Graphical Models Beyond Standard Settings: Lifted Decimation, Labeling, and Counting

Dissertation

zur Erlangung des Grades eines

D o k t o r s d e r N a t u r w i s s e n s c h a f t e n

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Fabian Hadiji

Dortmund

2015

Dissertation an der Fakultät für Informatik, TU Dortmund
Tag der mündlichen Prüfung: 26.10.2015
Dekan: Prof. Dr.-Ing. Gernot A. Fink

Gutachter:
Prof. Dr. Kristian Kersting
Prof. Sriraam Natarajan, PhD

Acknowledgments

I am thankful for the experiences that I was able to make and I am very grateful for meeting so many interesting and smart people during the time working on my PhD. I am very happy that the scientific exchange, as well as collaborations, with various people hold beyond the course of working on this thesis. Among all the people that I have met, I want to thank a few in particular.

First, I want to thank my supervisor Kristian Kersting for giving me the opportunity to start my PhD in the STREAM group at Fraunhofer IAIS. Besides being an inspiring supervisor with great advices, Kristian also brought a team of excellent researchers and great personalities together. It has been a great experience to work with the other members of the STREAM group and office mates. Kristan has always been very supportive and it was a pleasure to work with him on various papers and projects. On top of that, he is a person that is fun to work with.

None of the content in this thesis would have been possible without all the co-authors involved in working on the papers and writing the papers themselves. I have worked closely together with Babak in particular on various papers. The discussion with him were very resourceful and his vital contributions improved my own work greatly. Besides his contributions to research papers, I also want to thank Sriraam for his support and feedback during the writing of this thesis. A big thank you also belongs to Alejandro, Anders, Christian B., Christian T., Martin, Rafet, and Youness. These people contributed in unique ways to all of the published papers by bringing in their thoughts, ideas, and point of views.

The same holds for other colleagues with whom I have — unfortunately — not co-authored any papers: Mirwaes, Xiao, and Marion. Nevertheless, I have learned a lot from them. This also includes playing table football and I would certainly not underestimate the influence on research of winning matches in FIFA.

I also want to explicitly thank Christian T. and Anders. By hiring me at GameAnalytics, Christian widened my research interested to include analyzing user behavior in games and beyond. I met Anders through GameAnalytics which also resulted in fruitful collaborations up to this day. Both are still very helpful and I continue to enjoy collaborating with them in various areas.

I also want thank all institutes involved while working on my thesis which include the Fraunhofer IAIS, the University of Bonn, and the TU Dortmund University. Here, I also want to mention the University of Massachusetts Amherst. The research visit to Andrew McCallum's research group has been another outstanding experience. Working together with such NLP experts, and in particular together with Sebastian Riedel, has been a pleasure.

There is another former colleague with whom I have not written any papers but received more support and help from than anybody else. Besides that, we share memories that go beyond research and work. Thank you for everything, Anja.

Besides all people I have met in past years at work, there is also a number of important people outside of academia that have supported me all my life. I want to thank my parents, family, and friends for their support and making life as great as it has been. I would certainly not write these lines without their continuous help and love.

Abstract

The influence of Artificial Intelligence (AI) and Machine Learning on our everyday lives has been growing constantly over the past decades. Applications in both areas have attracted much attention and several of them depend on Probabilistic Graphical Models (PGMs). Furthermore, the emerging interest in methods originating from the Statistical Relational Learning (SRL) community has added to this underlying trend over the past ten years. In many cases, the combination of logic and probability is superior to relying on only either of the two. However, with problems becoming larger and more complex, inference and learning are still major issues in PGMs and in particular in the context of models derived from SRL formalisms. On the other hand, many problems in SRL are specified in such a way that symmetries arise from the underlying model structure. Exploiting these symmetries during inference, which is referred to as "lifted inference", has lead to significant efficiency gains.

The idea of lifted inference is not solely applicable to relational probabilistic models though. We provide several enhanced versions of known algorithms that show to be liftable too and thereby apply lifting in "non-standard" settings. For example, we show how to use lifted message passing in a decimation procedure for solving satisfying problems. By doing so, we extend the understanding of the applicability of lifted inference and lifting in general.

The majority of research on lifted inference has so far been concentrating on PGMs with binary random variables. Certainly, these models are not the universal remedy to all problems in AI and in fact represent only a small subset of all PGMs. In many cases, we wish to model variables over a large domain, or even infinite range, even if this introduces new challenges to the complexity. Often, common formalisms become unsuitable and even efficient approximative inference is intractable. However, we can still transfer ideas from SRL and lifting to other approaches that are more suitable in this context. In this regard, we show that algorithms solving multiclass labeling problems can benefit from SRL-like weighted rules and lifted inference — which are both non-standard techniques in this area. Here, weighted rules induce sparse and symmetric problems from which Label Propagation, an effective labeling algorithm, can benefit.

We use our novel Label Propagation approach in combination with an innovative Web-based data harvesting pipeline to label author-paper-pairs with geographic information in online bibliographies. This results is a large-scale transnational bibliography containing affiliation information over time for roughly one million authors. By counting the number of migration hops of researchers, we can infer mobility patterns in migration from this database. We find distributions fitting the data well which have also been found describing human and social phenomena in other areas. In this line of research, we also show that the Web provides a rich set of additional data sources where similar patterns can be found in the spread of information and content on the Web. These findings share one commonality: all datasets were build upon counting.

Although counting is done literally everywhere, mainstream PGMs have widely been neglecting count data. In the case where the ranges of the random variables are defined over the natural numbers, crude approximations to the true distribution are often made by discretization or a Gaussian assumption. To handle count data with positive and negative dependencies, we introduce Poisson Dependency Networks (PDNs). PDNs are defined over a set of local probability distributions and we obtain a joint distribution via sampling, allowing us to answer various probabilistic queries. Additionally, PDNs do not only allow to describe dependencies between count random variables but also allow to make predictions for unseen instances. Hence, we provide a new class of non-standard PGMs naturally handling count data.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

Providing machines with some kind of intelligence and enabling them to operate in a smart manner has been a longstanding goal of mankind and the field of *Artificial Intelligence (AI)*. We can clearly see that AI and also Machine Learning are affecting our everyday lives more and more, and their ample capabilities have just begun to have an impact. Until today, robotics has been widely recognized as the driving force behind AI and Machine Learning but there is a much wider range of applications that can and will benefit from these techniques: for instance those within the predominant era of *Big Data*. In this context, not only areas such as medicine and biology, but also whole branches of computer science like computer vision, natural language processing, and information retrieval, to name only a few, have been revolutionized by AI. This again has been facilitated by today's mass storage capabilities which make storage fast and cheap. Besides these areas of research, we are starting to see many applications throughout the industry enjoying the great advantages of Machine Learning. Both, eCommerce and mobile commerce for instance, are increasingly depending on Machine Learning. Recommender systems, as a common and wide-spread example, help to objectify advertisement in these branches to leverage the effectiveness of newsletters and banners. The foundation of detecting latent patterns — for example in data of human behavior — in any of the aforementioned sciences and applications would not have been possible without the help of statistical learning.

Although a lot of progress has been made over the past decades, much remains to be done in the future if we are to reach the goals we all imagine — namely those of applying intelligent algorithms into a real-world context. Due to the low costs of data storage and computational power, many companies have started gathering tons of data. They are, however, unable to discover latent knowledge and information due to the missing expertise, which also prevents them from generating revenue from the available data. There seems to be an agreement that large amounts of data are a crucial key to AI. However, it still remains to be answered if Big Data is really the skeleton key to all AI problems. This is where Machine Learning is starting to play a fundamental role by learning descriptive models from data and to use the derived models to make predictions on unseen instances. Nevertheless, Big Data also poses new challenges for Machine Learning in terms of complexity. In many of these cases, we do not see a single instance in isolation but instead a set of objects all interacting with each other.

Often, several of these objects are of the same type, such as customers in databases throughout the retail industry. In these cases, we are interested in predicting characteristics of customers: for example their churn or buying behavior [157, 85, 204].

One crucial effect that is often present in these kind of situations is *autocorrelation* [102] which is the statistical dependence between such interacting entities or objects of the same type. Friends for example may share similar interests. If one person buys a specific item, it could be worthwhile recommending this item to their friends. Here, friendship as such implies a relational structure between objects, or database entries, and thereby introduces dependencies. This holds in many other situations and the data we observe on an everyday basis is inherently relational, making the common assumption in Machine Learning of identically and independently distributed instances not suitable. Instead, this type of data asks for collective classification and regression. Jensen et al. [103] discuss this issue in particular for relational probabilistic models and conclude that collective approaches can greatly reduce the error in prediction with comparable results to non-relational approaches in the absence of autocorrelation.

Modeling links, such as the aforementioned example of friendship, is a common pattern in relational databases which belong to the most widespread form of data storage representing structured information. The field of Inductive Logic Programming [44] is concerned with learning logical rules from such databases that hold throughout the entire knowledge base. It must be noted however that real-word data often underlies uncertainty which additionally makes modeling and learning far more complex. For example, intelligent customer relationship management and marketing databases wish to store uncertain information on customers such as churn and buyer conversion likelihoods. And again, we can assume that the decisions made by customers related to each other are not independent. Models describing relational data under uncertainty have become more popular than ever and this trend is also visible in the increasing usage of probabilistic databases [216]. Early work in this area already combined relational query languages with probabilistic models [221], allowing developers familiar with SQL to specify complex probabilistic models. However, accessible and fast inference in these problem settings is crucial for the applicability of such modern AI and Machine Learning approaches. A broader application of such technologies will hopefully not only facilitate further advances in AI and computer science in general, but also trigger new data storages that are capable of handling Big Data under uncertainty.

There is a wide a range of approaches and algorithms capable of learning from observations which also provide extensions that carry over to the relational or multivariate setting. One particular class of models that can learn from different types of data and represent general interactions between observations, are *Probabilistic Graphical Models (PGMs)*. PGMs model the probability distribution of the observed data and allow to collectively infer missing values from the underlying joint distribution. The impact of PGMs on AI has been huge and has also been valued by the ACM recently by awarding Judea Pearl the Turing Award in 2011. Pearl's work on Bayesian Networks, a subset of PGMs, and Belief Propagation, which is an inference algorithm within these models, commenced a new era within probabilistic reasoning. A significant part of the work at hand is also based on Belief Propagation and the idea of message passing for inference.

Large-scale databases with millions of customers comprise difficult challenges for learning and inference in PGMs with theoretical results limiting both in the exact case. Correspondingly, various approaches have been presented to make inference tractable. One, namely the exploitation of symmetries in the data, has been receiving a lot of attention in recent years. Its general idea is to group indistinguishable objects together and to operate on the resulting clusters. Roughly speaking, this means that not every customer in a database is distinct in terms of the underlying problem and many customers exhibit the same characteristics. So, instead of making predictions on an individual level, we can now work on the clusters of customers. This can reduce the problem size by orders of magnitude and hence significantly reduce the required time for inference. In the *Statistical Relational Learning (SRL)* community, this approach is known as *lifting* or *lifted inference.* So far, lifted inference has been focusing on relational PGMs with mostly binary random variables and several of these lifted inference approaches are based on message passing. However, lifting is also useful in other settings and binary PGMs often limit the expressiveness or make statements overly complex. To further highlight these current limitations of PGMs and lifting, we now present three examples to motivate the contributions of this thesis.

1.1 Illustrating "non-standard" PGMs and Lifted Inference

Our focus is on PGMs and lifted inference approaches in "non-standard" settings. We will now provide three illustrative examples which will highlight our contributions and exemplify our understanding of these "non-standard" settings.

Lifted SAT — Lifted Message Passing Beyond Probabilistic Inference

One of AI's key challenges in moving ahead is closing the gap between logical and statistical AI. While logical AI has mainly focused on complex representations of knowledge, statistical AI has concentrated on the aspects of uncertainty. However, intelligent agents and systems must be able to handle both, namely the complexity and the uncertainty of the surrounding real world. While extending relational data with probabilities has been pursued in many research areas, PGMs have become one of the most successful approaches to combine logic with probabilities and have greatly contributed to the recently emerging fields of SRL [66] and *Statistical Relational AI (StarAI)*[1] [116].

Lifted inference algorithms, exploiting the relational structure of models, have rendered large and previously intractable probabilistic inference problems quickly solvable. In these cases, the logical representation is merged into the PGM whereupon the probabilistic inference algorithm is applied. In this regard it is worthwhile asking if the idea of lifting is applicable to algorithms solving logical problems that operate on similar concepts as probabilistic inference algorithms. As the following example suggests:

lifted message passing is not limited to probabilistic inference algorithms.

And taking this one step further, we will now indicate how instances of the *Boolean satisfiability problem (SAT)* [193] can be solved more efficiently based on lifted inference, hence, connecting logical and statistical AI in a "non-standard" setting.

SAT is one of the well known NP-complete [41] problems and efficiently solving particular classes of SAT instances contributes in many ways. Problems from knowledge representation, planning, scheduling, model checking, and many more, can all be reduced to SAT. A SAT problem poses the question if a given formula in propositional logic evaluates to `True`. Here, we assume that the formula is given in *conjunctive normal form (CNF)*, i.e., a conjunction of

[1] `www.starai.org`

1	2	3	4
2	?	?	?
3	?	?	?
4	?	?	?

(a) Problem

1	2	3	4
2	4	1	3
3	1	4	2
4	3	2	1

(b) Possible Solution

Figure 1.1: (a) Normalized Latin square of size four. Any solution fills the missing cells, denoted by a "?", in such a way that each row and each column uses each number (color) exactly once. One possible solution is depicted in (b).

disjunctions. This assumption is very common and does not impose any restrictions, since any formula can be converted into CNF. A solution to a SAT problem is an assignment to variables such that the underlying Boolean formula evaluates to `True`. Similar to lifted inference in relational models, we can exploit symmetries in the structure of propositional formulas when solving SAT problems and thus solve the underlying problem more efficiently. However, in contrast to many relational lifted inference approaches, we tackle the problem from the opposite point of view and follow a bottom-up approach.

We will now give a concrete example of a SAT problem and indicate how it can be solved. Figure 1.1a shows an example of a Latin square[2] which is akin to the popular Sudoku game. Similar to Sudoku, in a Latin square the goal is to label every unknown cell in such a way that each column and each row uses every number exactly once. In Figure 1.1a, this is further highlighted by using different colors for each number. The depicted example has a size of four with an additional constraint being that the first row and column are already labeled. Such a Latin square with a natural ordering in its first column and row is also called *normalized*[3]. We can now define a set of propositional clauses that describe the labeling problem more precisely based on the following constraints:

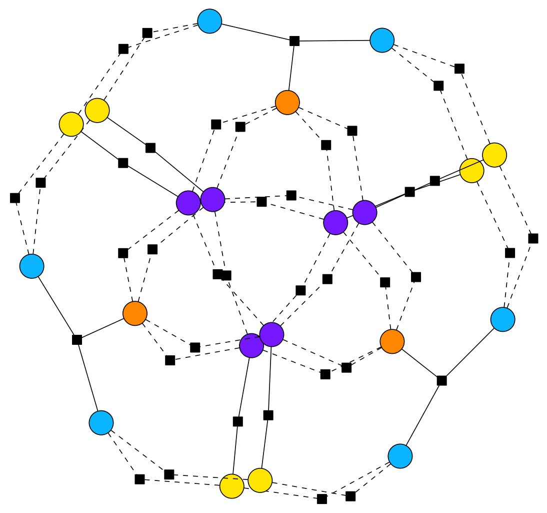
- Every cell is labeled with exactly one number.
- Every number occurs only once in every row.
- Every number occurs only once in every column.

Finding a solution to a Latin square can also be considered as finding a solution of a CNF respecting these specific constraints. Each variable in the CNF corresponds to the labeling of a specific cell with a particular color. While a normalized Latin square of size four has 21 variables, this number grows cubically with the size of the problem. Hence, one cannot simply try all combinations to find a solution to the problem we are trying to solve.

If we look at the formulas of a CNF representing a Latin square, we see that the clauses for each cell look similar and all cells are constraint likewise. As we will see below, this insight can be advantageous in iterative problem solving procedures which can be used to find solutions to SAT problems. This can be illustrated by representing the CNF as a graph which comes with the benefit of giving us a "neighborhood"-concept and hence some kind of relational information. To be more precise, we define the CNF by means of a factor graph (see Section 2.1.1 for a

[2]`en.wikipedia.org/wiki/Latin_square`

[3]Other Latin squares with different constraints can be found at `http://www.cs.ubc.ca/~hoos/SATLIB/Benchmarks/SAT/QG/qg.descr.html`



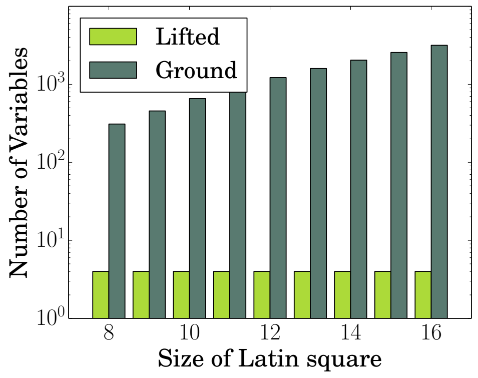


Figure 1.2: (left) Latin square of size four represented as a factor graph. The colors indicate clusters of variables that are constrained the same way. (right) Comparison of ground and lifted Latin squares for reasonable sizes. The ground problems grow cubically while the number of lifted variables remains constant.

detailed description of factor graphs). Figure 1.2(left) shows the factor graph representation for the Latin square of size four. Each variable is depicted as a circle and each square represents a clause. Each clause is connected to its variables via an edge. Dashed edges indicate that a variable appears negated in a clause.

Looking at the graph more closely, we see that the factor graph is symmetric and one can form different clusters of variables which are distinguished by colors. Note that these colors do not correspond to the colors which are used to color the Latin square itself in Figure 1.1. Symmetries in the factor graph arise from the fact that variables assigning the same label to different cells in the Latin square are constrained in the same way by the propositional formulas. An algorithm that finds such clusters based on sending messages in the graph will be explained in Section 2.4.1. For now, however, keep in mind that identically colored variables in the factor graph behave the same way. We will now indicate how the knowledge about these clusters speeds up the inference process.

Common approaches to solving SAT problems choose a variable heuristically, set its value to either `True` or `False`, simplify the CNF, and choose the next variable to clamp. Usually, the heuristic needs to be evaluated for each variable to make a locally optimal choice. Having a symmetric problem as the one described here, we can group the variables in such a way that we only have to evaluate the heuristic for each cluster because we know that each variable in a cluster behaves identically. For our current example, all variables in a cluster of the same color behave identically. Hence, we only need to apply the heuristic once to a single variable from this cluster.

This might not be impressive in the case of such a small problem. Therefore, let us have a look at Latin squares of increasing size. Figure 1.2(right) shows a comparison of ground and lifted Latin squares for realistic problem sizes. We see that the number of ground variables (dark green) increases cubically as stated above (note the log-scale of the plot). In contrast, the

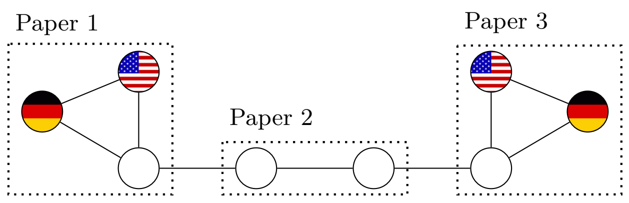


Figure 1.3: Extract of an online bibliography enriched with geographic information. A node corresponds to an author-paper-pair and dotted boxes refer to a paper written by the enclosed authors. The labels reflect the geographic location of an author while having written the particular paper and edges connect either co-authors or nodes of the same author for different papers.

number of lifted variables (light green) remains constant which is quite remarkable. Nevertheless, this is a common observation made in the lifted inference literature. In the case of probabilistic inference, it is often observed that the ground run time grows exponentially while the lifted run time only increases polynomially with growing problem sizes. At this point we should also mention that the level of compression drastically decreases once we clamp variables and simplify the model. In the worst case, we can no longer lift the model after simplification and detecting clusters becomes an unnecessary overhead. This aligns with setting evidence in PGMs where lifted inference for the unconditioned probability is fast but with evidence being set, the potential for lifting decreases.

The concept of lifting will be used throughout various parts of this thesis and will be applied in new areas apart from probabilistic inference in relational PGMs. We will return to the problem of *lifted satisfiability* in Section 3.1 where we will discuss and evaluate different problem instances in greater detail. More specifically, we will show that lifted inference can speed up SAT solving but also asks for new algorithms exploiting symmetries efficiently when variables are clamped. In this context, we will also discuss the problem of clamping in detail and we will propose algorithms that are capable of exploiting symmetries after setting evidence (see Section 3.2 and Section 3.3).

Lifted Labeling — Label Propagation on Lifted Matrices

The first example showed how instances of logical problems contain symmetries and illustrated how these symmetries can be exploited during problem solving. The idea of coloring the CNF is tightly connected to the message passing algorithm that is used in the decimation framework. However, this computational view on lifting can also introduce additional complexity as it requires changes to the messages being sent around (see Chapter 3). When looking at other problems and algorithms, we will see that symmetries in the problem structure itself can be exploited in such a way that the inference algorithm remains unaffected. This way, lifting can be seen as a preprocessing step. In a nutshell, the underlying graph can be lifted by algebraic means and the inference procedure, a sequence of matrix-matrix-multiplications, remains unchanged but now uses lifted matrices. In contrast to much of the existing literature, we show that

> *lifting is not limited to binary variables and does not necessarily require the modification of the ground algorithm.*

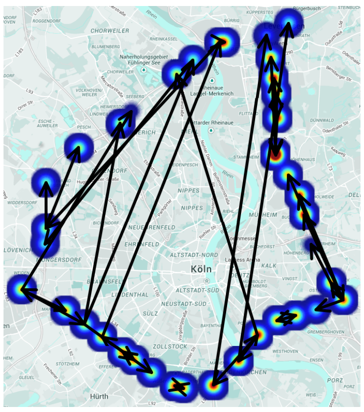


Figure 1.4: The prediction of traffic flows can be naturally realized by graphical models over count random variables. Dark red spots indicate a high traffic flow and the black arcs indicate how traffic segments influence each other.

We will now explain this idea for a labeling task with possibly thousands of labels. Let us assume that we want to label entities in a database. For example, one is interested in labeling the authors in a bibliography with their affiliations while writing a particular paper. Figure 1.3 illustrates this idea for a small database extract. Here, each node corresponds to an author of a paper while country flags are used as an approximation to an author's affiliation. Unknown affiliations are left blank. The dotted boxes enclose authors of the same paper. Each edge in the graph connects either two nodes corresponding to authors of the same paper, or two nodes of the same author but for different papers. The edges are supposed to model a similarity between nodes. Taking a closer glance, the presented graph encodes the intuition that neighboring nodes are more likely to share the same label. With this information we can now propagate the initial labels throughout this graph based on the edges. This is achieved by representing the graph and the initial labeling as matrices. Iteratively multiplying the label matrix with the similarity matrix results in a distribution over the labels for each node. This algorithm is known as Label Propagation [260].

In symmetric examples such as the one depicted in Figure 1.3, we can lift both of the involved matrices by algebraic operations returning two compressed datastructures. Interestingly, these lifted matrices can directly be used for the multiplications instead. The same label distribution as in the ground case can be read off the label matrix of reduced size. This idea of Lifted Label Propagation will be discussed in Section 5.3 in more detail. However, using a Gaussian similarity function, as it is often done, results in a dense similarity matrix, hence, limiting the applicability of Label Propagation in large-scale settings. Therefore, we resort in Section 5.1 to SRL concepts and define the similarity matrix based on logic formulas akin to relational PGMs.

Count Data — Graphical Models Beyond Multinomial and Gaussian Random Variables

The previous example has already shown a labeling task where the alphabet can be fairly large, albeit, still finite. In many other cases, we are confronted with variables defined over the natural numbers, i.e., discrete but infinite. Most of the existing PGMs are either multinomial or Gaussian, and thus ill-suited to represent the natural numbers appropriately in many situations.

	Range	Inference	Learning	Lifting
Chapter 3	$\{0, 1\}$	BP, SP, WP, LM	max. LL	Color Passing
Chapter 5	$\{0, \ldots, d\}$	LP	min. entropy	SAUCY
Chapter 7	$\mathbb{N}$	MCMC	max. LL via GTB	*ongoing work*

Table 1.1: This thesis covers different types of probabilistic graphical models and algorithms along with it. This table gives a brief overview on the approaches used in each technical chapter.

Extending PGMs to variables defined over the natural numbers is necessary to represent a large class of problems adequately.

To underline the necessity and benefits of such models, let us present a new example from the domain of traffic flow prediction. Figure 1.4 shows a map extract displaying the vehicular traffic from 38 stationary detectors located on the Cologne orbital freeway stretch adding up to 50 km in Germany. Each detector counts the number of cars passing by per minute and is naturally represented as a count random variable. Here, dark red detectors indicate high traffic and possibly a network congestion. An observation of the entire network for a certain point in time can be seen as one sample from a joint probability distribution of count random variables. Modeling this distribution requires to learn the dependencies between individual traffic segments.

Existing PGM approaches to such problems either discretize the count data into bins or follow a Gaussian approximation. Both techniques solely approximate the true nature of the counts. We therefore introduce in Chapter 7 a new class of PGMs based on Poisson distributions. These *Poisson Dependency Networks (PDNs)* are capable of learning dependencies between count random variables and inference in these models is based on a simple sampling procedure.

The black edges in Figure 1.4 feature the dependencies between detectors learned by a PDN. One can easily see that the dependencies of a highway segment do not solely depend on neighboring segments close by, but also on critical highway segments in other regions. Besides describing the traffic flow in the network, we can now also make predictions for missing datapoints or future situations. For example, detectors may often fail due to technical issues or construction sites. When these detectors are offline, we can use PDNs to fill in the missing counts. The ideas of modeling traffic based on counts has been further pursued in [96] and within the collaborative research center SFB 876, project B4[4].

1.2 Outline and Summary of Contributions

This thesis makes several contributions to inference and learning in PGMs by applying ideas from relational PGMs and lifted inference to non-standard settings. The examples in the previous section have already illustrated our understanding of non-standard settings and we will now summarize the content of this thesis in greater detail. Table 1.1 gives an overview how the models in the main technical chapters differ and how different inference and learning approaches are used accordingly. As we have seen in the previous three examples and as it is further shown in Table 1.1, the main technical chapters focus on random variables with different ranges, i.e., binary, multiclass, or count random variables. Depending on the range of

[4] `sfb876.tu-dortmund.de/SPP/sfb876-b4.html`

the variables, different inference and learning approaches will be used in each chapter. The idea of lifting is a recurring concept and will be discussed for each case. The content and contributions of Chapters 2 to 7 can be summarized as follows:

Chapter 2 To prepare the ground for what is to follow, Chapter 2 gives an overview on PGMs, including inference and learning algorithms. Chapter 2 furthermore explains the idea of lifted inference in detail because the specific contributions of Chapter 3 rely on the idea of lifted message passing. In this context, Chapter 2 explains probabilistic inference based on message passing and describes one of the most prominent lifted message passing algorithms in detail. This chapter will also comment on more advanced lifting techniques, such as informed Lifted Belief Propagation published in:

> K. Kersting, Y. El Massaoudi, B. Ahmadi, and **F. Hadiji**. Informed Lifting for Message-Passing. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

Since much of the lifted inference literature focuses on relational PGMs, Chapter 2 will also summarize the most well known formalisms in this area.

Chapter 3 While much of the early literature on lifted inference was primarily focusing on lifted variants of Variable Elimination and Belief Propagation, Chapter 3 introduces lifted versions of message passing algorithms beyond lifted Belief Propagation. We extend the understanding of lifting by introducing additional lifted versions of existing inference algorithms including Warning Propagation and Survey Propagation. Warning Propagation and Survey Propagation are message passing algorithms used to solve SAT problems; the lifted counterparts were published in:

> **F. Hadiji**, K. Kersting, and B. Ahmadi. Lifted message passing for satisfiability. In *Working Notes of the AAAI Workshop on Statistical Relational AI (StarAI)*, 2010.

As the introductory examples in the previous section already indicated, lifting is most effective in the unconditioned case, i.e., when no evidence is given or no variable has been clamped yet. However, iterative approaches to solving SAT instances clamp one or more variables in every iteration and therefore ask for lifting algorithms that can efficiently handle incremental evidence. To this end, we introduce a framework that is fitted to such sequential clamping approaches. This framework was previously published in:

> **F. Hadiji**, B. Ahmadi, and K. Kersting. Efficient Sequential Clamping for Lifted Message Passing. In *Proceedings of the 34th Annual German Conference on Artificial Intelligence (KI)*, 2011.

In particular, the proposed decimation approach tries to lift a clamped model more efficiently by adapting the unconditioned lifted model. To do so, it makes use of additional information from the network structure in form of shortest path distances. This Shortest-Path-Sequences lifting has been published in:

> B. Ahmadi, K. Kersting, and **F. Hadiji**. Lifted Belief Propagation: Pairwise Marginals and Beyond. In *Proceedings of the 5th European Workshop on Probabilistic Graphical Models (PGM)*, 2010.

Message passing algorithms as such work in an iterative way by exchanging messages to optimize an algorithm-specific criterion. In certain cases, this optimization obeys a monotonic behavior which can be interpreted as a process approaching evidence. We cut this process short and set the evidence during the message passing already, before the algorithm has completely converged. We show that such a likelihood maximization algorithm is liftable, yet requires a new lifting approach to handle the new evidence during the message passing itself. The lifted version of this maximum a posteriori inference algorithm together with the adapted lifting algorithm was published in:

> **F. Hadiji** and K. Kersting. Reduce and Re-Lift: Bootstrapped Lifted Likelihood Maximization for MAP. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, 2013.

Combining these contributions, we extend the applicability of lifting to new areas and establish lifting in these settings as a novel approach to reduce run time. We furthermore propose new lifting schemes that cope with incrementally growing sets of evidence to exploit symmetries even more effectively.

Chapter 4 The algorithms in Chapter 3 are based on message passing. When lifting these algorithms, the ground message passing algorithms have to be modified to respect the lifted model. Chapter 4 presents an alternative view on probabilistic inference without message passing. Instead, it reviews the well known semi-supervised Label Propagation algorithm. Label Propagation is used to label a graph based on a matrix representation of the similarity between nodes and a set of initially known labels. This algorithm features a probabilistic interpretation as well and returns a distribution over labels for each node in the graph. While Chapter 3 handles binary PGMs only, Label Propagation can naturally process large label alphabets and hence solve multiclass labeling problems. In the next step, we review connections between graph theory and lifted probabilistic inference that have been established recently, to ultimately motivate a lifted version of Label Propagation in the subsequent chapter.

Chapter 5 Standard Label Propagation commonly uses a Gaussian function to model the similarity between nodes. This results in a dense similarity matrix which does not scale to large graphs with millions of nodes. Relational PGMs have been shown to easily describe dependencies between related objects by using first-order logic as a template language. We apply these ideas to Label Propagation and define similarity matrices based on weighted logical rules. Such a rule-based approach for Label Propagation entails several advantages. First, logical rules allow an intuitive problem definition as knowledge can be encoded more easily with the help of domain experts. Second, logical rules often lead to much sparser similarity matrices compared to the standard setting where a Gaussian function is used. As our empirical results show, this relational Label Propagation still produces high accuracy labels. At this point, one may reflect upon whether Label Propagation is liftable too, especially when using a rule-based similarity function. In this regard we show that the interplay of graph theory and lifting algorithms can be applied to Label Propagation and we prove that Label Propagation can be lifted in terms of algebraic operations; the Lifted Label Propagation algorithm has recently been accepted in:

> **F. Hadiji**, M. Mladenov, C. Bauckhage, and K. Kersting. Computer Science on the Move: Inferring Migration Regularities from the Web via Compressed Label Propagation. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

Lifted Label Propagation results in a much faster labeling algorithm with less memory consumption. Furthermore, lifting becomes a single pre-processing step that does not require modifying the ground version. Hence, Lifted Label Propagation can be combined with any other method making Label Propagation more efficient.

Chapter 6 In Chapter 5, we use lifted relational Label Propagation to label large-scale online bibliographies with geographical attributes. Starting with a freely available online bibliography, we harvest additional information from the Web to add initial seed affiliation of authors to our database. Label Propagation is then used to fill in the missing affiliation information, resulting in a database of several million papers and authors augmented with geographical tags. Having such a vast resource at hand, Chapter 6 will point out that several interesting patterns and regularities can be found in this dataset. We show how to derive several statistical regularities and patterns from these geo-augmented bibliographies by fitting simple distributions to the data based on the maximum likelihood estimation principle. Our findings encompass distributions describing the migration behavior of researchers, as well as a general model for the academic job market. Excerpts of these patterns and regularities have been included in the IJCAI-15 publication referred to above and a full analysis has been made publicly available at:

> **F. Hadiji**, K. Kersting, C. Bauckhage, and B. Ahmadi. GeoDBLP: Geo-Tagging DBLP for Mining the Sociology of Computer Science. *arXiv preprint arXiv:1304.7984*, 2013.

Such an analysis of human behavior is applicable to various other kinds of datasets and we further highlight how we have used such techniques to describe the attention to Internet memes [14] and the spread of viral videos [15].

Chapter 7 While Chapter 3 focuses on binary variables and Chapter 5 deals with multiclass problems, Chapter 6 furthermore highlights that we are still missing the setting with variables defined over infinite ranges. This gap is filled in Chapter 7 where we take a closer look at random variables defined over the natural numbers and count data in general. While Chapter 6 fits distribution to count data for descriptive purposes, Chapter 7 presents models that allow a prediction conditioned on observations. Although counting is key to describing various observations and statistics, there is little work on PGMs for random variables defined over the natural numbers so far. We therefore introduce a different class of PGMs, namely *Poisson Dependency Networks (PDNs)*, that is tailored towards count data based on Poisson distributions, as previously existing graphical models were mainly focusing on discrete and finite data or Gaussian random variables. Although PDNs consist of a set of local conditional Poisson distributions, we show how to sample from the joint distribution with the help of a Gibbs sampler. Besides answering probabilistic queries, these samples can also be used to predict counts of unseen instances. In order to efficiently learn PDNs, we resort to *Gradient Tree Boosting (GTB)*. GTB has successfully been used in a large variety of applications where a stepwise optimization is performed by summing together the output of simple basis functions. Besides the additive case, which relies on a step-size specified by the user, we also present the first approach to multiplicative GTB to improve convergence. This multiplicative GTB shows to be an effective alternative in many cases that does not require the tuning of a step-size. This new class of PGMs has recently been published in:

> **F. Hadiji**, A. Molina, S. Natarajan, and K. Kersting. Poisson Dependency Networks: Gradient Boosted Models for Multivariate Count Data. *Machine Learning Journal*, 2015.

We further touch upon relational extensions for this model class and also describe how lifting could be implemented to motivate future work in this area.

We conclude in Chapter 8 by summarizing the main aspects of this thesis, presenting the lessons learned during the development of this thesis, and suggesting possible directions for future work. A list of selected publications in which the author was involved can be found in Section 1 of the Appendix which also details the author's specific contributions to each paper.

The reader may notice that the chapters are not equally dense. In particular, Chapter 3 is comparably more extensive. This primarily rises from the fact that much of this thesis was motivated by the initial work on lifting as described in Chapter 3. The work therein started to look at lifting in "non-standard" settings, however, still focusing on message passing algorithms in binary PGMs. While this work evolved, many of the ideas presented in Chapters 4 to 7 were developed, extending PGMs and ideas from lifting by additional dimensions. Therefore, we propose as a future research question to investigate lifting in the probabilistic count models presented in Chapter 7 which in combination would once again emphasize the core contributions of this thesis.

CHAPTER 2

From Probabilistic Inference to Lifted Message Passing

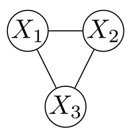
There are a couple of definitions and ideas that return in each of the following chapters and will be summarized in the current chapter. This mainly focuses on the definition of PGMs and particular instantiations of this formalism. It also includes the main challenges in PGMs and state-of-the-art approaches for solving them. A discussion on inference and learning in PGMs requires basic ideas from probability theory. Similarly, relational PGMs depend on terminology from logic. We will now describe these ideas and concepts in greater detail.

2.1 Probabilistic Graphical Models

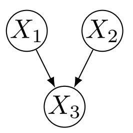
In this section, we give a short introduction to PGMs and define our notation. For a more detailed review of PGMs, we refer the interested reader to [124, 237]. Generally, PGMs combine ideas from graph theory and probability theory. Let us assume that we are given a graph $G = (V, E)$ where $V = \{V_1, V_2, \ldots, V_n\}$ is a set of vertices and $E \subseteq V \times V$ is a set of edges.

Each edge e_{ij} connects two vertices $V_i, V_j \in V$. In the most general form, we consider undirected edges between the nodes. In order to define a PGM, we further assume that we have a random variable X_i associated with every vertex $V_i \in V$. Each variable X_i is defined over a range $\mathcal{X}_i$. Throughout this thesis we will consider different ranges, starting from binary ones, e.g., $\mathcal{X}_i = \{0, 1\}$, over discrete but finite ones, e.g., $\mathcal{X}_i = \{0, 1, \ldots, d\}$, to discrete and infinite ones, e.g., $\mathcal{X}_i = \mathbb{N}$. We will also touch upon continuous ranges along the way, i.e., $\mathcal{X}_i = \mathbb{R}^+$, because the Gaussian setting for instance is related to the discrete case in many ways and various concepts carry over. Let x_i represent a possible instantiation of random variable X_i, i.e., $x_i \in \mathcal{X}_i$. We define a vector of n random variables as $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ and $\mathbf{x}$ as its instantiation. We now associate a *potential function* or *compatibility function* ϕ_c with every clique c of the graph G. A clique is a fully connected subset of vertices in G and we define the set of all cliques in G as C. By $\mathbf{X}_c$, we refer to the variables contained in clique c. Based on these potential functions, we can formulate the following factorization of a joint probability distribution:

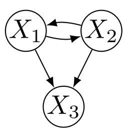
$$p(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \prod_{c \in C} \phi_c(\mathbf{x}_c) \; . \tag{2.1}$$



(a) Example MRF



(b) Example BN



(c) Example DN

Figure 2.1: A visual comparison of the different probabilistic graphical models that are covered in this thesis.

Here, each function ϕ_c is a non-negative function over a subset of the variables $\mathbf{X}_c$, i.e., $\phi_c : \mathcal{X}_1 \times \ldots \times \mathcal{X}_{|\mathbf{x}_c|} \to \mathbb{R}^+$. $Z = \sum_{\mathbf{x} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n} \prod_{c \in C} \phi_c(\mathbf{x}_c)$ is a normalization constant, ensuring that $p(\cdot)$ is a valid probability distribution. The product of positive functions, or possibly even local probability distributions, in Equation (2.1) together with a graph defines what is known as a *Markov Random Field (MRF)*. A small example MRF is depicted in Figure 2.1a. This relationship between a graph and a probability distribution goes back to the theorem of Hammersley-Clifford [89, 19]. The theorem of Hammersley-Clifford states that we can find such a factorization over cliques for every positive probability distribution, i.e., where we have $\forall \mathbf{x} : p(\mathbf{X} = \mathbf{x}) > 0$. This highlights the great expressiveness of MRFs and gives the theoretical foundations. Furthermore, the factorization of the graph describes the conditional independencies of the distribution. By this, the graph G, or equivalently the MRF, tells us that variable X_i is independent of all other variables in the graph given its direct neighbors $\mathrm{nb}(i)$, i.e., $P(x_i|\mathbf{x}_{\setminus i}) = P(x_i|\,\mathrm{nb}(i))$. Here, $\mathbf{x}_{\setminus i}$ refers to a random vector $\mathbf{x}$ with variable X_i removed. This is the so called *local Markov-property* and X_i's neighbors are also called *Markov blanket*.

It is important to note that C does not have to be the set of maximal cliques because any set of non-maximal cliques can be converted to a set of maximal cliques. Additionally, we also consider a single node as a clique which often simplifies the model specification. In fact, every MRF can be equally represented by a pairwise MRF, i.e., a model with pairwise potential functions only. We also want to note that the distribution in Equation (2.1) can be equivalently represented as a log-linear model:

$$p(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \prod_c \exp(w_c \cdot g_c(\mathbf{x}_c)) = \frac{1}{Z} \exp\left[\sum_c w_c \cdot g_c(\mathbf{x}_c)\right], \tag{2.2}$$

where the features $g_c(\cdot)$ are arbitrary functions over (a subset of) the configuration $\mathbf{x}$ and w_c is a feature weight. We will come back to this model below where we describe MRFs based on weighted logical rules (see Section 2.3.1). In these cases, each g_c corresponds to a function returning the truth value of a logical rule and w_c defines its weight.

