{0,1\}^n\setminus\mathcal{B}_\ell} \rho\kappa p(x)\phi(x)$$

It follows that

$$\frac{1}{\rho\kappa}\mathbb{E}_{p'_s}[\phi] - \epsilon\eta_\phi \le \mathbb{E}_p[\phi] \le \rho\kappa\mathbb{E}_{p'_s}[\phi] + \epsilon\eta_\phi$$

as desired. □

Proof of Proposition 4. Let $i^* = \log_2 |S|$ and $2^{\hat{i}-1}$ be the output of $\mathcal{A}$. We will show that $\hat{i}$ is within $[\lfloor i^*\rfloor, \lfloor i^*\rfloor + 2]$ with probability at least $3/4$.

Fix any $i \le i^*$. Then $\mathbb{E}[|h_t^{-1}(0) \cap S|] = |S|/2^i \ge 1$. Weak $(\mu^2, 4)$-concentration implies that $\Pr[|h_t^{-1}(0) \cap S| = 0] \le 1/4$. Chernoff bound applied to the T underlying independent 0-1 indicator random variables then implies that a majority of the T sets will be empty with probability at most $\exp(-T/8)$. It follows that with probability at least $(1-\exp(-T/8))^{i^*} \ge (1-\exp(-T/8))^n \ge 1-n\exp(-T/8)$, the majority of the T sets for *all* $i \le i^*$ will simultaneously be non-empty. Thus, for $T \ge 8\ln(8n)$, we have that with probability at least 7/8, all $i \le i^*$ will behave correctly.

Fix any $i \ge i^* + 2$. Here we can simply use Markov's inequality to infer that $\Pr[|h_t^{-1}(0) \cap S| \ge 1] \le 1/4$. From the same Chernoff bound based argument as above, it follows that for $T \ge 8\ln(8n)$, with probability at least 7/8, all $i \ge i^* + 2$ will behave correctly.

By union bound, it follows that the output $2^{\hat{i}-1}$ of $\mathcal{A}$ will be in the range $[2^{\lfloor i^*\rfloor - 1}, 2^{\lfloor i^*\rfloor + 1}]$ with probability at least $1 - 1/8 - 1/8 = 3/4$. □

Proof of Proposition 5. From Chebychev's inequality, $\Pr[|X - \mu| \ge \sqrt{\delta}\sigma] \le \delta$, which implies the claimed strong correlation. For showing the desired weak correlation, we use Cantelli's one-sided inequalities. For the first case, $\Pr[X \leq \mu - \sqrt{\delta - 1}\sigma] \leq 1/(1 + (\delta - 1)) = 1/\delta$. The second case works similarly. $\square$

Proof of Proposition 6. From Chernoff's bound, $\Pr[X \leq \mu + \sqrt{k}] = \Pr[X \leq (1 + \frac{\sqrt{k}}{\mu})\mu] \leq \exp(-\frac{k}{3\mu})$. Thus, $k \geq (3 \ln \delta)\mu$ suffices to bound this probability by $1/\delta$. The other side, $\Pr[X \leq \mu - \sqrt{k}]$, similarly leads to $k \geq (2 \ln \delta)\mu$ as the condition to bound the probability by $1/\delta$. Combining the two, we get the desired result for weak concentration. The result for strong concentration follows by using the union bound to obtain $\exp(-\frac{k}{3\mu}) + \exp(-\frac{k}{2\mu})$, which is less than $\exp(-\frac{k}{c\mu})$ for any $c > 3$. $\square$

Proof of Proposition 7. This follows from observing that pairwise independence implies $\sigma^2 = |S|/2^m(1 - 1/2^m) < \mu$ and then applying Prop. 5. $\square$

Proof of Proposition 8. The first two observations are straightforward. For the third, let S and T be sets with $|T| = |S| + 1$. Given $y_1, y_2 \in \{0,1\}^m$, let $f(x_1, x_2)$ denote $P[H(x_1) = y_1, H(x_2) = y_2]$. Then

$$\begin{aligned}
\sum_{x,y \in T, x \neq y} f(x,y) = \frac{1}{|T| - 2} \sum_{z \in T} \sum_{x,y \in T \setminus \{z\}, x \neq y} f(x,y) \\
\leq \frac{1}{|T| - 2} \sum_{z \in T} |S|(|S| - 1)\frac{\epsilon}{2^m} \\
= \frac{|T|}{|T| - 2}|S|(|S| - 1)\frac{\epsilon}{2^m} \leq |T|(|T| - 1)\frac{\epsilon}{2^m}
\end{aligned}$$