Using yet another reformulation of the MRF definition, we obtain a *Gibbs distribution* which is often used in statistical Physics:

$$p(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \exp\left[-\sum_c \mathcal{E}_c(\mathbf{x}_c)\right] = \frac{1}{Z} e^{-\mathcal{E}(\mathbf{x})}, \tag{2.3}$$

where $\mathcal{E}(\mathbf{x}_c) > 0$ is the energy associated with the variables in clique c. By setting $\mathcal{E}_c(\mathbf{x}_c) = -\ln(\phi_c(\mathbf{x}_c))$, we obtain again our original form of the MRF. Energy-based formulations become interesting when one seeks for the most likely configuration of the distribution because maximizing the probability now equals minimizing the total energy.

We have now seen different representations of MRFs which all assumed some kind of potential functions, or more specifically in case of the log-linear representation, the MRF was based on feature functions g_c and the corresponding weights w_c (a.k.a *canonical parameters*). However, as we will see next, we can also parameterize the MRF in a different way. One can also view the distribution defined by the factorization in terms of a *mean parameterization* which is immediately related to the inference problems introduced in Section 2.2. We can define for each ϕ_c a mean parameter $\mathbb{E}[\phi_c(\mathbf{X}_c)] = \sum_{\mathbf{x}_c} \phi_c(\mathbf{x}_c) p(\mathbf{x}_c)$ which is then used to define a vector of *mean parameters* over all potential functions $(\mathbb{E}[\phi_1(\mathbf{X}_1)], \ldots, \mathbb{E}[\phi_{|C|}(\mathbf{X}_{|C|})])$. We can now also define the set of all realizable mean parameters as:

$$\mathcal{M} = \left\{ \boldsymbol{\tau} \in \mathbb{R}^{|C|} \mid \boldsymbol{\tau} = \sum_{\mathbf{x}} \phi(\mathbf{x}) p(\mathbf{x}) \text{ for some } p(\mathbf{x}) \geq 0,\ \sum_{\mathbf{x}} p(\mathbf{x}) = 1 \right\} .$$

Here, $\phi = (\phi_c,\ c \in C) \in \mathbb{R}^{|C|}$ is the vector of all potential functions and $\mathcal{M}$ defines a convex polytope. We can assume without loss of generality that an MRF is pairwise, i.e., it contains only unary and pairwise feature functions. A procedure that transforms MRFs of arbitrary arity to a pairwise form is given in [237, E.3]. For such a pairwise MRF over $G = (V, E)$, the mean parameters correspond to the marginal probabilities $p(X_i)$ and $p(X_i, X_j)$ for connected variables. In this case, we have

$$\mathcal{M}(G) = \left\{ \boldsymbol{\tau} \in \mathbb{R}^{|C|} \mid \exists p \text{ s.t. } \tau_i = p(x_i)\ \forall i \in V \text{ and } \tau_{ij} = p(x_i, x_j)\ \forall (i,j) \in E \right\} , \tag{2.4}$$

and refer to $\mathcal{M}(G)$ as the *marginal polytope.* These marginals, and marginals over any subsets of variables, are of great interest because one often wants to answer probabilistic queries in the form of $p(\mathbf{X}_a = \mathbf{x}_a)$. Unfortunately, the number of constraints required to describe $\mathcal{M}(G)$ grows exponentially and therefore obtaining the mean parameterization efficiently is not possible in most cases. Indeed, determining the mean parameterization from the canonical representation amounts to probabilistic inference and is one of the main challenges in PGMs. The mapping in the opposite direction has an interesting interpretation as well and can be seen as parameter learning [237]. To reduce the number of required constraints, and hence the complexity, linear program relaxations have been proposed that focus on local constraints only. In particular, with the constraints

$$\sum_{x_i} \tau_i = 1\ \forall i \in V\ , \sum_{x_j} \tau_{ij} = \tau_i\ , \sum_{x_i} \tau_{ij} = \tau_j\ \forall (i,j) \in E\ , \tag{2.5}$$

we can define the *local marginal polytope*

$$\mathcal{M}_L(G) = \left\{ \boldsymbol{\tau} \in \mathbb{R}^{|C|} \mid \text{Conditions of Equation (2.5) hold.} \right\} ,$$

which outer bounds $\mathcal{M}(G)$. But more importantly, $\mathcal{M}_L(G)$ requires only a linear number of constraints in the number of edges. The marginal poltytopes $\mathcal{M}(G)$ and $\mathcal{M}_L(G)$ will be of greater interest in Section 3.3 where we will refer to an algorithm that adds additional constraints to the polytope, in order to implement a fast approximate inference algorithm. For more information on the mean parameterization and the marginal ploytope, we refer to [237, Section 3.4].

Besides the undirected MRFs, there is another very popular class of PGMs, namely *Bayesian networks (BNs)* [124]. Here, the joint probability distribution is represented by means of a directed, acyclic graph. Again, there is a node in the graph associated with each variable in the probability distribution. However, due to the fact that the graph is directed, we now have a set of parents for each node, referred to $\mathrm{pa}(X_i)$, instead of a Markov blanket as in MRFs. Each variable X_i is conditionally independent of its non-descendants in the network given its parents. This means that the marginal conditional probability distribution of each variable is specified by a probability table over the variable itself and its parents. The joint distribution of a BN is then defined as:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(X_i |\, \mathrm{pa}(X_i)) \; . \tag{2.6}$$

We saw for undirected models that the theorem by Hammersley and Clifford established the connection between an undirected graph and a probability distribution. Similarly, we have for BNs the representation theorem [124, Theorem 3.1 and 3.2] that connects a directed, acyclic graph with the factorization in Equation (2.6). An example of a BN is depicted in Figure 2.1b. Looking again at Equation (2.1), we can construct an MRF based on the conditional probability tables of an BN. We represent each conditional probability $p(X_i |\, \mathrm{pa}(X_i))$ by a potential function over $X_i \cup \mathrm{pa}(X_i)$, resulting in $Z = 1$. However, the structure of the MRF is not identical to the BN's graph, as we have to connect all parents of a node. This conversion is often referred to as *moralization* and the resulting graph is called a *moral graph*. For example, if we moralize the BN in Figure 2.1b, we obtain the undirected model in Figure 2.1a which is completely connected. This example already highlights that we lose some of the independence information when transforming a BN into an MRF. The nodes X_1 and X_2 are marginally independent, however, it is not possible anymore to read off this information from the completely connected MRF. For more information on the connection between BNs and MRFs, as well as the encoded independencies, we refer the reader to [124, Section 4.5]. Nevertheless, every probability distribution defined by a BN can be encoded as an MRF as well.

Although BNs already impose restrictions on the graphical structure of the problems compared to MRFs, parameter and structure learning is still a tough challenge. Hence, models were proposed that combine advantages of both models but fail to define a consistent probability distributions in closed form. Nevertheless, these models still perform well on various tasks and reduce complexity of learning and inference significantly. One example of such models are *Dependency Networks (DNs)* [90] which we find in between BNs and MRFs. DNs are defined as a set of local conditional probabilities that may contain cycles where each local conditional probability distribution is learned independently. An example of a DN is depicted in Figure 2.1c. We will describe DNs in larger detail in Chapter 7, as these models play a central role for our approach of handling count data, i.e., random variables over the natural numbers. Chapters 3 and 5 are mainly concerned with MRFs, although the work in Section 3.3 is solving the inference problem by transforming the original MRF into a mixture of simple BNs. While Chapter 3 is mainly concerned with binary MRFs, Chapter 5 looks at multinomial MRFs which however have an equivalent Gaussian representation in the case of binary variables.

As we have seen above, we can use graphs to formulate BNs, MRFs, and DNs. However, in many cases a different type of graph is more expressive as we will see next. This formalism, called *factor graph*, can represent all three types of models and hence factor graphs represent the main formalism to visualize graphical models in this thesis.

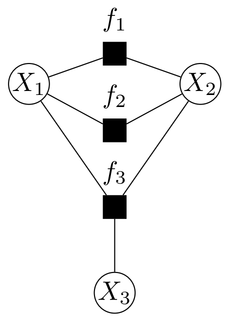


Figure 2.2: Example factor graph. Circles denote variable nodes and squares denote factors.

2.1.1 Factor Graphs

As we have seen in the previous section, there are different ways of defining graphical models. MRFs are represented by undirected graphs, while directed graphs are used for BN. However, there is also a formalism that is capable of representing both types of models, namely factor graphs [128]. Hence, factor graphs present an important formalism in this work, as several of our problems can be formulated in a unified way. Additionally, the factor graph formalism is more expressive than undirected graphs, since it can distinguish between pairwise and higher-order interactions. In a undirected graph, cliques of size two cannot be distinguished from cliques of size three. For example, the MRF shown in Figure 2.1a could very well represent a pairwise factorization $p(\mathbf{x}) = \phi_1(x_1, x_2)\phi_2(x_1, x_3)\phi_3(x_2, x_3)$, as well as a single clique $\phi_1(x_1, x_2, x_3)$. However, this circumstance would not be visualized appropriately with the corresponding underlying graph. More formally, a factor graph is a bipartite graph $G = (V, E)$ that expresses the factorization of a joint probability distribution for $p(\mathbf{X})$ as in Equation (2.1). Here, V is the set of nodes, which contains a variable node for each variable $X_i \in \mathbf{X}$ and a factor node for each $\phi_c \in C$. There is an edge $(i, c) \in E$ between the variable node for X_i and the factor node corresponding to ϕ_c if and only if the variable is in the range of the function associated with the factor. Since G is bipartite, there are no edges connecting two variable nodes or two factor nodes. In our figures, we will be denoting variables nodes as circles and factor nodes as squares. An example of a factor graph is depicted in Figure 2.2. Additionally, factor graphs also provide a probabilistic view on CNFs as we have seen in Figure 1.2: the marginals of the variables in a factor graph representing a CNF give the probability of the variable being `True`, respectively `False`, when all solutions to the problem in G are chosen uniformly, i.e., unweighted.

2.2 Inference and Learning in Graphical Models

This thesis covers different facets of inference and learning in PGMs. Therefore, we will now summarize the main challenges in this area. Assuming that the structure of a PGM and its parameters are already given, we are facing different inference tasks in order to answer probabilistic queries. However, in other settings we are starting solely with a dataset of empirical observations for which we first need to learn an appropriate PGM. Resulting form these different scenarios, we distinguish the following main challenges:

- **Probabilistic Inference**: Answering probabilistic queries for one or more variables is referred to as *probabilistic inference.* We can further split probabilistic inference into the following two subclasses:
  - *Marginal Inference*: In many situations, we are interested in the marginal probability of a single node $p(X_i = x_i) = Z^{-1} \sum_{\mathbf{x}_{\setminus i}} \prod_c \phi_c(\mathbf{x}_c)$. In other cases, we might also need the probabilities of a subset of variables, i.e., $p(\mathbf{X}_c = \mathbf{x}_c)$, or the probability of an entire joint configuration $p(\mathbf{X} = \mathbf{x})$. Depending on the algorithm, different of these quantities are provided or can be read-off easily. In the case of BNs, the joint probability can be determined easily, while this requires the computation of the normalization constant Z in MRFs. For BNs we have $Z = 1$. As seen above, Z is defined as the sum over exponentially many configurations which already indicates the hardness of its computation. Indeed, it can be shown that exact inference, i.e., the decision problem $P(X_i = x_i) > 0$, is NP-complete [124]. The proof follows a reduction from the 3-SAT problem which is known to be NP-complete. In correspondence to this results, it can also be shown that computing the marginal probability $P(X_i = x_i)$ itself is #P-complete [191]. Here, the proof follows a reduction from the model counting problem for 3-SAT. It can also be shown that probabilistic inference amounts to weighted model counting [34] which gives yet another connection to SAT related problems and their weighted extensions. Due to the hardness of inference in general, one often resorts to approximate inference algorithms. Parameter learning typically relies on computing the marginal probabilities and hence, efficient learning is also most often done in an approximate fashion.
  - *MAP inference*: In other cases, we are not interested in the exact probability of a configuration but instead want to find the most likely one. The *maximum a posteriori (MAP)* inference amounts to finding the most probable assignment $\mathbf{x}^*$ to all unobserved variables, i.e., the *mode* of the possibly conditioned distribution. More formally, the task of MAP inference is defined as

    $$\mathbf{x}^* = \arg\max_{\mathbf{x}} p(\mathbf{x}).$$

    In general, this problem is NP-complete [203] as well. Still, MAP inference is a challenging and crucial task in solving different problems and it has applications in various domains such as image processing [251], entity resolution [206], and dependency parsing [144], among others. Here, the MAP assignment presents a solution to a specific problem and one is also required to use approximation algorithms in most situations. Important for MAP inference is the notion of *max-marginals* which are defined as follows:

    $$m(x_i) = \max_{\{\mathbf{x} | \mathbf{x}_i = x_i\}} p(\mathbf{x}) \tag{2.7}$$

    Many algorithms solve the MAP inference problem by calculating the max-marginals first. Having the max-marginals, one can decode the MAP configuration under certain assumptions [124, Section 13.1]. To be more precise, if each variable X_i achieves a unique max-marginal $x_i^* = \arg\max_{x_i \in \mathcal{X}_i} m(x_i)$, i.e., is unambiguous, then $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)$. We have seen above that the joint probability of an MRF can equally be formulated based on an energy function (see Equation (2.3)). Therefore, the problem of MAP inference can equally be formulated as energy minimization:

$$\begin{aligned}\arg\max_{\mathbf{x}} p(\mathbf{x}) &= \arg\max_{\mathbf{x}} \ln p(\mathbf{x}) = \arg\min_{\mathbf{x}} -\ln p(\mathbf{x}) \\ &= \arg\min_{\mathbf{x}} \ln(Z) + \mathcal{E}(\mathbf{x}) \\ &= \arg\min_{\mathbf{x}} \mathcal{E}(\mathbf{x}) \;.\end{aligned}$$

We can drop the normalization constant $\ln(Z)$ because we are just interested in a minimizing configuration. Moving to the log-domain has several advantages, e.g., we can avoid numerical instabilities of the multiplication and so on.

- **Learning**: We can further decompose the general learning task into two subclasses:
  - *Parameter Learning*: Given an MRF, we are required to choose the potential functions or parameters, e.g., the weights w_c in Equation (2.2). In *generative learning*, one typically uses a training dataset and obtains the optimal parameter values by maximizing the log-likelihood over this training dataset. This approach is called *maximum likelihood estimation (MLE)* and assumes in the most simple case a fully observed dataset. Often, this already requires several calls to an inference procedure which highlights the importance of efficient marginal inference. For efficiency reasons, one often optimizes the pseudo-log-likelihood [20] instead of the true log-likelihood. In several situations, we can split the variables $\mathbf{X}$ of an MRF in observed and latent variables, i.e., variables that are always given ($\mathbf{E}$) and variables that will be predicted based on the observed ones ($\mathbf{Y}$). Here, we can use *discriminative learning* that optimizes only the conditional log-likelihood $p(\mathbf{Y}|\mathbf{E})$ instead of $p(\mathbf{X})$. A more detailed discussion of generative and discriminative models can be found in [217]. In Chapter 7, we will learn the parameters of conditional probability distributions and touch upon the issue of parameter learning in greater detail. MLE is typically done in supervised learning, however, in some situations, we are only given one mega-example which is additionally not fully labeled. We find ourselves in such a situation in Chapter 5 where we use an semi-supervised algorithm to solve a graph labeling problem. In this case, learning can be done via entropy minimization instead of MLE, as we will describe in Section 5.5. This combines the knowledge of known labels and the structure of the entire problem, however, it also introduces the assumption that parameters leading to low entropy are desirable.
  - *Structure Learning*: The most challenging problem is structure learning because it also subsumes parameter learning as a subroutine. Here, we start solely with a given dataset and do not know the structure of the MRF in advance. Many structure learning algorithms start with an empty network or a network consisting of trivial potentials. In an iterative procedure, the existing potentials are refined or new potentials are generated based on the existing ones. A score function determines which new potentials are added to the model. Once the refinement process cannot improve the model anymore, the learning algorithm stops. Due to its high complexity, structure learning is often neglected and a structure is proposed in joint work with domain experts. With an exception in Chapter 7, we assume a given structure as well. The models proposed in Chapter 7 suggest a simple and efficient local learning

approach in which local models are learned independently. These local models can then be combined via MCMC methods to infer the structure and dependencies of the joint probability distribution. This approach circumvents the hard problem of finding a consistent structure immediately and instead reconstructs the joint distribution based on the local models.

In the following two subsections, we will briefly summarize algorithms that aim to solve the challenges above but we will focus on marginal and MAP inference. Often, only slight modifications have to be made to a marginal inference algorithm in order to apply it to MAP inference. We will present examples of exact and approximate inference algorithms and give a high level view without many details or examples. In later chapters, we will discuss a few algorithms selectively and explain them in greater detail while also relating these approaches to our own technical contributions. This overview focuses on inference algorithms in PGMs with multinomial variables. Inference algorithms for PGMs with Gaussian or Poisson distributed variables are going to be discussed in later chapters together with our contributions in that area.

2.2.1 Exact Inference

We begin by briefly covering the most prominent exact inference algorithms before discussing approximate algorithms. When we are interested in the marginal distribution of one or more variables $\mathbf{X}' \subseteq \mathbf{X}$, we can calculate $p(\mathbf{x}')$ via

$$p(\mathbf{x}') = \sum_{\mathbf{y}} p(\mathbf{y}, \mathbf{x}') = \sum_{y_1} \dots \sum_{y_{|\mathbf{Y}|}} \prod_{c} \phi_c(\mathbf{x}_c) \; .$$

We can avoid summing over all configurations of $\mathbf{Y} = \mathbf{X} \setminus \mathbf{X}'$ explicitly by a clever reordering of the terms. The products in the sum over the joint configurations are based on the factorization. The reordering is possible due to the fact that not each factor ϕ_c is defined over all variables in $\mathbf{Y}$. This allows us to eliminate one variable $Y \in \mathbf{Y}$ at a time to compute $p(\mathbf{x}')$ at the end. This idea was originally presented for BNs in [257], however, it is generally applicable to factor graphs, and the algorithm is known as *Variable Elimination (VE)*. The elimination of a variable is based on summing out a variable Y from a factor ϕ_c which is also know as factor marginalization. Summing out a variable from the entire factorization requires first the computation of the factor product of all factors containing Y. We can then sum out the variable from this newly constructed factor, creating yet a new factor not dependent on Y anymore. This new factor is used in the elimination of the following variables and the efficiency of VE relies on this dynamic programming paradigm. The runtime of the algorithm depends on the elimination ordering of the variables. Finding an optimal ordering of the variables is an NP-hard problem though. Nevertheless, several heuristics have been presented that achieve good results empirically. VE can be used for MAP inference as well. This requires replacing the sum my the max-operator in the elimination process. Further details on VE can also be found in [124].

As a second example of an exact inference algorithm, we will now describe the *Junction Tree (JT)* algorithm. The description of the VE algorithm was already based on the idea of factor manipulations. The idea behind the JT algorithm is similar but uses the notion of message passing. We will see through the entire theses that the idea of message passing occurs frequently in inference algorithms and makes description of these convenient. We need to introduce a few definitions before we are ready to explain the JT algorithm. Let us recall that we have defined an undirected graph $G = (V, E)$ with nodes V and edges E. A *cluster graph* is a graph where nodes correspond to subsets of variables in V and two nodes are connected by an undirected edge if and only if the intersection of their subsets is non-empty. Correspondingly, we define a *cluster tree* as cluster graph having a tree structure. A cluster tree has the *running intersection property*, if for all cluster C_i and C_j with $X \in C_i$ and $X \in C_j$, we have $X \in C_k$ for every C_k on the path between C_i and C_j. A cluster tree satisfying the running intersection property is called a *clique tree* or *junction tree*. Interestingly, applying VE to an MRF over G induces a cluster graph over V which is necessarily a tree and which also satisfies the running intersection property. The root of this tree is a cluster containing X_i if VE was run to compute $p(X_i)$. This tree implicitly defines a direction and VE can be seen as a message passing on this cluster tree where messages are passed from the leafs upwards to the root. A message send from C_i to C_j corresponds to a manipulated factor akin to VE. This factor is constructed by multiplying C_i's incoming messages and C_i's initial potential, and then summing out all variables not in the intersection of C_i and C_j. After all messages have been propagated to the root, we can do a similar message passing the other way around. From this upward and downward message passing we obtain a calibrated clique tree where each clique has incoming messages from both sides. The clique tree can be seen as another reparameterization of the MRF that allows to read off marginal probabilities of the cliques easily, and hence, also the calculation of single node marginals. Compared to VE, JT is the method of choice, when we want to compute the marginals of several variables because we can reuse computations along the edges. The message passing implies that we can implement inference efficiently with a junction tree. However, we have not discussed yet, how we obtain the tree in general. In particular, how to construct a junction tree without running VE first. To this end, we call a graph *triangulated* if every cycle of length four or longer has a chord. A *chord* joins two nodes in the cycle that are not adjacent. It can now be shown that a graph has a junction tree if and only if it is *triangulated*. Therefore, we need to triangulate the graph first. Unfortunately, finding the optimal triangulation is NP-hard. Once we have a triangulated graph, we can read off the maximal cliques from this graph and construct a new undirected graph based on these maximal cliques. We then set the weight of an edge connecting C_i and C_j as the size of the intersection of both cliques. Running an maximum spanning tree algorithm on this graph gives us the desired tree for running the JT algorithm. The efficiency of the JT algorithm depends on the *treewidth* of the junction tree. The treewidth is the size of the maximal clique over all possible triangulations of a graph and the JT algorithm is exponential in that number. In cases where we are interested in MAP inference, we can adapt the JT algorithm in a similar way as VE is adapted for this task.

Both, VE and JT are only feasible if the network is sparse, i.e., has low treewidth. Therefore, we will now turn our attention to approximate inference algorithms as these are the only ones applicable in large-scale settings. In the following subsection, we will introduce some of the most prominent examples of approximate inference algorithms. This will lead us to other message passing algorithms as well. In particular, we will see *Belief Propagation (BP)* which is an exact message passing algorithm if the underlying graph is a tree.

2.2.2 Approximate Inference

In most settings, in particular when using relational PGMs, exact inference algorithms are not feasible. Therefore one often resorts to approximate inference algorithms. One class of approximate inference algorithms are sampling based methods. There is a large choice of sampling approaches including *rejection sampling*, *importance sampling*, and *Markov chain Monte Carlo (MCMC)* methods, to name only a few. The general idea behind sampling algorithms is to obtain a set of samples drawn from the distribution $p(\mathbf{X})$. As a representative of the MCMC methods, we will now briefly describe Gibbs sampling and we refer to [21, Chapter 11] and [124, Chapter 12] for a general overview of sampling methods. *Gibbs sampling* is a very simple yet widely applied technique. Gibbs sampling [64] is an MCMC method and serves as a simple approximate inference technique for problems where the conditional probability of a variable given its Markov blanket can be computed easily. Starting with an initial value of the unobserved variables, we iterate over all of these variables in a pre-defined, or alternatively random, order. For each variable, we sample a new value based on its conditional probability distribution given its neighbors. In many BNs and MRFs, every sampling step can be done fast because one can easily sample from the conditional distribution if all parents, respectively neighbors, are observed. Iterating once over all variables in the network is often referred to as a *sweep*. These sweeps are then computed for a fixed number of iterations. The more iterations we make, the less influence does the initial state of the variables have and we are approximating the true posterior distribution. To reduce the bias of the initial state, a burn-in phase is usually incorporated into the sampling process. From the MCMC algorithms' point of view, the Gibbs sampler is a particular instance of the *Metropolis-Hastings* algorithm with a constant acceptance probability equal to one, i.e., all samples are accepted. Gibbs sampling will be used in different parts of this thesis because the MRFs in use allow to easily sample from the required conditional distributions. For example, we will see in Chapter 7 how Gibbs sampling is used to generate samples for count random variables that are assumed to be conditionally Poisson distributed. Another interesting MCMC approach worth mentioning is *slice sampling*. This sampling algorithm also finds applications in PGM inference and is referred to in Section 2.3.1 again.

While MCMC methods have been among the most popular approximate inference algorithms for a long time, an increasing interest in alternatives to MCMC has been seen starting in the late nineties of the last century. A large class of these inference algorithms are message passing based approaches. These algorithms exchange messages between nodes in the graph, or respectively between nodes and factors in a factor graph, until a convergence criterion is met. Probably the most famous algorithm in this group is *Belief Propagation (BP)* [177]. BP is exact on tree structured problems and has also shown good performance on loopy graphs [158]. Similar to VE and JT, BP can easily be adapted to compute max-marginals, i.e., to solve MAP inference problems, which is referred to as max-product BP as opposed to sum-product BP for marginal inference. We will describe BP in Section 2.4.1 in detail because it is of major importance for this thesis. There exists an extension to BP for Gaussian MRFs as well which is referred to as *Gaussian BP (GaBP)* [246]. It was shown that running GaBP on specific graphs amounts to running a power iteration approach. We will shed more light into this connection in Chapter 5 because we will use this power iteration method for running Label Propagation, a graph labeling algorithm, in a large-scale labeling task.

Variational Inference approaches approximate a complex target distribution by a simpler distribution for which running inference is tractable. The goal is then to find a distribution from a family of simpler distributions that is as close as possible to the original one. Hence, the task of inference can be seen as an optimization problem where the objective is the distance between the true distribution and the approximation. Probabilistic queries are then answered by using the simpler distribution as a proxy. Variational methods include approaches such as

Expectation Propagation [150] and *Mean Field* approximations [105]. However, also BP was later understood to fall into this paradigm and described in terms of the Bethe approximation to the free energy [253]. In general, variational inference subsumes a lot of different algorithms and ideas, and as Jordan et al. [105] state “there is as much *art* as there is *science* in our current understanding of how variational methods can be applied to probabilistic inference”. Therefore, a full coverage of variational inference is beyond the scope of this introduction and we refer the reader to [105] and [124, Chapter 11] for more details.

For approximate inference algorithms it is worthwhile to explicitly point out several MAP inference algorithms. For example, Boykov et al. [25] use graph cuts to find minimal energy assignments for a class of MRFs which are frequently found in computer vision problems. The MAP problem for multinomial random variables has an equivalent formulation as an integer linear program over the marginal polytope $\mathcal{M}(G)$ (see Equation (2.4)). Of course, it has the same complexity as the original problem and hence it is intractable. The relaxation of this problem does not require the indicator variables to be integer anymore. Nevertheless, these linear program relaxations come with the benefit that solutions are guaranteed to be optimal if the solution found is integer. Although the linear program can be solved in polynomial time, it is often impossible to use standard solvers due to the huge number of constraints necessary to specify the problem. Therefore, several approaches have been introduced that make use of the specific structure of the linear program for MAP.

One example of such an approach is *Tree-Reweighted BP (TRWBP)* [239] which is a message passing algorithm akin to max-product BP. However, the messages are slightly altered and extended by edge-appearance probabilities. If all of these probabilities are one, then TRWBP reduces to standard BP. The intuition of TRWBP is to reformulate the original MAP problem into a convex combination of tree-structured problems. Each subproblem is represented as a spanning tree and the appearance probability of each edge is based on the distribution over the trees. We have already mentioned above max-product BP and its correctness for tree-structured problems. Therefore, TRWBP can be seen as a combination over a tractable subclass of problems. TRWBP is especially interesting because it finds the same solutions as a linear program relaxation and hence connects a well known message passing algorithm in the spirit of BP with linear programs.

A comparison of TRWBP and CPLEX, a well known yet commercial linear program solver, can be found in [251]. Yanover et al. show that TRWBP is not only faster than CPLEX for many examples, but also capable of solving large instances which CPLEX cannot handle. Lastly, there is also a sum-product variant of TRWBP presented by Wainwright et al. [238]. Yarkony et al. [252] present a TRWBP variant that is based on covering trees (CT). By using covering trees instead of spanning trees, the number of parameters in TRWBP is reduced while the same bounds on the MAP are achieved. However, it is necessary to introduce copy-nodes which have to be aligned by means of optimization to obtain a consistent solution.

As described, the CT-based approach is capable of reducing the number of required parameters compared to the original TRWBP by using a single tree instead of a distribution over spanning trees. Globerson and Jaakkola [70] present with *Max Product Linear Programming (MPLP)* an approximate MAP inference algorithm that also relies on the linear program relaxation but is parameter free and still guaranteed to converge. It is again a message passing algorithm and in its structure similar to max-product BP. It can be seen as block coordinate descent in the dual of the linear program relaxation. In each iteration, block coordinate descent fixes all variables except for a subset and optimizes over this subset.

Approaches such as TRWBP and covering trees can see seen as two instances of a more general paradigm. The idea of decomposing the original MRF into tractable sub-structures can be found in various other inference and learning approaches as well, and is the fundamental idea in *dual decomposition* [212], also known as *Lagrangian relaxation*. Here, the original problem is decomposed and inference is run repeatedly on the sub-problems. The local solutions to each sub-problem are then combined into one global solution.

Instead of a decomposition of the original problem, we can also further constrain the feasible joint probability distributions in $\mathcal{M}(G)$. By doing so, we obtain approaches such as the likelihood maximization approach in [129]. We will return to this algorithm in Section 3.3 where we will explain it in detail and also describe the message updates precisely.

So far, we have made roughly the distinction between marginal and MAP inference algorithms. However, there exist also approaches that somewhat combine the ideas of both inference types. For example, the *Perturb-and-MAP* random fields by Papandreou and Yuille [175]. In this approach, noise is injected into the factors and afterwards MAP inference is run. By doing this in multiple rounds, uncertainty is induced into the MAP solutions and the process approximates Gibbs sampling. Hence the results can be used for parameter learning and probabilistic reasoning. After this brief overview on inference algorithms, we will now summarize some of the ideas that have been proposed to combine logics and PGMs. These template languages describe PGMs compactly and also promote the concept of lifted inference which will be explained afterwards.

2.3 Probabilistic Relational Models

We will now describe a class of approaches for defining and representing PGMs. Most of them can be understood as a template language for PGMs based on logical rules. These *relational PGMs* do not specify dependencies on the object level, i.e., propositional level, but instead model the dependencies on the object-type level, i.e., first-order level. This means that variables in PGMs get typed and objects of the same type get the same parameters (parameter tying). Having these models and languages at hand, it will be much easier to describe certain problem classes later on. Many of these models can be related to the field of SRL, and Getoor and Taskar [66] give an excellent introduction to SRL. Getoor and Taskar summarize the necessary background on graphical models and discuss several approaches based on directed and undirected models. Another more recent overview of several relational probabilistic models is presented in [121]. Kimmig et al.'s survey is centered around the general concept of parametric factors, i.e., *par factors*, and the corresponding *par factor graphs*. Par factor graphs [180] generalize the concept of factor graphs (see Section 2.1.1) by defining potentials over vectors of random variables. Additionally, a set of constraints can be defined on the grounded random variables for each par-factor. Therefore, par factor graphs also relate to the ideas of parameter tying because a set of grounded variables shares the same potential function. We will now briefly discuss several of these relational approaches and explain a specific one in greater detail, namely *Markov Logic Networks (MLNs)*, as these will be used throughout this thesis for different experiments.

As before, we can distinguish directed and undirected approaches, in addition relational extensions to Dependency Network have been proposed as well. Similar to Section 2.1, we will start off by discussing relational undirected models. Before doing so, we will briefly provide a minimal background on first-order logic because the basic principles of logic are required to formulate and understand approaches such as MLNs. We have seen an example of logic already in Section 1.1. Propositional logic was used to define solutions to Latin squares. A drawback of propositional logic is the fact that we had to define rules for each cell separately. Instead, first-order logic aims at reasoning about groups of objects of the same type. Due to this, one often speaks about "lifting"[1] a propositional model to the first-order case.

We will restrict our introduction to first-order logic to the terminology that is required for defining relational PGMs, which is essentially based on the contents given in [188, Section 3]. A first-order formula is constructed using constants, variables, functions, and predicates. However, here we will restrict ourselves to finite, function-free first-order logic. *Constants* represent objects in our domain and are typed. For example, one type of objects could be "person" with objects `Anna`, `Bob`, and so on. A logical *variable* `x`, which is also typed, ranges over the domain of objects and can take on values from this domain. *Predicates* define attributes of objects or relations between objects depending on the arity. For example, a unary predicate `smokes` specifies the smoking habits of a single person but `friends` models friendship between two persons. We assume typed predicates where each argument can have a different type.

To construct formulas, we apply logical operators and quantifiers recursively to atoms. An *atom* is a predicate applied to a tuple of terms and a *term* can be a constant or a variable. A *ground term* is a term not containing any variables and correspondingly, a ground atom is a predicate whose arguments are all ground terms. Hence, `x` and `Anna` are terms and `friends(x, Anna)` is an atom, while `friends(Bob, Anna)` is a ground atom. Assuming that F_1 and F_2 are formulas, we have the following operators $\neg F_1$ (negation), $F_1 \wedge F_2$ (conjunction), $F_1 \vee F_2$ (disjunction), $F_1 \Rightarrow F_2$ (implication) and $F_1 \Leftrightarrow F_2$. An atom is a formula and an operator applied to one or more formulas is again a formula. A formula F is *satisfiable* if there exists at least one world in which F evaluates to true. A *world* assigns a truth value to each possible ground atom. Lastly, variables in a formula F can be universally or existentially quantified. An universally quantified formula $\forall \mathtt{x} : F$ is true if and only if F holds for every possible instantiation of `x`. An existentially quantified formula $\exists \mathtt{x} : F$ is true if there exists at least one instantiation of x for which F holds. We have already seen in the introduction of this thesis that propositional satisfiability problems are most often specified in conjunctive normal form. This also holds for first-order formulas and requires rewriting implications and equivalences, but also includes the removal of existential quantifiers, i.e., Skolemization. With this minimal introduction to logic and the basics of probability theory from above, we are now ready to discuss relational probabilistic models.

In the following, we will see different graphical models where variables usually correspond to predicates and model the true state of a logical predicate with probabilities. Although several of the approaches discussed in this section rely on logic, we will start with a logic-free undirected relational approach called *Conditional Random Fields (CRFs)* [130]. CRFs can be seen as a template based approach for constructing MRFs. CRFs apply the idea of parameter tying and explicitly divide the variables into latent and observed variables. This separation of the variables leads to discriminative training which optimizes the conditional likelihood of the latent labels given the observed features. To name a few examples, CRFs have successfully been used in various sequence labeling tasks such a named entity recognition and part of speech tagging where parameters are tied over the vocabulary.

[1]Lifted inference originates from the idea to run inference on relational probabilistic models, without fully grounding the first-order model.

Relational Markov Networks (RMNs), introduced in [221], extend MRFs to work with relational data and formulate dependencies on the relational level. To this end, Taskar et al. use *relational clique templates* which are defined over a set of typed object variables, a constraint over the objects formulated as boolean formula, and a set of attributes of the objects. These clique templates can be mapped to relational queries, such as SQL-queries, and are used to construct the underlying propositional MRFs. Each clique template has a potential and corresponds to a set of cliques in the MRF. Inspired by CRFs, RMNs also use discriminative learning for conditional MRFs. During training, the conjugated gradient method is combined with BP to approximate the gradient. Datasets such as WebKB, a corpus of hypertext documents, are well suited to be represented by RMNs due to their relational structure. RMNs allow collective classification, for example of documents, in this dataset.

A different approach to relational MRFs based on logical rules are MLNs [188]. Instead of SQL, an MLN uses *first-order logic (FOL)* to define a set of rules which are then used to instantiate a ground MRF. Compared to classical *Inductive Logic Programing (ILP)* approaches, the formulas in an MLN are weighted and factors in the corresponding ground MRF are constructed based on the weighted formulas. Section 2.3.1 will explain MLNs in detail.

We will now shift our focus to directed relational models. *Relational Bayesian Networks (RBNs)*[2] [67, 59] use relational schemata over classes and relations to describe skeletons with fixed and probabilistic attributes. An RBN then defines a probability distribution over all possible instantiations of a skeleton which is again a ground BN and, hence, the conditional probability tables are defined over the parent set of a variable. In RBNs however, parents can also be aggregates of random variables, such as the mean value of a set. Friedman et al. apply standard inference and learning algorithms from the propositional case to the instantiated RBNs. However, in parameter learning the concept of parameter tying is used to ensure that each attribute is identically parameterized and additional constraints are imposed on the structure learner to guarantee an acyclic network. Neville and Jensen note in [167] that RBNs do not learn autocorrelation well because dependencies between variables of the same type would induce cycles in the grounded network.

In *Bayesian Logic Programs (BLPs)* [113, 111] one component consists of Bayesian clauses over Bayesian predicates in *Prolog* and the second component is a set of conditional probability tables associated with these Bayesian clauses. The domain of a Bayesian predicate does not necessarily have to be binary but instead can have any finite range. Next to the conditional probability tables, combining rules are required if a ground atom is head in several clauses. These combining rules return a single combined conditional probability table. The BLP can now be used to derive a directed dependency graph where nodes correspond to ground atoms. Edges are added based on the clauses whenever one node corresponds to the head of a clause and the other node is contained in the body. This dependency graph represents a BN and hence defines the joint probability distribution of the BLP.

We have already discussed the advantages and disadvantages of MRFs and BNs in the previous section which motivated Heckerman et al. to introduce DNs. *Relational Dependency Networks (RDNs)* [167] are an relational extension of Heckerman et al.'s DNs which use relational probability tables. Each local conditional probability distribution is learned with

[2]The name Relational Bayesian Networks was originally introduced by Jaeger in [100]. However, this formalism does not act as a template language for the instantiation of ground BNs. Instead, the states of the random variables correspond to instantiations of relations. So this is not immediately a probabilistic relational model in the spirit of the other approaches mentioned in this section.

$$\lambda_1 : \texttt{smokes(x)} \Rightarrow \texttt{cancer(x)}$$
$$\lambda_2 : \texttt{friends(x,y)} \Rightarrow (\texttt{smokes(x)} \Leftrightarrow \texttt{smokes(y)})$$

Table 2.1: Formulas used in the Smokers-and-Friends-MLN.

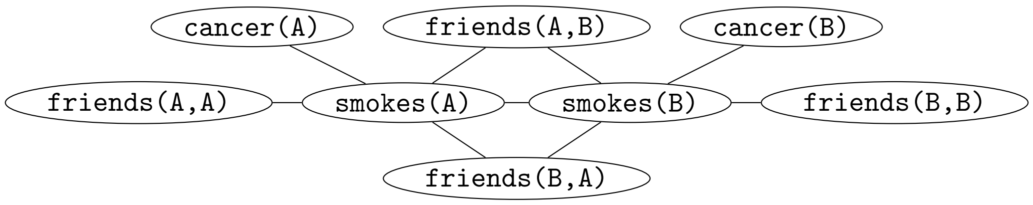


Figure 2.3: Smokers-and-Friends MLN.

a relational learner such as a Relational Bayesian Classifier [169] or a Relational Probability Tree [168]. This local pseudo-likelihood optimization makes learning in the relational case tractable. Again, RDNs only approximate the joint probability distribution and are not guaranteed to be consistent. Similar to the non-relational case, a consistent RDN is a model for which the conditional probabilities can be derived from the joint distribution using rules of probability theory. Consistent RDNs are equivalent to the RMNs mentioned above. In RNDs, inference is also done via Gibbs sampling on the grounded graph.

The concepts of relational PGMs are better understood with help of an example, therefore we will now pick up on MLNs, discuss these in detail, and present a small running example.

2.3.1 Markov Logic Networks

One of the most well established frameworks in the area of probabilistic models are MLNs [188][3]. MLNs are probabilistic relational models that are defined by weighted first-order formulas. More precisely, an MLN is a set of pairs (F_i, w_i) where F_i is a first-order formula and $w_i \in \mathbb{R}$ is the formula's weight. In addition, we have a set of typed constants $C = \{C_1, \ldots, C_{|C|}\}$. We can now use the MLN to define an MRF. We add one binary variable to the MRF for every ground predicate in the MLN, e.g., `smokes(Anna)`. The value of the random variable models the truth state of the ground predicate. Due to the finite set of constants, this results in a finite set of random variables. Furthermore, we add one potential ϕ_i to the MRF for each possible grounding of a formula, e.g., `smokes(Anna)` $\Rightarrow$ `cancer(Anna)`. The potential is defined as $\exp(w_i \cdot g_i)$ where g_i is a feature function for the formula f_i. Hence, g_i is defined over all ground predicates appearing in the formula and evaluates to 1 if the formula is satisfied and to 0 otherwise. Essentially, MLNs serve as a template engine for MRFs that allow to easily model independencies based on logical rules. The resulting joint probability of the MRF is a log-linear model and looks as follows:

$$p(\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_i w_i \cdot n_i(\mathbf{x})\right) ,$$

[3]A reference implementation of MLNs, including various inference and learning algorithms, is available online: `alchemy.cs.washington.edu`

λ_1 : `smokes(x,t)` $\Rightarrow$ `cancer(x,t)`

λ_2 : `friends(x, y, t)` $\Rightarrow$ (`smokes(x, t)` $\Leftrightarrow$ `smokes(y, t)`)

λ_3 : `cancer(x, t)` $\Leftrightarrow$ `cancer(x, succ(t))`

λ_4 : `smokes(x, t)` $\Leftrightarrow$ `smokes(x, succ(t))`

λ_5 : `friends(x, y, t)` $\Leftrightarrow$ `friends(x, y, succ(t))`

Table 2.2: Formulas used in the Smokers-and-Friends DMLN.

with $n_i(\mathbf{x})$ corresponding to the number of true groundings of formula F_i. We will now give a small toy example of an MLN that will be used below for experiments on synthetic datasets.

Example 2.1. *One of the most prominent toy examples is the Smokers-and-Friends MLN. Intuitively, it models a social network containing friendships between persons and their smoking habits. It also describes the influence of friendships on smoking habits and the implication of smoking causing cancer. The minimal rules that define this MLN are depicted in Table 2.1. The MRF that is obtained by grounding this MLN in the presence of two persons only is shown in Figure 2.3. Given a partial observation of the predicates, MAP inference asks for the assignment to the remaining variables. A marginal query such as* $p($`cancer(Anna)`$= 1)$ *returns the probability that "Anna" has cancer given a specific network.*

The truth state of a predicate may also depend on the time. For example, a person is usually not born having cancer but instead sickens while aging. Similarly, a smoker can quit smoking in the future. Both examples require the predicates to be extended by a discrete time step, e.g., `smokes(x, t)`. Although the additional time step t allows us to specify the predicates for specific points in time, we still need another mechanism that allows us to connect predicates of two subsequent time steps. To achieve this, a successor function `succ(t)` is added which maps point t to $t + 1$. The resulting framework is called *Dynamic MLNs (DMLNs)* and the Smokers-and-Friends MLN from above can be extended as follows.

Example 2.2. *The introduction of a time step and the successor function allows us to formulate the Smokers-and-Friends MLN over time. The new rules are depicted in Table 2.2. Besides the rules from the static MLN extended by a time step (1 & 2), we have three new rules based on the* `succ`*-function. These rules define the behavior of the predicates over subsequent time steps.*

In Chapter 3, several experiments will be based on MLNs. Additionally, we will introduce a graph construction algorithm for Label Propagation in Section 5.1 that is based on weighted logics similar to MLNs. We will now briefly discuss inference and learning in MLNs.

Inference and Learning in MLNs

The most straightforward way to run inference in MLNs is to ground the network and then run a standard marginal or MAP inference algorithm from the MRF literature, such as BP or Gibbs sampling. Interestingly, MAP inference in MLNs can equally be achieved by solving a weighted MAX-SAT problem. However, several other algorithms specifically designed for MLNs have been introduced in recent years as well. One of the most well known ones is MC-

SAT [181] which combines ideas from satisfiability solving with MCMC techniques. Technically speaking, MC-SAT makes use of slice sampling and samples a new state with the help of SampleSAT [243]. An interesting MAP inference approach is the cutting plane based algorithm introduced by Riedel [190]. Inspired by cutting plane methods from Operations Research, the algorithm avoids instantiating the full ground MLN and instead only adds ground formulas as long as the current solution can be further improved. By doing so, the corresponding MRF remains often much smaller and inference is more efficient. This cutting plane algorithm is already one approach avoiding the instantiation of the entire MRF. Ground MLNs quickly become very large and hence staying on the first-order level is desirable. Lifted inference, which will be explained in the next section, tries to avoid shattering the model as much as possible to remain on the lifted level and is therefore a popular technique to speed up inference algorithms for MLNs. Looking from a different angle, lifted inference tries to exploit symmetries by avoiding to ground identical propositional parts and grouping together indistinguishable variables and formulas. Evidence, however, can lead to models where lifted inference reduces completely to ground inference because evidence destroys symmetries in many situations. Lifted inference and handling evidence in PGMs will be of major interest throughout different chapters of this thesis. A different approach to speeding up inference in MLNs is the Frog algorithm presented by Shavlik and Natarajan [200]. Potentially, Frog could be combined with lifted inference and is particular useful when subsequent queries for the same MLN are necessary. The algorithm specifically looks at the evidence and by doing so, reduces the size of the grounded MLN. It exploits the fact that the count of formulas satisfied by the evidence remains unchanged for subsequent queries. Compared to lifting, a key advantage of this approach is its independence from the inference algorithm and it can be seen as a general preprocessing step.

Running inference assumes that the structure and parameters are given. However, in many situations it is not trivial to choose appropriate weights. In the extreme case, it is even impossible to specify the structure, i.e., the logical rules, of an MLN. Instead, we are only given a relational database representing a single mega training example. Based on this training database, we can use classical ILP techniques to infer rules for our MLN. For example, in early work on MLNs, Claudien [45] was used by Richardson and Domingos [188] to learn a set of first-order clauses. However, it was shown by Kok and Domingos [123] that directly optimizing a pseudo log-likelihood score often gives better results than a purely logic-based approach. Their approach starts with all unit clauses and then adds additional literals to the clauses as long as the log-score improves. This is a very common approach in MLN structure learning. A search algorithm proposes new candidate formulas by altering the current ones, and a score function determines which of the new formulas to add to the MLN. Recently, there has also been observed a surge in structure learning approaches based on decision and regression trees. Decision trees [118, 140] have the advantage that learning is done very fast and each path to a leaf can be seen as a feature combining different variables and values. In a similar line of work, Khot et al. [119] use gradient tree boosting to learn the parameters and the structure of MLNs. Gradient tree boosting elegantly combines parameter and structure learning with the help of regression trees. We will follow the idea of gradient tree boosting in Section 7.3 where we will show that pseudo log-likelihood optimization and tree-based learning also work well for learning probabilistic count models based on Dependency Networks. We already touched upon the idea of lifted inference and indicated how lifting can reduce run time in PGMs. We will now review several different lifted inference approaches and present Belief Propagation in detail.

2.4 Lifted Inference

Although approximate inference techniques are already quite efficient, many graphical models contain a lot of symmetries which are not exploited by those algorithms. Recent years have witnessed a surge of interest in lifted probabilistic inference [110, 121], handling whole sets of indistinguishable objects together. The terms "lifting" and "lifted inference" are being used across different fields in computer science and have different meanings within each field. Here, we refer to *lifting* as the general concept of compressing a model, e.g., an MRF, and applying a possibly modified inference algorithm on the lifted model. In most cases, the compression is a pre-processing step and can be achieved by different means. One should note however that many lifted inference algorithms require an adaption of the ground algorithm to run on the lifted level. Taking a slightly different approach, there are lifted inference algorithms that integrate the compression of the model into the actual inference algorithm (see Section 2.4.2 for such *online* lifted approaches).

Lifting inference essentially follows an upgrade paradigm. That is, most lifted inference approaches take a well known propositional inference algorithm and adapt it to run on a more compact representation that exploits symmetries in the graphical structure of the original model for a given query at hand. Especially for relational models, such as MLNs, lifting has been proven successful in rendering large, previously intractable, probabilistic inference problems quickly solvable by handling whole sets of indistinguishable objects together. Generally, there are different lines of work and we mainly distinguish here two of them. Lifting can be done "top-down" — starting at the most lifted level and shattering the lifted model against itself and the evidence until a modified inference approach can run safely on the model [180, 46, 209] — or in a "bottom-up" fashion — starting at the propositional level and lifting that model to obtain a representation as compact as possible [114, 171]. As we will see, the advantage of bottom-up lifting is that these approaches can also be applied to problem formulations for which a first-order representation does not exist. For example, images in computer vision are presented in a propositional form but still may contain symmetries that are desirable to be exploited during inference. On the other hand, for relational representations the model has to be grounded first which can already result in intractable problem sizes. Top-down algorithms handle these models more elegantly but struggle with problems naturally presented in propositional form. Fundamental to all lifting approaches is the concept of *counts*. These counts allow a lifted variant to simulate the calculations of its ground version and summarize the symmetries in the model. However, we will also see in Chapter 5 that counts can be integrated in the model right away and hence it is possible to use a vanilla version of the ground algorithm. Several of the inference algorithms discussed in the previous section also have lifted versions. As before, we will start with the description of lifted versions of exact inference algorithms and then continue with the approximate ones.

Exact Lifted Inference

Section 2.2.1 described VE as an example of an exact inference algorithm. In [180, 46, 149, 219] lifted versions of VE have been presented, also known as First-Order Variable Elimination (FOVE). For example, Poole [180] shows how to reason over a group of objects instead of only individual ones. All of these FOVE approaches can be considered as top-down approaches. Using bisimulation to compress the underlying graph structure, Sen et al. [197] follow a bottom-up approach and focus on queries in probabilistic databases. Another class of exact algorithms are knowledge compilation approaches such as Gogate and Domingos [73] and Van den Broeck et al. [230]. For example, Van den Broeck et al. compile weighted first-order theories, such as MLNs, into first-order deterministic decomposable negation normal form (d-DNNF) circuits which can then be used in lifted weighted model counting. However, compared to the propositional case, conditioning, i.e., integrating observed knowledge, of first-order d-DNNFs is more complex and not always feasible. Van den Broeck and Davis [228] prove that conditioning on propositions and unary relations can be done efficiently while conditioning higher-arity relations is #P-hard. These exact inference approaches are often extremely complex, so far do not easily scale to realistic domains, and hence have only been applied to rather small and artificial problems. Therefore, we will now turn our attention to approximate lifted inference algorithms which are also of greater interest for this thesis.

Approximate Lifted Inference

Similar to [197], Sen et al. [198] introduced an approximate lifting approach based on approximate bisimulation which is also based to the VE algorithm. It essentially groups together random variables if they have similar computation trees, the tree-structured "unrolling" of the underlying graphical model rooted at the nodes. The simplest and most efficient lifted inference approaches are lifted BP approaches which easily scale to realistic domains. Motivated by Jaimovich et al. [101], Singla and Domingos [209] developed the first lifted BP variant tailored towards MLNs. Subsequently, Kersting et al. [114] generalized Singla and Domingos' approach to any factor graph over finite random variables. Similar to Sen et al., lifted BP approaches also group together random variables that have identical computation trees, but now use the more efficient *Lifted Belief Propagation (LBP)* update equations on the lifted factor graph. We will discuss the lifted versions of BP together with its ground algorithm in Section 2.4.1 in greater detail.

While initial work on lifted inference was mainly focusing on lifted VE or lifted BP, approaches lifting MCMC algorithms have been recently presented as well. This includes in particular lifted Gibbs sampling. Motivated by the fact that inference in graphical models can be seen as a special case of probabilistic theorem proving, Gogate and Domingos [73] provide an algorithm for approximate probabilistic theorem proving based on sampling. Gogate and Domingos reduce the probabilistic theorem proving to lifted weighted model counting and show how to combine the algorithm with importance sampling. This yields an unbiased estimator for the exact ground weighted model count. Gogate et al. [74] further improve on this lifted importance sampling approach by exploiting additional symmetries and introducing an informative proposal distribution.

Niepert [171] presents a different approach to lifted MCMC by introducing *orbital Markov chains*, and an *orbital Gibbs sampler* in particular. To find symmetries in a PGM, an undirected auxiliary graph is constructed and colored. It is assumed that the PGM is defined based on weighted clauses and the auxiliary graph is then constructed based on properties of these weights and clauses. It is shown that the automorphism group of the colored graph allows to read off the permutation group of the original PGM which defines the symmetries in the model. The permutation group induces an orbit partition and it is shown that variables, respectively factors, in the same orbit have similar properties as clustered variables, respectively factors, in lifted BP. More precisely, they send and receive identical messages and also have identical marginal distributions. However, the approach that is taken by lifted BP to partition the nodes does not return an orbit partition. In Section 2.4.1, we will describe lifted BP's partitioning algorithm in greater detail.

Another lifted Gibbs sampler was proposed by Venugopal and Gogate [234] who lift the blocked Gibbs sampling. Compared to a regular Gibbs sampler, the blocked Gibbs sampler groups variables together in the ground case, i.e., the blocks, and samples all variables from a given block instead of single variables. Venugopal and Gogate form clusters of atoms such that exact probabilistic theorem proving in each cluster is feasible. It is also shown that the clusters can be constructed in such a way that the lifting rules of probabilistic theorem proving can be applied, and hence, lifting in the sampling approach is achieved.

A recent lifted sampling approach for MLNs based on importance sampling was presented by Venugopal and Gogate [236]. To generate samples from the proposal distribution, Venugopal and Gogate use compressed MLNs [235], which avoid grounding the full MLN. To avoid the full grounding in the computation of the importance weight as well, a second sampler is used to sample a finite number of groundings for each formula.

Van den Broeck and Niepert [229] extend the idea of sampling based lifted inference to models with only approximate symmetries. A mixture of Markov chains is presented where the base, i.e., ground, Markov chain is mixed with a chain operating on approximate symmetries. Van den Broeck and Niepert discuss different heuristics for finding approximate symmetries which, however, is still an ongoing research question.

There is also a variety of lifted variational inference approaches. For example, Choi and Amir [38] present relational hybrid models where random variables can either be discrete or continuous. To compute marginal probabilities in these models, a lifted relational variational inference algorithm is used that first finds a variational approximation of the model and then uses latent variable elimination to calculate the posterior distribution. Alternatively, a lifted MCMC algorithm is suggested when the variable elimination approach is assumed to require too much time.

As in the case of ground inference, we will now also specifically discuss a few of the lifted MAP inference algorithms. The lifted MAP approach presented by Apsel and Brafman [9] identifies groups of variables that are assigned the same value in a solution and exploits the fact that these variables can be merged. Consequently, only one of them has to be considered. This idea is very similar to the idea of lifting a factor graph as in lifted BP, however, Apsel and Brafman approach the problem of finding the clusters differently and particularly work on the relational level, i.e., top-down.

Bui et al. [30] follow a bottom-up approach to MAP inference and use the theory of group actions to formally define symmetries in graphical models. Bui et al. exploit the symmetries found in approximate variational MAP inference, e.g., local linear program relaxations.

Mladenov et al. [151] presented the first approach to lifted linear programming which is particularly interesting because it allows to run an off-the-shelf linear program solver. In a follow up work, Mladenov et al. [152] also showed how to obtain lifted message passing based on linear programming by reparameterization of ground MRFs. It is shown how double edges prevent the application of standard message passing algorithms on lifted MRFs in the case of linear programming relaxations and the idea of *linear programming equivalence* between MRFs is introduced. Two MRFs are linear programming equivalent if they share the same lifted graph. Given an MRF to solve, the proposed approach then tries to find a more compact linear programming equivalent MRF by means of reparameterization. This is essentially done by resolving the double edges in the lifted program, resulting in a compact reparameterized ground MRF which can be solved with off-the-shelve ground solvers.

Instead of a linear program relaxation and similar to the ideas in [190], Noessner et al. [172] compile MAP queries in MLNs to integer linear programs and apply cutting plane inference. Additionally, cutting plane aggregation is used to exploit symmetries in the model by aggregating ground formulas and compiling the corresponding constraints more compactly. Essentially, Noessner et al. group clauses that have the same weight and share sub-clauses. Next to linear programs and integer linear programs, Sarkhel et al. [194] present an approach where the task of MAP inference in MLNs is encoded as an integer polynomial program. Each variable in the program represents an assignment to a set of indistinguishable variables. This clustering can reduce the search space and hence decrease the run time of the solver. The integer polynomial program is solved by conversion to an integer linear program, followed by running probabilistic theorem proving for MAP inference.

Besides the lifted MAP algorithms seen so far, we will present a novel approach based on message passing in Section 3.3. This new lifted inference algorithm is based on the likelihood maximization approach presented in [129]. Our approach can be seen as a bottom-up approach which is not tied to any of the relational model frameworks and can be run on any discrete MRF. Nevertheless, we face the same issues with evidence in lifting akin to many of the previously described approaches. Generally, handling evidence plays a central role in lifted inference. With more incoming evidence, the models typically become less symmetric and lifting potential decreases. In Section 3.2 and Section 3.3, we will present different approaches to handle evidence in lifted inference. An interesting point of view on evidence and lifting is presented in [201]. As by Sheldon et al. described, aggregated evidence on all objects, e.g., in the form of contingency tables, can be seen as lifted evidence. Sheldon et al. refer to it as "evidence at the population level".

When being used as a subroutine, lifted inference can also speed up parameter learning [4, 232] because the ground inference algorithm can be replaced by a more efficient lifted inference algorithm. If one is also required to learn the model structure, it can be beneficial to consider only liftable models during structure learning as suggested in [233]. After this overview of several recent lifted inference approaches, we will now describe one particular algorithm in greater detail, namely *Lifted Belief Propagation (LBP)*. LBP can be seen as the origin for several algorithms developed in the work at hand.

2.4.1 Belief Propagation and Its Lifted Versions

One of the most well known message passing approaches is BP [177] which is also known as the *sum-product*-algorithm. BP is a simple message passing algorithm for marginal and MAP inference in MRFs. In the case of MAP inference, it is also referred to as *max-product* BP. Given a probability distribution over some random variables as in Equation (2.1), sum-product BP computes the single node marginal probabilities and the marginal probabilities over the variables in each clique. Similarly, the algorithm can also compute the max-marginals of the variables and factors with only slight modifications to the messages. The algorithm can be described elegantly via messages passing in a factor graph [128]. The computed marginal probability distributions are exact and BP runs in linear time if the factor graph is cycle-free, i.e., the graph is a tree. Otherwise, BP is still a well-defined algorithm that approximately solves these problems. Running BP on a cyclic graph is often referred to as *Loopy BP*. However, in this thesis, we do not explicitly distinguish between running BP on a tree or a cyclic graph, hence, we use the term BP for its variant on trees, as well as the loopy approximation. In cases where we need to explicitly distinguish both version, we will make it clear in the context.

Although BP has no guarantees of convergence, nor of giving the correct results, in practice it often does perform well, and can be much more efficient than other methods [158]. We will now describe BP in terms of operations on a factor graph.

BP works by sending messages between variable nodes and their neighboring factor nodes, and vice versa. The message from a variable i to a factor a is

$$\mu_{i\to a}(x_i) = \prod_{b\in \mathrm{nb}(i)\setminus\{a\}} \mu_{b\to i}(x_i)$$

where $\mathrm{nb}(i)$ is the set of factors i appears in. The message from a factor to a variable is

$$\mu_{a\to i}(x_i) = \sum_{\neg\{i\}} \left(f_a(\mathbf{x}_a) \prod_{j\in \mathrm{nb}(a)\setminus\{i\}} \mu_{j\to a}(x_j) \right) \tag{2.8}$$

where $\mathrm{nb}(a)$ are the arguments of factor a, and the sum is over all configurations of $\mathbf{x}_a$ having x_i fixed, denoted as $\neg\{i\}$. The messages are usually initialized to 1. Upon convergence, or a fixed number of iterations, one can calculate the unnormalized belief $b(\cdot)$ of each variable X_i based on these messages:

$$b(x_i) = b_i = \prod_{a\in \mathrm{nb}(i)} \mu_{a\to i}(x_i)\ .$$

In a similar way, one can calculate $b(\mathbf{x}_a) = b_a$, i.e., the marginal beliefs for the variables involved in factor a. Evidence, i.e., variables for which the state is known in advanced, can directly be incorporated into the factors. We set $f_a(\mathbf{x}_a) = 0$ for states $\mathbf{x}_a$ that are incompatible with the evidence. Algorithm 1 shows how these message updates are used in the BP algorithm. The messages are iteratively computed in a fixed order for a previously defined maximum number of iterations. If the messages converge earlier, i.e., the maximum difference in the messages falls below a threshold, the algorithm is terminated.

If one is interested in MAP inference, one can simply change the formula sent from factor to a variable as follows:

$$\mu_{a\to i}(x_i) = \max_{\neg\{i\}} \left(f_a(\mathbf{x}_a) \prod_{j\in \mathrm{nb}(a)\setminus\{i\}} \mu_{j\to a}(x_j) \right)$$

The only difference is the replacement of the sum by the max-operator. For this reason, the message equations are also referred to as *max-product.* Instead of the marginals of the distribution, the max-marginals are now calculated.

As mentioned above, BP is not necessarily correct on graphs with cycles. We will now just briefly comment on known theoretical insights on BP's quality and approaches to increase its approximation quality. It has been shown that BP can only converge to stationary points of the Bethe free energy. To increase the accuracy of BP, Generalized Belief Propagation has been introduced [253] which allows to converge to other free energy approximations as well. For more details, we refer the reader to [253] and the references in there. One should also note that BP does not only provide us with the pseudo-marginals. Resulting form Bethe's free energy approximation [254], it is also possible to calculate an estimate on the partition function Z based on BP's beliefs.

```
Input: Factor graph $G$, maximum iterations $T$, convergence threshold $\pi$
Output: Vector $\mathbf{b}$ containing variable beliefs $b(x_i)$ and factor beliefs $b(\mathbf{x}_a)$
while $t < T$ and $\epsilon > \pi$ do
    for $a \leftarrow 0$ to $|\mathbf{F}| - 1$ do
        for $i \in \mathrm{nb}(a)$ do
            calcNewMessage($a$, $i$);                    /* $\mu_{i \to a}$ and $\mu_{a \to i}$ */
        end
    end
    for $a \leftarrow 0$ to $|\mathbf{F}| - 1$ do
        for $i \in \mathrm{nb}(a)$ do
            updateMessage($a$, $i$);                     /* Possibly apply damping */
        end
    end
    $\epsilon \leftarrow -\infty$;        /* Stores the maximum difference between iterations */
    for $i \leftarrow 0$ to $|\mathbf{X}| - 1$ do
        $b_i^{new} \leftarrow$ beliefV($i$);
        $\epsilon \leftarrow \max(\epsilon, \|b_i^{new} - b_i\|_\infty)$;
        $b_i \leftarrow b_i^{new}$
    end
    for $a \leftarrow 0$ to $|\mathbf{F}| - 1$ do
        $b_a^{new} \leftarrow$ beliefF($a$);
        $\epsilon \leftarrow \max(\epsilon, \|b_a^{new} - b_a\|_\infty)$;
        $b_a \leftarrow b_a^{new}$
    end
    $t \leftarrow t + 1$;
end
return $\mathbf{b}$
```

Algorithm 1: BP with parallel updates.

Besides the fact that BP is not correct on loopy graphs, it is also known that BP has problems with near deterministic factors as they induce long distance dependencies. Contrary to that general statement, BP has been successfully applied to SAT problems [126] and solved hard random k-SAT problems. The key to the success in such problem is to effectively cope with the problem of convergence. One particular simple but effective approach to improve the convergence behavior of BP is *damping*. Here, the message in iteration t is replaced by a weighted average of the messages from the previous iteration and the current one, i.e.,

$$\mu_{i \to a}(x_i)^{(t)} = (1 - \epsilon) \cdot \mu_{i \to a}(x_i)^{(t)} + \epsilon \cdot \mu_{i \to a}(x_i)^{(t-1)}$$

This slight modification of the messages often prevents oscillation and helps in cases with near-deterministic factors [126].

There are several other aspects of BP that have an impact on its convergence. First, the update schedule of the messages plays an important role in convergence. In the *flooding* or *parallel* message protocol, which is often used and well performing, messages are passed from each variable to all corresponding factors and back at each step. I.e., first a pass over all variables is made to compute the messages send to the neighboring factors and in a subsequent

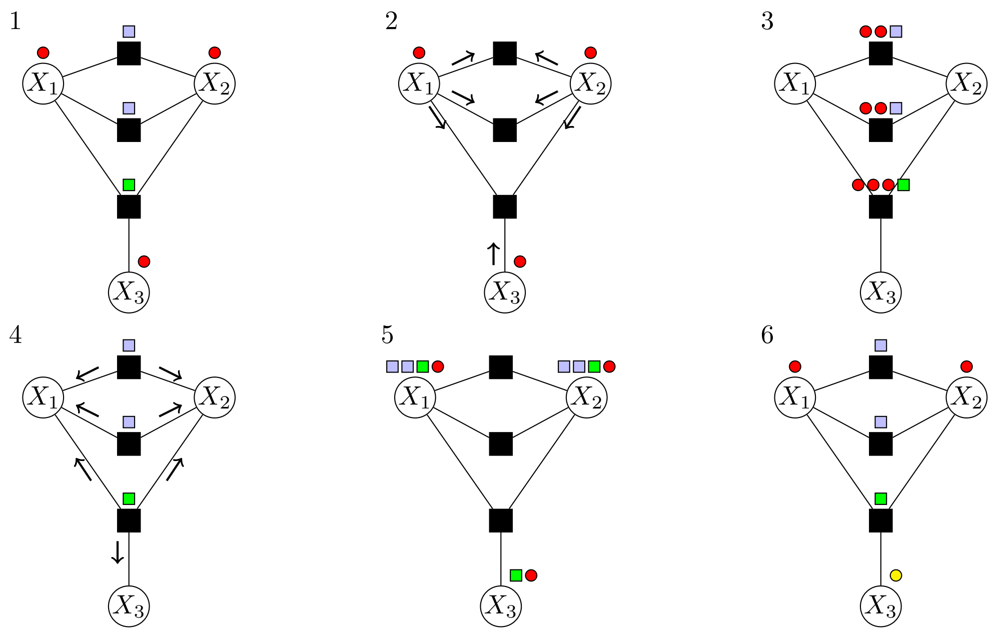


Figure 2.4: Lifting a factor graph without evidence by running *Color Passing (CP)*. From left to right, top to bottom, the steps of running CP on the factor graph in Figure 2.2. The colored small circles and squares denote the groups and signatures produced during the algorithm.

pass over all factors, the messages are calculated the other way around. Here, BP iterates over the variables in a predefined order and the new marginals of the current iteration are not used before the next iteration. In a *sequential* update schedule, the updated messages of the current iteration also affect the calculation of the other messages in the same iteration. In a *sequential random* update schedule, the order of the variables is not predefined and instead randomly chosen in each iteration.

Although BP is already quite efficient, many graphical models produce inference problems with a lot of symmetries which are not exploitable by BP. For example, looking again at the factor graph depicted in Figure 2.2 and assuming that the factors f_1 and f_2 are identical, the graph is symmetric and variables X_1 and X_2 are distributed identically. Technically, this also requires that X_1 and X_2 are exchangeable in the three factors, i.e., the tables of the factors also exhibit symmetries. We will further discuss this issue with an additional example in Figure 2.6. To exploit such symmetries, different lifted versions of BP have recently been proposed [209, 114] that can make use of the knowledge about symmetry in the graph. Here, we focus on *Counting Belief Propagation*, the approach by Kersting et al., because the algorithm directly runs on factor graphs while the approach by Singla and Domingos specifically focuses on MLNs, and hence requires a relational formalism. The factor graph representation allows us to apply the ideas of lifting to a wider range of problems as we will see later. Although the approach by Kersting et al. was originally referred to as Counting BP, we will refer to it as *Lifted Belief Propagation (LBP)* in the following to have a uniform naming scheme for lifted inference algorithms. The acronym LBP was originally introduced in Singla and Domingos's work. Nevertheless, if nothing else is mentioned, we will refer to Kersting et al.'s lifted extension of BP whenever we write LBP because it is more generally applicable.

Let us now illustrate LBP in detail. LBP runs in two steps:

```
Input: Factor graph $G$, evidence $\mathbf{Y}$
Output: Lifted factor graph $\mathfrak{G}$
colorsV $\rightarrow$ initVariableColors($G$, $\mathbf{Y}$);
colorsF $\rightarrow$ initFactorColors($G$);
while colors change do
    signatures $\leftarrow$ [ ];
    for $a \leftarrow 0$ to $|\mathbf{F}| - 1$ do
        signatures[$a$] $\leftarrow$ [ ];
        for $i \in \text{nb}(a)$ do
            signatures[$a$].append(colorsV[$i$]);
        end
        signatures[$a$].append(colorsF[$a$]);
    end
    colorsF $\leftarrow$ updateFactorColors(signatures);
    signatures $\leftarrow$ [ ];
    for $i \leftarrow 0$ to $|\mathbf{X}| - 1$ do
        signatures[$i$] $\leftarrow$ [ ];
        for $a \in \text{nb}(i)$ do
            signatures[$i$].append(colorsF[$a$]);
        end
        signatures[$i$].append(colorsV[$i$]);
    end
    colorsV $\leftarrow$ updateVariableColors(signatures);
end
return createLiftedFactorGraph();
```

Algorithm 2: Color Passing as proposed by Kersting et al. [114]

1. In a first step, which is called *lifting*, LBP automatically groups together nodes and factors that send and receive the same messages during an iteration of BP.

2. Then, it runs a modified BP on the lifted, i.e., compressed, graph.

As shown empirically by Singla and Domingos as well as by Kersting et al., this can significantly speed up inference. More importantly, Kersting et al. have shown that the lifting step can be viewed as a *Color Passing (CP)* approach. This view abstracts from the type of messages being sent and, hence, highlights one of the main contributions of this thesis: *CP can be used to lift other message passing approaches as well.* Before we will show how to lift other message passing algorithms, we will describe CP in more detail.

CP simulates message passing, keeping track of the nodes and factors which send the same messages, and groups these nodes and factors together correspondingly. Specifically, let G be a given factor graph with variable and factor nodes as described in Section 2.1.1. Initially, all variable nodes fall into $d + 1$ groups (one or more of these may be empty) — known states s_1, ... , s_d, and *unknown* — represented by colors. For ease of explanation, we will represent the groups by colored circles, say red. Since we are not dealing with evidence here, all variables receive initially the same color. All factor nodes with the same associated potentials also fall into one group represented by colored squares. For the factor graph in Figure 2.2 the situation

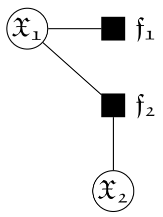


Figure 2.5: The resulting lifted factor graph $\mathfrak{G}$ after running CP as illustrated in Figure 2.4. We use fraktur letters such as $\mathfrak{G}$, $\mathfrak{X}$, and $\mathfrak{f}$ to denote the lifted graphs, nodes, and factors. For details, see main text.

is depicted in Figure 2.4 (step 1). Here, we assume that the potential functions of the upper two factors are identical (light blue) and differ from the lower factor's one (green). Now, each variable node sends a message to its neighboring factor nodes saying "I am of color red" (step 2). Each factor node sorts the incoming colors into a vector according to the order the variables appear in its arguments. The last entry of the vector is the factor node's own color, represented as light blue, respectively green, squares in Figure 2.4 (step 3). Based on these signatures, the colors of the factors are newly determined and sent back to the neighboring variables nodes (step 4), essentially saying "I have communicated with my neighbors". The variable nodes stack the incoming signatures together and form unique signatures of their one-step message history (step 5). Variable nodes with the same stacked signatures, i.e., message history, are grouped together. To indicate this, we assign a new color to each group (step 6). In our running example, only variable node X_3 changes its color from red to yellow. This process is iterated until no new colors are created anymore.

Algorithm 2 shows the pseudo code for the CP algorithm. The algorithm is very similar to the BP algorithm, however, instead of calculating beliefs, new colors for each factor and each node are calculated in every iteration. While Algorithm 1 shows BP with the flooding protocol, the CP in Algorithm 2 follows a parallel protocol as well. The calculation of new colors is not made explicit in the pseudo code and we will now describe the functionality of Line 12 and Line 21. Essentially, all factors obtain the same color that have the same color signature. In cases where the positions of the variables do not matter in the factor function, i.e., the associated function or table is symmetric, we can sort the incoming colors. However, this is not generally possible. For the new variable colors, we can always sort the incoming colors in the signature. We will touch upon this issue later again when we describe a CP variant specifically tailored for CNFs and lifted satisfiability in Section 3.1.

The final lifted factor graph, denoted by $\mathfrak{G}$, is constructed by grouping all nodes with the same color into so called *clusternodes* $\mathfrak{X}_i$ and all factors with the same color signatures into so called *clusterfactors* $\mathfrak{f}_a$. Additionally, we define the function $\mathfrak{X}(X_i)$ which returns the clusternode or color of ground variable X_i. Correspondingly, we define for ground factors the function $\mathfrak{f}(f_a)$. Clusternodes, respectively clusterfactors, are sets of nodes, respectively factors, that send and receive the same messages at each step of carrying out the message passing on G. These sets form a partition of the nodes in G. In our case, variable nodes X_1, X_2 and factor nodes f_1, f_2 are grouped together into clusternode $\mathfrak{X}_1 = \{X_1, X_2\}$ and clusterfactor $\mathfrak{f}_1 = \{f_1, f_2\}$. The others sets are singletons ($\mathfrak{X}_2$ and $\mathfrak{f}_2$) as shown in Figure 2.5. Equivalently, function $\mathfrak{X}(.)$ evaluates to the same value for X_1 and X_2, i.e., $\mathfrak{X}(X_1) = \mathfrak{X}(X_2) = \mathfrak{X}_1$. Now, we are essentially ready to run Step 2, i.e., the modified message passing on the lifted factor graph.

Before we explain the lifted message updates in detail, we will comment how CP distinguishes from related approaches. The algorithm to lifted BP that was presented by Singla and Domingos in [209] differs from the one explained above in several ways. In first place, it was defined for MLNs and makes use of the first-order formulas over which an MLN is defined. On the one hand, this makes the approach possibly more efficient because it does not always require a full grounding of the network and remains on the lifted level all the time. On the other hand, the approach by Kersting et al. is more general as it operates directly on the factor graph and hence can be applied in many more settings. It can be used to lift problems that are not defined based on FOL formulas. Additionally, one can also view the CP algorithm more from a graph-theoretic point of view. As shown in [151], CP finds the *coarsest equitable partition (CEP)* of the graph which is also known as the total degree partition. We will further discuss this context in Chapter 4 where we describe how the lifting can be detached from the message passing. Furthermore, there are several other approaches for finding the CEP. For example, instead of applying the CP algorithm on the graph, one can also find the CEP by defining a nonlinear continuous optimization problem and solve this problem instead [117].

For the modified BP on the lifted graph $\mathfrak{G}$, the basic idea is to simulate BP carried out on G now on $\mathfrak{G}$. An edge from a clusterfactor $\mathfrak{f}_a$ to a clusternode $\mathfrak{X}_i$ in $\mathfrak{G}$ essentially represents multiple edges in G. Let $\mathrm{c}(a,i)$ be the number of identical messages that would be sent from the factors in the clusterfactor $\mathfrak{f}_a$ to each node in the clusternode $\mathfrak{X}_i$ if BP was carried out on G. The message from a clustervariable $\mathfrak{X}_i$ to a clusterfactor $\mathfrak{f}_a$ is

$$\mu_{i\to a}(\mathfrak{x}_i) = \mu_{a\to i}(\mathfrak{x}_i)^{\mathrm{c}(a,i)-1} \cdot \prod_{b\in \mathrm{nb}(i)\setminus\{a\}} \mu_{b\to i}(\mathfrak{x}_i)^{\mathrm{c}(b,i)} \ ,$$

where $\mathrm{nb}(i)$ now denotes the neighbor relation in the lifted graph $\mathfrak{G}$. The $\mathrm{c}(a,i)-1$ exponent reflects the fact that a clustervariable's message to a clusterfactor excludes the corresponding factor's message to the variable if BP was carried out on ground graph G. I.e., we prevent double counting the message $\mu_{a\to i}$. In a similar fashion, the belief of a variable can be calculated on the lifted graph. The unnormalized belief of $\mathfrak{X}_i$, which is identical to any of the beliefs for nodes in $\mathfrak{X}_i$, is computed as follows:

$$b_i(\mathfrak{x}_i) = \prod_{a\in \mathrm{nb}(i)} \mu_{a\to i}(\mathfrak{x}_i)^{\mathrm{c}(a,i)} \ .$$

Evidence is incorporated as in standard BP by setting $\mathfrak{f}(\mathfrak{x}) = 0$ for states $\mathfrak{x}$ that are incompatible with the observations.

However, we advocate here in favor of a more general form of the lifted messages to account for several issues. Compared to the equations presented in [114], this more general form will allow to compare LBP messages more easily to other lifted algorithms. In first place, we want to stress the fact that the position of a variable in a factor needs to be distinguished. The function associated with a factor does not necessarily have to be commutative. Secondly, there are two different types of counts. One count is responsible for keeping track of identical factors connected to a variable, the other one counts identical variables in the range of a factor. We will begin by explaining the issue of positions in factors by a small example factor graph.

Example 2.3. *Looking at Figure 2.6(middle), we see a small factor graph with three variables and three factors. It is important to note that in this example the factor functions are not commutative (Figure 2.6(left)), i.e., it matters if a variable is f_i's first or second*

X_i	X_j	f_a
0	0	1
1	0	0.7
0	1	1.5
1	1	2

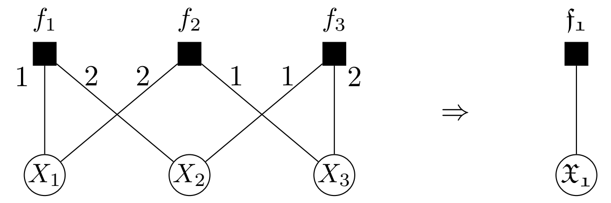


Figure 2.6: Simple factor graph highlighting the requirement of distinguishing positions in lifting. All factors f_a are identical, however, in this example they are not commutative, i.e., $f_a(0,1) \neq f_a(1,0)$. Therefore, it matters if a variable appears as the first or the second argument in the function. The edge-labels indicate the position of a variable in the factor.

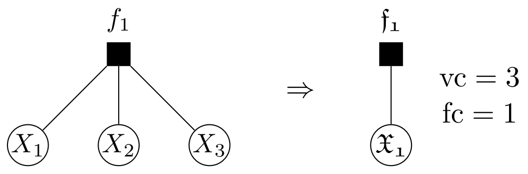


Figure 2.7: Simple factor graph highlighting the requirement of having two different kinds of counts opposed to just a single count. The factor in this graph is connected to three variables of the same type, i.e., vc = 3. In contrast, each variable is only connected to a single factor, i.e., fc = 1.

argument. A simple example could be a factor representing the logical rule $X_i \rightarrow X_j$. Looking at the graphical representation of the lifted graph (Figure 2.6(right)), one cannot see that different messages for the edges at different positions were exchanged in the ground graph.

To remedy this issue, we will add an indicator of the position to the formulas. In the LBP algorithm, the loop calculating the new messages has to be extended to iterate over all possible positions of a variable in a particular factor. The more general form of the message send from a clusternode to a clusterfactor looks as follows:

$$\mu_{i \rightarrow a,p}(\mathfrak{x}_i) = \prod_{b \in \mathrm{nb}(i)} \prod_{p' \in \mathrm{pos}(b,i)} \mu_{b \rightarrow i,p'}(\mathfrak{x}_i)^{\mathrm{fc}(b,i,p') - \delta_{ab}\delta_{pp'}} \tag{2.9}$$

with pos(b, i) representing all different positions a variable $\mathfrak{X}_i$ can take on in a factor $\mathfrak{f}_b$. Instead of a multiplicative term in front of the product over the neighbors, we integrate the self-loop into the product directly by ranging over "nb(i)". Note that factor $\mathfrak{f}_a$ is not excluded from the range anymore. This saves us a separate handling of the positions for the self-loops. This also moves the prevention of double counting (previously "c$(a, i) - 1$") into the product term and we only have to subtract in cases where we send a message to the same variable (δ_{ab}) at the same position ($\delta_{pp'}$). Also note that we have renamed the counts from "c" to "fc". The reason for this renaming is that we have to distinguish two types of counts. We will underline this requirement with the help of a small example as well.

Example 2.4. *Figure 2.7(left) depicts a very simple factor graph with a single factor f_1 over three variables. If we assume that positions do* not *matter in this factor, the lifted factor graph looks like the one shown in Figure 2.7(right). The counts must now reflect that f_1 is connected to three variables of the same type in the ground network. However, each* *of the variables is only connected once to the factor. Therefore, we distinguish the variable count* vc $= 1$ *from the factor count* fc $= 3$.

We will now give the formula for the lifted message from a clusterfactor to a clustervariable which highlights the usage of the variable count vc. Again, we will be distinguishing different positions of a message:

$$\mu_{a \to i,p}(\mathfrak{x}_i) = \sum_{\neg\{i\}} \left(\mathfrak{f}_a(\mathfrak{x}_a) \prod_{j \in \mathrm{nb}(a)} \prod_{p' \in \mathrm{pos}(a,j)} \mu_{j \to a,p'}(\mathfrak{x}_j)^{\mathrm{vc}(a,j,p') - \delta_{ij}\delta_{pp'}} \right)$$

Here, the *variable count* $\mathrm{vc}(\mathfrak{i}, \mathfrak{a})$ is the count of identical variables connected to a ground factor represented by $\mathfrak{a}$. Factor a received $\mathrm{vc}(\mathfrak{i}, \mathfrak{a})$ identical messages from its neighbors. Similarly, the *factor count* $\mathrm{fc}(\mathfrak{a}, \mathfrak{i})$ in Equation (2.9) represents the number of equal ground factors that send messages to a variable at a given position. It counts the number of neighbors of type a that ground variable i was connected to in the ground graph G. It counts the number of identical messages that variable i was receiving. Introducing positions and different counts into the formulas explicitly, makes it more difficult to read the formula on the one hand, however, on the other hand it becomes easier to verify the correctness of messages on the lifted graph and makes the complexity of the messages more obvious.