This finishes the proof. $\square$

Proof of Lemma 1. By Theorem 8, the hash functions $h^i_{A,b}$ from $\mathcal{H}^{f*}$ in the inner loop at iteration i are $(\epsilon, 2^{i+2})$-AU, with $\epsilon < \frac{31}{5(2^{i+2}-1)}$ by construction.

Let $S = \{\sigma_1, \sigma_2, \cdots, \sigma_{2^{i+2}}\}$, $X = |\left(h^i_{A,b}\right)^{-1}(\mathbf{0}) \cap S|$. Notice $|S| = 2^{i+2}$ and $\mathbb{E}[X] = 2^{i+2}/2^i = 4$.

By by Corollary 2 and Theorem 7, X is weakly $(\mu^2, 9/4)$-concentrated.

Then by weak concentration

$$\Pr[w_i \geq b_{i+2}] = \Pr[w_i \geq w(\sigma_{2^{i+2}})] \geq \Pr[X \geq 1]$$
$$= 1 - \Pr[X \leq 0] \geq 1 - 4/9 = 5/9$$

Similarly, we have from Markov's inequality

$\Pr[w_i \leq b_{i-2}] \geq 3/4 \geq 5/9$.

Finally, using Chernoff inequality (since $w_i^1, \cdots, w_i^T$ are i.i.d. realizations of w_i)

$$\Pr\left[M_i \leq b_{i-2}\right] \geq 1 - \exp(-\alpha' T) \tag{A.5}$$

$$\Pr\left[M_i \geq b_{i+2}\right] \geq 1 - \exp(-\alpha' T) \tag{A.6}$$

where $\alpha' = 2(5/9 - 1/2)^2$, which gives the desired result

$$\Pr\left[b_{i+2} \leq M_i \leq b_{i-2}\right] \geq 1 - 2\exp(\alpha' T)$$
$$= 1 - \exp(-\alpha^* T)$$

where $\alpha^* = \ln 2\alpha' = 2(5/9 - 1/2)^2 \ln 2 > 0.0042$ $\square$

BIBLIOGRAPHY

[1] D. Allouche, S. de Givry, and T. Schiex. Toulbar2, an open source exact cost function network solver. Technical report, INRIA, 2010.

[2] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[3] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

[4] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. In *Proc. of the 34th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 724–733, 1993.

[5] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction for the web. In *Proc. of the 20th International Joint Conference on Artifical Intelligence (IJCAI)*, pages 2670–2676, 2007.

[6] C. Barrett, A. Stump, and C. Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`, 2010.

[7] R. J. Bayardo and J. D. Pehoushek. Counting models using connected components. In *Proc. of the 17th National Conference on Artifical Intelligence (AAAI)*, pages 157–162, 2000.

[8] M. Bellare, O. Goldreich, and E. Petrank. Uniform generation of NP-witnesses using an NP-oracle. *Information and Computation*, 163(2):510–526, 2000.

[9] R. Bellman. *Adaptive control processes: A guided tour*. Princeton University Press, Princeton, NJ, 1961.

[10] E. Berlekamp, R. McEliece, and H. Van Tilborg. On the inherent intractability of certain coding problems. *Information Theory, IEEE Transactions on*, 24(3):384–386, 1978.

[11] D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific Belmont, MA, 1997.

[12] C. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[14] R. Bryant, R. H. Katz, and E. D. Lazowska. Big-data computing: Creating revolutionary breakthroughs in commerce, science and society. Technical report, CRA, 2008.

[15] J. Cai and X. Chen. A decidable dichotomy theorem on directed graph homomorphisms with non-negative weights. In *Proc. of the 51st Symposium on Foundations of Computer Science (FOCS)*, 2010.

[16] M. Carreira-Perpinan and G. Hinton. On contrastive divergence learning. In *Proc. of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, page 17, 2005.