As mentioned above, there are different update schedules for the BP equations. In the case of LBP, we always assume a parallel schedule because otherwise we cannot guarantee that messages are always identical in the ground and lifted factor graph.

This idea of lifting BP via CP does not only apply to the multinomial variant of BP. In fact, Ahmadi et al. [3] have shown that GaBP can be lifted in a similar way, resulting in a lifted BP algorithm for Gaussian MRFs.

Before moving on, we want to comment on the run time of CP. One can see that CP is a form of the 1-dimensional *Weisfeiler-Lehman* algorithm. It follows from the arguments in [202] that the complexity of one iteration of CP is linear in the number of edges of the factor graph. This makes use of efficient sorting based on *counting sort*. Hence, for h iterations, we have a run time of $\mathcal{O}(h|E|)$. Depending on the structure of the network, we can require up to $|V|$ iterations.

2.4.2 Advanced Lifting Techniques

In many situations, exact lifting is either not applicable or not well suited. In other settings we are required to repeatably lift a network and want to make use of the previous runs of CP. For these and other scenarios, advanced lifting techniques have been developed and are of interest of current research. We will here discuss some these approaches.

Approximate Lifting

In many applications, in particular if LBP is not applied to relational models without evidence, the underlying factor graph is not perfectly symmetric. For this reason, different approaches to approximate lifting have been proposed. The most simple way of approximating the exact lifting is by *early stopping* the CP.

Instead of running CP until convergence, we stop the exchange of new colors early. By doing so, we prevent that the model is completely shattered to the ground case. It is interesting to note that early stopping can lead to fractional counts. For example, after running CP on

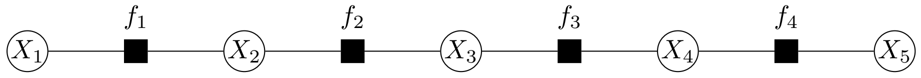


Figure 2.8: A chain-MRF leads to fractional counts when Color Passing is stopped early

the graph shown in Figure 2.8 for two iterations, we obtain for the variables a coloring of $(1, 2, 2, 2, 1)$, similarly we obtain the coloring of $(1, 2, 2, 1)$ for the factors. If we now extract the lifted graph from this coloring, we see that we obtain fractional counts. This is due to the fact that not all variables in the cluster with color 2 are connected to factors with the same color. This information has not been propagated yet. To be more precise, variables X_2, X_3, and, X_4 have the same colors but are connected in some cases to a factor with color 1 and in other cases to factors of color 2. In the end, this leads to a count of 2/3 for $\mathfrak{f}_1$ and 4/3 for $\mathfrak{f}_2$. However, the sum of the counts is sill correct, i.e., 2.

Akin to early stopping based on CP, de Salvo Braz et al. [47] presented Anytime LBP. Anytime LBP follows the top-down lifting paradigm and shatters the model iteratively. Starting from a query predicate, Anytime LBP adds neighboring predicates of the query in each iteration to the model which can be seen as the unrolling of the computation tree. This can be done on the lifted level until newly added neighbors require a shattering.

In [115], we presented an approach to approximate lifting which also refines the lifting in each iteration of inference and is close to the approach by de Salvo Braz et al. [47]. However, our approach works in a bottom-up fashion and is again independent of a first-order formalism. Intuitively, the proposed algorithm *informed Lifted Belief Propagation (iLBP)* interleaves lifting with BP iterations. More precisely, iLBP first runs a single iteration of CP on the ground graph and this is followed by a single iteration of BP on the resulting lifted graph. The graph is now lifted again based on the BP messages of this first iteration, i.e., variables and factors are clustered together if their difference in beliefs is small. This process is iteratively repeated until now new clusters arise. Essentially, iLBP is replacing CP's syntactic criterion for comparison by a real-valued similarity measure. iLBP is particularly well suited in situations where standard LBP is not efficient because the CP poses an overhead. This can be for example the case when BP converges in very few iterations. In iLBP, nodes can be grouped together that have different computation trees but similar beliefs. By adjusting the definition of "difference in messages", we can force the algorithm to produce more or fewer clusters. iLBP finds smaller factor graphs, hence further speeding up inference, without a decrease in performance, as it has been shown empirically on different datasets. For example. iLBP has been successfully used on the content distribution problem in peer-to-peer networks.

More recently, Singla et al. [211] further analyzed early stopping in their top-down lifting approach from [209] and also introduced noise-tolerant hypercubes. Similar to iLBP, the noise-tolerant hypercubes allow for marginal errors in the clusternodes by forming coarser factor graphs. I.e., variables are clustered together that are separated in the original lifted BP approach. In the framework of bisimulation, Sen et al. [198] use binning of factors that are within an user-specified distance to approximate lifting.

It has been shown in [4] that shattering a large graph into pieces gives opportunities for lifting in an online training settings. This piecewise approach breaks long-range dependencies and removes asymmetries that prevented lifting of the entire graph. Instead of training on the entire graph, the pieces are used as mini-batches and lifting can be applied to these.

Online Lifted Inference

In many applications one has to apply lifted inference to a sequence of problems with only little differences between each of the subproblems. Here, it is desirable to avoid lifting from scratch for each subproblem. One particular instance of such a setting is the case where only the evidence changes in each subproblem. A few algorithms have been proposed that try to become more efficiently by exploiting symmetries across inference instances [2, 3, 165]. So far, these algorithms have concentrated to efficiently obtain a lifted data structure for subsequently following inference instances with changes in knowledge, i.e., evidence. We will make use of such ideas and extend these for our purposes in Chapter 3. In contrast to existing approaches approaches, we will show in Section 3.3 how changing evidence during a single inference run can be exploited as well.

We have seen in this section how probabilistic inference algorithms can benefit from lifting. In particular, we have seen how BP can be lifted by using modified update equations running on a compressed factor graph. Before we will present our contributions in form of new lifted message passing algorithms and modified lifting procedures in Chapter 3, we will briefly summarize quality measures that will be used throughout this thesis for comparison and evaluation.

2.5 Quality and Error Measures

To measure the quality of an assignment in MAP inference, one often uses a *Log-Likelihood (LL)* based score. For an MRF as defined above with $p(\mathbf{X}=\mathbf{x}) = \frac{1}{Z}\prod_{c\in C}\phi_c(\mathbf{x}_c)$, we define the log score as:

$$\text{LL\_SCORE}(\mathbf{x}) = \ln\left(\prod_{c\in C}\phi_c(\mathbf{x}_c)\right) = \sum_{c\in C}\ln\left(\phi_c(\mathbf{x}_c)\right) \tag{2.10}$$

This score ignores the normalization constant and can be seen as the unnormalized joint probability of an assignment. A different approximation is taken, when using a pseudo log-likelihood score:

$$\text{PLL\_SCORE}(\mathbf{x}) = \sum_{i=1}^{n}\ln p\left(X_i = x_i \middle| \operatorname{nb}(X_i) = \mathbf{x}_{\operatorname{nb}(X_i)}\right) \ , \tag{2.11}$$

where $\operatorname{nb}(X_i)$ are the neighbors of X_i and hence define the Markov blanket of X_i. The probability of X_i given its Markov blanket can be easily calculated by local information. Instead of the absolute value, a normalized version of the PLL_SCORE is often given.

For comparing two probability distributions, it is also useful to consider measures specifically defined for that purpose. One particular example of such a measure is the *Kullback-Leibler divergence*. Assuming an unknown distribution $p(\mathbf{x})$ and a learned distribution $q(\mathbf{x})$ approximating p, the KL divergence is defined as:

$$\mathrm{KL}(p||q) = \sum_{\mathbf{x}} p(\mathbf{x})\ln\left(\frac{q(\mathbf{x})}{p(\mathbf{x})}\right) \tag{2.12}$$

One should note that the KL divergence is not symmetric and therefore the distinction of p and q is important. The KL divergence also has a nice information theoretic interpretation. It measures the information lost when approximating p by q. For a more detailed discussion we refer to [124, A.1.3]. For example, we will use the KL divergence in Chapter 6 to compare the fit of different distributions on observed data and pick the best one based on the minimum divergence.

In other situations, we want to compare a set of learned parameters or predicted regression values to a gold-standard. Here, the *Root-Mean-Square Error (RMSE)* is a performance measure which is often used:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{x}_i - x_i)^2} \tag{2.13}$$

where $\hat{x}_i$ amounts to the gold-standard value and x_i is the learned or predicted value. We will see in Chapter 7 that we can easily extend this measure to the multivariate case if we want to measure the quality of predictions over n variables in m datacases.

In binary classification tasks, we wish to use yet another error measure. One can consider using the *accuracy*, i.e., the number of correctly classified instances divided by the total number of instances. However, especially with highly unbalanced classes, this is not appropriate. For example, in Chapter 7, we will evaluate different algorithms on a network discovery task on very sparse graphs. Here, an algorithm always returning `False` will achieve fairly high accuracy. A better score in such settings is the *F1-score*:

$$\text{F1\_SCORE} = 2\frac{p \cdot r}{p + r} \tag{2.14}$$

which is the harmonic mean of the *precision* p and *recall* r. Recall is the number of true positives divided by the sum of true positives and false negatives. Precision is the number of true positives divided by the sum of the true positives and false positives.

CHAPTER 3

Lifted Message Passing Beyond Lifted Belief Propagation

Inference in probabilistic graphical models remains one of the major challenges in AI and Machine Learning. Due to its theoretical complexity, exact inference is not tractable in most cases. But even approximate inference is too slow in various cases and a lot effort has been put into speeding up approximate inference as well. This chapter will pick up the ideas from lifted inference in Section 2.4 and further contribute to that area of research. Lifted message passing has so far been concentrating on probabilistic inference. We will add to this development and present a lifted version of an existing MAP inference algorithm (Section 3.3). However, we will also show that message passing algorithms for tasks beyond probabilistic inference are liftable as well. More precisely, we show how popular message passing algorithms for satisfiability solving can be lifted (Section 3.1). In addition, we will present several improvements over existing lifting algorithms that apply to probabilistic inference as well as satisfiability solving (Section 3.2). The contributions of this section have been published mainly in [82, 83, 81].

Although this chapter covers several different aspects of lifting and corresponding inference algorithms, we can list a handful of papers that were most influential to the work that is to follow:

- Kersting et al. [114]: Color Passing for the lifting of a factor graph and Lifted Belief Propagation based on the compressed network.
- Braunstein et al. [27]: Decimation framework with ground Warning Propagation and ground Survey Propagation.
- Kumar and Zilberstein [129]: Likelihood Maximization approach for MAP inference that works as an iterative optimization procedure.

In particular, the work by Kersting et al. was very influential for this thesis and its contents have already been described in detail in Section 2.4.1.

We proceed as follows. To begin with, we return to the idea of lifted satisfiability from the introduction (cf. Section 1.1) and explain in Section 3.1 the concepts of lifted satisfiability in greater detail. This will also motivate the requirement of an adapted compression algorithm for factor graphs to fully exploit lifting in SAT. This algorithm is introduced in Section 3.2 where we also explain how to integrate it into the lifted SAT framework and how it can be applied beyond satisfiability. In Section 3.3, we will show that iterative optimization algorithms based on lifting can often benefit from re-lifting during inference and we present a lifted version of Kumar and Zilberstein's Likelihood Maximization algorithm. Before viewing lifted inference from a different angle in Chapter 4, we summarize the contributions of the current chapter.

3.1 Lifted Satisfiability

We now return to our example from the introduction and give more details on concepts behind lifted satisfiability. Besides the fact that solving satisfiability problems is key to all kinds of problems, SAT solving, or sampling satisfying assignments, also plays a key role in some inference algorithms for relational probabilistic models such as MC-SAT [181]. Using message passing algorithms for satisfiability has been investigated from different points of view. Besides algorithms tailored specifically for SAT, approaches based on BP have been used as well. The AAAI conference in 2008 also featured a tutorial dedicated to satisfiability based on message passing[1]. This tutorial summarizes several approaches and gives a good introduction to this topic. Having observed the success and impact of lifted inference in relational models, we will now show that lifting is also beneficial in SAT solving by developing lifted versions of two message passing algorithms for satisfiability, namely *Warning Propagation (WP)* and *Survey Propagation (SP)*. In addition, we will show that a number of simplifications are possible compared to the lifted version of BP when the underlying problem is a CNF.

The main drawback of existing lifted inference approaches, such as LBP, is the fact that they are not tailored towards combinatorial problems. In many real-world applications, however, the problem formulation does not fall neatly into the pure probabilistic inference case. Problems may very well have a component that can be well-modeled as a combinatorial problem, hence, taking it outside the scope of standard LBP approaches. Besides that, satisfiability of Boolean formulas is one of the classical AI problems and the idea of "reduction to SAT" is a powerful paradigm for solving problems in different areas. Ideally, lifted inference should be efficient here, too, exploiting as many symmetries as possible. Indeed, driven by the same question of reasoning with both probabilistic and deterministic dependencies, but for the non-lifted case, Poon and Domingos [181] developed MC-SAT that combines ideas from MCMC and satisfiability. It still proceeds by first fully instantiating the first-order theory and then essentially staying at the propositional level. LazySAT [207] is a lazy version of WalkSAT taking advantage of relational sparsity. Recently, Poon *et al.* [182] have shown that this idea of lazy inference goes beyond satisfiability and can be combined with other propositional inference algorithms such as BP. However, the link has not been explored on the lifted (message passing) level.

In this section, we therefore revisit lifted message passing algorithms this time for combinatorial problems. Specifically, we focus on Boolean satisfiability problems consisting of a formula F representing a set of constraints over n Boolean variables, which must be set such that all constraints are satisfied. It is one of the most — if not the most – studied problems in computer science and AI and several other NP-complete problems can be reduced to it. SAT has applications in several important areas such as automated deduction, cryptanalysis, modeling biological networks, hardware and software verification, planning, and Machine

[1]Lukas Kroc, Ashish Sabharwal, and Bart Selman: "Satisfied by Message Passing" (`www.aaai.org/Conferences/AAAI/2008/aaai08tutorials.php`)

Learning, among others. Hence, pushing the boundary of practical solvability of SAT has far reaching practical consequences.

In fact, we develop a lifted version of an exciting algorithm for solving combinatorial problems, namely *Survey Propagation (SP)*. SP was introduced by Mézard et al. [147] and can easily solve SAT problems with one million variables in a few minutes on a desktop computer. SP has particularly attracted a lot attention because it was able to solve difficult problems. A SAT problem is generally accepted to be difficult if it lies in the crossover regime between the "SAT" and "UNSAT" classes. With an increasing number of constraints, the problem is more likely to be in the "UNSAT" class and the exact crossover point for k-SAT has so far been only estimated by numerical experiments. Braunstein and Zecchina [26] and Maneva et al. [141] have shown that SP can be viewed as a form of BP, hence, somehow suggesting that lifting SP is possible. As noted by Kroc et al. [125], SP's remarkable effectiveness — the performance is clearly beyond the reach of mainstream SAT solvers — has created the impression that solving hard combinatorial instances requires SP but BP would have little success. Kroc et al. [126], however, have shown that "the gap between BP and SP narrows" as the number of variables per clause increases. This is also supported by Montanari et al. [153] showing good performance of a combination of BP and *Warning Propagation (WP)*, a message passing algorithm for SAT that is simpler than SP but also less powerful. Therefore, we also show how to lift WP and provide a complete picture of lifted message passing algorithms for SAT, which is the main contribution of the current section. Let us again summarize the individual contributions of this section:

(C3.1) We apply the idea of lifted inference to the problem of SAT solving, thus exploiting the structure of combinatorial problems by techniques known from probabilistic inference in relational models.

(C3.2) We provide a lifted version of WP and derive the lifted update equations in detail.

(C3.3) In the same spirit as lifting WP, we also provide a lifted version of the more powerful SP algorithm.

These contributions significantly advance the understanding of lifted message passing and put an unified, lifted treatment of combinatorial and probabilistic problems within reach.

We proceed as follows. We first provide background on Boolean formulas, show how factor graphs can be used to represent those, and then discuss (lifted) message passing for SAT. Specifically, we show how LBP's Color Passing algorithm, which compresses the original factor graph, can be used for SAT problems and show how a simplified version can be used for lifting WP and SP. We then introduce both *Lifted Warning Propagation (LWP)* and *Lifted Survey Propagation (LSP)*, i.e., the modified message passing equations used on the lifted factor graph. Again, we argue that they are simpler than the ones for LBP. Before concluding, we present the results of our experimental evaluation where we show that running lifted message passing within the standard decimation approaches for SAT can achieve remarkable efficiency gains.

3.1.1 CNFs and Factor Graphs

Without loss of generality, we assume that a Boolean formula is represented in CNF. Every Boolean formula can be converted to CNF with help of De Morgan's law and the distributivity law. However, in practice this can lead to an exponential blow up of the formula. A CNF is a conjunction of disjunctions of Boolean literals. A literal is either a negated or unnegated

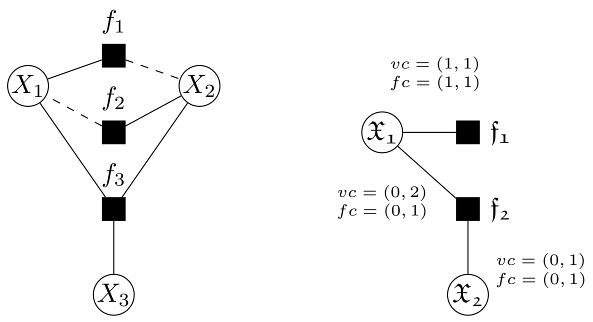


Figure 3.1: (left) $(X_1 \vee \neg X_2) \wedge (\neg X_1 \vee X_2) \wedge (X_1 \vee X_2 \vee X_3)$ represented as a factor graph. Circles denote variable nodes and squares denote clauses. A dashed line indicates that the variable appears negated in a clause; otherwise we use a full line. (right) The resulting lifted CNF after running the Color Passing algorithm as illustrated in Figure 2.4.

propositional variable. Specifically, a CNF consists of n variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ with $x_i \in \{\texttt{False}, \texttt{True}\}$ and m clauses $C = \{C_1, \ldots, C_m\}$ constraining the variables. A clause C_a is defined over a subset of the variables $\mathbf{X}_a = \left\{X_{a,1}, \ldots, X_{a,|C_a|}\right\}$ and — this will turn out to be convenient for formulating the message passing update formulas — a corresponding list of their signs which is denoted as $s_a = \left\{s_{a,1}, \ldots, s_{a,|C_a|}\right\}$ with

$$s_{a,i} = \begin{cases} -1, & \text{if } X_{a,i} \text{ unnegated} \\ 1, & \text{if } X_{a,i} \text{ negated} \end{cases} \tag{3.1}$$

A solution to a CNF is an assignment to all variables in $\mathbf{X}$ such that all clauses in C are satisfied, i.e., evaluate to `True`. As an example, consider the following CNF, consisting of three variables, three clauses, and seven literals:

$$(X_1 \vee \neg X_2) \wedge (\neg X_1 \vee X_2) \wedge (X_1 \vee X_2 \vee X_3) \ .$$

A possible solution to that formula is $X_1 = \texttt{True}$, $X_2 = \texttt{False}$, $X_3 = \texttt{True}$.

Every CNF can be represented as a factor graph (see [128]). The factor graph representing the example CNF from above is shown in Figure 3.1(left). For CNFs, there is a one-to-one mapping between the variables in the CNF and the variable nodes in the factor graph, as well as the clauses and factor nodes. Therefore, we will not distinguish between logical and probabilistic variables explicitly. We have a factor f_a for each clause $C_a \in C$. We use a dashed line between a variable X_i and a factor f_a whenever the variable appears negated in a clause, i.e., $s_{i,a} = 1$, otherwise a full line. The factor functions $f_a(\mathbf{x}_a)$ are defined in a such a way that they return a value of 1 if the corresponding clause evaluates to `True` for $\mathbf{x}_a$ and 0 otherwise. With this definition, the normalization constant Z amounts to the number of solutions of the CNF and each satisfying assignment of $\mathbf{X}$ has the same probability, i.e., $1/Z$.

Before deriving lifted message updates for WP and SP in the next sections, let us first illustrate lifting in the case of CNFs. As we will see, CP and its outcome, the lifted factor graph, are simpler for the CNF case compared to the LBP case. As we have seen above, we can use the CP algorithm to lift a ground factor graph. Of course, this is also possible for a factor graph representing a CNF. Again, for the lifted satisfiability approach, CP on a factor graph representing a CNF is the first step. For such a factor graph, the domain of each variable has size $d = 2$ and the initial coloring of the factors just distinguishes clauses with different number of negated and unnegated variables. I.e., two clauses fall into the same group if both have the same number of positive and negative literals because the logic operator $\vee$ is commutative. But more importantly, as we will show next, the lifting can actually be simplified in the case of CNFs.

First, we can employ a more efficient color signature coding scheme. The initial grouping of the factors solely depends on the zero state of their corresponding clause. Additionally, the factor nodes do not have to sort the incoming colors according to the positions of the variables, instead only the sign of a variable matters, i.e., only two positions exist. One position for the variables appearing unnegated in the clause and a second position for the negated variables. We have seen in the case of the BP lifting that we have to distinguish arbitrarily many positions (see Example 2.3 for details).

Second, we can employ simplified counts. Because there are only two positions, we store two values: counts for the negated position and for the unnegated position. We do not have to manage a complex structure with counts for different positions as in the case of LBP anymore. Reconsider Figure 3.1(right). Here, the count associated with the edge between $\mathfrak{X}_1$ and $\mathfrak{f}_1$ is $(1, 1)$ because $\mathfrak{X}_1$ represents ground variables that appeared positive and negative in ground factors represented by $\mathfrak{f}_1$. Second, there is the *variable count* vc. It corresponds to the number of ground variables that send messages to a factor at each position. In the ground network, the factor f_3 is connected to three positive variables, but two of them are represented by a single clusternode in the lifted network. Hence, we have $vc = (0, 2)$ for the edge between $\mathfrak{X}_1$ and $\mathfrak{f}_2$ because $\mathfrak{f}_2$ is now the representative of f_3. So, for lifted CNFs, all counts in the factor graph are of dimension two only.

Having the lifted CNF at hand, we can now run LBP on this lifted graph. However, there are other ground message passing algorithms available that are more appropriate for solving CNFs. We will show that these algorithms can be lifted too. First, we will describe the approach based on LBP, before we will then derive the lifted versions of WP and SP which are designed to operate on factor graphs for CNFs directly.

3.1.2 Belief Propagation for Satisfiability

The introductory example in Section 1.1 already referred to a heuristic used in solving SAT problems which is applied to each variable. In the introduction, however, this heuristic was left unspecified. One possible heuristic can be the single node marginal probabilities $p(X_i)$. If we chose the potentials of the factors as described above, running sum-product BP on a factor graph representing a CNF has a nice and intuitive explanation. The beliefs of a state give the approximate marginal probability of a variable being in that state in a solution to the underlying CNF. Additionally, the partition function estimates the number of solutions for a given CNF. Of course, these values are only guaranteed to be correct for tree-structured CNFs. We will now exemplify, how results can diverge from the true values on a loopy graph.

Example 3.1. *Looking at the CNF in our running example (cf. Figure 3.1(left)), we can find the corresponding truth tables of the clauses in Table 3.1. One can easily verify that the CNF has three solutions in total and each variable has a probability of* $2/3$ *being* `True` *in a solution. However, as already discussed, BP is not exact on loopy graphs. The beliefs computed by sum-product BP diverge highly from the true marginals (also shown in Table 3.1). Therefore, it is of little surprise that the estimated partition function is not exact either. The computed value is* $Z = 2.00$ *as apposed to the correct value of 3. On the other hand, the*

	x_1	x_2	x_3	f_1	f_2	f_3	$\prod f_i$
	0	0	0	1	1	0	0
	1	0	0	1	0	1	0
	0	1	0	0	1	1	0
	1	1	0	1	1	1	1
	0	0	1	1	1	1	1
	1	0	1	1	0	1	0
	0	1	1	0	1	1	0
	1	1	1	1	1	1	1
$p(x_i = 1)$	0.67	0.67	0.67				
sum-product BP $b(x_i = 1)$	0.98	0.98	0.50				
$m(x_i = 1)$	0.50	0.50	0.50				
max-product BP $b(x_i = 1)$	0.50	0.50	0.50				

Table 3.1: Exemplifying BP on a CNF. The marginals give the probabilities of each variable for being in a specific state in a satisfying assignment. One can see that the BP approximation is of low quality in this example.

computed max-marginals are correct but give less insights than the marginal probabilities. Of course, we can apply LBP on this factor graph as well, to obtain the marginals more efficiently. We can see from the table that the beliefs of X_1 and X_2 are identical. This matches our expectation when looking at the lifted CNF in Figure 3.1(right).

Although BP is not exact, it can still be used to solve satisfiability problems. In particular, the approximate BP marginals have been used successfully in decimation procedures to find a satisfying solution [153]. Decimation procedures make iteratively use of the BP marginals. As we can see from Example 3.1, the marginals are not exact, but at least the tendency of X_1 and X_2 towards the `True` state is correct. Besides this, BP can also be used for approximative model counting [127]. We will now describe the concept of BP-guided decimation in detail.

3.1.3 Decimation Based SAT Solving

The example in the previous section already shows that having solely BP at hand is usually not sufficient to solve complex CNFs. Nevertheless, we can run BP, or similar message passing algorithms as we will see in the next sections, on this problem and use the fixed points of the algorithm to obtain insights about satisfying configurations. However, in most cases the information about the fixed points is not sufficient to obtain an entire satisfying configuration. Still, the output of the algorithm, i.e., the beliefs, will point us in the right direction to find a solution in many cases. This idea was already motivated by the example in Example 3.1 and Table 3.1. The general paradigm of applying this idea iteratively is known as *decimation* and will now be explained in detail. Many complex inference tasks, such as finding a satisfying assignment to a Boolean formula or computing joint marginals of random variables, can be cast into a sequence of simpler ones. This is achieved by selecting and clamping variables one at a time and running inference after each selection. SAT problems can be solved using a decimation procedure based on BP [153]. For SAT problems, decimation is the process of iteratively assigning a truth value to k variables (often a single variable only) in $\mathbf{X}$. This results in a factor graph which has these variables clamped to a specific truth value. We can then simplify the factor graph G accordingly, obtaining a smaller factor graph representing a simplified formula with $n - k$ variables. Now, we repeatedly decimate the formula in this manner, until all variables have been clamped. In the end, either a satisfying assignment has been found or a contradiction has been detected. This procedure heavily depends on the strategies for choosing variables to fix and the corresponding states. In a BP guided decimation, BP is run on the (simplified) factor graph and a variable is picked based on the approximated marginals. Usually, the variable with the highest probability for one state is picked and clamped accordingly. This variable is often referred to as the most *magnetized* variable. This popular approach can be seen as a general idea of turning the complex inference task into a sequence of simpler ones by selecting and clamping variables one at a time and running inference again after each selection.

Example 3.2. *Looking at our running example in Figure 3.1 and the marginal probabilities in Table 3.1, we would pick either X_1 or X_2 if the decimation was using BP as heuristic. In case X_1 was picked, we would set $X_1 =$ `True` and simplify the formula. In this simple case, we would already obtain a formula of the form (X_2) and would not need a second iteration of BP to completely solve the problem.*

The concept of decimation can be lifted as well. In the following sections, we will describe how an efficient *lifted decimation* procedure can be realized. In first place, we will be using lifted message passing to speed up each iteration. However, this requires lifting from scratch in each iteration, possibly canceling some of the benefits of lifting. Therefore, we will also introduce a more efficient lifting algorithm in the second step that makes use of the current lifting when producing the lifted graph for the following iteration. Additionally, BP is not the only message passing algorithm that can be used to obtain insights about the state of a variable in a satisfying configuration. We will see how Survey Propagation is capable of computing indicators for each variable upon which one can pick a variable to clamp as well. We will propose a lifted variant of this algorithm which also requires the introduction of counts. As described above, and shown in Example 3.2, fixing variables often allows a simplification of the formula. This simplification can be casted into a message passing algorithm as well and amounts to applying Warning Propagation on the factor graph. Having Warning Propagation at hand, we implement the decimation completely by means of a factor graph and message passing. BP and Survey Propagation can be combined with Warning Propagation to simplify the CNF after having variables clamped. We will show that Warning Propagation is liftable as well and can be run on the lifted factor graph, yielding a decimation with all operations on the lifted level.

3.1.4 Lifted Warning Propagation

Clauses consisting of a single literal only, i.e., factors with an edge to exactly one variable X_i only, are called *unit clauses.* A unit clause fixes the truth value of X_i. The process of fixing all variables appearing in unit clauses and simplifying the CNF correspondingly is called *Unit Propagation* [256]. Casted into the message passing framework [27, 153], it is known under the name of *Warning Propagation (WP).*

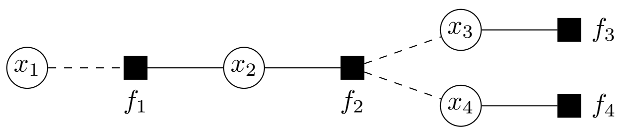


Figure 3.2: CNF containing unit clauses depicted as factor graph.

Intuitively, a message $\mu_{a\to i} = 1$ sent from a factor to a variable says: "Warning! The variable i must take on the value satisfying the clause represented by the factor a." As shown by Braunstein et al. [27], this can be expressed mathematically as follows:

$$\mu_{a\to i} = \prod_{j\in \mathrm{nb}(a)\setminus\{i\}} \theta(\mu_{j\to a} s_{a,j})$$

where nb(a) is the set of variables that are constrained by a and $\theta(x) = 0$ if $x \leq 0$ and $\theta(x) = 1$ if $x > 0$. $s_{a,j}$ is the sign of an edge and defined in Equation (3.1). The message from a variable to a factor is:

$$\mu_{i\to a} = \left(\sum_{b\in \mathrm{nb}_+(i)\setminus\{a\}} \mu_{b\to i}\right) - \left(\sum_{b\in \mathrm{nb}_-(i)\setminus\{a\}} \mu_{b\to i}\right)$$

where $\mathrm{nb}_+(i)$, respectively $\mathrm{nb}_-(i)$, is the set of factors in which i appears unnegated, respectively negated. The exchange of messages is repeated until the messages converge or a maximum number of iterations has been reached. If the message passing converges, a set of warnings can be read off this fixed point.

To be more precise, let us assume that $\mu^*_{a\to i}$ are the fixed point warnings. We can now calculate for every variable the *local field* H_i and the *contradiction number* c_i which are defined as follows:

$$H_i = -\sum_{b\in \mathrm{nb}(i)} s_{b,i}\cdot \mu^*_{b\to i},$$

and

$$c_i = \begin{cases} 1, & \text{if } \left(\sum_{b\in \mathrm{nb}_+(i)} \mu^*_{b\to i}\right)\left(\sum_{b\in \mathrm{nb}_-(i)} \mu^*_{b\to i}\right) > 0 \\ 0, & \text{otherwise} \end{cases}$$

The local field indicates the preferred state of a variable. If $H_i > 0$, X_i should be set to `True`, and for $H_i < 0$, X_i prefers `False`, otherwise no preference is expressed by the warnings. However, it can happen that a variable receives conflicting warnings which is indicated by the contradiction number. The contradiction number is non-zero if and only if a variable receives warnings for both possible states.

The initialization of the messages can either be done at random and each message is set to $\mu_{a\to i} \in \{0, 1\}$ with equal probability. Alternatively, we set all messages to zero which shows to be beneficial in the case of lifting and exploiting symmetries. Similar to BP, we have to specify a schedule which is used to update the messages. Here, we will use a parallel schedule and a fixed order of the variables.

Example 3.3. *To illustrate the effectiveness of WP, we consider a small example. Given the CNF* $(\neg X_1 \vee X_2) \wedge (X_2 \vee \neg X_3 \vee \neg X_4) \wedge (X_3) \wedge (X_4)$ *(see Figure 3.2 for the corresponding factor graph). One can easily verify that this CNF constrains variables* X_2*,* X_3*, and* X_4 *to be positive to satisfy the problem and* X_1 *can take on any value. Assume that all messages are* *initially being set to* 0. *Let us take a look at the WP messages sent from factors* f_3 *and* f_4 *to its neighbors. Since these factors have only a single neighbor, the product is over the empty set, and hence the factors can only be satisfied by the neighbors taking on the satisfying state. This results in a warning* $\mu_{a\to i} = 1$ *being sent from* f_3 *and* f_4 *to its neighbors. This warning now effects the message being send from* f_2 *to* X_2. *Now, the product over the neighbors of* f_2 *excluding* X_2 *is equal to one, indicating a warning being sent from* f_2 *to* X_2. *Factor* f_2 *informs* X_2 *that it has to take on the value satisfying* f_2. *On the other hand, the message* $\mu_{f_1\to X_1}$ *remains zero, since* f_1 *is automatically satisfied by* X_2 *and does not further constrain* X_1. *The algorithm converges to a fixed point after two iterations. Calculating the local fields, we obtain* $H_1 = 0$, $H_2 = -(-1 \cdot 1) = 1$, $H_3 = -((1 \cdot 0) + (-1 \cdot 1)) = 1$, *and similarly* $H_4 = 1$. *This highlights how WP can solve the CNF in this case similar to unit propagation but as a message passing formulation.*

If we run sum-product BP on this graph, the marginals and partition function are exact because of the tree structure. We can read off the solution from the beliefs in a similar way. The marginals of X_2, X_3, *and* X_4 *are all equal to* $(0.0, 1.0)$, *indicating that these variables have to be* `True` *in a solution. Additionally, the marginal of* X_1 *is equal to* $(0.5, 0.5)$ *which shows in this problem that* X_1 *can take on both values with equal probability. Lastly, the partition function correctly amounts to* $Z = 2$ *because there are only two solution.*

WP is also guaranteed to converge if the underlying factor graph is a tree. Additionally, if at least one of the contradictory numbers c_i is equal to 1 in the tree case, the represented problem is not satisfiable.

Example 3.3 also exemplifies how symmetries in an CNF lead to identical messages. We can see that the messages being sent from X_2 to f_1 and from X_3 to f_1 are identical. The same holds for the messages the other way around. We can now apply the CP algorithm to any CNF and use modified WP messages on the lifted graph. The *Lifted Warning Propagation (LWP)* equations take the following form[2]:

$$\mu_{a\to i,p} = \prod_{j\in \mathrm{nb}(a)} \prod_{p'\in \mathrm{pos}(a,j)} \theta(\mu_{j\to a,p'} s_{p'})^{\mathrm{vc}(a,j,p')-\delta_{ij}\delta_{pp'}}$$

with $\mathrm{pos}(a, j) \subseteq \{0, 1\}$ referring to a negated or unnegated position of a variable X_j in a clause f_a. Furthermore, we have $s_{p'} = 1$ if $p' = 0$ and $s_{p'} = -1$ if $p' = 1$. This is a position dependent version of the $s_{a,j}$ from above. Note that the set of positions $\mathrm{pos}(a, j)$ contains only those positions with a variable count (vc) greater than zero. Here, the variable count corresponds to the number of ground variables that send messages to a factor at each position. Because there are only two positions — a variable may occur at any position in negated or unnegated form — we store two values: counts for the negated position and for the unnegated position. The Kronecker deltas take care of the "$\setminus\{i\}$" in the product's range of the ground formula. They reduce counts by one when a message is sent to the node itself at the same position, i.e., they prevent double counting of messages.

To exemplify the idea of positions and variable counts, reconsider the lifted factor graph in Figure 3.1(right), "pos" contains a non-zero value for both positions for the variable count of edge $\mathfrak{X}_1 - \mathfrak{f}_1$. This is due to the fact that the factors f_1 and f_2 contain a negated and an unnegated variable in the ground factor graph. On the other hand, for the edge $\mathfrak{X}_1 - \mathfrak{f}_2$ the variable counts are only positive for the unnegated position because f_3 contains only unnegated variables in the ground network. In the ground network, the clause f_3 was connected to three

[2]We define $0^0 = 1$ in cases where $\theta(x) = 0$ and $\mathrm{vc}(a, j, p') - \delta_{ij,pp'} = 0$.

positive variables, but two of them are now represented by a single clusternode ($\mathfrak{X}_1$) in the lifted network. Hence, we have variable count vc $= (0, 2)$ for the edge between $\mathfrak{X}_1$ and $\mathfrak{f}_2$. Remember that ground factor f_3 is now represented by the clusterfactor $\mathfrak{f}_2$. Intuitively, this already indicates that the lifted formula simulates WP on the ground network. It even becomes more clear when we run it on the ground network shown in Figure 3.1(left). In this case, there is exactly one position with a variable count greater than zero for each neighbor j of factor a. In other words, pos(a, j) is a set containing only a single position for which the count is always 1. In turn, the second product can be dropped, none of the $\delta_{..}$ can be 1, and both formulas coincide. Similarly, one can intuitively check that the following formula in the other direction is correct for the lifted case:

$$\mu_{i\to a,p} = \left(\sum_{\substack{b\in \mathrm{nb}(i),\\ \mathrm{fc}(b,i,1)>0}} \mu_{b\to i}\left(\mathrm{fc}(b,i,1) - \delta_{ab}\delta_{p1}\right) \right) - \left(\sum_{\substack{b\in \mathrm{nb}(i),\\ \mathrm{fc}(b,i,0)>0}} \mu_{b\to i}\left(\mathrm{fc}(b,i,0) - \delta_{ab}\delta_{p0}\right) \right)$$

Again, the δs take care of the "$\setminus\{a\}$" in the original ground formulation and fc is a factor count. The factor count represents the number of equal ground factors that send messages to the variable at a given position. The "$\mathrm{fc}(b, j, 1/0) > 0$" simulates the ground sets "$\mathrm{nb}_{+/-}(j)$".

The role of the factor count can again be understood best when looking at Figure 3.1. Here, the factor count associated with the edge between $\mathfrak{X}_1$ and $\mathfrak{f}_1$ is $(1, 1)$ because $\mathfrak{X}_1$ represents the ground variables corresponding to X_1 and X_2. Both ground variables were involved once negated and once unnegated in the two factors represented by $\mathfrak{f}_1$. This leads to a positive factor count for both positions. Using this correspondence, the proof that applying the lifted formulas to the lifted graph gives the same results as WP applied to the ground network follows essentially from the one for the LBP in [209].

If we are interested in solving CNFs, we make use of WP in every iteration. By fixing one or several variables, other variables in the formula may now be implied directly by unit clauses. WP allows us to find these variables easily and at lower computational costs then running a more expensive inference algorithm such as BP. Of course, we could just run unit propagation on the simplified formula. However, with help of the message passing formulation, we were able to develop a lifted variant that directly operates on the lifted factor graph. This enables us now to do the simplification on the lifted level, getting us closer to a completely lifted decimation.

3.1.5 Lifted Survey Propagation

After we have seen the WP algorithm and its lifted version, we will now shift our focus to *Survey Propagation (SP)* [147, 27] and its lifted equations. SP has been very successful in solving large and difficult random CNFs. The difficulty of an k-CNF is often measured by the ratio of the number of clauses and the number of variables. A CNF with only few constraints and many variables is said to be under-constraint and it is likely that the problem has many solution where one of them can easily be found. Contrary, a CNF with a very large number of clauses is over-constraint and more likely to be unsatisfiable. It is conjectured that this ratio splits the space of all k-SAT problems in mainly satisfiable and unsatisfiable instances. The threshold or crossover point is often referred to as a_c. The ratio a_c plays an important role in k-SAT satisfiability and its theoretical analysis. Usually, SAT solvers have the longest running time in the region around a_c. There have been various experimental studies to determine a_c, e.g., in [43] for random 3-SAT problems where Crawford and Auton estimate a_c to be around 4.258.

However, we are interested in applying SP to non-random instances in order to speed up the message passing with help of lifting. SP has a nice interpretation in terms of warnings being sent in WP. SP assumes that WP has multiple fixed points and surveys represent the probability of a warning being send. More precisely, the intuition of a message $\mu_{a\to i}$ sent from a factor to a variable is that of a "survey" which represents the probability that a warning, as in WP, is sent from factor a to variable i. In comparison to WP, where the value of a warning was binary only, i.e., $\mu_{a\to i} \in \{0, 1\}$, the corresponding message in SP is real $\mu_{a\to i} \in [0, 1]$. The ground SP message being sent from a factor to a variable looks as follows [27]:

$$\mu_{a\to i} = \prod_{j\in \text{nb}(a)\setminus\{i\}} \left[\frac{\mu^u_{j\to a}}{\mu^u_{j\to a} + \mu^s_{j\to a} + \mu^*_{j\to a}}\right]$$

This message is a ratio of three different types of messages being sent from variables to factors. Besides the messages for the unsatisfied (μ^u) and the satisfied (μ^s) case, there exists also a message for the arbitrary/undecided case (μ^*). This is often referred to as the "don't care"-state. The three messages from variables to factors are defined the following way:

$$\begin{aligned}
\mu^u_{i\to a} &= \left[1 - \prod_{b\in \text{nb}^u_a(i)} (1-\mu_{b\to i})\right] \prod_{b\in \text{nb}^s_a(i)} (1-\mu_{b\to i}) \\
\mu^s_{i\to a} &= \left[1 - \prod_{b\in \text{nb}^s_a(i)} (1-\mu_{b\to i})\right] \prod_{b\in \text{nb}^u_a(i)} (1-\mu_{b\to i}) \\
\mu^*_{i\to a} &= \prod_{b\in \text{nb}(i)\setminus\{a\}} (1-\mu_{b\to i})
\end{aligned}$$

where $\text{nb}^u_a(i) = \text{nb}_+(i)$ and $\text{nb}^s_a(i) = \text{nb}_-(i) \setminus \{a\}$ if $s_{a,i} = 1$, while $\text{nb}^u_a(i) = \text{nb}_-(i)$ and $\text{nb}^s_a(i) = \text{nb}_+(i) \setminus \{a\}$ if $s_{a,i} = -1$. Similar to BP and WP, the initialization of messages can be done at random or with an initial fixed value. The different update schedules for BP which were describe in Section 2.4.1 can also be used for SP. In [27] for example, sequential random updates are suggested. If the message passing converges, we want to read off assignments from the surveys in the same spirit as it can be done from the warnings in WP. For SP, we calculate positive and negative biases of the variables:

$$\begin{aligned}
W^+_i &= \frac{b^+_i}{b^+_i + b^-_i + b^*_i}, \\
W^-_i &= \frac{b^-_i}{b^+_i + b^-_i + b^*_i},
\end{aligned}$$

where b^+_i, b^-_i, and b^*_i are defined as follows:

$$\begin{aligned}
b^+_i &= \left[1 - \prod_{a\in \text{nb}_+(i)} (1-\mu^*_{a\to i})\right] \prod_{a\in \text{nb}_-(i)} (1-\mu^*_{a\to i}) \\
b^-_i &= \left[1 - \prod_{a\in \text{nb}_-(i)} (1-\mu^*_{a\to i})\right] \prod_{a\in \text{nb}_+(i)} (1-\mu^*_{a\to i}) \\
b^*_i &= \prod_{a\in \text{nb}(i)} (1-\mu^*_{a\to i})
\end{aligned}$$

Each bias expresses the tendency of a variable to be either in the negated or unnegated state. Looking at the absolute difference of both biases, one can find the most biased variable which has the strongest tendency to be in a particular state.

It is difficult to give a simple example of SP akin to Example 3.3 for WP. On trees, it has been shown that SP reduces to WP [27], and is therefore exact. For cyclic graphs with $a_c < 3.9$, it has been reported that SP converges to a set of trivial messages, i.e., "don't care" states [27]. Therefore, meaningful examples quickly grow large and we now continue with the equations for *Lifted Survey Propagation (LSP)* and explain how the ground equations from above are rewritten to run on the lifted level. Essentially, the SP equations can be lifted in a similar way as for WP. That is, we apply rewriting rules such as "$\backslash\{i\}$"$\mapsto$"$\delta_{ij}\delta_{pp'}$" to the ground SP equations. This yields the following LSP equations. A message sent from factor a to a variable i at position p is defined as follows:

$$\mu_{a\to i,p} = \prod_{\substack{j\in \mathrm{nb}(a),\\ p'\in \mathrm{pos}(a,j)}} \left[\frac{\mu^u_{j\to a,p'}}{\mu^u_{j\to a,p'}+\mu^s_{j\to a,p'}+\mu^*_{j\to a,p'}}\right]^{\mathrm{vc}(a,j,p')-\delta_{ij}\delta_{pp'}}$$

Messages sent from variable i to factor a at position p becomes:

$$\mu^u_{i\to a,p} = \left[1-\prod_{\substack{b\in \mathrm{nb}(i),\\ p'\in \mathrm{pos}(b,i),p'\neq p}} (1-\mu_{b\to i,p'})^{\mathrm{fc}(b,i,p')-\delta_{ab}}\right] \prod_{\substack{b\in \mathrm{nb}(i),\\ p'\in \mathrm{pos}(b,i),p'=p}} (1-\mu_{b\to i,p'})^{\mathrm{fc}(b,i,p')-\delta_{ab}}$$

$$\mu^s_{i\to a,p} = \left[1-\prod_{\substack{b\in \mathrm{nb}(i),\\ p'\in \mathrm{pos}(b,i),p'=p}} (1-\mu_{b\to i,p'})^{\mathrm{fc}(b,i,p')-\delta_{ab}}\right] \prod_{\substack{b\in \mathrm{nb}(i),\\ p'\in \mathrm{pos}(b,i),p'\neq p}} (1-\mu_{b\to i,p'})^{\mathrm{fc}(b,i,p')-\delta_{ab}}$$

$$\mu^*_{i\to a,p} = \prod_{\substack{b\in \mathrm{nb}(i),\\ p'\in \mathrm{pos}(b,i)}} (1-\mu_{b\to i,p'})^{\mathrm{fc}(b,i,p')-\delta_{ab}\delta pp'}$$

Indeed, we have to be a little bit more careful. For messages from factors to variables, we use exactly the same rules. For the messages from variables to factors, however, we use slightly different rules. The set "$b \in \mathrm{nb}^{u/s}_a(i)$" maps to "$b \in \mathrm{nb}(i), p' \in \mathrm{pos}(b,i), p' \neq p$", respectively to "$\ldots, p' = p$", for the message sent at position p. This ensures that we consider exactly those neighbors of i which tend to make the clause a unsatisfied, respectively satisfied. In the same way as in WP, we can now apply these lifted equations to the lifted factor graph. In the following, we will investigate different SAT solving scenarios using message passing algorithms.

3.1.6 Lifted Decimation

We have described the idea of BP-based decimation already in Section 3.1.3 and we also indicated that SP could be used as an alternative inference algorithm to pick variables to clamp. The first way of lifting the decimation is lifting the factor graph before running the inference algorithm. Then we run the corresponding lifted variant of the inference algorithm on the compressed graph. After picking one or more variables to clamp based on the beliefs or surveys, we clamp the variables in the ground factor graph, lift the modified graph again and run LWP on the lifted graph to simplify the graph subsequently. This process is iterated until

```
input: A factor graph $G$, a list of query vars $\mathbf{Y}$, e.g., the variables in $G$
output: An assignment $\mathbf{a}$ to the variables in $\mathbf{Y}$
$\mathbf{a} \leftarrow \emptyset$;
$\mathfrak{G} \leftarrow$ lift($G$);
while $\mathbf{Y} \neq \emptyset$ do
    $\mathbf{b} \leftarrow$ runLiftedInference($\mathfrak{G}$); // run LBP or LSP
    $\mathbf{y}_t \leftarrow$ pickVarsToClamp($\mathbf{b}$); // based on beliefs or surveys
    $\mathbf{a} = \mathbf{a} \bigcup \mathbf{y}_t$;
    $\mathbf{Y} = \mathbf{Y} \setminus \mathbf{Y}_t$;
    $G \leftarrow$ clamp($G$, $\mathbf{y}_t$);
    $\mathfrak{G} \leftarrow$ lift($G$);
    $\mathfrak{G} \leftarrow$ simplify($\mathfrak{G}$); // based on LWP
    if containsContradition($\mathfrak{G}$) then
        return None;
    end
end
return $\mathbf{a}$;
```

Algorithm 3: Lifted Decimation

either a solution has been found, or alternatively a contradiction has been detected during the simplification. The pseudo code for the *lifted decimation* is depicted in Algorithm 3. Especially with LBP, LSP, and LWP at hand, we can execute every step now on the lifted level. We will analyze the performance gains by lifting in the next section.

3.1.7 Experiments

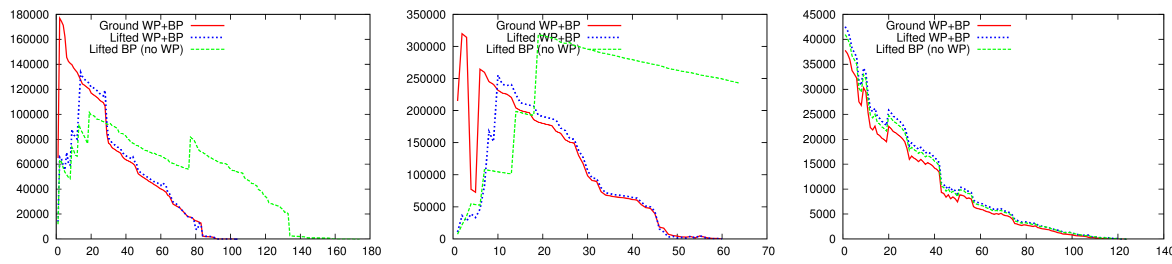
Our intention in this section is to empirically investigate the correctness of the lifted message updates for WP and SP, and compare their performance to the corresponding ground versions. We want to investigate whether message passing for SAT can benefit from lifting. Therefore, we define the following questions for this section:

(Q3.1) Do ground and lifted decimation return the same results?

(Q3.2) Does lifting speed up the decimation procedure for SAT solving?

To this aim, we implemented LWP and LSP in C++ using the LIBDAI library [155] for the underlying data structures. Based on the C++ code, we implemented the (lifted) decimation in Python which uses the C++ message passing as a subroutine. More precisely, we implemented the decimation approaches as described in Section 3.1.3 and Section 3.1.6. As a proof of concept, we compared the performances of lifted message passing approaches with the corresponding ground versions on three CNFs from a standard benchmark [75]. We use the decimation to measure the effectiveness and efficiency of the lifted algorithms.

To assess performance, we report run times. In order to abstract from the underlying physical hardware and programing language, run time is measured by the number of messages being sent. For the typical message sizes, e.g., for binary random variables with low degree, computing color messages is essentially as expensive as computing the actual messages of the inference algorithm. Therefore, we view the run time as the sum of both types of messages, i.e., color



(a) `2bitmax_6` (b) `ls8-norm` (c) `wff.3.150.525`

Figure 3.3: Total number of messages sent (y-axis) in each iteration (x-axis) of decimation for (lifted) WP+BP and lifted BP on several CNF benchmarks. (a) Lifted WP+BP saves 13% of the messages ground WP+BP sent. (b) Lifted WP+BP saves 16% of the messages ground WP+BP sent. (c) A small lifting overhead occurs on this random CNF.

and (modified) propagation messages, treating individual message updates as atomic unit time operations. BP messages were damped by a factor of 0.4, and the convergence threshold was set to 10^{-3}. We used parallel update schedules for (L)WP, (L)BP, and (L)SP. While the (L)BP messages were initialized uniformly, the (L)WP and (L)SP messages were initialized with zero.

We evaluated (lifted) WP+BP decimation on a circuit synthesis problem (`2bitmax_6`). The formula has 192 variables and 766 clauses. The resulting factor graph has 192 variable nodes, 766 factor nodes, and 1,800 edges. The statistics of the different variants are shown in Figure 3.3a. As one can see, lifting yields significant improvement in efficiency especially in the first iterations: LIFTED WP+BP reduces the messages sent by 13% compared to GROUND WP+BP when always fixing the most magnetized variable, i.e., the variable with the largest difference between negated and unnegated marginals. Additionally, the ground and the lifted version found the same solution to the problem.

Then, we investigated a Latin square construction problem of size eight (`ls8-norm`). The formula has 301 variables and 1,601 clauses, resulting in a factor graph with 3,409 edges. The statistics of running lifted message passing are shown in Figure 3.3b. Again, the lifting yields improvement in efficiency: LIFTED WP+BP saved 16% of the messages sent by the ground version. Both variants were also able to find the same solution.

Finally, we also applied the lifted message passing algorithms to a random 3-CNF, namely `wff.3.150.525`. The CNF contains 150 variables, 525 clauses and 1,575 edges. We did not expect compression on this problem instance because random structures typically do not provide any significant symmetries. As expected, no lifting was possible as shown in Figure 3.3c.

We also ran ground and lifted SP on these three problem instances. Due to the properties of the CNFs, SP always converged to a paramagnetic state, i.e., the trivial solution, in the first iteration. In such cases the problem is usually passed to a simpler SAT solver such as WALKSAT. Nevertheless LSP saved roughly 45% of the messages the ground version sent on the `2bitmax_6` problem. On the Latin square, LSP sent only 44% of messages ground SP sent. As expected, the lifting produced again a small overhead on the random 3-CNF for LSP.

To summarize, questions **Q3.1** and **Q3.2** where answered in favor of the introduced algorithms and the experimental results clearly show that a unified lifted message passing framework, including BP, WP, and SP, is indeed useful for SAT. Not only are significant efficiency gains obtainable by lifting, but lifted WP-guided BP clearly outperforms unguided (lifted) BP. On top of that, the assignments returned by the lifted decimation were identical to the results of the ground decimation.

3.1.8 Summary

In this section, we presented the first lifted message passing algorithms for determining the satisfiability of Boolean formulas. Triggered by the success of *Lifted Belief Propagation (LBP)* approaches, our algorithms construct a lifted CNF, represented as a lifted factor graph. The clusternodes and clusterfactors correspond to sets of variables and clauses that are sending and receiving the same messages. On these lifted data structures, we can then apply *Lifted Warning Propagation (LWP)*, respectively *Lifted Survey Propagation (LSP)*, yielding (approximate) information on satisfying assignments upon converge. The experimental results validate the correctness of the resulting lifted approaches as they coincide with the ground results. Significant efficiency gains are obtainable when employing lifted instead of standard WP, respectively SP. In particular, lifted decimation approaches based on LBP and LWP, or alternatively LSP and LWP, are faster than "just" using LBP to determine the satisfiability of Boolean formulas.

Of course, with decades of research in SAT solving, lifted SAT is not the first approach exploiting symmetries in CNFs. Symmetries have been studied in propositional and first-order logic for a long time and have already been related to graph isomorphism decades ago, e.g. [42] and further references in there. In SAT solving, symmetries are often considered to be undesirable because a search-based solver should avoid checking two symmetric branches. To do so, in symmetry breaking, e.g. [8], additional predicates are inserted that break symmetries but preserve the satisfiability of the CNF. Instead, the framework we presented in this section clusters symmetries to reduce run time. In addition, it nicely integrates itself into the lifted inference paradigm and benefits from existing implementations and ongoing research. Relating the idea of symmetry breaking back to probabilistic inference, one should further investigate an operation the other way around, i.e., adding evidence to the factor graph which makes the problem more symmetric without heavily changing the underlying probability distribution.

The SP experiments are by far not exhaustive yet and it is still an open question how LSP performs on other difficult but symmetric problems. For additional experiments, we require more suitable large-scale CNFs that contain appropriate symmetries. In particular, running lifted decimation based on SP+WP on structured problems, while enforcing that non-trivial solutions are returned by SP, can make this approach significantly more applicable.

Several extensions have been proposed to SP in recent years, extending SP to the maximum satisfiability (Max-SAT) or weighted Max-SAT case. For example, the *SP-y* algorithm [12] generalizes SP to handle Max-SAT problems. The *Relaxed Survey Propagation* (RSP) algorithm introduced in [36] allows to run a modified SP algorithm on weighted Max-SAT instances. Using extensions of SP that can handle weighted SAT problems, such as RSP, are also interesting for lifting because they could be used in inference on relational models such as MLNs. In particular, MLNs with deterministic and soft clauses asks for a tight integration of lifted SAT and lifted probabilistic inference. In many real-world applications, the problem formulation does not fall neatly into one of the classes. The problem may have one component that can be well-modeled as a pure SAT problem, while it contains another part better modeled as a probabilistic inference problem. We suggest to partition a problem into corresponding subnetworks, run the corresponding type of lifted message passing algorithm on each subnetwork, and combine the information from the different subnetworks. For the non-lifted case, this is akin to Duchi et al. [52]'s COMPOSE approach that has been proven to be successful for problems involving matching problems as subproblems.

Looking at the lifted decimation framework more closely, we can already identify a possible drawback of the naive lifting. As we have mentioned above, the approaches presented here apply the lifting as pre-processing step in every iteration. As we will see in Section 3.2, the lifting can be integrated more tightly into the decimation which yields a more elaborated lifted decimation approach, however, also requiring a more complex lifting scheme. Our current decimation approach does not take backtracking strategies into account. Many SAT solvers though make extensive use of backtracking which allows reverting earlier decisions and exploring other paths to the solution. Naturally, one could integrate backtracking in our approach as well. But more interestingly, one would like to investigate how well backtracking can benefit from lifting, i.e., connecting it again closer to symmetry breaking. Nevertheless, this section focused on the feasibility of lifted satisfiability. Naturally, many concepts and improvements from the past years or even decades could be included in the decimation as well, to further improve the results and reduce run time additionally.

As a last note, it is interesting to observe that WP can be seen as a simplified version of max-product BP. More precisely, WP is an instance of the *min-sum* algorithm with only binary variables and a particular initialization of the messages. We obtain the min-sum algorithm by running the message passing on the energy function based formulation of the MRF (Equation (2.3)). If we assume that $\mathcal{E}_c(\mathbf{x}_c) = 0$ if the constraint c is satisfied and 1 otherwise, it can be shown that the messages in the energy minimization form are equivalent to the WP updates if the messages are initialized with values from $\{0, 1\}$. Further details can be found in [146, Chapter 14.3]. Bearing this in mind, the existence of LWP can be derived from LBP, however, we showed in this section that we can also lift the simpler WP equations directly. Similarly, it has also been shown in [26] that the SP equations can be derived from the sum-product BP equations.

3.2 Sequential Clamping

As we have seen in the previous section and as it has been reported in the literature, BP and other lifted message passing algorithms can be extremely fast at computing approximate marginal probability distributions over single variable nodes. These marginals were used in the decimation procedure to solve SAT problems. They allow us to determine the magnetization of a variable or alternatively to sample a variable according to its marginal distribution. This computation works similarly well for marginals over neighboring nodes in the underlying graphical model. However, if we want to solve an entire CNF, sample a complete assignment to all variables, or sample an arbitrary subset of the variables, (lifted) message passing approaches leave space for improvement. For instance, in the BP+WP-guided decimation as described above, or in sampling configurations from a joint configuration [153, 146], one is essentially computing these probabilities in the same network but with changing evidence in a sequential procedure with re-lifting in every iteration.

While we have already discussed the importance of decimation approaches in satisfiability, obtaining samples from a joint distribution is equally important to AI and another example of a task fitting into the decimation framework. For example, this kind of sampling is often used in parameter learning. Using a decimation framework, we do not set the variable to its most magnetized state but instead sample a state according to its marginal distribution. We then clamp the variable in the network to the sampled state and continue as before.

In such cases where we want to apply lifted message passing to the same network with sequentially changing evidence, we want to avoid the naive solution that recomputes the lifted network in each step from scratch. This can possibly cancel the benefits of lifted inference and we also ignore information from the previous iterations. One can see this handicap in Algorithm 3, Line 9, where the graph is lifted in every iteration of the decimation.

As we will see in this section, we can do considerably better. Especially in cases where only additional evidence is added in each iteration, we want to exploit the knowledge from the previous lifting, to obtain the current lifted network more efficiently. Generally, we are interested in approximate lifted inference algorithms in such sequential scenarios that use information from previous iterations and scale to realistic domain sizes. This section makes a number of important and novel contributions to both the lifted BP and the decimation literature:

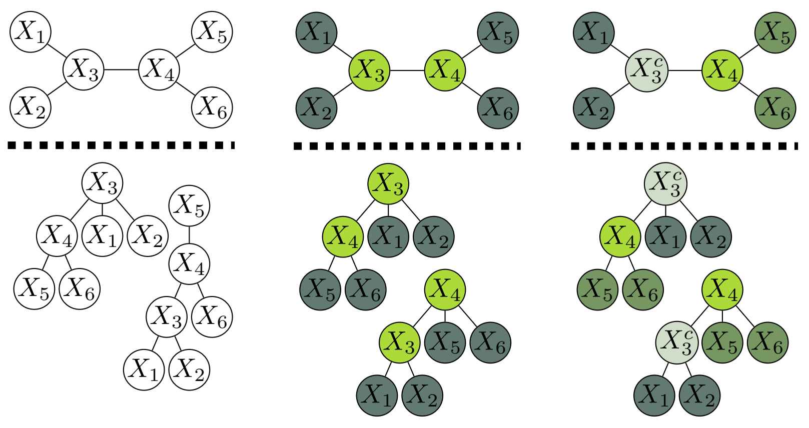
(C3.4) We present Shortest-Path-Sequences lifting for Lifted Belief Propagation (SPS-LBP). A scalable lifted inference approach for approximating solutions of complex inference tasks that is based on shortest paths sequences between variables in the factor graph. It does not lift in every iteration from scratch but instead uses the already known liftings from previous iterations. SPS-LBP avoids the repetitive lifting for each subtask by efficiently computing the corresponding lifted network directly from the one already known for the unconditioned case.

(C3.5) Besides demonstrating the application of SPS-LBP on lifted SAT solving, we also show its usefulness on another lifted inference task, namely lifted message passing guided sampling. The experimental results highlight that significant efficiency gains are obtainable compared to ground inference and naive lifting in each iteration.

(C3.6) By the first and second contribution, we continue to develop a lifted decimation framework and integrate the lifting more tightly into the decimation. The application to sampling also shows that lifted decimation is interesting beyond SAT solving.

There has been done some prior work on related problems and we have briefly touched upon online lifting already in Section 2.4.2. One particular lifted inference approach that is similar to the upcoming ideas, is the work by Nath and Domingos [165]. They essentially memorize the intermediate results of previous liftings. For new evidence, they make a warm start from the first valid intermediate lifting. So far, their approach has not been used for lifted sampling, nor lifted satisfiability. Additionally, the algorithm is only defined for MLNs and an MLN-specific lifted network construction, whereas our approach applies to any factor graph by relying on the more general Color Passing algorithm. Furthermore, the approach presented in this section gives a clear characterization of the core information required for sequential clamping for lifted message passing: the shortest paths sequences connecting the variables in the network.

We will now explain *Shortest-Path-Sequences (SPS)*-lifting in detail and provide a running example along with it. We will also give an in-depth analysis of it and sketch the soundness of SPS-lifting. As our experimental results will highlight, SPS-lifting is capable of saving further messages sent in the decimation compared to the naive approach.



(a) (b) (c)

	$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{X}_3$	$\mathbf{X}_4$	$\mathbf{X}_5$	$\mathbf{X}_6$
$\mathbf{X}_1$	0	2	1	2	3	3
$\mathbf{X}_2$	2	0	1	2	3	3
$\mathbf{X}_3$	1	1	0	1	2	2
$\mathbf{X}_4$	2	2	1	0	1	1
$\mathbf{X}_5$	3	3	2	1	0	1
$\mathbf{X}_6$	3	3	2	1	1	0

(d)

Figure 3.4: (a) Original factor graph. (b) Prior lifted network, i.e., lifted factor graph without clamped variables. (c) Lifted factor graph when X_3 is clamped, indicated by the "c" superscript. Factor graphs are shown (top) with corresponding colored computation trees (bottom). For the sake of simplicity, we assume identical factors but omit them in the graphical representation. The shades in (b) and (c) encode the clusternodes. (d) Shortest path distances of the nodes. The i-th row will be denoted as dist_i in the text.

3.2.1 Lifted Sequential Inference

When we turn a complex inference task into a sequence of simpler tasks, we are repeatedly answering slightly modified queries on the same graph. Because LBP and LWP generally lack the opportunity of adaptively changing the lifted graph and using the updated lifted graph for efficient inference, they are doomed to lift the original model in each of the k iterations again from scratch as it was already depicted in Algorithm 3. Each Color Passing run scales $\mathcal{O}(h \cdot |E|)$ where $|E|$ is the number of edges in the factor graph and h is the number of iterations required. Hence, we can spend up to $\mathcal{O}(k \cdot h \cdot |E|)$ time just on lifting if we clamp k variables in a sequential fashion. The previous section already showed how BP-guided decimation fixed one variable after another. Depending on the nature of the CNF, or more generally on the structure of the factor graph, this can require up to $k = n$ iterations.

Let us now consider BP-guided sampling which can also be casted into the framework of decimation. We will explain its idea in greater detail and show how it poses similar issues for lifting due to changing evidence. When we want to sample from the joint distribution over k variables, this can be reduced to a sequence of one-variable samples conditioned on a subset of the other variables [146]. Thus, to get a sample for $\mathbf{X} = (X_1, \ldots, X_k)$, we first compute $p(X_1)$, then $p(X_2|X_1)$, $\ldots$, $p(X_k|X_1, \ldots X_{k-1})$. This problem can also be casted into the framework in Algorithm 3 and one should keep in mind that a CNF represented as a factor graph defines a joint distribution where each solution has equal probability. Let us now exemplify the idea of BP-guided sampling with a small example.

Example 3.4. *Assume we want to sample from the joint distribution $p(X_1, \ldots, X_6)$, given the network in Figure 3.4a(top). Further assume that we begin our sequential process by first computing $p(X_3)$ from the prior lifted network, i.e., the lifted network when no evidence has been set (depicted in Figure 3.4b(top)). After sampling a state x_3, we want to compute $p(X_{\setminus 3}|x_3)$ as shown in Figure 3.4c(top).*

To do so, it is useful to describe BP and its operations in terms of its *computation tree (CT)*, see e.g. [97]. The CT is the unrolling of the (loopy) graph structure where each level i corresponds to the i-th iteration of message passing. Similarly, we can view Color Passing as a *colored computation tree (CCT)*. More precisely, one considers for every node X the CT rooted in X but now each node in the tree is colored according to the nodes' initial colors (cf. Figures 3.4a to 3.4c(bottom)). Each CCT encodes the root nodes' local communication patterns that show all the colored paths along which node X communicates in the network. Consequently, CP groups nodes with respect to their CCTs: nodes having the same set of rooted paths of colors (node and factor names neglected) are clustered together.

Example 3.5. *For instance, Figure 3.4a(bottom) shows the CTs rooted in X_3 and X_5. Because their set of paths are different, X_3 and X_5 are clustered into different clusternodes as indicated by different colors in Figure 3.4b(top). For this prior lifted network, the light green nodes exhibit the same communication pattern in the network which can be seen in identical CCTs in Figure 3.4b(bottom), and were consequently grouped together. Now, when we clamp the node X_3 to a value x_3, we change the communication pattern of every node having a path to X_3. Specifically, we change X_3's, and only X_3's, color in all CCTs where X_3 is involved, as indicated by the "c" in Figure 3.4c. This affects nodes X_1 and X_2 differently than X_4, respectively X_5 and X_6, for two reasons:*

1. *they have different communication patterns as they belong to different clusternodes in the prior network*
2. *more importantly, they have different paths connecting them to X_3 in their CCTs*

The shortest path is the shortest sequence of factor colors connecting two nodes. Since we are not interested in the paths but whether the paths are identical or not, these sets might as well be represented as colors. Note that in Figure 3.4 we assume identical factors for simplicity. Thus in this case path colors reduce to distances. In the general case, however, we compare the paths, i.e., the sequence of factor colors.

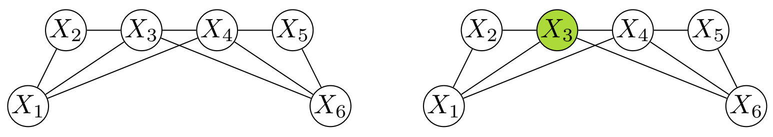
Example 3.6. *The prior lifted network can be encoded as the vector $l = (0, 0, 1, 1, 0, 0)$ of node colors. Thus, to get the lifted network for $p(X_{\setminus 3}|x_3)$, as shown in Figure 3.4c, we only have to consider the vector dist_3 of shortest paths distances to X_3 (see Figure 3.4d) and refine the initial clusternodes correspondingly. This is done by*

1. *the element-wise concatenation of two vectors: $l \oplus \mathrm{dist}_3$*
2. *viewing each resulting number as a new color*

For our example, we obtain:

$$(0, 0, 1, 1, 0, 0) \oplus (1, 1, 0, 1, 2, 2) =_{(1)} (01, 01, 10, 11, 02, 02) =_{(2)} (3, 3, 4, 5, 6, 6) ,$$

which corresponds to the lifted network for $p(X_{\setminus 3}|x_3)$ as shown in Figure 3.4c. Thus, having the shortest path matrix, we can directly update the prior lifted network in linear time without taking the detour through running CP on the ground network. Now, we run inference, sample a state $X_4 = x_4$ afterwards, and compute the lifted network for $p(X_{\setminus\{3,4\}}|x_4, x_3)$ to draw a sample for $p(X_1|x_4, x_3)$. Essentially, we proceed as before: compute $l \oplus (\mathrm{dist}_3 \oplus \mathrm{dist}_4)$.



	$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{X}_3$	$\mathbf{X}_4$	$\mathbf{X}_5$	$\mathbf{X}_6$
$\mathbf{X}_1$	0	1	1	1	2	2
$\mathbf{X}_2$	1	0	1	2	3	2
$\mathbf{X}_3$	1	1	0	1	2	1
$\mathbf{X}_4$	1	2	1	0	1	1
$\mathbf{X}_5$	2	3	2	1	0	1
$\mathbf{X}_6$	2	2	1	1	1	0

(c)

Figure 3.5: Toy example of a graph where a single run of Shortest-Paths-lifting fails to return the correct lifting.

However, the resulting network might be suboptimal in cases when more than one variables is clamped and variables from the same initial cluster are sampled identically, i.e., take on the same value in the sample.

Example 3.7. *The concatenation in the previous example assumed $x_3 \neq x_4$ and, hence, X_3 and X_4 cannot be in the same clusternode. For $x_4 = x_3$, they could be placed in the same clusternode because they were in the same clusternode in the prior network. If X_3 and X_4 are clamped, this can be checked by* $\text{dist}_3 \odot \text{dist}_4$*, the element-wise sort of two vectors. In our case, this yields $l \oplus (\text{dist}_3 \odot \text{dist}_4) = l \oplus l = l$: the prior lifted network.*

In general, we compute

$$l \oplus (\bigoplus_{\mathfrak{r}}(\bigoplus_{s} \text{dist}_{\mathfrak{r},s})) \ ,$$

where

$$\text{dist}_{\mathfrak{r},s} = \bigodot_{i\in\mathfrak{r}:x_i=s} \text{dist}_i \ ,$$

with clsuternodes $\mathfrak{r}$ and the truth values s. For an arbitrary network, however, the shortest paths might be identical although the nodes have to be split, i.e., they differ in a longer path, or in other words, the shortest paths of other nodes to the evidence node are different. Consequently, we apply the shortest paths lifting iteratively. Let CN_E denote the clusternodes given the set E as evidence. By applying the shortest paths procedure, we compute $CN_{\{X_1\}}$ from $CN_\emptyset$. This step might cause initial clusternodes to be split into newly formed clusternodes. To incorporate these changes in the network structure the shortest paths lifting procedure has to be iteratively applied. Thus in the next step we compute $CN_{\{X_1\}\cup\Delta_{X_1}}$ from $CN_{\{X_1\}}$, where Δ_{X_1} denotes the changed clusternodes of the previous step. This procedure is iteratively applied until no new clusternodes are created. We exemplify this issue by introducing another example where a single run of shortest paths lifting fails, however, an iterative application returns the correct lifting.

Example 3.8. *The initial lifting of the graph depicted in Figure 3.5a can be encoded as $l = (0, 1, 2, 2, 1, 0)$. If we now clamp variable X_3 as shown in Figure 3.5b, we can use the distances in Figure 3.5c to concatenate l and* dist_3*:*

$$(0, 1, 2, 2, 1, 0) \oplus (1, 1, 0, 1, 2, 1) = (01, 11, 20, 21, 12, 01) = (0, 1, 2, 3, 4, 0) \ .$$

Yet, this lifting is not correct, as we have to distinguish X_1 and X_6. We also observe that the clusternodes of X_2, X_4, and X_5 have changed. Therefore, we now have to iteratively apply the shortest paths lifting based on the nodes that changed in the previous iteration:

$$\begin{aligned}
&(0,1,2,3,4,0)\\
\oplus&(1,0,1,2,3,2)\\
\oplus&(1,2,1,0,1,1)\\
\oplus&(2,3,2,1,0,1)\\
=&(0112,1023,2112,3201,4310,0211) = (0,1,2,3,4,5)\ .
\end{aligned}$$

Since we are now at the ground level anyhow, we can stop iterating.

The description above together with Example 3.8 essentially sketch the proof of the following theorem. Originally, the theorem in a slightly different way, together with its proof, were presented by Ahmadi et al. [2].

Theorem 3.1. *If the shortest path colors among all nodes and the prior lifted network are given, computing the lifted network for $p(\mathbf{X}|X_k,\dots,X_1)$, $k>0$, takes $\mathcal{O}((k+h)\cdot n)$, where n is the number of nodes and h is the number of required iterative applications of the concatenation. Furthermore, running LBP produces the same results as running BP on the original model.*

Proof. Assuming a graph $G=(V,E)$. We have seen above that the concatenation is a linear operation in the number of nodes. When k nodes have been set, we have to concatenate k distance vectors. However, this concatenation can result in supernodes being changed. Consequently, this requests the concatenation of additional nodes. This iterative application can result in h additional concatenations.

When we set new evidence for a node $X \in V$ then for all nodes within the network the color of node X in the *CCTs* is changed. If two nodes $Y_1, Y_2 \in V$ were initially clustered together and belonged to the same clusternode, i.e., $\mathfrak{X}(Y_1)=\mathfrak{X}(Y_2)$, they have to be split if the *CCTs* differ. Now, we have to consider two cases:

1. If the difference in the *CCTs* is on the shortest path connecting X with Y_1 and Y_2, respectively, then shortest path lifting directly provides the new clustering.
2. If the coloring along the shortest paths is identical, the nodes' *CCTs* might change in a longer path. Since $\mathfrak{X}(Y_1)=\mathfrak{X}(Y_2)$ there exists a mapping between the paths of the respective *CCTs*. In particular $\exists Z_1, Z_2$, s.t. $\mathfrak{X}(Z_1)=\mathfrak{X}(Z_2)$ from a different clusternode, i.e., $\mathfrak{X}(Z_i)\neq\mathfrak{X}(Y_i)$, and

$$Y_1,\dots,\underbrace{Z_1,\dots,X}_{\Delta_1} \in CCT(Y_1), Y_1,\dots,\underbrace{Z_2,\dots,X}_{\Delta_2} \in CCT(Y_2)\ ,$$

   and $\Delta_1 \in CCT(Z_1) \neq \Delta_2 \in CCT(Z_2)$ are the respective shortest paths for Z_1 and Z_2. Thus, by iteratively applying shortest-path lifting as explained above, the evidence propagates through and we obtain the new clustering. □

In fact, SPS lifting can be quite fast and we will now show in our experimental evaluation how the lifting time can be decreased for tasks with evidence arriving sequentially, e.g., lifted satisfiability or lifted sampling of joint configurations. If, however, lifting does not pay off, computing the pairwise distances may produce an overhead.

```
input: A factor graph $G$, list of query vars $\mathbf{Y}$
output: An assignment $\mathbf{a}$ to the variables in $\mathbf{Y}$
$\mathfrak{G} \leftarrow$ compress($G$);
$D \leftarrow$ calcDistances($G$);
$\mathbf{a} \leftarrow \emptyset$;
while $\mathbf{Y} \neq \emptyset$ do
    $\mathbf{b} \leftarrow$ runLiftedInference($\mathfrak{G}$);
    $\mathbf{y}_t \leftarrow$ pickVarsToClamp($\mathbf{b}$);
    $\mathbf{a} = \mathbf{a} \bigcup \mathbf{y}_t$;
    $\mathbf{Y} = \mathbf{Y} \setminus \mathbf{Y}_t$;
    clamp($\mathfrak{G}$, $\mathbf{y}_t$);
    adaptLifiting($\mathfrak{G}$, $D$); // SPS-lifting
    // only for SAT
    simplify($\mathfrak{G}$); // based on LWP
    if containsContradition($\mathfrak{G}$) then
        return None;
    end
end
return a
```

Algorithm 4: Lifted Decimation using SPS-lifting

3.2.2 Experiments

After we have described the idea of SPS-lifting, we will now show how it can be integrated into an improved lifted decimation framework and we want to investigate the following questions:

(Q3.3) Can we further support the results from the previous section and show that lifted decimation solves satisfiability problems more efficiently?

(Q3.4) Does lifting improve sequential inference approaches beyond lifted satisfiability?

(Q3.5) Is SPS-lifting even more beneficial than naive lifting based on CP?

Therefore, we run experiments on two AI tasks in which sequential clamping is essential. Namely, BP-guided decimation for satisfiability problems and sampling configurations from MLNs. With SPS-lifting, both tasks essentially follow the decimation strategy shown in Algorithm 4. However, in the case of sampling, we do not run LWP and check for contradictions. Additionally, the variable is not clamped based on the magnetization. Instead, we sample the value of the variable based on its marginal belief. The previous section already contained initial experiments for lifted satisfiability, however, the lifted decimation was implemented in a naive way. This initial investigation of **Q3.2** will now be extended to additional problem instances and supported by a secondary set of experiments based on a different inference task.

Lifted Satisfiability

As indicated above, our lifted satisfiability based on decimation fits well into the sequential setting. However, we now use a more elaborated variant compared to the one depicted in Algorithm 3. It distinguishes most importantly from avoiding the entire re-lifting from scratch

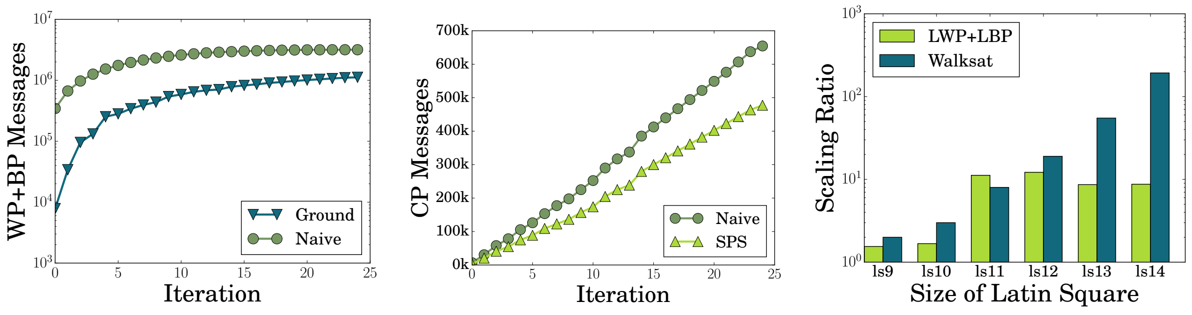
CNF Name	**# Iters.**	**Ground**	**Naive**	**SPS**	**Walksat**	
ls8-normalized	26	3.17	1.12	0.95	540	
ls9-normalized	13	5.47	1.65	1.47	1,139	
ls10-normalized	14	10.27	1.84	1.59	1,994	
ls11-normalized	26	38.82	11.51	10.64	4,500	
ls12-normalized	35	60.83	13.15	11.57	10,351	structured
ls13-normalized	21	55.39	9.99	8.21	30,061	
ls14-normalized	22	83.30	10.22	8.30	104,326	
2bitmax_6	55	2.35	1.25	1.05	379	
5_100_sd_schur	53	111.19	75.98	64.91	1,573,208	
wff.3.100.150	54	0.19	0.26	0.22	17	
wff.4.100.500	78	1.73	2.04	1.89	33	random
wff.3.150.525	126	6.36	6.76	6.56	284	

Table 3.2: Total messages sent (in millions) in SAT experiments and the number of average flips needed by Walksat (last column).

in each iteration. We first run LBP on the lifted graph and use its marginals to fix the next variable. Based on this clamping, the lifting is updated using SPS-lifting. We then run LWP on the modified lifted factor graph to clamp directly implied variables as before. Again, LWP is also used to detect possible contradictions. When LWP finds a contradiction, the algorithm stops and does not return a satisfying configuration. Otherwise, we continue by running LBP again.

We continue to compare the performance of lifted message passing approaches with the corresponding ground versions on the previously used CNF benchmark from [75]. We use the decimation as described to measure the effectiveness of the algorithms. To assess performance, we again report the number of messages sent. As before, for the typical message sizes, e.g., for binary random variables with low degree, computing color messages is essentially as expensive as computing the actual messages. Therefore, we report both color and (modified) BP messages, treating individual message updates as atomic unit time operations. We use the parallel message protocol for (L)BP and (L)WP where messages are passed from each variable to all corresponding factors and back at each step. The convergence threshold was set to 10^{-8} for (L)BP and all messages were initialized uniformly. In the case of (L)WP, all messages were initialized with zero, i.e., no warning being sent initially. As mentioned above, it is usually necessary to iteratively apply the SPS-lifting to obtain the correct adapted lifted graph. The number of required iterations, however, can be high if long paths occur in the network. Therefore, we use the SPS-lifting only once but then continued with standard CP to determine the new lifting. This can still save several passes of CP because the SPS-lifting provides us with a good head start.