[17] S. Chakraborty, K. Meel, and M. Vardi. A scalable and nearly uniform generator of SAT witnesses. In *Proc. of the 25th International Conference on Computer Aided Verification (CAV)*, 2013.

[18] S. Chakraborty, K. Meel, and M. Vardi. A scalable approximate model counter. In *Proc. of the 19th International Conference on Principles and Practice of Constraint Programming (CP)*, pages 200–216, 2013.

[19] A. Darwiche. *Modeling and reasoning with Bayesian networks.* Cambridge University Press, 2009.

[20] L. De Moura and N. Bjørner. Z3: An efficient SMT solver. In *Proc. of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 337–340. Springer, 2008.

[21] R. Dechter. Reasoning with probabilistic and deterministic graphical models: Exact algorithms. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 7(3):1–191, 2013.

[22] G. Desjardins, A. Courville, and Y. Bengio. On tracking the partition function. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2501–2509, 2011.

[23] M. Dyer, A. Frieze, and R. Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *Journal of the ACM*, 38(1): 1–17, 1991.

[24] S. Ermon, J. Conrad, C. P. Gomes, and B. Selman. Playing games against nature: optimal policies for renewable resource allocation. In *Proc. of the 26th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010.

[25] S. Ermon, C. P. Gomes, and B. Selman. Collaborative multiagent gaussian inference in a dynamic environment using belief propagation. In *Proc. of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1419–1420, 2010.

[26] S. Ermon, J. Conrad, C. P. Gomes, and B. Selman. Risk-sensitive policies for sustainable renewable resource allocation. In *Proc. of the 22nd Inter-*

national Joint Conference on Artificial Intelligence (IJCAI), pages 1942–1948, 2011.

[27] S. Ermon, C. P. Gomes, A. Sabharwal, and B. Selman. Accelerated Adaptive Markov Chain for Partition Function Computation. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[28] S. Ermon, C. P. Gomes, and B. Selman. A message passing approach to multiagent gaussian inference for dynamic processes. In *Proc. of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1277–1278, 2011.

[29] S. Ermon, C. P. Gomes, and B. Selman. Uniform solution sampling using a constraint solver as an oracle. In *Proc. of the 28th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 255–264, 2012.

[30] S. Ermon, C. P. Gomes, B. Selman, and A. Vladimirsky. Probabilistic planning with non-linear utility functions and worst-case guarantees. In *Proc. of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 965–972, 2012.

[31] S. Ermon, R. Le Bras, C. P. Gomes, B. Selman, and R. B. van Dover. SMT-aided combinatorial materials discovery. In *Theory and Applications of Satisfiability Testing (SAT)*, pages 172–185, 2012.

[32] S. Ermon, Y. Xue, C. P. Gomes, and B. Selman. Learning policies for battery usage optimization in electric vehicles. In *Proc. of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, pages 195–210, 2012.

[33] S. Ermon, C. P. Gomes, A. Sabharwal, and B. Selman. Embed and project: Discrete sampling with universal hashing. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2085–2093, 2013.

[34] S. Ermon, C. P. Gomes, A. Sabharwal, and B. Selman. Optimization with parity constraints: From binary codes to discrete integration. In *Proc. of the 29th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2013.

[35] S. Ermon, C. P. Gomes, A. Sabharwal, and B. Selman. Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *Proc. of the 30th International Conference on Machine Learning (ICML)*, 2013.

[36] S. Ermon, Y. Xue, C. P. Gomes, and B. Selman. Learning policies for battery usage optimization in electric vehicles. *Machine learning*, 92(1):177–194, 2013.

[37] S. Ermon, C. P. Gomes, A. Sabharwal, and B. Selman. Low-density parity constraints for hashing-based discrete integration. In *Proc. of the 31st International Conference on Machine Learning (ICML)*, pages 271–279, 2014.

[38] J. Feldman, M. J. Wainwright, and D. R. Karger. Using linear programming to decode binary linear codes. *Information Theory, IEEE Transactions on*, 51(3):954–972, 2005.

[39] B. Felgenhauer and F. Jarvis. Enumerating possible sudoku grids. *Mathematical Spectrum*, 2005.

[40] R. Gallager. Low-density parity-check codes. *Information Theory, IRE Transactions on*, 8(1):21–28, 1962.