We evaluated (Lifted) WP+BP decimation on different CNFs, ranging from problems with about 450 up to 78,510 edges. The CNFs contain structured problems as well as random instances. The statistics of the runs are shown in Table 3.2. As one can see, naive lifting already yields significant improvement, further underlining the experiments from the previous section. When applying the SPS-lifting, we can do even better by saving additional messages in the compression phases. The savings in messages are visible in running times as well. Looking



(a) Ground vs. Lifted (b) Naive vs. SPS-lifting (c) Comparison to WALKSAT

Figure 3.6: Experimental results for lifted decimation on Latin squares. (a) Comparison of ground decimation with naive lifted decimation on `ls8-normalized`. (b) Comparison of naive lifting with SPS-lifting on `ls8-normalized`. (c) Comparison of the growth in computational costs on increasing problem sizes measured relative to the smallest problem.

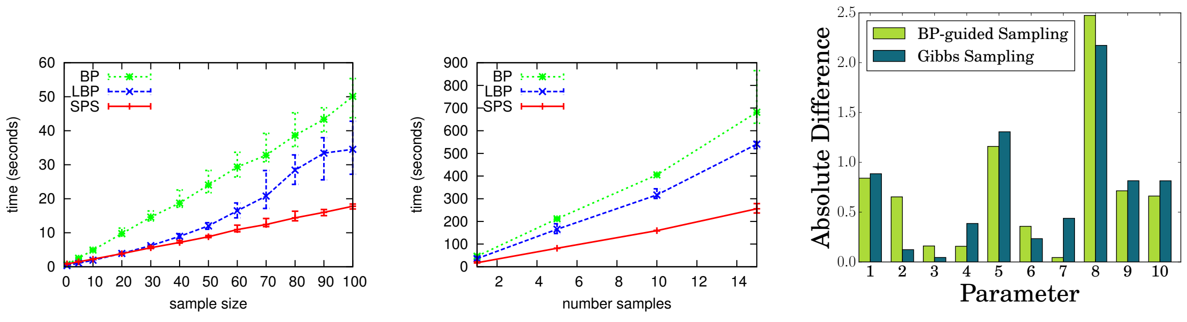
at Figure 3.6a, we compare ground decimation with its lifted counterpart. In the lifted case, we only send 33% of the total ground messages. When using the SPS-lifting, we can save up to an additional 10% of messages sent in the compression phase (Figure 3.6b). In the decimation, we always clamp the most magnetized variable which is the variable having the largest difference between the probability of the `True` and `False` state. We also applied the lifted message passing algorithms to random CNFs (last three rows in Table 3.2). As expected, no lifting was possible because random instances do usually not contain symmetries. In our experiments, we were able to find satisfying solutions for all problems which validates the effectiveness of the (lifted) decimation.

Although we are not aiming at presenting a state-of-the-art SAT solver, we solved all problems using Walksat [196] as well. Hence, we report results measured in variable flips in Table 3.2 too. Although Walksat requires fewer flips than we send messages, one can see that our lifted decimation strategy still scales well. In Figure 3.6c we have compared the computational effort on increasing problem sizes for Walksat and our lifted decimation. The results indicate that our approach can handle large problem instances without employing complex heuristics and code optimization but exploiting symmetries in the problems.

In combination with the results from the previous section, **Q3.3** and **Q3.5** have clearly been answered in favor of our algorithms for the task of lifted SAT solving. The next experiments will show that this also holds in the case of lifted sampling.

Lifted Sampling

We investigated BP, LBP and SPS-LBP for sequentially sampling a joint configuration over a set of variables, i.e., for a sequence of one-variable samples conditioned on a subset. Thus, to get a sample for $\mathbf{X} = \{X_1, \ldots, X_n\}$, we first compute $p(X_1)$, then $p(X_2|X_1)$, ..., $p(X_n|X_1, \ldots X_{n-1})$ as shown in Algorithm 4 for the SPS-case. In contrast to the decimation for SAT solving, the procedure of picking a variable to clamp is solely based on the index of the variables. Additionally, we now sample a state from the computed BP marginals instead of clamping a variable to the most magnetized state. Figure 3.7 summarizes the results for the "Smokers-and-Friends" dynamic MLN with ten people over ten time steps. This MLN over time was described in Section 2.3.1.



(a) (L)BP-guided sampling for a varying sample size

(b) (L)BP-guided sampling for a varying number of samples

(c) Absolute difference of the learned parameters

Figure 3.7: Experimental results for Lifted BP guided sampling.

In our first experiment, we randomly chose 1, 5, 10, 20, 30, . . ., 100 "cancer" nodes over all time steps and sampled from the joint distribution. As one can see in Figure 3.7a, LBP already provides significant improvement compared to BP, however, as the sample size increases, the speed-up is lower. The more evidence we have in the network, the less lifting is possible. SPS-LBP comes with an additional reduction in runtime as we do not need to perform the lifting in each step from scratch.

In our second experiment, we fixed the sample size to 100, i.e., we sampled from the joint distribution of all `cancer(x,t)` predicates for all persons `x` in the domain and all time steps `t`. We drew 1, 5, 10, and 15 samples and the timings are averaged over five runs. Figure 3.7b shows that LBP is only slightly advantageous compared to BP, as the sample size is 100, especially in the later iterations we have lots of evidence and long chains to propagate the evidence. Repeatedly running CP almost cancels the benefits. SPS-LBP on the other hand, shows significant speed-ups.

To evaluate the quality of the samples, we drew 100 samples of the joint distribution of all variables using the BP-guided approach and Gibbs sampling respectively. We learned the parameters of the model maximizing the conditional marginal log-likelihood using scaled conjugate gradient. Figure 3.7c shows the absolute difference of the learned weights from the model the datacases were drawn. To summarize the error into a single number, we calculated the RMSE (see Equation (2.13)). The RMSE for the BP parameters was 0.31 and for Gibbs parameters 0.3. As one can see, parameter learning with BP-guided samples performs as good as with samples drawn by Gibbs sampling.

This experiment for lifted sampling completes our evaluation of the SPS-lifting for sequential clamping and also answers **Q3.4** in favor of our lifted decimation. We have seen that lifting, and in particular SPS-lifting, speeds up the sampling of joint configurations and confirms the positive answer to question **Q3.5**. At the same time, BP-guided sampling achieves comparable quality to an alternative approach base on Gibbs sampling.

3.2.3 Summary

In this section, we presented an adaption to the original Color Passing algorithm for lifting. To avoid the lifting from scratch in each iteration of the naive realization, we employed an efficient sequential clamping approach. The SPS-lifting also gives a novel characterization of the main information required for lifting in terms of shortest paths in a given network. The experimental results were conducted on two tasks for lifted inference, namely Boolean satisfiability and sampling joint configurations from MLNs. These experiments validate the correctness of the proposed lifted decimation framework and demonstrate that instantiations can actually be faster than "just" using standard lifting.

Since lifting SAT solvers themselves and the exploitation of efficient sequential lifting for them is a major advance, one should further look into the integration of SPS-lifting for Survey Propagation because SP has proven to be very powerful for solving certain classes for CNFs. Initial experiments were conducted in the previous section already, but for structured CNFs, we are often facing the problem of trivial surveys returned by SP. One should investigate if the choice of a variable to clamp can directly influence the results of SP and its convergence.

As mentioned above, in the lifted sampling experiments, the variable to clamp was solely picked based on its index in the problem formulation. Other algorithms use the idea of conditioning to construct approximate inference algorithm based on BP, e.g., CBP-BBP presented in [53], and use more elaborated methods to select a variable to clamp. Both, lifting the CBP-BBP-algorithm, as well incorporating their ideas on picking a variable to clamp into our decimation framework are interesting avenues for future research and still open questions.

Lastly, if we are clamping the factor graph over the iterations, we will most likely end up with a factor graph with almost all variables being set and little opportunities for lifting. Hence, one should also look for heuristics that stop compressing when almost all variables are clamped and lifting is too costly.

While this section showed how lifting can be efficiently exploited over several runs of inference with changing evidence, we will now turn our attention to an atomic inference run. For iterative MAP inference algorithms, one should also consider adapting the lifting over the iterations of message passing to possibly speed up run time.

3.3 Bootstrapped Inference with Lifting

The previous sections focused on marginal inference and we will turn our attention now to MAP inference. It was already mentioned that MAP inference is an important, yet challenging, problem and it was discussed how an approximate MAP assignment can be obtained by max-product BP. In this section, we will go into further detail how an approximate MAP assignment can be obtained by other means than BP. Specifically, we will consider a known *Likelihood Maximization (LM)* approach [129] which was empirically shown to have a convergence rate often significantly higher than approaches such as MPLP [70]. LM increases a lower bound on the MAP value while being fairly simple to implement. Intuitively, Kumar and Zilberstein approximate the original distribution by a simpler one and show that this approximation often yields satisfying results. This approximation imposes the constraint on the model distribution that any probability over subsets of variables is equal to the product of their single node marginals. Kumar and Zilberstein derive *Expectation Maximization (EM)* [48] equations to optimize the objective and show that their algorithm can be implemented using a message passing paradigm. LM quickly finds high quality solutions and yet the message updates are remarkably simple. Specifically, as our first contribution, we show that LM is liftable in a bottom-up fashion akin to the lifting of BP and the message passing algorithms for satisfiability presented in the previous section.

Specifically, we introduce lifted LM equations that are used on the lifted problem formulation. Here, we also obtain the lifted model by applying a variation of the CP algorithm to the ground model. This contribution further extends the family of known lifted MAP approaches. However, one common issue in lifting is evidence which often destroys symmetries in the model. The more evidence is given, the more propositional the problem becomes, ultimately canceling the benefits of lifted inference. It is easy to generate evidence cases that actually cancel the benefits of lifted inference completely. Hence, the main contribution of this section is to show, how to employ additional structure for optimization provided by LM, to further speed up lifted inference. In particular, we will introduce the concept of *Pseudo Evidence (PE)*.

For LM, it has been recognized that pseudo marginals may converge quickly when forcing updates to be as greedy as possible [223]. We interpret this greedy update rule as ultimately inducing Pseudo Evidence: if a pseudo marginal is close to one state, clamp the corresponding variable accordingly. As we will show empirically, this can already considerably speed up inference in the propositional case and reduce the number of messages being sent. However, the model could potentially be simplified even further over the inference iterations, but the naive *Lifted Likelihood Maximization (LLM)* approach cannot make use of this new evidence to speed up lifted inference in later iterations. Consequently, we propose an efficient LLM version for updating the structure of the lifted network with Pseudo Evidence over the iterations.

To do so, one is tempted to just lift the network again when new Pseudo Evidence is "observed" using any online lifted inference approach and related approaches (see Section 2.4). However, we can do significantly better. Existing online approaches simply compute a more shattered model, should the new evidence require to do so. But note that the current lifted model is always valid for the remaining iterations because we always set entire clusters as Pseudo Evidence. Therefore we can simply lift the current lifted model and obtain models monotonically decreasing in size. Indeed, this is akin to Bui et al. [31] who show how soft evidence affects lifting. Bui et al. exploit symmetries before soft evidence is obtained. So, in contrast to existing lifted inference approaches, we add evidence over the iterations and use this additional evidence for lifting. Indeed, this bootstrapped[3] LM using Pseudo Evidence is akin to lifted "relax, compensate and then recover" (RCR) [231] because messages to clamped variables do not have to be calculated anymore and these edges can be deleted. However, whereas RCR first relaxes and then compensates, we just "reduce and re-lift" over the iterations.

To summarize, this section makes the following contributions:

(C3.7) We show that Kumar and Zilberstein's LM can be lifted using a Color Passing algorithm.

(C3.8) For MAP inference, we exploit Pseudo Evidence to save message computations without decreasing the quality of the solution drastically.

(C3.9) By employing bootstrapped re-lifting, we compress the problem additionally in later iterations.

(C3.10) We show how to re-lift solely on the lifted level without resorting to the ground model.

Our experimental results on MLNs, Ising models, image segmentation and entity resolution show that *Bootstrapped Lifted LM (BLLM)* can yield considerable efficiency gains compared to standard LM and naive LLM.

We proceed as follows. We start off by reviewing the LM approach from [129]. We then develop its naive lifted variant. Afterwards, we show how to obtain Pseudo Evidence and how

[3]Here, we refer to bootstrapping as the capability of self-improvement during the inference stage. In learning settings, bootstrapping also refers to iteratively improved learning where the learner adds previously learned concepts to the training set, e.g. [156].

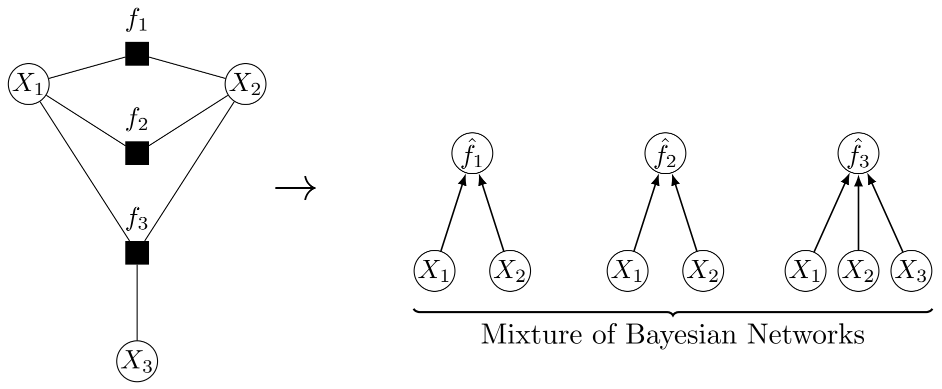


Figure 3.8: Kumar and Zilberstein's LM approach relies on the fact that a factor graph is decomposed into a mixture of Bayesian networks. Here, the factor graph composed of three variables is split into three Bayesian networks, each containing a reward variable $\hat{f}_a$ for the corresponding factor.

it can be used to efficiently re-lift the already lifted model. Before discussing the new approach and putting it into the context of the previous sections, we will present our experimental evaluation.

3.3.1 Lifted LM

Inspired by the equivalence between MAP inference and solving a linear program, Kumar and Zilberstein place a constraint on the probability distribution in the definition of the marginal polytope $\mathcal{M}(G)$ (see Equation (2.4)). More details on the marginal polytope can be found in Section 2.1. More precisely, they constrain the original distribution in such a way that the following holds

$$p(\mathbf{x}) = \prod_{i=1}^{n} p_i(x_i)$$

and afterwards maximize over the resulting set of mean parameters. This set of parameters defines an inner bound of the marginal polytope. Kumar and Zilberstein show that this maximization problem can be reformulated as likelihood maximization in a finite-mixture of simple *Bayesian networks (BNs)*. The hidden variables in the mixture are the X_i in the original MRF. The potentials f_a are incorporated via "binary reward variables" $\hat{f}_a$ and the conditional probability distribution of $\hat{f}_a$ is proportional to f_a, i.e., for every potential f_a in the MRF, there exists a BN with a reward variable and its parents being x_a. Figure 3.8 shows how an example factor graph is transformed into a mixture of BNs. The optimization of this new objective cannot be done using linear programming techniques anymore because the restrictions on the probability distribution are now non-linear. Therefore, Kumar and Zilberstein introduce an EM approach that monotonically increases the lower bound of the MAP. While in the original work the equations for pairwise MRFs were derived only, we will here give the EM message passing equations in an adapted form for arbitrary factor graphs. A message sent from a factor a to one of its neighboring variables i is defined as:

$$\mu_{a\to i}(x_i) = \sum_{\neg\{i\}} f_a(\mathbf{x}_a) \prod_{j\in \text{nb}(a)\setminus\{i\}} p_j(x_j),$$

as before, $\neg\{i\}$ denotes all possible instantiations of the variables in the domain of factor a with variable X_i fixed to x_i and $\mathrm{nb}(a)$ denotes the variables connected to factor i. Recalling the BP message from a factor to a neighboring variable in Equation (2.8), the LM message looks very similar. However, we do not have a message in the other way around involved but instead solely the current beliefs $p_j(x_j)$ of the marginal probabilities of the neighboring variables. This simplifies the message passing. The M-Step updates these beliefs in every iteration as follows:

$$p_i'(x_i) = \frac{1}{Z_i} p_i(x_i) \sum_{a \in \mathrm{nb}(i)} \mu_{a \to i}(x_i) \; ,$$

where Z_i is a normalization constant for variable X_i. Again as in BP, or the other message passing algorithms we have seen so far, the process of sending messages is iterated until convergence. In the case of LM, the stopping criterion is defined over the difference of the beliefs in two subsequent iterations. For what follows, it is important to note that a modification of this M-Step is common in order to speed up convergence: the so called *soft greedy M-Step*, originally proposed in [223], is used to weight the current expectation stronger. Its usefulness has been shown empirically in the literature.

We now derive the formulas for the lifted variant of LM. As we have explained above (see Section 2.4.1), lifting can be viewed as a color passing procedure that finds symmetries in a given factor graph by sending color messages. For BP, the idea of the CP algorithm was to find variables that have the same marginal distributions under the message passing algorithm following a parallel update schedule. Similarly, this held for warnings in WP and surveys in SP. For the lifted LM, this means that one identifies variables that have the same optimizing distribution in the EM algorithm.

If we want to run the LM algorithm on the lifted model, we have to adapt the messages sent from factors to variables and the M-step, to reflect merged groups of variables and factors, i.e., nodes with the same color after running CP. Again, this is achieved by adding counts to the formulas. We save messages sent from factors to nodes by observing that factors connected to k variables of the same color only need to calculate a single message instead of k messages because all k messages are identical. With a similar argument, we multiply the incoming messages by k in the M-step if a variable is connected to k factors of the same color. In a lifted representation, the variables would only appear in one single Bayesian network instead of k different nets. Hence, the lifted message equation of the M-Step is:

$$p_i'(\mathfrak{x}_\mathrm{i}) = \frac{1}{Z_i} p_i(\mathfrak{x}_i) \sum_{a \in \mathrm{nb}(i)} \mu_{a \to i}(\mathfrak{x}_i) \cdot \mathrm{vc}(a, i), \tag{3.2}$$

where $\mathfrak{x}_i$ is a clustervariable in the lifted network. As opposed to the ground case, the lifted equation sums over the lifted network ($a \in \mathrm{nb}(i)$) and introduces the count $\mathrm{vc}(a, i)$. This count is the number of times, ground variable X_i is connected to factors with the color of f_a in the ground network. This M-Step has only to be done once for every distinct variable color. This provides an intuitive interpretation of what the equations do in the case of *Lifted Likelihood Maximization (LLM)*: LLM can be thought of sorting the Bayesian networks of the mixture (cf. Figure 3.8) into buckets and each bucket corresponds to a clusterfactor. Initially, every Bayesian network is assigned to a bucket according to the conditional probability table of $\hat{f}_a$. Then variables compute their color signatures as before. The Bayesian networks are now put into different buckets depending on the colors of their child nodes.

3.3.2 Bootstrapped LM

While lifting is an exact method to speed up inference, we will now introduce an additional, approximate modification to the LM approach that reduces run times even further. Adjusting the current estimation of a variable's distribution by a small amount should only have a small effect on its direct neighbors and the effect on other variables decays with the distance. For example, such an effect of errors in BP-messages has been analyzed in [97]. On the other hand, if we clamp a variable in the graph, we do not have to care about messages to that variable anymore and additionally, potentials over this variable can be simplified. On top of that, the influence on the other variables does not necessarily have to be negative, instead these variables can converge faster to their correct maximizing state. We will now describe, how we make use of this observation.

When LM iteratively computes the new beliefs p'_i, the uncertainty about the MAP state decreases over the iterations. Once the probability for one state is above a threshold π, one can assume that it will not change its state in the final MAP assignment anymore. Since LM essentially implements a gradient-based solver for the underlying mathematical MAP program [129], being close to one state makes it very unlikely ever turning to a different one. In such cases, we fix the distribution in such a way that all states have zero probability except for the most likely state which is set to one. More formally, if $x_i^* = \arg\max_{x_i} p'_i(x_i)$ and $p'_i(x_i^*) > \pi$, we set

$$p'_i(x_i) = \begin{cases} 1, & \text{if } x_i = x_i^* \ , \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

We call such states *Pseudo Evidence (PE)* because they do not belong to knowledge that is available beforehand but instead becomes available during the inference. Therefore, *Bootstrapped LM (BLM)* will refer to the LM approach that adds evidence over the iterations. More importantly, PE has major implications on future message updates:

- It simplifies the MRF because it cancels out impossible states from the potentials.
- It allows skipping messages to these variables because for clamped variables the M-step is obsolete.

Recall that the idea of PE is inspired by the soft greedy rule for the M-Step in LM and also reduces the number of iterations required to converge. On top of that, we can now combine PE and lifting. We can re-lift the network during inference based on this new evidence. Thereby we obtain a more compact lifted network. This is intuitively the case because PE often introduces additional independencies in the graphical model which can be exploited via lifting. However, as we will see below (Figure 3.9), standard CP is not always capable of exploiting all of these newly introduced independencies. Hence, fully exploiting PE for lifting requires an adapted form of CP. Before we explain this bootstrapped lifted LM, we want to give some intuition into the approximative nature of the maximization when PE is present. Essentially there are two errors that can be introduced by PE:

(**A.1**) In rare cases a variable will change its state again, even though its belief was already above the threshold π. Therefore, we also propose a variant of BLM where we do not directly clamp a variable on observing a belief above π for the first time but instead we only mark it for clamping. We clamp the variable if we see that its state remains the same for the following T iterations and proceed as before. To estimate the effect of this lag, we simply count the number of state changes during the iterations for different T. However, as we will show in our experiments below, these state changes occur rarely.

(**A.2**) Clamping a variable can be seen as introducing an error to the messages. But as shown in [97] for BP, the influence of the error on other variables in the network decays with the distance to the clamped node. Additionally, this influence does not have to be negative. In fact, it can also lead to faster convergence to the correct state. Optimally, one would clamp a variable only if the effect on neighboring variables is negligible or even positive. Determining the exact influence of PE is difficult but we estimate it by comparing the final MAP solution obtained by BLM to the result of standard LM. One can think of more elaborated ways to measure the effect of clamping. For example, comparing the messages exchanged between direct neighbors with and without PE, one can calculate the error made by setting the variable. Inspired by the idea of backtracking, one could even calculate the change in beliefs of the neighboring nodes and omit the clamping if the direction of the gradient changes. However, this requires even more computational effort and dispossesses LM one of its key features, namely its simplicity.

In the experimental section, we will empirically show that issues A1 and A2 do not have critical influence on the quality. Instead, BLM results are of high quality and we do not use more enhanced ways for clamping which requires higher computational demand.

3.3.3 Bootstrapped Lifted LM

We now combine PE and lifting. After setting PE, we may now get more opportunities for lifting since variables and factors may fall into the same cluster though being in different ones in earlier iterations. A naive lifting approach takes the ground network, clamps it according to the PE, uses CP or any online variant to lift it, and continues message passing on this new lifted network. Although this approach sounds promising, we will now give an example to illustrate the downside of this naive approach.

Example 3.9. *Rows 1-3 in Figure 3.9 depict the naive approach of combining PE and lifting based on CP. The figure assumes a binary, pairwise MRF with identical pairwise potentials, i.e., $f_{ij}(x_i, x_j)$ is the same for all pairs $(i, j) \in E$. The line color of the nodes denotes different unary potentials $f_i(x_i)$. In our example, all unary potentials are identical (red) except for the one associated with the variable in the upper left corner (green). However, for simplicity let us further assume that these priors are not converse but only different in their parameter strength. For example, green could encode something like $(0.2, 0.8)$ while red encodes $(0.25, 0.75)$. The first row in Figure 3.9 shows the ground network without any PE. After running CP, we get the graph shown in the center column (on the right hand side of the $\rightarrow$) with nodes colored by six different gray shades according to CP. This graph can be compressed to the lifted network next to the equality sign (right column). The lifted network contains only six nodes as opposed to nine ground nodes. Now assume that at some point we obtain identical PE for all four corners (depicted as stars instead of circles for the variable nodes). Running CP on the ground network with the new evidence in mind, we obtain the very same lifting as in the previous iteration, namely a network containing six variables (Figure 3.9, second row). As we will explain now, this behavior of CP is suboptimal. Since the upper left corner is set, its unary factor does not have any influence on any message being sent in future iterations. This also holds for the other three corners. This means that their unary factors can actually be neglected and all four corner nodes should receive the*

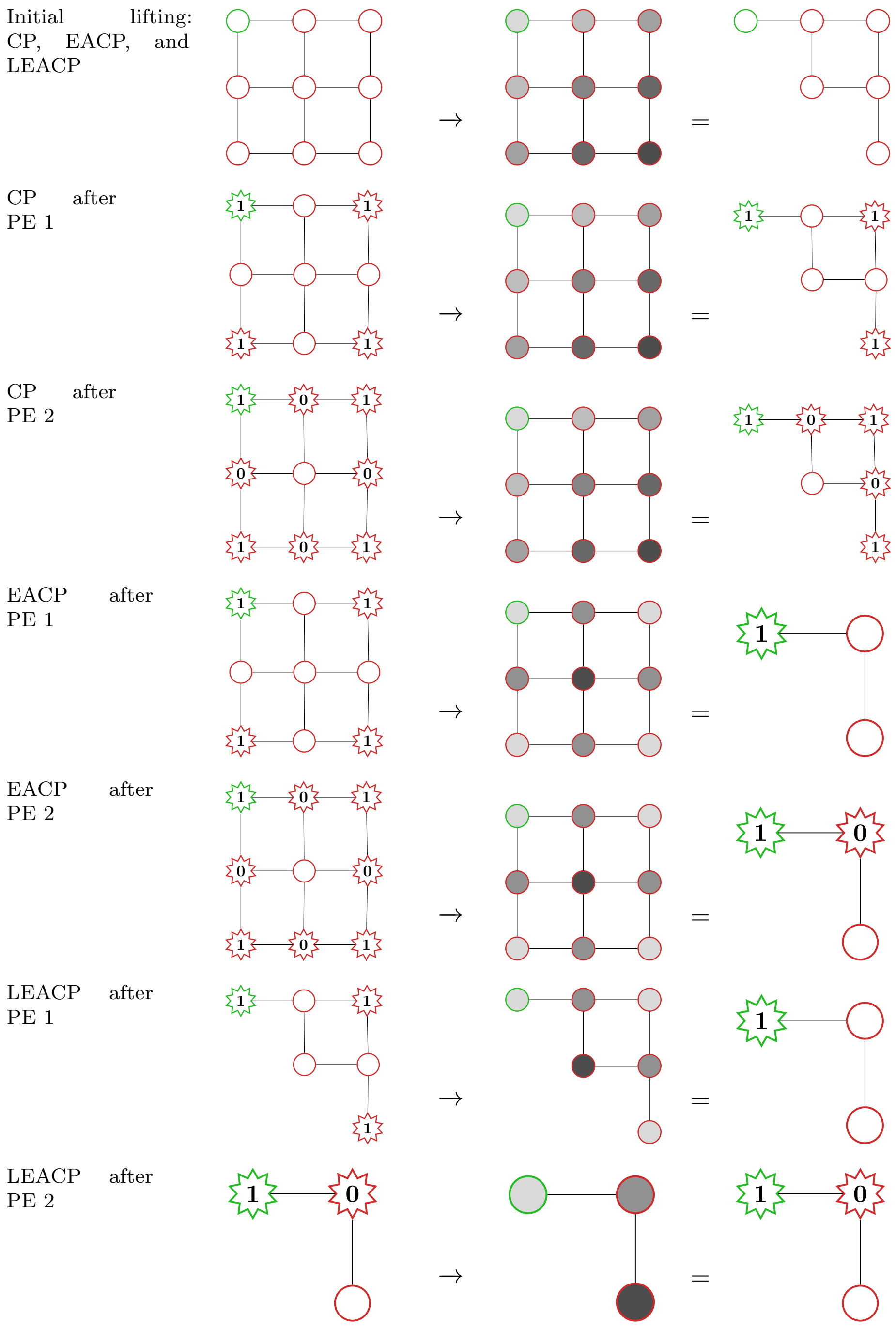


Figure 3.9: Re-lifting a network with Pseudo Evidence which is *not* optimally lifted via standard Color Passing once Pseudo Evidence is present.

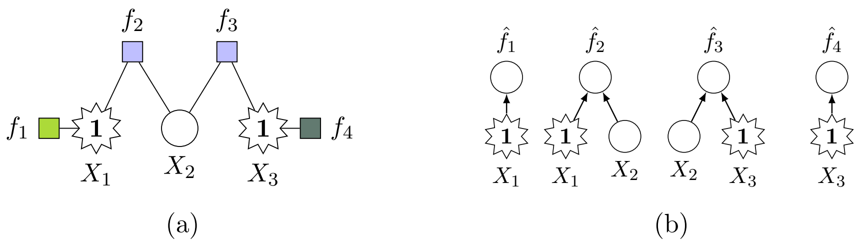


Figure 3.10: Factor graph and corresponding Bayes nets. X_1 and X_3 are clamped. f_1 and f_4 are different (denoted by different colorings of the factor nodes) while f_2 and f_3 are identical (same color of factor nodes).

same color in the final coloring of the graph. We will below show how to modify CP in order to recognize the new symmetries induced by PE. However, let us first finish our running example. In the third row of Figure 3.9, we assume that all remaining unclamped variables are now being fixed except for the center variable. Again, CP is not able to exploit additional symmetries and the compressed model simply remains the initial lifting.

To sum up, the lifting obtained from CP is not always optimal if evidence is present. To overcome this issue, we will now propose an adapted CP algorithm and we will show how to modify CP to achieve a higher compression due to "edge-deletion" induced by PE. *Evidence Aware Color Passing (EACP)* is similar to the original CP but capable of returning a lifted network that respects PE. We begin by illustrating the underlying idea by an example.

Example 3.10. *Assume the factor graph in Figure 3.10a has identical and symmetric pairwise potentials f_2 and f_3. The unary factors of X_1 and X_3 are different, denoted by the different colors of the factor nodes f_1 and f_2. We also assume that X_1 and X_3 have just been set to 1 and we stick to the notation that variables with PE are denoted as stars. Figure 3.10b shows the BNs associated with the factor graph. The message sent from f_2 to X_2 ($\mu_{f_2 \to X_2}$) is calculated as follows:*

$$\begin{aligned}\mu_{f_2\to X_2}(x_2) &= \sum_{x_1} f_2(x_1,x_2)\cdot p(x_1)\\ &= f_2(0,x_2)\cdot \underbrace{p(0)}_{=0} + f_2(1,x_2)\cdot \underbrace{p(1)}_{=1}\\ &= f_2(1,x_2)\end{aligned}$$

One can see that this message is affected by the evidence on X_1. The message reduces to the value of $f_2(1,x_2)$ because the probability of X_1 being in state 0 becomes zero due to the PE, i.e., $p(x_1 = 0) = 0$. Similarly, the message sent from f_3 to X_2 can be simplified to $\mu_{f_3\to X_2} = f_3(1,x_2)$. In a network without PE, f_2 and f_3 send different messages to X_2 because these messages are dependent on the current beliefs of X_1 and X_3 which are different due to their unary factors. Since the messages are now identical, we can put f_2 and f_3 into the same cluster, i.e., the corresponding BNs can be put into the same bucket. The key insight is that standard CP assigns too many colors, i.e., factors that actually send identical messages are colored differently. CP initializes X_1 and X_3 identically, but f_1 and and f_4 have different colors which are propagated to X_1 and X_3 in the initial color exchange.

```
   input: A factor graph $G$
   output: Variable and factor colors
1  initialize_colors();
2  propagate_evidence();
3  while not converged do
4      foreach $f_k$ do
5          if free_vars($f_k$) > 1 then
6              build_signature($f_k$);                 /* As in standard CP */
7          end
8      end
9      factor_colors = new_colors();
10     foreach $X_i \in \mathbf{X}$ do
11         if not_clamped($X_i$) then
12             build_signature($X_i$);                 /* As in standard CP */
13         end
14     end
15     variable_colors = new_colors();
16 end
```

Algorithm 5: The EACP algorithm also accounts for existing evidence during the color passing and thus finds more compact liftings.

Essentially, evidence creates independencies in the underlying factor graph and blocks influence thereby. If there is no flow between two nodes due to evidence, their colors should not be affected by each other. To overcome this problem, EACP differs from CP in several aspects and produces a coloring of the ground factor graph that respects the PE. The pseudo code for EACP is given in Algorithm 5. In line 2, the evidence of the variables is propagated to the neighboring factors. This includes pseudo and observational evidence. This step allows us to distinguish identical factors that are connected to variables with different evidence. In turn, it can distinguish f_2 and f_3 in Figure 3.10a if X_1 and X_3 had been set to different states. The other two main modifications compared to CP are as follows. In line 5, factors only create new color signatures if they are connected to more than one unclamped variable. Otherwise they keep their current color. Secondly, in line 11, evidence variables never receive new colors. We now show that EACP indeed returns a correct lifting.

Theorem 3.2. *EACP is sound, i.e., it returns a valid lifting of the ground network.*

Proof sketch. All evidence variables that have the same range and are set to the same state keep their colors because their distribution is fixed. They all have the same influence and do not have to be distinguished. Nodes that have not been set behave as in standard Color Passing. Without loss of generality, let us assume pairwise factors but the argument generalizes to the arbitrary case. If one variable of a factor is clamped, messages to the other variable become independent of any of the marginal distributions; they solely depend on the potential of the factor. Hence, two factors with identical potentials receive the same colors initially and do not have to adapt their colors; the messages they send are identical which satisfies the requirement for the same color. By propagating the evidence of the variables before the actual Color Passing, we ensure that factors can be distinguished that are connected to PE variables in different states. □

```
input: A factor graph $G$
output: An approximate MAP assignment
$\mathfrak{G} \leftarrow$ compress($G$);                          /* $G$ is the input graph */
while not converged do
    calc_deltas();                                             /* $\mu_{f \to X}(x)$ */
    calc_marginals();                                          /* Equation (3.2) */
    clamp_variables();                                         /* Equation (3.3) */
    if new pseudo evidence then
        leacp();
        average_marginals();
    end
end
return calc_map();                                             /* MAP assignment */
```

Algorithm 6: Bootstrapped Lifted Likelihood Maximization

The main idea is now to replace CP with EACP. As our experiments will demonstrate, considerably higher compression rates are obtainable because we exploit the PE. This is also illustrated in rows four and five in Figure 3.9. Here, EACP obtains better compression than CP after PE is observed. After clamping the corner variables, EACP returns a model containing three variables instead of six. However, we can still do considerably better and become more efficient because EACP operates on the ground network. As we will show now, it can also operate on the current lifted network.

Lifted EACP (LEACP) functions similar to EACP but it requires some modifications of the color signatures. We have to distinguish variables that are connected to a different number of factors of the same color. This is exactly the information contained in the counts $\mathrm{vc}(a, i)$. LEACP now simulates the ground color signature for a variable X_i by appending $\mathrm{vc}(a, i)$ times the color c of a factor f_a, i.e.,

$$\langle \underbrace{c, c, \ldots, c}_{|\langle\cdot\rangle| = \mathrm{vc}(a,i)} \rangle \, .$$

This idea is only correct for lifted networks that have non-fractional counts. This makes it non-trivial to employ other approximate lifting approaches such as informed lifting [115] and early stopping [210]. An example of fractional counts in early stopping lifting was shown in Figure 2.8. To see that LEACP returns a valid lifting, first note that setting PE will never require a cluster in the lifted network to be split. This is simply because all variables in a cluster have the same expectation of their marginal distribution so that we will always set an entire cluster to evidence. In turn, the variables in that cluster will remain to be indistinguishable in all later iterations and we will never have to split them. This is clearly a distinction from the decimation-based framework (Section 3.1.6) where usually just a subset of the variables, in the extreme case only a single variable, were clamped in each cluster. This essentially proves the soundness of LEACP.

Theorem 3.3. *LEACP is sound, i.e., it returns a valid re-lifting of the current lifting containing additional evidence.*

Proof. Every partition is valid for all following iterations. A partition remains valid after setting evidence because we always set an entire cluster, i.e., all ground variables in a cluster are set to the same state. This means that we never require more colors than we have clusternodes in the current lifting. Only merging two clusternodes is a possible action which reduces the number of required colors. We can merge two clusters $\mathfrak{X}_i$ and $\mathfrak{X}_j$ iff all corresponding ground variables behave the same in the ground network. Similar to the BP message passing in the lifted network, we now have to simulate the color signatures in the case of Color Passing. Here, the counts come into play which then produce the same color signature as in the ground case. Via Color Passing on the lifted graph and adapting the color signatures by the counts, we obtain the same color signatures as if we were running CP on the ground graph. □

Putting all pieces together, we obtain *Bootstrapped Lifted LM (BLLM)* as summarized in Algorithm 6. BLLM exploits all enhancements we have introduced above. It uses standard CP to obtain an initial lifting (line 1) for a factor graph without PE. It then runs on the lifted level until PE is encountered (lines 3-5). In cases, where PE is set, it uses LEACP to re-lift the model on the lifted level (line 7).

To complete the discussion of BLLM, we have to explain one final issue. Indeed, LEACP returns a valid lifting. However, when two clusternodes are merged, they may have different beliefs about the current estimation of the marginals. Hence, we need to calculate a common marginal in the re-lifted network to continue with the iterative process. Reconsider Figure 3.9. When the PE for all four corner variables is encountered, their neighboring variables have different marginals but BLLM puts them together into one cluster. We propose to to simply take the average of the marginals of the involved variables to make use of the calculations of previous iterations (line 8 in Algorithm 6). Intuitively, we do not initialize the variables again since we already know their tendency and they should get the same beliefs in the end. Although this is not exact, it does not sacrifice the estimation quality as we will validate in our experimental results below and it also justifies the re-lifting. On top of that, it is also likely to reduce the variance of the estimates by this averaging. BLLM saves messages compared to the ground inference by avoiding the calculation of indistinguishable messages. Additionally, LEACP is more efficient than EACP and CP because it solely operates on the lifted network to obtain the lifting for the following iterations. That is we never have to return to the ground factor graph again after the initial lifting. This is also illustrated in rows six and seven in Figure 3.9.

Finally, we want to comment on the time complexity of EACP and LEACP. The idea of bootstrapped lifting only works if the repeated call to lifting is efficient. Since (L)EACP involves only linear-time modifications of CP, each of their iterations also runs in linear time (see Section 2.4.1 for details on the run time of CP).

3.3.4 Experiments

Here, we investigate the question whether our bootstrapped and lifted LM variants can be considerably faster than the baselines without decreasing performance. We will answer the following main questions:

(Q3.6) Does LM benefit from PE?

(Q3.7) Can PE combined with lifting speed up LM even further?

(Q3.8) How does the quality of B(L)LM's results compare to standard LM?

To do so, we implemented the following set of algorithms:

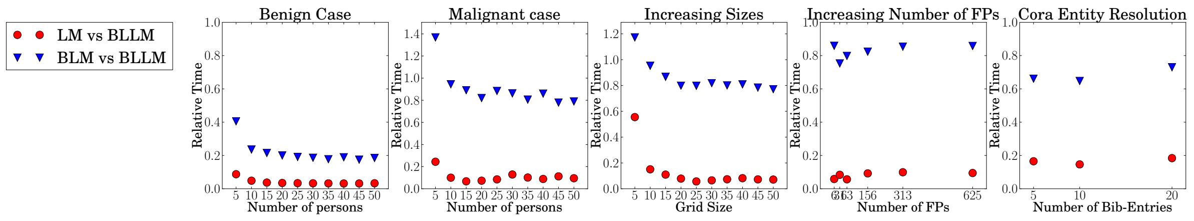


Figure 3.11: Relative run time of BLLM compared to LM and BLM.

- LM: the propositional approach by Kumar and Zilberstein
- LLM: the naively lifted LM using only standard Color Passing
- BLM: Bootstrapped LM using PE but no lifting
- BLLM: the bootstrapped lifted LM as shown in Algorithm 6

LLM, BLM, and BLLM are the contributions from this section and will be compared against LM. To this aim, we evaluated them on lifting benign (friendly) and malignant (unfriendly) models. In particular, we consider synthetic Ising grids and relational models, as well as real-world models for simple image segmentation and relational entity resolution.

Before we present the quantitative and qualitative results, we give some further technical notes on the implementation and the parameters used. Since clamping a single variable is unlikely to achieve significant gains in lifting, we re-lift after we have obtained PE for a batch of $b\%$ of the original number of variables. In our experiments we set b to 10. We stop the algorithm after 2,000 iterations if the algorithm does not converge earlier. The other parameters are set to $\pi = 0.9$ (PE-threshold) and $T = 0$ (lag-parameter).