[41] D. Ginley, C. Teplin, M. Taylor, M. van Hest, and J. Perkins. *Combinatorial materials science*. McGraw-Hill Companies, 2005.

[42] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[43] P. Godefroid, M. Y. Levin, D. Molnar, et al. Automated whitebox fuzz testing. In *Proc. of the 16th Network and Distributed System Security Symposium*, 2008.

[44] P. Godefroid, M. Y. Levin, and D. Molnar. Sage: Whitebox fuzzing for security testing. *Queue*, 10(1):20:20–20:27, Jan. 2012. ISSN 1542-7730.

[45] V. Gogate and R. Dechter. SampleSearch: A scheme that searches for consistent samples. In *Proc. 10th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

[46] V. Gogate and R. Dechter. SampleSearch: Importance sampling in presence of determinism. *Artificial Intelligence*, 175(2):694–729, 2011.

[47] V. Gogate and P. Domingos. Approximation by quantization. In *Proc. of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 247–255, 2011.

[48] O. Goldreich. Randomized methods in computation. *Lecture Notes*, 2011.

[49] C. P. Gomes. Computational sustainability: Computational methods for a sustainable environment, economy, and society. *The Bridge*, 39(4):5–13, 2009.

[50] C. P. Gomes, A. Sabharwal, and B. Selman. Near-uniform sampling of combinatorial spaces using XOR constraints. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

[51] C. P. Gomes, A. Sabharwal, and B. Selman. Model counting: A new strategy for obtaining good bounds. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 54–61, 2006.

[52] C. P. Gomes, J. Hoffmann, A. Sabharwal, and B. Selman. Short XORs for model counting: From theory to practice. In *Theory and Applications of Satisfiability Testing (SAT)*, pages 100–106, 2007.

[53] C. P. Gomes, W. J. van Hoeve, A. Sabharwal, and B. Selman. Counting CSP solutions using generalized XOR constraints. In *Proc. of the 22nd National Conference on Artificial Intelligence (AAAI)*, 2007.

[54] N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum. Church: A language for generative models. In *Proc. of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 220–229, 2008.

[55] J. M. Gregoire, M. E. Tague, S. Cahen, S. Khan, H. D. Abruna, F. J. DiSalvo, and R. B. van Dover. Improved fuel cell oxidation catalysis in Pt1-xTax. *Chemistry of Materials*, 22(3):1080, 2010.

[56] A. Griggio. A Practical Approach to Satisfiability Modulo Linear Integer Arithmetic. *Journal on Satisfiability, Boolean Modeling and Computation*, 8: 1–27, 2012.

[57] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[58] T. Hazan and T. Jaakkola. On the partition function and random maximum a-posteriori perturbations. In *Proc. of the 29th International Conference on Machine Learning (ICML)*, 2012.

[59] G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[60] IBM ILOG. IBM ILOG CPLEX Optimization Studio 12.3, 2011.

[61] R. Jeroslow. On defining sets of vertices of the hypercube by linear inequalities. *Discrete Mathematics*, 11(2):119–124, 1975.

[62] M. Jerrum and A. Sinclair. The markov chain monte carlo method: An approach to approximate counting and integration. In *Approximation Algorithms for NP-hard Problems*, pages 482–520. PWS Publishing, Boston, MA, 1997.

[63] M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[64] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT Press, 2009.

[65] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2): 498–519, 2001.

[66] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–224, 1988.

[67] R. Le Bras, T. Damoulas, J. M. Gregoire, A. Sabharwal, C. P. Gomes, and R. B. van Dover. Constraint reasoning and kernel clustering for pattern

decomposition with scaling. In *Proc. of the 17th International Conference on Principles and Practice of Constraint Programming (CP)*, 2011.

[68] C. J. Long, J. Hattrick-Simpers, M. Murakami, R. C. Srivastava, I. Takeuchi, V. L. Karen, and X. Li. Rapid structural mapping of ternary metallic alloy systems using the combinatorial approach and cluster analysis. *Review of Scientific Instruments*, 78, 2007.

[69] C. J. Long, D. Bunker, V. L. Karen, X. Li, and I. Takeuchi. Rapid identification of structural phases in combinatorial thin-film libraries using x-ray diffraction and non-negative matrix factorization. *Review of Scientific Instruments*, 80, 2009.