Smokers-and-Friends-MLN

For our first experiment, we generated MLNs of varying domain sizes, i.e., for an increasing number of persons (see Section 2.3.1 for more details on the S&F-MLN). More precisely, this means that we increased the domain size of the person predicate in the MLN from 5 up to 50. Resulting in factor graphs with up to 2,600 variables, 5,510 factors, and 10,150 edges. We call this setting the *benign* S&F-MLN because it does not contain any evidence and hence is well suited for lifting. The first plot in Figure 3.11 shows that BLLM only requires a fraction of the time compared to LM (red circles). Additionally, the blue triangles highlight that lifting reduces the running required by BLM significantly. For a clearer picture, we have omitted the running times of LLM. But for this problem, LLM outperforms BLM while BLLM is the fastest method among all. This is not surprising, as this scenario is very well suited for lifting.

We now add 25% random evidence to the MLNs by randomly choosing predicates and their state. We call this setting the *malignant* S&F-MLN because the random evidence prevents CP from lifting the network initially. The results are depicted in the second plot in Figure 3.11. Again, BLLM only requires a fraction of LM's running time. But compared to the benign case, lifting BLM does not help as much as before. Nevertheless, lifting is still beneficial for the larger problems and BLLM is capable of exploiting symmetries in models for which standard CP resorts to the ground network. We see that lifting is an overhead in the small problem instances but as the problems become larger, BLLM becomes faster than BLM, exploiting symmetries

	S&F-MLNs Benign Case									
	5	10	15	20	25	30	35	40	45	50
(L)LM	103.5	362.0	775.5	1344.0	2067.5	2946.0	3979.5	5168.0	6511.5	8010.0
B(L)LM	**103.5**	**362.0**	**775.5**	**1344.0**	**2067.5**	**2946.0**	**3979.5**	**5168.0**	**6511.5**	**8010.0**
	S&F-MLNs Malignant Case									
	5	10	15	20	25	30	35	40	45	50
(L)LM	85.2	333.0	687.3	1234.4	1854.9	2682.3	3655.6	4744.0	5895.6	7251.8
B(L)LM	**85.2**	**333.0**	**687.3**	**1234.4**	**1854.9**	**2682.3**	**3655.6**	**4744.0**	**5895.6**	**7251.8**
	Ising Grids of Varying Sizes									
	5	10	15	20	25	30	35	40	45	50
(L)LM	73.1	309.1	499.1	892.0	1991.6	2447.8	3095.0	3227.0	5391.7	5480.4
B(L)LM	**72.8**	**294.4**	**481.4**	**859.4**	**1913.0**	**2352.4**	**2997.2**	**3118.9**	**5201.2**	**5248.8**
	Ising Grids of Varying Field Parameters									
	6	31	63	156	313	625				
(L)LM	1693.9	1808.8	1991.6	1566.9	1093.0	1748.1				
B(L)LM	**1663.0**	**1739.8**	**1913.0**	**1520.4**	**1061.5**	**1681.8**				

Table 3.3: The log-scores (the higher, the better) show that the B(L)LM versions of the algorithms achieve high quality results. Bold log-scores are in 95% of the standard LM score.

obtained from Pseudo Evidence. For these malignant MLN problems, LLM basically reduces to the ground LM case as the evidence destroys initial symmetries. We now also want to compare the quality of the obtained MAP configurations. To this end, we computed the log-likelihood score, as defined in Equation (2.10), of the obtained assignments. As shown in Table 3.3, all solutions obtained by BLLM are qualitatively as good as the results produced by LM.

Ising Models

The next set of experiments were conducted on Ising grids. Ising grids are pairwise MRFs with $x_i \in \{-1,+1\}$ and

$$p(\mathbf{x}) = \frac{1}{Z}\prod_{i\in V} f_i(x_i) \prod_{(i,j)\in E} f_{ij}(x_i,x_j)$$

The unary potentials are $f(x_i) = \exp(x_i w_i)$ where w_i is called the *field parameter* and is drawn from $[-1,1]$. Pairwise potentials are defined as $f(x_i,x_j) = \exp(x_i x_j w_{ij})$ with the *interaction parameter* w_{ij}. We call the grid *attractive* if all $w_{ij} > 0$. We consider different Ising models in the classical sense which have a single interaction parameter w_{ij} only. Motivated by applications in image processing, we use attractive interaction parameters and a limited set of field parameters instead off drawing all w_i randomly.

The first problem set consists of grids of increasing size. We generated grids from 5×5 up to 50×50 with a fixed ratio of different field parameters equal to 10%. We generated ten grids for every size and averaged the results over these runs. The results are shown in the third plot in Figure 3.11. BLLM is by far the fastest method and lifting of BLM achieves around 20% speed up on the larger instances.

The second experiment keeps a fixed grid size of 25×25 and varies the number of field parameters. For every grid, we generated a set of field parameters beforehand. The size of this set is chosen relative to the number of variables in the grid. The fourth plot in Figure 3.11 again shows that BLLM is the fastest method and requires less than 10% of LM's running time. Similar to the malignant S&F-MLN, lifting can here speed up BLM only little. This can be

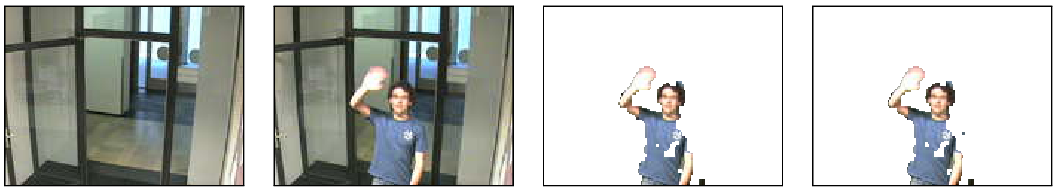


Figure 3.12: Input and output for the image segmentation task. In summary, BLLM outperforms all other variants in terms of run time while achieving accurate results.

explained by the fact that the effects of unary factors vanish once a variable is set and pairwise potentials with only a single free variable can be clustered for all distinct values of the clamped variables. On the other hand, standard lifting does not help in this scenario and requires the same time as the ground variant plus an small overhead due to the lifting at the beginning. Table 3.3 contains average log-scores for a qualitative comparison of both experiments on Ising grids. The results of BLLM are close to LM's solutions and always above 95% of the LM score.

To obtain another qualitative measure, we compared the log-scores of LM and BLM with max-product BP on different grids. One observes that the scores of LM and BLM are often close to the ones obtained by max-product BP. For example, on a grid with 156 field parameters, we have LM and BLM both with log-scores of 391 while max-product BP achieved a score of 393. However, when the strength of the interaction parameters increases, we observe that the quality of LM solutions decrease compared to max-product BP, e.g., we obtain a log-score of 2,544 for LM and 3,162 for max-product BP (LM is not above 95% of the BP-score anymore). A similar observation has already been made in the original work. Kumar and Zilberstein state that the LM approach can tend to poor performance in cases where the gap between the minimum and maximum reward is large.

In general, these random grids are not well suited for exact lifted inference as there is no structural symmetry. However, BLLM is still able to perform better than BLM. This clearly shows the benefits of exploiting PE and shows that lifting can become beneficial even in settings that are adversarial towards lifting in general.

Image Segmentation

Many tasks in computer vision can be encoded as binary pairwise grids, e.g., simple forms of foreground/background classification. To show that the BLLM algorithm actually results in high quality solutions on real world problems, we ran our algorithm on an instance of an image segmentation task. We have used Ising models in the previous experiment already in a set of synthetic experiments. Now, we will use such binary discrete MRFs to label pixels as foreground or background in an image. Having variable $x_i = -1$ assigns the corresponding pixel to the background and correspondingly, for $x_i = 1$ the pixel belongs to the foreground. As described in [154], the unary factors $f_i(\cdot)$ are based on the difference of two video camera images and defined as follows:

$$f_i(x_i) = \exp(x_i w_i) = \exp\left(x_i w \tanh \frac{h_i - c}{r}\right)$$

Here, h_i is the absolute difference in hue of pixel i in the two images. The other parameters, i.e., w, c, and r can be fined tuned based on a training set. This function penalizes a large

	Cora Entity Resolution		
	5	10	20
(L)LM	0.98	-103.6	-689.2
B(L)LM	**0.98**	-116.5	**-702.8**

Table 3.4: The log-scores on the Cora dataset (the higher, the better) also show that the B(L)LM versions of the algorithms achieve high quality results on a real-world problem dataset. Bold log-scores are in 95% of the standard LM score.

difference in the input and the reference image. The pairwise factors $f_{ij}(\cdot)$ encode the intuition that neighboring pixels are more likely to belong to the same part of the image:

$$f_{ij}(x_i, x_j) = \exp(x_i x_j w_{ij})$$

with $w_{ij} > 0$. These pairwise factors are defined as in attractive Ising models and act as a smoothing function. We used the scripts with their default parameters contained in libDAI [155] to generate a factor graph for this task and we also used their example images. The inputs were the first two images depicted in Figure 3.12. The script calculates the difference between both images to define the unary factors as described above. The potentials of the pairwise factors are defined such that neighboring pixels are more likely to belong to the same segment. The third and fourth image in Figure 3.12 show the results after running inference (LM on the left, BLLM on the right). The images show that the result obtained by BLLM is almost as good as the one obtained by LM. One has to look very carefully to see differences which visually validates the PE based approach. In fact, the score achieved by BLLM is only slightly lower compared to LM.

Relational Entity Resolution - Cora

We conduct our final experiments on the Cora entity resolution dataset. We obtained the dataset from the Alchemy repository[4] and also used Alchemy for weight learning. We did not evaluate on the entire dataset since subsets are already sufficient to highlight our results. Instead, we randomly sampled k bib-entries and added all predicates describing their properties to the evidence. We ran experiments for $k = 5$, 10, and 20. The largest problem instance contains 1,110 variables, 32,965 factors, and 71,209 edges. The results in Figure 3.11, 5th plot, show that BLLM clearly outperforms LM in terms of runtime, while the scores are very similar (see Table 3.4). Here, lifting achieves good performance increases over BLM again. This also holds for the naive LLM which is again omitted in the plot for clearness. One should not get confused about the negative values for the datasets with 10 and 20 bibliographic entries. This is due to the fact that the Cora-MLN has positive and negative weights. If not sufficiently many clauses with positive weights are satisfied, the sum of the log-weights becomes negative.

We also analyzed the behavior of BLM on the Cora problem with respect to the nature of the approximation as motivated in A1 and A2 above (see Section 3.3.2). We have observed that state changes, as described for A1, during the iterations almost never occur. One has to set the lag-parameter $T > 50$ to observe any state changes at all. Similarly, as one can already expect from the log-scores, the approximate MAP solutions are very similar. The differences

[4]`alchemy.cs.washington.edu`

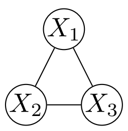


Figure 3.13: If we assume that unary and pairwise factors are identical, this fully connected graph is compressed to a single node if Color Passing is applied.

seem to result from variables changing their states only but not due to wrong influence on neighboring variables that results form setting PE.

The experimental results clearly show that LM is boosted by lifting and PE can indeed considerably reduces run times without sacrificing accuracy. Moreover, BLLM scales well and is most beneficial on relational models. Nevertheless, BLLM can also improve state-of-the-art on propositional models that so far were challenging for lifted inference approaches. To summarize, questions **Q3.6**–**Q3.8** were all answered in favor of BLLM.

3.3.5 Summary

In this section, we introduced the idea of *Pseudo Evidence (PE)* for MAP inference. We showed that it can considerably speed up MAP inference and provides novel structure for optimization. Specifically, we bootstrapped lifting using PE and empirically demonstrated that this "reduce and re-lift" concept can decrease run time even further and achieve lifting in situations where standard lifting fails completely.

Our approach exploits additional symmetries by adding evidence during the optimization. Essentially, we try to fix broken symmetries by PE. The fact that evidence breaks symmetries is well known and other authors have dealt with that issue as well. For example, Van den Broeck and Darwiche approximate evidence by low-rank boolean matrix factorization to speed up lifted inference. It is shown how a large class of binary relations can be encoded with the help of unary relations only. More precisely, the unary relations are represented by boolean vectors and their product represents the original binary relation. This vector-product can be extended to the matrix-product case as well. Based upon this insight, approximate boolean matrix factorization is used to find a decomposition of the binary evidence relation. Instead of a decomposition, our approach iteratively adds evidence and could be combined with a decomposition approach to further compress the evidence, including the Pseudo Evidence.

It is important to note that LLM and BLLM currently rely on the parallel protocol for message updates. We have seen this requirement for the other lifted algorithms as well and we cannot run asynchronous updates because the Color Passing schedule relies on parallel updates. This restriction prevents us from solving problems such as the one depicted in Figure 3.13 correctly. Think of a fully connected pairwise MRF with all unary and pairwise potentials being identical. We further assume distractive pairwise potentials with $w_{ij} < 0$. Any of the CP algorithms will reduce the graph to one node. If we now run LM on this graph, we will end up with marginals $(0.5, 0.5)$. An optimal assignment would consists of any combination with the states not being identical, e.g., $(-1, 1, 1)$. We cannot read off any optimal assignment from our calculated marginals, although the marginals are computed correctly in terms of the underlying model. Besides using a decimation approach based on conditioning, one heuristic would be to add small noise to every unary potential. In this case, however, we could not lift the model anymore. On the other hand, the parallel update protocol results in satisfying solutions in many applications and also allows easy parallelization.

Another limitation induced by lifting is the fact that we do not have the complete freedom to randomly initialize the EM algorithm. Similar to the message passing algorithms for satisfiability solving, e.g., SP in Section 3.1.5, LM can benefit from random initialization. However, in lifted LM all variables have identical message updates that fall into the same cluster, i.e., we would initialize all these variables identically. We additionally rely on the fact that we always set all variables in one cluster as PE. If clusters were only partially clamped, we are back in the decimation framework and we have to use methods such as the sequential clamping described in Section 3.2.

We used in our experiments only exact CP and Lifted EACP also provided an exact extension for BLLM. Combining BLLM with informed lifting approaches described in Section 2.4.2 is an interesting avenue for future work. At the beginning, these algorithms will be more efficient than BLLM but once evidence has been set, only BLLM can exploit the additional symmetries. It would also be interesting to use approximate lifting techniques, such as early stopping. However, as mentioned above, this is not obvious to implement due to fractional counts that may arise. Fractional counts prevent us from using the LEACP algorithm for lifting because signatures cannot be copied in the current form.

After PE has been set, it is also possible to further analyze the factors. Some configurations are now impossible and hence factors can be reduced. This might also introduce new symmetries on the factor level which can be exploited in the upcoming iterations. Similarly, setting PE can introduces conditional independencies between variables. In the extreme case, this can result in variables for which the entire Markov blanket is set. It is not necessary anymore to update these variables in future iterations and one can therefore immediately clamp these variables based on the Markov blanket. Additionally, one can also imagine to calculate the conditional probabilities exactly for other small sets of variables that are independent from the rest of the network and for which this calculation is feasible. This is a common pattern in many approximate inference algorithms to increase the quality of the results. While these two adjustments focus more on the implementation of each iteration, there are other, more general, additional research questions. For example, one should also consider connecting PE to the use of extreme probabilities used for lifting [38]. In general, exploring PE for other inference and learning approaches is highly interesting. In Section 5.4, we will show how bootstrapping via PE also works for Label Propagation.

3.4 Interim Conclusion

This chapter has contributed to the research in lifted inference in several ways and presented lifting approaches in non-standard settings. For example, we have presented the first approach to solving satisfiability problems based on lifted message passing algorithms (**C3.1**). The idea of lifted satisfiability fits well into the emerging trend of Statistical Relational AI which focuses on applying lifted inference and SRL-methods to classical, general, AI tasks. We have also seen that CP is a general technique to lift a given factor graph that can lead to more compact problem presentations for algorithms besides BP, including WP (**C3.2**), SP (**C3.3**), and LM (**C3.7**). In many applications, we either want to re-lift a factor graph with changing evidence (see Section 3.2), or we want to exploit Pseudo Evidence and re-lift iteratively as it arrives (see Section 3.3). Both cases require an adapted lifting to benefit the most from lifted inference.

Naive approaches, that re-lift every time from scratch, often completely cancel the benefits of lifting. Resulting from these observations, we have presented alternative lifting approaches for both scenarios, namely SPS-lifting (**C3.4**) and LEACP (**C3.10**).

The lifted message passing algorithms presented in this chapter ran some form of the CP algorithm, hence, the implementation of CP and the number of iterations influence the run time of those algorithms strongly. Lately, the correspondence between CP and the so called *coarsest equitable partition (CEP)* has been investigated. The results, e.g. [151], show that the partition obtained by CP amounts to the CEP. CEPs are frequently computed in algorithms for graph isomorphism testing and therefore have been studied for a long time. This research has resulted in highly efficient implementations for CEP computations. However, we cannot directly use these algorithms but have to design an initial color encoding that respects variables and factors, as well as the potentials and positions in factors. Nevertheless, all lifted message passing algorithms in this section can benefit from the most efficient lifting algorithms. In the case of CNFs, one can expect the initial color encoding to be simplified similar to the approach presented in this chapter for CP (see Section 3.1). Chapter 4 and Chapter 5 will show how the notion of CEPs can detach lifting from CP and present lifting from an algebraic point of view. This framework will not explicitly require the modification of the ground inference algorithm by means of counts.

The fact that LBP, LWP, LSP, and LLM all require an adaption of the ground algorithm, represents one common drawback of these lifted message passing algorithms. The introduction of different counts and explicit positions make the algorithms more complex and to some extend cumbersome to implement. More elegant lifted inference approaches only preprocess the data and then run an unmodified algorithm on the lifted model. For example, Mladenov et al. [152] show how to avoid the adaption of the message passing algorithm for lifted MAP inference based on linear program relaxations in pairwise MRFs. Their approach follows the idea of a reparametrized model which is used instead of the original MRF.

The approaches we have used in this chapter all relied on exact lifting. First, initial factors were only clustered if their potentials were identical based on a syntactical comparison. Second, variables and factors were only clustered if their color signatures exactly matched. To show the complete picture of lifted inference, one should actually employ approximate lifting algorithms such as the ones described in Section 2.4.2. We can argue that such algorithms will work better than standard lifting in several cases as it has been shown previously in the literature. For example, in the case of grids with identical attractive interaction parameters and field parameters sampled from a small range. While exact CP will not recognize that the marginals are possibly very similar, informed lifting would still achieve lifting. This also holds for LEACP in the case of BLLM. Still, the underlying ideas are rather orthogonal but it is an interesting avenue to combine both.

All of the experiments in this section focused on binary MRFs which included CNFs, grounded MLNs, and Ising grids for image segmentation, among others. However, non-binary problems, such as Potts models, are of great interest as well. We will see in the next chapters how lifting is applied to a labeling tasks with finite discrete random variables that have a large but finite domain. Motivated by such applications, generalizing BLLM to MRFs with non-binary variables is an interesting avenue and the restriction to binary variables is currently rather of technical nature. In general, BLLM is not limited to the binary case and CP works with finite ranges anyhow. Following up this research question, it is natural to ask for lifting approaches in infinite domains or hybrid models as well. Besides the work on lifted GaBP or the lifted variational inference for hybrid models in [38], we will discuss the case of lifting for count random variables, i.e., random variables over the natural numbers, in Chapter 7.

In general, this chapter touched upon several aspects that will return in the rest of this thesis. For example, we will not only lift Label Propagation in Chapter 5 but also show how bootstrapping is applicable to Label Propagation as well. However, this chapter looked specifically at lifted message passing algorithms which relied on the CP algorithm and the modification of the ground algorithm. These assumptions will be relaxed in the next chapters.

CHAPTER 4

From Lifted Message Passing to Algebraic Lifting

Although the techniques presented in the previous chapter are not necessarily restricted to binary random variables, the experiments were focused on MRFs with such variables, such is most of the related literature on lifting and SRL. However, many applications do not neatly fall into the category of binary variables. Let us reconsider the example of image segmentation in Figure 3.12 where a simple foreground/background segmentation was performed based on two images of a video camera. This approach for segmentation was inherently based on the fact that two images, e.g., from a video camera stream, are available to construct the difference of both images. If only a single image is at hand, this approach is not appropriate. If we were additionally to distinguish more than just foreground and background, two possible states per variable were not sufficient either. For example, in interactive image segmentation as described in [241], the task is to label different objects in an image based on a rough initial labeling by the user. Here, a human annotator starts to roughly label the image, i.e., marks pixels from different objects. This can typically cover more than a single object and hence requires more than two labels. More precisely, for each pixel, represented by a variable X_i, the label is $x_i = j$ if the pixel i belongs to the j-th object and $x_i = 0$ if the pixel is background. For example, this allows to distinguish two persons in an image which need to be separated from the background. In computer vision, further examples can be found that require more than two labels, such as image denoising, depth estimation, and stereo matching [220]. If we want to model color intensities or disparities in an image, our label alphabet increases fast because the possible states are the number of possible colors for a pixel and this is usually at least 128.

Such a semi-automated segmentation task requires us to move from the binary Ising model to the multinomial Potts model [183] with $\mathcal{X}_i = \{0, \ldots, d-1\}$. The Potts model allows us to deal with discrete, finite random variables that have potentially a large domain. The pairwise factors allow us to define the potentials in such a way that neighboring pixels are more likely to have the same label. A simple Potts model assumes that the pairwise potentials $f_{ij} = \exp(\delta_{x_i,x_j} w_{ij})$ have identical parameters w_{ij}. Here, δ_{x_i,x_j} is the Kronecker delta which is $\delta_{x_i x_j} = 1$ if $x_i = x_j$ and 0 otherwise (cf. Section 3.3.4 for the Ising case). However, for the image segmentation task, we want to encode in the model that neighboring pixels with similar colors should have identical labels. Hence, we cannot restrict the model to have a constant interaction coefficient if we want to consider the differences in hue of neighboring pixels. If each w_{ij} is individually parametrized, we obtain a Generalized Potts Model as described in [24]. If the hue of pixel i is defined as h_i and correspondingly the hue of pixel j as h_j, we obtain the following pairwise factor:

$$f_{ij}(x_i, x_j) = \exp(\delta_{x_i,x_j} w_{ij}) = \exp\left(\delta_{x_i x_j} w(h_i, h_j)\right) \ ,$$

with $w(\cdot,\cdot)$ measuring the similarity between color intensities of two pixels. Assuming we have a set of initial labels for pixels L, we can incorporate this knowledge via the unary factors. This requires us to define the unary factors in such a way that the initial labels are propagated through the image:

$$f_i(x_i) = \begin{cases} 1, & x_i \in \mathbf{L} \\ 0, & \text{otherwise} \end{cases}$$

We do not attach unary factors to unlabeled pixels because they can take on any label. However, if we model the factors for such a problem in a naive way, the size of the underlying factor graph can quickly increase. In a labeling task where each unknown instance takes on d different labels, each pairwise factor has to model d^2 states. Of course, by omitting zero-states, one can often represent factor tables more compactly but factors still grow quadratically in the size of d.

Although there is much literature on solving Potts models efficiently with the help of approximation algorithms, e.g., graph cuts in [25], computing the marginal probabilities or the MAP configuration in such a discrete MRF is still expensive. This naturally becomes even worse in tasks with a very large number of labels, i.e., the multiclass case, or with problems where the structure is not limited to a lattice. As we have discussed in the previous chapter already, approximate inference algorithms in MRFs often have the additional shortcoming that the results may be inaccurate or the algorithms do not converge. Therefore, we seek for alternative approaches that scale for problem sizes in the dimension of millions of nodes and thousands of labels. Naturally, this asks for approaches that only have to store a similarity function for each pair of nodes where it is desirable that this can be stored in a sparse way for problems not densely connected. We will discuss the definition of neighbors and appropriate similarity functions below in greater detail. For now, let us assume an arbitrary similarity function between two instances that is $w : \mathcal{X}_i \times \mathcal{X}_j \rightarrow \mathbb{R}$. We often write w_{ij} for shorthand instead of $w(x_i, x_j)$. Together with the structure of the problem, such a similarity functions shall help us modeling problems where similar instances are supposed to have the same label.

One particular method solving this problem is the *Label Propagation (LP)* algorithm [260]. This semi-supervised learning algorithm assumes a row-normalized similarity matrix and a second matrix encoding the distribution over labels for each node. We refer to this second matrix as the label matrix which contains one column for each possible label. If a node is initially labeled, this is reflected in the label matrix. Now, the initial labels are propagated through the graph by multiplying the label matrix in place with the similarity matrix in an iterative process. The process is constrained in such a way that the initial labels remain unchanged. Upon convergence, a distribution of the labels for the unlabeled nodes can be read off the label matrix. The propagation of the labels often works well even in cases where only few labels are initially known because the similarity matrix specifies the structure of the entire weighted graph. This algorithm has several interesting properties as we will describe next. This will additionally connect LP to PGMs and naturally asks for lifting this algorithm.

Although, we are interested in a multiclass labeling problem, we can still split the problem into multiple one-vs-rest subproblems and solve each of them separately. Afterwards, we can join the subproblems again to obtain the same labeling as in the mutliclass approach. Looking at binary subproblems, the algorithm can be formulated as an energy minimization problem which is equivalent to a Gaussian random field.

In particular, as shown by Zhu et al. [261], the LP algorithm solves the following minimization problem if we assume binary labels:

$$\mathcal{E}(f) = \frac{1}{2} \sum_{i,j \in E} w_{ij}(f(i) - f(j))^2 \; . \tag{4.1}$$

Here, we assume that $f : X \to \mathbb{R}$ with the constraint that $f(i) = x_i$ if X_i belongs to the initially known labels. I.e., the function takes on the label value for the labeled instances. With this optimization criterion, neighboring nodes with the same label to not contribute to the objective that is to minimize. Hence, unary factors that were enforcing the initial labels in the Potts model are now handled by the constrained label function f. It has been shown that the function f, solving the problem in Equation (4.1), has several desirable properties. For example, f is *harmonic*, i.e., the label score of each unlabeled node is the average of its neighbor's label scores [261]. Since the problem has a proper definition in terms on a continuous MRF, one can also use GaBP to solve the problem.

Besides the fact that the optimization criterion amounts to a Gaussian MRF, the energy function is close to the discrete approach based on Potts models as well. Let us recall that an MRF can be equally formulated based on an energy function and MAP inference amounts to minimizing this energy (cf. Chapter 2). In many problems, like the ones above from the field of computer vision, the energy function can be split into *data* and *smoothness* components [25]:

$$\mathcal{E}(\mathbf{x}) = \mathcal{E}_{\mathrm{d}ata}(\mathbf{x}) + \mathcal{E}_{smooth}(\mathbf{x}) \; .$$

This formulation still corresponds to a Potts model. In the image segmentation example above, the data-part encodes the initially known labels. If we are in the binary case with $x_i, x_j \in \{0, 1\}$, we can equally write $(x_i - x_j)^2$ instead of $\delta_{x_i x_j}$. If we now integrate the data part into the pairwise factors, we arrive at a model that looks as follows:

$$\mathcal{E}(\mathbf{x}) = \mathcal{E}_{smooth}(\mathbf{x}) = \sum_{i,j \in E} w_{ij}(x_i - x_j)^2 \; .$$

So, we have seen that LP optimizes an objective that is akin to the problems we have seen in the previous chapter. However, LP has an elegant implementation in terms of matrix multiplications that is guaranteed to converge to the harmonic solution, i.e., in comparison to BP approaches on MRFs, we do not have to worry about inaccurate beliefs or oscillation. On top of that, it naturally handles multiclass problems.

We are interested in multiclass problems because we want to label a large-scale online bibliography with geo-tags as motivated by the example in Section 1.1. Here, the label alphabet can easily consist of several thousand possible locations. However, it is also clear that millions of nodes and thousands of labels result in similarity matrices that are impossible to handle in the standard case. Therefore, Chapter 5 will present several enhancements to the original LP algorithm that will allow us to label a graph with more than five million nodes with one of more than four thousand labels. In particular, this will make use of a similarity function based on logical rules and a lifted version of LP. Since our inference is not based on message passing anymore, this also asks for a different kind of lifting possibly without passing around colors.

```
def label_propagation(W, Y):
    # W is a similarity matrix
    # Y is a label matrix
    T = preprocess(W)    # row normalization
    Y_old = Y.copy()
    iters = 0
    while True:
        Y = T * Y
        max_diff = np.abs(Y-Y_old).max()
        iters += 1
        if max_diff < th or iters == max_iters:
            break
        Y_old = Y.copy()
    return iters, Y
```

Algorithm 7: Label Propagation by Zhu and Ghahramani implemented in Python.

To pave the way for lifted LP in the next chapter, we will now discuss the necessary background and related work. We can divide the related literature into two main categories. The first part is mainly concerned with the aspects of modeling labeling problems and running LP. This also features a short discussion of efficient LP implementation and different variants of LP. The second part focuses on lifted inference but this time from a different angle. While the previous chapter focused on message passing based lifting, we will here describe how lifting can be seen in terms of graph theory and algebraic operations.

4.1 Label Propagation

Label Propagation (LP) [260] is a semi-supervised learning algorithm which operates on a similarity graph. In this setting only few nodes in the graph are labeled with one of d labels but we have access to a similarity function between all node pairs that is encoded in the edges. A missing edge indicates that two nodes do not have anything immediately in common. LP now propagates node labels along the edges to compute a distribution over the labels for the unlabeled nodes. We define the LP graph as an undirected graph $G = (V, E)$ with nodes V and edges E. Every edge $e_{ij} \in E$ between two nodes i and j contains a weight w_{ij} that is proportional to the similarity of the nodes. Originally, a similarity function of the following form was suggested:

$$w_{ij} = w(x_i, x_j) = \exp\left(-\sum_k \frac{(x_{i,k} - x_{j,k})^2}{\sigma_k^2}\right)$$

Here, x_i is a feature vector of node V_i and $x_{i,k}$ represents feature k of node i. One can see immediately that this similarity function results in a completely connected graph, making this function intractable for large problem sizes. Therefore, we will below introduce an alternative approach to connect two nodes and to define the weights of each edge based on logical rules, resulting in a sparse LP-graph.

```
def clamp_evidence(T, labels):
    for i in labels:
        for j in range(T.shape[1]):
            if j == i:
                T[i, j] = 1.0
            else:
                T[i, j] = 0.0
```

Algorithm 8: Updates the normalized affinity matrix T in such a way that the push-back phase is integrated into affinity matrix itself.

Independent of the similarity function, we can build a symmetric similarity matrix $W \in \mathbb{R}^{n \times n}$ based on the edge weights. Let us define the diagonal degree matrix D with $D_{i,i} = \sum_j W_{i,j}$. We now row-normalize the affinity matrix by multiplying it with the inverse of D:

$$T = D^{-1}W.$$

Now, the LP algorithm performs the following matrix-matrix-multiplication until convergence:

$$Y^{t+1} = TY^t \; , \tag{4.2}$$

where $Y^t \in \mathbb{R}^{n \times d}$ is the label matrix. A simple Python implementation is shown in Algorithm 7. At convergence, row i in Y^* corresponds to a distribution over the possible labels for node i. Initially, this matrix, i.e., Y^0, is all zero except for the rows of the nodes that are labeled in advance. For the known labels, the cell in the corresponding column is set to 1, i.e., we set a cell $y_{i,j}$ to 1 if we know that node i has label j. After every iteration, a *push-back* phase clamps the rows of the initially labeled nodes in Y^t to their original distribution as in Y^0.

We apply a minor modification to the original algorithm by encoding the push-back phase directly into the affinity matrix. We adapt the similarity matrix in such a way that we do not need the push-back phase in each iteration explicitly. Intuitively, this is achieved in such a way that the matrix multiplication leaves the rows with existing evidence in Y unchanged. This works by setting all values to zero except for the diagonal element in each row for which evidence is present. The diagonal element is set to 1 in these rows. This procedure is also shown in Algorithm 8. We use this modification because it simplifies the implementation in distributed settings where we only have to distribute the affinity matrix but not the initial labels. Additionally, this makes the matrix sparser, and hence, the implementation becomes more efficient when using a linear algebra library supporting sparse matrices.

When using LP as in Equation (4.2), LP tends to converge slowly which is the reason why the iterative process is often terminated after a fixed number of iterations. At convergence, the labels of the unknown nodes are read off the label matrix, i.e., the label of node i is given by

$$y_i^* = \operatorname*{arg\,max}_{0 \leq j \leq d} Y_{i,j}^* \; .$$

However, we still require an efficient implementation for multiplying a sparse matrix with a dense matrix. Our LP implementation is based on LAMA[1], a highly efficient `C++` linear

[1]`www.libama.org`

algebra library. LAMA is specifically designed for parallel scalability and supports multi-core processors, GPU accelerated computations, and distributed memory clusters as well.

Regarding an implementation of LP, it can be shown that the labeling problem solved by LP can be equally formulated as a set of linear equations. Hence, we cannot only use standard solvers for linear programs to solve the problem but also use GaBP on the MRF described above. For multiclass labeling problems, this requires a one-vs-all approach to calculate the label scores.

There exist also slight modifications of the original LP algorithm. One example is the Jacobi-method inspired version in [16, Algorithm 11.2]. This algorithm adds a regularization term for numerical stability and its convergence properties follow the convergence of the Jacobi-method. Another algorithm with yet a different cost criterion was presented in [258]. Several of the label propagation algorithms can also be seen as a form of message passing because the updated distribution of a node is calculated based on a function of its neighbors. As stated in [16], the convergence of LP, and both algorithms mentioned in this paragraph, can be expected to be at worst in the order of $\mathcal{O}(kn^2)$ where k is the number of neighbors of a node in the graph. However, the practical average convergence depends on different properties of the graphs and often, the final labeling does not change anymore although the scores have not converged below a specified threshold yet. I.e., class agreement is often achieved early and we do not necessarily have to run LP until convergence.

Zhu [259] discusses different LP implementations and compares LP based on Equation (4.2), a conjugate gradient approach, and GaBP. Although GaBP often converges quite fast on small problems tested by Zhu, it is beneficial for us to use LP in the form of Equation (4.2). As we will see below, it allows us to lift LP without modifications of the implementation (see Section 5.3). Additionally, as discussed in the previous paragraph, we do not necessarily have to run LP until convergence.

As roughly sketched at the beginning of this chapter, we can also model multiclass labeling problems with multinomial MRFs [241], i.e., Potts models. Although multinomial MRFs are in general very powerful, they might not be best suited for the task of solving multiclass labeling problems at hand. As mentioned above, a naive implementation can result in large factor graphs and on top of that, solving the MAP inference problem in the resulting MRF has to be done approximately for problems of realistic size. We have already discussed the issues with solvers for this problem. For example, BP often suffers from convergence problems and inaccurate beliefs.

4.2 Lifting and Equitable Partitions

In many cases, a naive implementation of standard LP is slow. Besides an efficient parallelized implementation, possibly based on sparse matrices, there are various other modification that have been introduced to reduce the run time of LP because graph problems with several million nodes and thousands of labels call for a very efficient inference approach. *Lifted Label Propagation (LLP)*, which will be described in Section 5.3, is akin to lifted probabilistic inference (see Section 2.4). While LP is based on matrix-matrix-multiplications, we have described above already that we could also implement LP via GaBP and in turn use lifted GaBP to exploit symmetries in LP [166]. However, this does not allow the usage of an out-of-the-box GaBP implementation because changes to the GaBP algorithm are required to account for the lifted model [3]. This is very similar to the modifications that we have seen in the previous chapters for LBP and other lifted message passing algorithms. Counts are integrated into the messages, to reflect the structure of the lifted graph. Moreover, such a lifted LP approach is based on matrix inversion and requires several re-liftings which is impractical for graphs at massive scale.

However, having a problem specified in terms of a weighted graph and an inference algorithm in terms of matrix multiplications, we now take a look at lifting from the graph algebraic point of view. For this purpose, it is useful to define the notion of symmetries in graphs. A graph is symmetric if we can find an isomorphism to itself. Such an isomorphism is called *automorphism*. An automorphism of a graph induces a partition of the nodes and intuitively, the nodes in the same class are indistinguishable. Having an automorphism for a graph with n nodes, we can represent this mapping with help of an $n \times n$ matrix which commutes with the matrix specifying the graph. Unfortunately, there is no polynomial time algorithm known that decides whether two graphs are isomorphic [71]. A common intermediate step in determining graph isomorphisms is the calculation of the *coarsest equitable partition (CEP)* and it has been shown by Mladenov et al. [151] that the partition found by the Color Passing algorithm is identical to the CEP. The CEP also groups the nodes of the graph into classes and can equally be represented as a matrix. However, the CEP can be coarser than the partition induced by an automorphism [88].

In our lifted LP approach, we will directly lift the LP graph by calculating its CEP and we prove in the next chapter that we can then run vanilla LP on the compressed graph without any modifications to the LP algorithm itself. More specifically, we will use SAUCY[2] [108] to find the CEP of the graph which partitions the graph into equivalence classes. This is similar to the Color Passing approach that was run on the factor graph to lift BP. However, SAUCY does not follow a color passing approach. Instead, it suggests to look at the problem of lifting in terms of algebraic operations and facilitates the notion of a matrix for the partitioning. SAUCY is not only extremely fast at computing the CEP but the resulting fractional automorphism also allows to construct the lifted matrix very efficiently. The advantages of using SAUCY in lifted inference algorithms have also been recognized in other approach such as [170]. For further information on equitable partitions of graphs and matrices, we refer the reader to [152] and the references in there.

Next to lifting, as presented in the next chapter, other approaches to speed up LP also avoid computations to decrease the computational costs. A recent approach was presented by Fujiwara and Irie [60], who reduce the run time by updating only the scores of a subset of labels in each iteration. Also similar in spirit to LLP, and the general idea of lifted inference, are the ideas of Alexandrescu and Kirchhoff [6]. They proposed to merge identically labeled nodes to speed up LP, whereas LLP intuitively clusters the entire graph. Actually, there are many more efficient LP approaches, see e.g. the references in [60] for an overview. However, any of these approaches can be used on top of our compressed graph.

The next chapter will introduce our relational lifted LP in greater detail and describe how this LP approach can be used to augment online bibliographies with geo-tags. We will motivate the need for our LLP approach by the existence of web-scale bibliographies where standard LP runs out of memory and any decrease in run time is most welcome.

[2] `vlsicad.eecs.umich.edu/BK/SAUCY`

CHAPTER 5

Lifted Label Propagation

The previous chapter has described how *Label Propagation (LP)* can be used as an alternative to message passing algorithms for labeling problems. We have described how LP can be implemented by means of a simple matrix-matrix-multiplication and we have also motivated to look at lifting from a perspective independent of a message passing algorithm such as Color Passing. Accordingly, the papers most related to this chapter are:

- Zhu and Ghahramani [260], Zhu et al. [261]: the Label Propagation algorithm, its matrix-multiplication-based implementation, and entropy minimization for parameter learning.
- Katebi et al. [108]: the graph theory based symmetry detection which is implemented in SAUCY. SAUCY will be a core part of lifted LP.

In this chapter, we will enhance LP by techniques that have already been shown beneficial in Chapter 3 but are “non-standard” approaches in the LP literature. The contributions of this chapter can be summarized as follows:

C5.1 We use logical rules to construct a sparse similarity matrix for LP and we show how this approach can be used to augment online bibliographies with high accuracy geo-tags.

C5.2 Standard LP does not exploit symmetries in the similarity matrix. Hence, we use lifting techniques to compress the similarity matrix, obtaining a faster LP variant. This *Lifted Label Propagation (LLP)* requires less memory and still achieves identical labeling scores as ground LP.

C5.3 We show that the idea of bootstrapped evidence from the previous Section 3.3 can also be used to speed up convergence of LP.

These contributions have been submitted mainly in [86] and we proceed as follows. We will start by introducing our logical rule based LP which is used to label author-paper-pairs with geo-tags in online bibliographies. Afterwards we will show that LP's run time can also suffer from unexploited symmetries in the underlying graph as we have seen it in Chapter 3 for message passing algorithms. We will then describe how bootstrapping and Pseudo Evidence from Section 3.3 can be used for LP and quickly touch upon parameter learning for LP with a similarity function based on logical rules. We summarize this chapter before we continue with an in-depth analysis of our geo-annotated bibliographies.

5.1 Logical Rules for Label Propagation

As described above, the originally proposed similarity function leads to a completely connected graph that is impractical for very large problem instances. We therefore propose an alternative approach to constructing the edges and their weights. Intuitively, the weight of an edge should still be proportional to the similarity between the nodes, however, we will now define the similarity of two nodes based on formulas defined over logical predicates. Only those nodes are connected via an edge where at least one relation holds and hence $W_{i,j} > 0$. Similarly to the definition of MLNs (see Section 2.3.1), we define a set of tuples (F_a, w_a). Here, F_a is a formula in first-order logic and $w_a \in \mathbb{R}$ is the weight of that rule. Along with these rules goes a set of constants $C = C_1, \ldots, C_n$. Each constant C_i corresponds to a variable $V_i \in V$. For now, we only use rules that are defined at most over two nodes which gives us the following weight for each edge:

$$W_{i,j} = \exp\left(\sum_a w_a \cdot f_a(C_i, C_j)\right) = \exp\left(\sum_a w_a \cdot f_a(i, j)\right) \ . \tag{5.1}$$

Here, the feature function f_a corresponds to a grounding of formula F_a, with f_a evaluating to 1 if the ground formula F_a is `true`, and 0 otherwise. We construct the LP-graph now based on (F_a, w_a) and the constants in C. The graph construction distinguishes from the Markov network construction in MLNs in various aspects:

- MRFs based on a grounded MLN contain one binary node for each grounding of each predicate. For the relational LP, the nodes can have any finite range and each node corresponds to a domain object.
- The edges in an grounded MLN are based on the variables appearing together in formulas. In the LP-graph, edges correspond to one or several satisfied formulas over the nodes involved.