[70] D. J. MacKay. Good error-correcting codes based on very sparse matrices. *Information Theory, IEEE Transactions on*, 45(2):399–431, 1999.

[71] N. Madras. *Lectures on Monte Carlo Methods*. American Mathematical Society, 2002.

[72] B. Milch, B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov. BlOG: Probabilistic models with unknown objects. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1352–1359, 2005.

[73] J. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173, 2010.

[74] K. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Proc. of the 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1999.

[75] B. Narasimhan, S. Mallapragada, and M. Porter. *Combinatorial materials science*. John Wiley and Sons, 2007.

[76] Y. Naveh, M. Rimon, I. Jaeger, Y. Katz, M. Vinov, E. Marcu, and G. Shurek. Constraint-based random stimuli generation for hardware verification. *AI Magazine*, 28(3):13, 2007.

[77] R. M. Neal. Slice sampling. *Annals of statistics*, pages 705–741, 2003.

[78] J. Park. Using weighted MAX-SAT engines to solve MPE. In *Proc. of the 18th National Conference on Artificial Intelligence (AAAI)*, pages 682–687, 2002.

[79] K. B. Petersen and M. S. Pedersen. The matrix cookbook. Technical report, Technical University of Denmark, 2008.

[80] P. Ravikumar and J. Lafferty. Quadratic programming relaxations for metric labeling and Markov random field MAP estimation. In *Proc. of the 23rd International Conference on Machine Learning (ICML)*, pages 737–744, 2006.

[81] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1):107–136, 2006.

[82] S. Riedel. Improving the accuracy and efficiency of MAP inference for Markov Logic. In *Proc. of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 468–475, 2008.

[83] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1):273–302, 1996.

[84] M. Simonovits. How to compute the volume in high dimension? *Mathematical programming*, 97(1):337–374, 2003.

[85] M. Sipser. A complexity theoretic approach to randomness. In *Proc. of the 15th ACM Symposium on Theory of Computing (STOC)*, pages 330–335, 1983.

[86] D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *Proc. of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 503–510, 2008.

[87] D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *Proc. of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2008.

[88] M. Soos, K. Nohl, and C. Castelluccia. Extending SAT solvers to cryptographic problems. In *Theory and Applications of Satisfiability Testing (SAT)*, 2009.

[89] J. Stern. Approximating the number of error locations within a constant ratio is NP-complete. In *Proc. of the 10th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 325–331, 1993.

[90] D. R. Stinson. On the connections between universal hashing, combinatorial designs and error-correcting codes. *Congressus Numerantium*, pages 7–28, 1996.

[91] L. Stockmeyer. On approximation algorithms for #P. *SIAM Journal on Computing*, 14(4):849–861, 1985.

[92] G. Strang. *Introduction to linear algebra*. Wellesley Cambridge Press, 2009.

[93] S. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 2011.

[94] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[95] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[96] R. B. Van Dover, L. Schneemeyer, and R. Fleming. Discovery of a useful thin-film dielectric using a composition-spread approach. *Nature*, 392 (6672):162–164, 1998.

[97] A. Vardy. Algorithmic complexity in coding theory and the minimum distance problem. In *Proc. of the 29th ACM Symposium on the Theory of Computing (STOC)*, 1997.

[98] M. J. Wainwright. Tree-reweighted belief propagation algorithms and approximate ML estimation via pseudo-moment matching. In *Proc. of the 8th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2003.

[99] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[100] L. Wang, S. Ermon, and J. E. Hopcroft. Feature-enhanced probabilistic models for diffusion network inference. In *Proc. of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, pages 499–514, 2012.

[101] W. Wei and B. Selman. A new approach to model counting. In *Theory and Applications of Satisfiability Testing (SAT)*, pages 324–339, 2005.

[102] M. Welling and G. Hinton. A new learning algorithm for mean field Boltzmann machines. In *Proc. of the 12th International Conference on Artificial Neural Networks (ICANN)*, 2002.

[103] M. Yannakakis. Expressing combinatorial optimization problems by linear programs. *Journal of Computer and System Sciences*, 43(3):441–466, 1991.

[104] C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation – an empirical study. *Journal of Machine Learning Research*, 7:1887–1907, 2006.