5.2 Geo-Tagging Bibliographies

We now present an application of our rule-based LP. In particular, we are interested in labeling online bibliographies with geographic information in order to conduct an analysis of the social behavior in the data. Geographic information about researchers over time and across countries is of great interest to different kinds of organizations and public authorities. Therefore, our goal is to create geo-annotated versions of large-scale bibliographies. Looking at bibliographies such as DBLP[1] [137], the goal is to tag every of the over five million author-paper-pairs with an affiliation and its geographic information. This would allow us to analyze movement data of researchers worldwide in the field of computer science. We are triggered by the observation that many collective human activities have been shown to exhibit universal patterns. However, the possibility of strong regularities underlying computer science researcher migration has barely

[1]`dblp.uni-trier.de`

been explored at global scale. Fortunately, in the Internet era the Web stores tons of data on researchers which is frequently updated. We demonstrate in this section and Chapter 6 that this information can be utilized to extract migration behavior of researchers and to learn models for the underlying process. Bibliographic sites on the Web, such as DBLP, are publicly accessible and contain millions of data records on publications. Papers are written virtually everywhere in the scientific world, and the affiliations of authors tracked over time could be used as proxy for migration. Unfortunately, many if not most of the prominent bibliographic sites do not provide affiliation information and there are no datasets available on the Web that immediately allow such an analysis. Instead, we first need to build a migration dataset, to conduct a large-scale investigation of migration afterwards. To this aim, we harvested data from different sources freely accessible on the Web and merged these into bibliographic databases. However, not all necessary affiliation information is available — it might actually be impractical to gather — and it is uncertain. Therefore, we have to rely on a Machine Learning algorithm to fill in the blank spots. This is precisely where our relational LP comes into play. Our relational LP is well suited to work on datasets based on bibliographies because the character of these databases is inherently relational. For example, the co-authorship relation allows to construct a social network of scientists and naturally the labels correspond to affiliations of an author-paper-pair.

Compiling the data in itself represents a significant advance in the field of quantitative analysis of research and migration patterns. Official and commercial records are often access restricted, incompatible between countries, and especially not registered across researchers. Instead, we present a general machinery for propagating geographical seed locations retrieved from the Web across online bibliographies. Next, we will describe how we gathered the data to construct a semi-labeled graph on which we run LP afterwards.

5.2.1 Harvesting Data

The Web provides several freely accessible bibliographies with millions of papers and authors. However, most of them do not contain affiliations or geo-information. For an extensive study of researcher's migration behavior this information is crucial though. Our goal is to label every author-paper-pair in a bibliography with the affiliation of that author and its geographic location. Although it is possible to manually, or semi-automatically, retrieve such labels, a full labeling of large databases, such as DBLP, is not practical with such methods. In addition, if we can build an effective automated machinery that helps us with this task, it is also much easier to update the database continuously with new papers arriving.

To this end, we assume an initial bibliography consisting of papers and theirs authors. We start by adding affiliation information to authors in our bibliographic database. To obtain the affiliations, we could take a look at every paper and extract the information from the title page of the paper. Clearly, this is not feasible for various reasons. Fortunately, there are other information sources on the Web that contain such information. One of these systems is the ACM Digital Library[2]. Unfortunately, ACM DL does not allow a full download of the data. Consequently, we retrieved the affiliation information of only a few author-paper-pairs randomly selected from ACM DL which we then matched with our initial bibliography. This gave us initial seed affiliations per author for different papers. In order to fill in the missing information, we want to resort to LP. To do so, we have to be a little bit more careful. First, the names of the affiliations in ACM DL are not in canonical form which results in a very large

[2]`dl.acm.org`

set of affiliation candidates. Secondly, although we have now partial affiliation information, we still lack exact geo-information of the organizations to identify cities, countries, and continents. Many of the affiliation names may contain a reference to the city or country but these pieces of information are not trivial to extract from the raw strings. Additionally, we want to have latitude and longitude values to enable further analysis and visualization.

Example 5.1. *Latitude and longitude data allow to calculate the exact distances between collaborators. One could postulate the hypothesis that the Internet-age has removed barriers and enabled long-distance collaborations. Hence, having this precise geo-information in our database allows to investigate such a hypothesis empirically.*

This geo-location issue can be resolved by using Google's Geocoding API[3]. Querying the API resulted in geo-tags for most of the affiliation strings. The remaining gap primarily rises from the fact that the Google API does not find geo-locations for all the retrieved affiliation strings. This is essentially because the strings contain information not related to the geo-location such as departments, e-mail addresses, among others. In any case, as our empirical results will show, this resulted in enough information to propagate the seed affiliations and in turn the geo-locations across the initial network of authors and papers.

5.2.2 Inferring Missing Data

Before we infer the missing author-paper-pairs, we revise our obtained affiliation data. To further increase the quality of our harvested affiliations, we hypothesize that there are actually not that many relevant organizations in computer science and these names need to get de-duplicated. This hypothesis is confirmed by services such as *MS Academic Search*[4] which currently lists only 13,276 organizations compared to our 150k+ names we obtained from crawling seed affiliations from ACM DL. Since we have the geo-locations attached to most of the affiliation strings now, we can use this information for a simple entity resolution which helps resolving this issue. More precisely, we clustered affiliations together for which the retrieved city coincide. Indeed, this approach does not distinguishing multiple affiliations per city such as *MIT* and *Harvard* which are both in Cambridge, MA, USA. However, our approach is simple and yet effective, and — as our empirical results show in Chapter 6 — the resolution is sufficient to establish strong regularities in the timing events.

Based on these known geo-locations, we fill in the missing ones by using the LP as described above. LP propagates the known cities to the unknown author-paper-pairs based on the similarity between the nodes. Correspondingly, the set of constants, C, consists of the author-paper-pairs and we have a node in the LP graph for each of these author-paper-pairs that we want to label with a city.

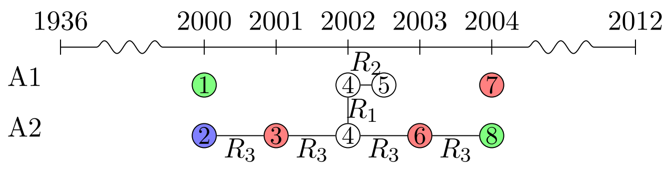
As we have mentioned above, a similarity function that is too dense will make the algorithm impractical, especially in large-scale bibliographies with millions of authors and papers. Hence, we resort to logical rules to formulate the similarity. These rules are based on relations such as co-authorship between the authors associated with the nodes. Specifically, in order to define the edges, we considered the following functions over the set of nodes that return facts about the nodes:

- `author`(i): returns the author of an author-paper node

[3] `developers.google.com/maps/documentation/geocoding`
[4] `academic.research.microsoft.com`

P	A	Y	Aff	Aff*
1	1	2000	g	g
2	2	2000	b	b
3	2	2001	r	r
4	1,2	2002	?,?	r,r
5	1	2002	?	r
6	2	2003	r	r
7	1	2004	r	r
8	2	2004	g	g

(a) Example database



(b) The graph for our rule-based LP

Figure 5.1: LP for geo-tagging bibliographies. Missing geo-tags from the example database (a) are estimated by propagating the known cities/geo-locations across the network of authors and papers (b).

- `paper(`i`)`: returns the paper of an author-paper-node
- `year(`i`)`: returns the the year of publication of an author-paper node.

Based on these functions, we can now define the following logic based rules that add a rule-specific weight w_a to every matching edge $e_{i,j}$. Initially, we set all edge weights $W_{i,j}$ to zero. The first rule R_1,

$$W_{ij} = W_{ij} + w_1 \text{ if } \texttt{paper(}i\texttt{)} = \texttt{paper(}j\texttt{)}$$

adds a weight between two nodes if the nodes belong to two authors that co-author the paper associated with nodes i and j. The second rule R_2,

$$W_{ij} = W_{ij} + w_2 \text{ if } \texttt{author(}i\texttt{)} = \texttt{author(}j\texttt{)} \wedge \texttt{year(}i\texttt{)} = \texttt{year(}j\texttt{)}$$

adds a weight whenever two nodes corresponds to different publications by the same author in the same year. And finally the third rule R_3,

$$W_{ij} = W_{ij} + w_3 \text{ if } \texttt{author(}i\texttt{)} = \texttt{author(}j\texttt{)} \wedge \texttt{year(}i\texttt{)} = \texttt{year(}j\texttt{)} + 1$$

fires when the nodes belong to two publications of the same author but written in subsequent years.

Example 5.2. *The construction process of the LP matrix and its corresponding graph is depicted in Figure 5.1b for the example publication database in Figure 5.1a. The example database is missing the affiliation information for papers 4 and 5 which is denoted by the "?" in the "Aff"-column. The graph for propagating the information is constructed as follows. There is a node in the graph for each pair of an author from column "A" and the corresponding paper in column "P". Two nodes are connected if they are written by the same author in the same or subsequent years or if two researchers co-author them. The colors of nodes indicate known cities and white nodes indicate unknown locations.*

We then run LP on the constructed graph, to get a distribution over the possible cities for every unlabeled node. Running LP on the graph for our running example in Figure 5.1b, we see that LP labels the unknown nodes. Looking at the last column "Aff*" in Figure 5.1a, the previously missing labels for papers 4 and 5 have been inferred and the graph is now completely labeled. This example shows a very simple case because the unlabeled nodes are only connected to red neighbors. Therefore, the color of the unlabeled nodes is deterministically determined because only red can be propagated to them. Usually, the situation is less clear. For example, if paper 6 for A_2 was green, we would have a probabilistic interpretation of the label scores for the nodes. The unlabeled nodes could be labeled with either green or red. In this situation, with paper 3 for A_2 labeled red and paper 6 for A_2 labeled green, there would be a tie between both colors because paper 4 of author A_2 is identically connected to the labeled nodes. We will now present experiments of the rule based LP on publicly available online bibliographies.

5.2.3 Experiments

With LP based on logical rules at hand, let us now turn towards filling in missing geo-tags in bibliographic databases. There are different choices as a starting point for the data harvesting process. Ultimately, we are interested in a bibliography covering all different scientific disciplines. To begin with, however, we focus on computer science. For an qualitative evaluation, we are interested in a dataset with as much ground truth as possible, to answer the following question:

Q5.1 Is the relational LP capable of producing meaningful geo-tags with high accuracy for our bibliography based on few seed locations?

In order to answer **Q5.1**, we require a manually curated dataset for which we have a relatively large amount of affiliations in advance. As we will see, the AAN bibliography — which is described in detail in the next section — serves as a good starting point. For all our experiments, we used the following weights for the rules described above $w_1 = 1$, $w_2 = 3$, and $w_3 = 2$. These rules were found by a grid search on a small subset of the data. Furthermore, all experiments were run on a Linux machine with 64GB RAM and 20 cores.

GeoAAN

The *ACL Anthology Network (AAN)* [184] is a comparatively small, but manually curated, dataset that contains affiliations for many authors and papers from the natural language processing community. In total, the dump in use from August 2013 contains 19,410 publications written by 15,397 authors from a time span across five decades. Although the dataset is manually curated, we cannot directly use the affiliations in the provided form. Many of the affiliation strings represent multiple affiliations for one author. Since we are interested in the geographical location of a researcher, we have to reduce these strings to a single organization. We split the affiliation strings and assume that the first mentioned affiliation is more likely the residential location of a researcher. After reducing the available affiliations to the city level, the resulting number of author-paper-pairs is 49,530 and 33,061 of these nodes are labeled with one of 802 cities. The LP graph G has a total of 145,594 edges, resulting in a very sparse matrix W, respectively T. By removing an increasingly number of labels from the graph, we construct test sets of different sizes which we use for the evaluation. We start by removing 10% of the labels, obtaining a graph with 55% of the nodes labeled. We then gradually add 10% of the nodes to the test set until only 6% of the nodes are labeled. We apply this dataset construction ten times, to allow for multiple re-runs of the experiment. Table 5.1 shows the average accuracy of the predicted labels for each test set when running LP for 200 iterations. Having access to only 36% of the labels or more, we can achieve an accuracy ≥ 0.80. As expected, reducing the number of labeled nodes slowly decreases the performance. With only 6% labeled nodes, we still achieve an accuracy of 0.58, which is a high performance on a multiclass labeling problem with

Labeled Nodes	Accuracy
6%	0.58
12%	0.67
18%	0.72
24%	0.75
30%	0.78
36%	0.80
42%	0.81
48%	0.82
55%	0.83

Table 5.1: Label accuracy for the AAN dataset with a varying number of initially labeled nodes. One should note that accuracy is a very challenging performance measure for a multiclass labeling problem with around 800 classes.

roughly 800 classes; the accuracy of random labels would be 0.00125. We will refer to the AAN dataset with augmented geo-tags as GeoAAN. These results clearly answer **Q5.1** in favor of our proposed relational LP approach. However, to infer global patterns for migration across various fields of computer science, neither the scope nor the size of GeoAAN are satisfactory. Instead, we want to use a bibliography, such as DBLP, with millions of papers. However, as we will describe next, one cannot simply just run LP on a dataset of that size. Therefore, we will describe a new LP approach that exploits symmetries in the LP-graph and splits the label matrix into chunks, in order to obtain label scores of over five million author-paper-pairs.

5.3 Lifted Label Propagation

Although the naive implementation of LP is just a simple matrix-matrix-multiplication, this becomes challenging with several million nodes. While W, and respectively T, are very sparse, Y^t becomes denser with every iteration. Resulting in a pure dense matrix if the graph was completely connected. Eventually, this presents an obstacle in terms of both, computation time and memory requirements.

Example 5.3. *We have observed in our experiments that W has often $< 1\%$ of the cells filled if we use logical rules as described in Section 5.2 to construct W. Which means that a problem of $5M \times 5M$ can easily be stored in RAM which is intractable with a dense (Gaussian) similarity function. However, with around 4,000 labels, the label matrix requires more than 160GB RAM with a 64bit-float representation.*

To alleviate some of this burden, we can exploit the latent symmetries in the structure of T. In our proposed approach, *Lifted Label Propagation (LLP)*, we do so by means of equitable partitions which is akin to lifted probabilistic inference in Chapter 3. If the graph contains symmetries, lifting makes every iteration of LP more efficient. As we are often facing the problem of slow convergence, LP should strongly benefit from lifting in the presence of symmetries.

The algorithm proceeds as follows (illustrated in Figure 5.2): we first partition the nodes according to their initial labels (Figure 5.2a). We then compute the *coarsest equitable partition*

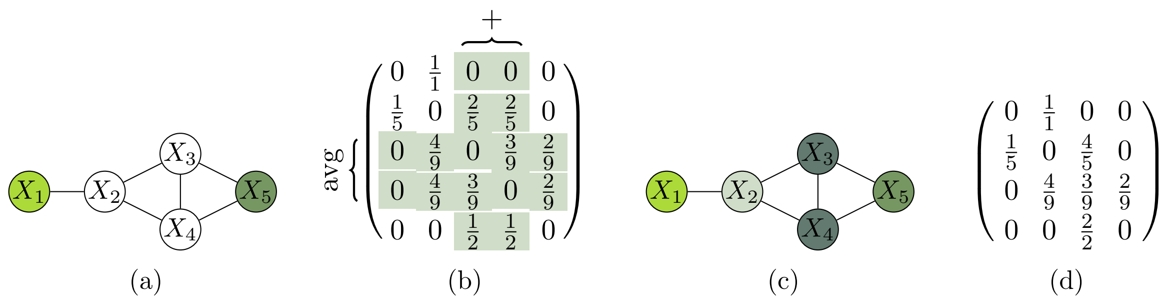


Figure 5.2: Toy example for LLP. The LP graph in Figure 5.2a contains two labeled nodes (green shades) and three unlabeled nodes (white). The corresponding similarity matrix is given in Figure 5.2b. Based on the CEP, the graph can be colored as in Figure 5.2c. The partition clusters X_3 and X_4 together. The corresponding compressed matrix is depicted in Figure 5.2d.

(CEP) of T which preserves the initial label partition. From the partition, we obtain a (hopefully) smaller quotient matrix $\mathfrak{T}$ by:

- replacing the set of all columns corresponding to nodes in the same class by their sum **(S.1)**;
- replacing the set of all rows of nodes in the same class by their average (Figure 5.2b) **(S.2)**.

We carry out the second step on Y^0 as well to obtain the compressed label matrix $\mathfrak{Y}^0$. Finally, we run LP with $\mathfrak{T}$ (Figure 5.2d) and $\mathfrak{Y}^0$ in place of T and Y^0. As we will show next, we can perfectly recover Y^k from $\mathfrak{Y}^k$ and thus the result of LP can be recovered from the result of LLP.

Theorem 5.1. *Running LP on the compressed matrix $\mathfrak{T}$ returns identical label scores as running LP on T.*

Proof. First observe that algebraically, $\mathfrak{T}$ and $\mathfrak{Y}^0$ can be written as $\mathfrak{T} = \widehat{B}TB$ and $\mathfrak{Y}^0 = \widehat{B}\mathfrak{Y}^0$, where B is an $n \times p$ matrix having $B_{ik} = 1$ if node i is in class k of the CEP and 0 otherwise (representing the summing of columns; S.1). $\widehat{B}$ is a $p \times n$ matrix defined as $\widehat{B}_{ki} = 1/|\text{class } k|$ if i is in class k and 0 otherwise (representing the averaging of rows; S.2). We first reference the following facts [79]: ($\clubsuit$) $\widehat{B}B = I_p$ ($\widehat{B}$ acts as left inverse of B); ($\heartsuit$) $B\widehat{B}T = TB\widehat{B}$ (the matrix $B\widehat{B}$ commutes with T since B comes from the CEP of T).

As a first step, we need to show that $Y^{t+1} = B\widehat{B}Y^{t+1}$. We proceed by induction. For the case of $t = 0$, let us note that $P = B\widehat{B}$ is a fractional automorphism obtained from the CEP which is a doubly stochastic matrix. Figure 5.3 shows P for our running example. We distinguish two cases, first assume row i in Y^0 contains a non-zero entry in column j, i.e., $Y^0_{ij} = 1$. There can only be a single non-zero cell by construction. If node i corresponds to class k, which has $|\text{class } k|$ members, P contains $|\text{class } k|$ non-zero entries P_{il} in row i which all have value $1/|\text{class } k|$. The columns l in P correspond to the rows in Y^0 that have class label k, and hence $Y_{lj} = 1$. Therefore, if we multiply Y^0 with P, Y^0_{ij} is a sum of $|\text{class } k|$ elements with value $1/|\text{class } k|$, i.e., 1, and it remains unchanged. In the second case we have $Y^0_{ij} = 0$ and all other members of that class k have $Y^0_{lj} = 0$ as well. Therefore, Y^0_{ij} remains unchanged too and we have $Y^0 = PY^0 = B\widehat{B}Y^0$. For the induction step, we have

$$Y^{k+1} = TY^k \overset{\text{ind.}}{=} TB\widehat{B}Y^k \overset{\heartsuit}{=} B\widehat{B}TY^k = B\widehat{B}Y^{k+1} \,, \tag{$\spadesuit$}$$

$$B \cdot \widehat{B} = P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 5.3: Fractional Automorphism obtained by the CEP in Figure 5.2

where the second equality follows from our induction hypothesis. Note that we omitted the discussion of the push-back operation, however, it can be shown that the above holds after the push-back as well. It follows from the same reasons that the integrated push-back works in the ground case. The only difference is that an entire cluster now remains unchanged instead of a single node.

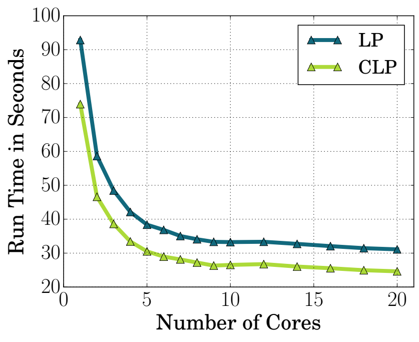
Finally, induction also shows that $Y^{k+1} = B\mathfrak{Y}^{k+1}$. We can show that $Y^0 = B\mathfrak{Y}^0$ by similar arguments as above. Each row in B has exactly one non-zero entry, namely $B_{ik} = 1$ representing the cluster membership of node i. All nodes in that cluster have the same label or no label at all. Similarly, each row in $\mathfrak{Y}^0$ also has at most one non-zero cell which exactly corresponds to the label of the class. Therefore, multiplying B and $\mathfrak{Y}^0$ produces exactly the ground label matrix Y^0 because $Y^0_{il} = \sum_j B_{ij}\mathfrak{Y}_{jl}$ can only be one if node i has label k. The induction shows

$$\begin{aligned} B\mathfrak{Y}^{k+1} &= B(\widehat{B}TB)\mathfrak{Y}^k \overset{\text{ind.}}{=} B\widehat{B}T(B\widehat{B})Y^k \overset{\heartsuit,\clubsuit}{=} B\widehat{B}TY^k \\ &= B\widehat{B}Y^{k+1} \overset{\spadesuit}{=} Y^{k+1} . \end{aligned} \qquad \square$$

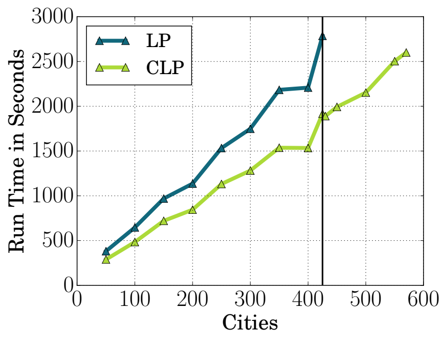
Observe now that $\mathfrak{T} \in \mathbb{R}^{p\times p}$ where p is the number of classes of the CEP of T. That is, we have one row and column per cluster instead of per node. Thus, if $p \ll n$, we need to solve a much smaller system. Moreover, using the highly efficient implementation of SAUCY [108], the CEP computation is done in $\mathcal{O}\left[(m+n)\ln n\right]$ time. So, even in case of little to no symmetry, there is only very little computational overhead due to symmetry detection. One striking advantage of the LLP approach is the fact that we do not have to adapt the original algorithm and can run standard LP implementations on the lifted matrix. This simplifies the implementation a lot compared to the lifted message passing approaches described in Chapter 3 and we can additionally use existing efficient parallelized LP implementations. On top of that, the construction of the compressed matrix is a simple linear algebra operation and straightforward to implement compared to the lifted factor graph construction for lifted message passing algorithms.

5.3.1 Experiments

We saw above that our rule-based LP returns high quality labels on a geo-tagging task for a small, manually curated, dataset. However, for an in-depth analysis of researcher's migration behavior, we would like to construct a database as large as possible. To this end, we want to use LLP to tag a large-scale dataset and demonstrate that it is significantly superior over LP in terms of run time and memory usage. Therefore, we pose the following two questions to be answered in this section:



(a)



(b)

Figure 5.4: (a) Run time of LP on the AAN dataset with an increasing number of available cores. (b) Due to the large size of DBLP, it is not possible to run LP with all cities in a single run. Instead, we have to split the cities into batches and run (L)LP on each batch separately. As one can see, LLP can use up to 570 cities in a single run while LP can only handle 425 cities at once.

Q5.2 Is LLP faster than its ground counterpart?

Q5.3 Does LLP require less memory than ground LP?

GeoAAN

Of course, LLP is not limited to cases where running LP is not feasible. Instead LLP also reduces run time on smaller problems that fit completely into main memory. Hence, we start off by giving results for LLP on the AAN corpus and investigate the reduction in run time due the compression of the LP graph on this dataset as well as the potential benefit of using a parallelized linear algebra library. After running SAUCY on the ground graph and using the returned CEP to construct the lifted LP graph, the lifted similarity matrix contains only 36,178 nodes, i.e., the size of the graph has been reduced to about 73% of the original size (see Figure 5.5(left)). We now ran both variants for 200 iterations on the compressed matrix and Figure 5.4a shows the run times of LP and LLP for a varying number of cores. Here, the run time measures the total time required by the script. In the case of LLP, this includes the compression of the matrix as well. By using LLP instead LP, we reduce the run time by 20%. Of course, the relative run time is further reduced with more iterations, as the overhead of the compression vanishes. If we solely look at the run time of the matrix-matrix-multiplications, we see that LLP requires up to 25% less time. One can also see that LP and LLP equally benefit from using additional cores.

GeoDBLP

To get closer to our final goal, i.e., investigating migration regularities in large-scale online bibliographies, we used our lifted relational LP to augment DBLP with geo-tags. Compared to AAN, DBLP is roughly 100 times larger but contains relatively speaking way fewer labels. More precisely, we use a DBLP dump from February 2012 which contains 1,894,758 papers written by 1,080,958 authors. While DBLP offers a best-effort entity disambiguation for authors, it does not contain any affiliation information at all, and hence lacks the required seed labels to run

	AAN	DBLP
Grnd Nodes	49,539	5,033,018
Lbld Nodes	30,025	478,365
Grnd NNZ	291,188	58,414,302
Lft Nodes	36,178	3,744,065
Lft NNZ	239,172	43,410,077

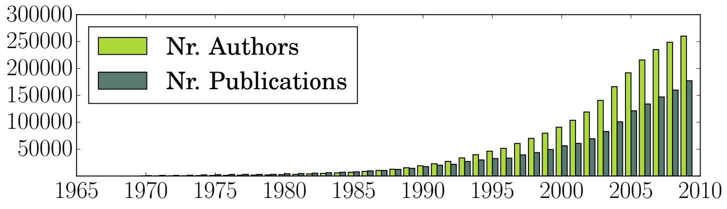


Figure 5.5: (left) Dataset statistics of AAN and DBLP. (right) The number of authors and publications per year for the DBLP dump in use. As one can see, DBLP has been growing constantly over the past decades from 1970 until 2010.

LP. Therefore, we used the affiliation information obtained from ACM (cf. Section 5.2.1) as seed affiliations. After matching DBLP and ACM papers, the enhanced DBLP dump contained 159,068 different affiliation names in total and 10% of the 5,033,018 author-paper-nodes were labeled with an affiliation. The resolution to the city level reduced the label alphabet to 4,350 cities and the goal is to label the remaining unknown nodes with one of these locations. The size of DBLP has been growing rapidly over the past years. Figure 5.5 shows the development of the number of authors and publications from 1965 until 2010. We omitted the year 2011 because DBLP has a certain latency such that the data for 2011 was not complete by early 2012. As one can see, the number of computer scientists, as well as their productivity, have been growing enormously over the past decades. The growth of the database further motivates us to develop a fast and efficient LP approach, so updating the database continuously does not form an obstacle.

A illustrated in Example 5.3, with a dataset of such size, running LP on a single machine is not easily possible anymore, even with modern hardware. Additionally, the current LAMA release cannot handle arbitrarily large matrices which additionally limits the maximum number of labels. One way to overcome this hurdle, is to split the label matrix into k chunks and do the multiplications separately. Afterwards, we can merge the results and obtain the final labeling. Using this approach, we compared LLP with LP to see how many cities each variant can handle in a single run. We started with the first 50 cities and added cities as possible. The results are depicted in Figure 5.4b for 200 iterations and show that LLP requires far less RAM because the matrices $\mathfrak{T}$ and $\mathfrak{Y}$ are much smaller than the uncompressed ones. We can now run LP with up to 570 cities in a single run which only requires eight machines in a distributed setting. On the contrary, LP can only handle 425 at a time and would require 11 machines. The run times in 5.4b exclude the time needed for the lifting which is negligible because we only have to compress T once and not for every chunk. The average total run time, including lifting, for 200 iteration on the DBLP dataset with LLP is about 5.40 hours. This is a lot faster than LP which took on average around 7.49 hours.

The effect of running rule based LP on the entire DBLP database with only initial seeds is shown in Figure 5.6. We refer to the resulting database as GeoDBLP. One can clearly see that the worldwide coverage increases substantially. These geo-locations of publications alone can already reveal interesting insights such as the most productive research cities in the world. Such statistics at full scale only become available by propagating the initial seed locations through the DBLP graph. According to GeoDBLP, the five most productive cities by the end of 2011 are Beijing, Singapore, Pittsburgh, Cambridge, and Los Angeles. The main focus of our analysis, however, will be the timing of migration. At this point, we want to mention that the resulting dataset is not perfect of course. For example, missing an elaborated

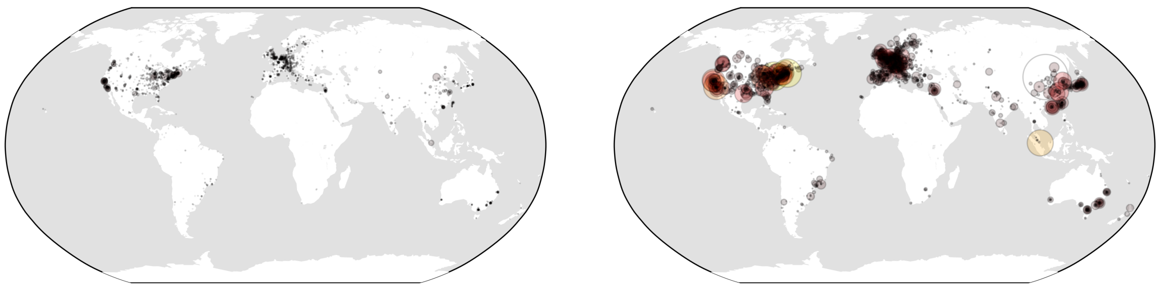


Figure 5.6: As one can see, running Label Propagation greatly improves the content of our database. The number of geo-tagged author-paper-pairs increased substantially, showing the publication activities across the world.

entity resolution on the affiliation's level and missing disambiguation on authors in DBLP can lead to unreasonable entries. The latter can lead to researchers with an unrealistic amount of publications assigned. Nevertheless, the analysis of the data in Chapter 6 will show that the timing patterns inferred from the database are fairly regular and can be explained intuitively. Additionally, it will be shown that the regularities found in DBLP are very similar to the ones founds in AAN which further supports the quality of the GeoDBLP dataset.

The experiments on the AAN and DBLP datasets clearly answered questions **Q5.2** and **Q5.3**, showing that LLP is superior over LP in terms of resources without any decrease in quality.

5.4 Bootstrapped Label Propagation

Bootstrapped inference using *Pseudo Evidence (PE)*, as presented in Section 3.3, is not restricted to MAP inference in graphical models. PE can also be incorporated into other algorithms. We investigated bootstrapped LP on a binary classification task as done in [62]. Here, a venue was to be predicted for papers in the CiteSeer$^{\mathbf{x}}$ citation network. Bootstrapped LP, implementing a kind of adaptive push-back, converged to solutions very similar to standard LP, but in only about half the number of iterations. Similar as the lag-parameter in BLM, we can incorporate a burn-in phase to achieve solutions even closer to standard LP.

5.5 Learning Weights of Logical Rules

When we labeled the online bibliographies above, we determined appropriate parameter values based on a small subset of the data. However, a proper Machine Learning based approach is desirable to find weights that optimize a predefined criterion. Originally, Zhu and Ghahramani [260] proposed the entropy minimization criterion for learning the parameters of the similarity function in LP. We will follow their approach and present now the gradient for the parameter learning in the case where the similarity matrix is constructed based on logical rules. Each parameter corresponds to a weight w_a of a rule. Another benefit of entropy minimization is related to the idea of bootstrapped LP. If the label scores quickly converge to a distribution with low entropy, we can start setting Pseudo Evidence early. Following [260], the optimization objective in entropy minimization is:

$$H = -\sum_{i,j} Y_{ij} \ln Y_{ij} \tag{5.2}$$

To simplify the derivation of the gradient, we will now assume without loss of generality that the normalized similarity matrix T and the label matrix Y can be written in the following form:

$$T = \begin{bmatrix} T_{ll} & T_{lu} \\ T_{ul} & T_{uu} \end{bmatrix} \text{ and } Y = \begin{bmatrix} Y_l \\ Y_u \end{bmatrix} ,$$

where l is the number of labeled nodes and u is the number unlabeled nodes, i.e., $n = l + u$. The contribution to H of nodes with a given label is zero, therefore, we can rewrite Equation (5.2) as follows:

$$H = -\sum_{i=l+1}^{l+u} \sum_{k=1}^{d} Y_{ik} \ln Y_{ik} \ .$$

Now, the partial derivative with respect to w_a is:

$$\begin{aligned} \frac{\partial H}{\partial w_a} &= -\sum_{i=l+1}^{l+u} \sum_{k=1}^{d} \frac{\partial Y_{ik}}{\partial w_a} \ln Y_{ik} + Y_{ik} \frac{1}{Y_{ik}} \frac{\partial Y_{ik}}{\partial w_a} \\ &= \sum_{i=l+1}^{l+u} \sum_{k=1}^{d} (-\ln Y_{ik} - 1) \frac{\partial Y_{ik}}{\partial w_a} \end{aligned}$$

To obtain a value for $\frac{\partial Y_{ik}}{\partial w_a}$, we have to resort to the fact that the Y_u can be written as:

$$Y_u = (I - T_{uu})^{-1} T_{ul} Y_l \ ,$$

which is a simple reformulation of Equation (4.2) based on algebraic rules and the fact that $Y_U = T_{uu} Y_u + T_{ul} Y_l$. Further details can be found in [260]. We now also follow [260], to obtain the derivative of Y_u with respect to w_a:

$$\frac{\partial Y_u}{\partial w_a} = (I - T_{uu})^{-1} \left(\left[\frac{\partial T_{uu}}{\partial w_a} \right] Y_u + \left[\frac{\partial T_{ul}}{\partial w_a} \right] Y_l \right)$$

Up to this point, the derivation of the gradient followed the work in [260]. Since we are using the similarity function based on logical rules, the rest of the derivation looks different:

$$\begin{aligned} \frac{\partial T_{ij}}{\partial w_a} = \frac{\partial}{\partial w_a} \frac{W_{ij}}{\sum_k W_{ik}} &= \frac{\frac{\partial W_{ij}}{\partial w_a} \sum_k W_{ik} - W_{ij} \sum_k \frac{\partial W_{ik}}{w_a}}{(\sum_k W_{ik})^2} \\ &= \left[\frac{\partial W_{ij}}{\partial w_a} - \frac{W_{ij}}{\sum_k W_{ik}} \sum_k \frac{\partial W_{ik}}{w_a} \right] / \sum_k W_{ik} \\ &= \left[\frac{\partial W_{ij}}{\partial w_a} - T_{ij} \sum_k \frac{\partial W_{ik}}{\partial w_a} \right] / \sum_k W_{ik} \end{aligned}$$

With the definition of our similarity in Equation (5.1), we obtain:

$$\frac{\partial W_{ij}}{\partial w_a} = \frac{\partial}{\partial w_a} \exp\left(\sum_k w_k \cdot f_k(i,j)\right) = W_{ij} \cdot f_a(i,j) \ ,$$

and lastly,

$$\frac{\partial T_{ij}}{\partial w_a} = \left[W_{ij} \cdot f_a(i,j) - T_{ij} \sum_k W_{ik} \cdot f_a(i,k) \right] / \sum_k W_{ik}$$

We can now use gradient descent to find the optimal parameter values minimizing the entropy. However, using this approach on our bibliography datasets is computationally challenging as it requires the inversion of the matrix $(I - T_{uu})$ in each iteration of the optimization. Even for smaller datasets, such as GeoAAN, this matrix has almost 20k rows and columns, and therefore learning is not efficient. In conclusion, further enhancements such as lifting should be incorporated. Alternatively, approximations such as piecewise learning [218] should be considered in LP learning as well.

5.6 Interim Conclusion

In this chapter, we have demonstrated how large-scale, transnational bibliographies can be constructed from the Web with the help of a small set of initial locations and AI techniques. In particular, we have shown how *Label Propagation (LP)* can be applied to infer missing locations when only a few seed locations are available. Such enriched bibliographies can be used to discover surprisingly simple and strong regularities as we will show in Chapter 6. To label large-scale graphs with several million nodes more efficiently, we introduced *Lifted Label Propagation (LLP)* which runs LP on a lifted graph by compressing the similarity matrix in a preprocessing step. We proved the correctness of LLP and demonstrated that LLP can significantly reduce the run time and memory consumption of LP without sacrificing performance at all.

We see several interesting avenues for future work. We have seen in Section 5.4 preliminary results on ground bootstrapped LP which looked very promising. Chapter 3 discussed various lifting approaches and also combined the idea of bootstrapping with lifting. It would be interesting to combine the ideas of LLP with bootstrapping as well. However, combining LLP with Pseudo Evidence requires to efficiently re-lift the graph iteratively and an algorithm that can handle small changes in the evidence efficiently. This essentially motivates to investigate an algorithm akin to LEACP for LLP.

In Section 5.1, we suggested to use logical rules to construct the affinity matrix for LP. Clearly, this is only one of several possibilities to define the similarity between two nodes but it worked well in combination with lifting. One should further investigate which alternative similarity functions also produce symmetric and sparse graphs that can be used to obtain high quality labels. A concrete example of another similarity function is based on the k nearest neighbors according to a predefined distance function. This graph construction approach should also go well along with LLP and produce symmetries as long as k is not too large. Additionally, this will also produces sparse matrices for small values of k that can still be used with large-scale problems.

In many applications — independent of the similarity function — we cannot expect very large LP graphs to be perfectly symmetric and lossless data compression is not applicable because of insufficient symmetries. The knowledge about labels is often not systematic and can break symmetries additionally as it has often been observed in lifting for probabilistic inference. Therefore, one should investigate approximate lifting for LP as well. For example, in the spirit of iLBP (cf. Section 2.4.2) or the work by Sen et al. [198]. Both approaches would suggest to cluster variables based on approximate label scores after a sufficient number of initial iterations of LP. Van den Broeck and Darwiche [227] have also shown how to use low-rank boolean matrix factorization to speed up lifted probabilistic inference in MLNs. One should investigate similar ideas for LLP too. In a piecewise approximation approach, we can partition the graph and hope that each partition allows for considerable lifting. We can try to partition the graph in such a way that the partitions are as symmetric as possible. We then run lifted inference and combine the results of the partitioning afterwards. The idea of partitioning large graphs for LP was recently introduced in [225] and the idea of lifting subgraphs is akin to the piecewise lifting by Ahmadi et al. [4].

So far, the proposed avenues for future work were focusing on algorithmic aspects of LP but the data harvesting pipeline also leaves room for further improvement. We currently do a very simple resolution based on the retrieved city of an affiliation. This can be improved by a more sophisticated entity resolution based on SRL approaches. One can easily adapt the entity resolution approach presented in Section 3.3.4 to the problem of resolving affiliations. Furthermore, we currently use only three rules to construct the LP graph. Looking deeper into the nature of the problem, one can come up with additional rules that should hold among nodes. Still, as we will see in Chapter 6, running LP on both datasets produces sufficiently accurate results for an in-depth analysis of migration patterns in GeoAAN and GeoDBLP.

LP presented by Zhu and Ghahramani [260] is only defined for the finite case. As we will see in the next chapters, many problems are defined most precisely by random variables over the natural numbers. For example, imagine a problem where we want to predict the number of publications of an author. Each node in the graph could represent an author and the labels would correspond to the number of publications in a certain timespan. Each author is connected to its co-authors or other researchers from the same affiliation. The idea of a harmonic solutions is appealing in this scenario as well. Essentially, the number of publications of an author amounts to the (weighted) average of their neighbors. Assuming that each variable is Poisson distributed, it might be sufficient to iteratively update the mean of the distribution based on the neighboring means. Chapter 7 will present a PGM following the idea of Poisson distributed random variables that can alternatively be used for the prediction of publication counts as we will see.

CHAPTER 6

Inferring Social Phenomena From the Web Based on Counts

6.1 Migration Analysis Based on Geo-Tagged Bibliographies

We will now motivate the global meaning of our geo-annotated bibliographies that we generated based on Label Propagation in Chapter 5 in detail and show how a statistical analysis reveals interesting patterns about science and society. So far, it was only possible to analyze the productivity of researchers by means of bibliographic databases and catalogs. Now, with geo-annotated bibliographies at hand, we can extend this analysis by an geographic dimension that did not exist at this scale beforehand. Here, we show that timing events of migration within different countries exhibit remarkable similarities. Specifically, we look at the distribution governing the data of researcher migration inferred from the Web. We are able to find statistical patterns and regularities that explain the migration of researchers in a plausible way. For instance, we show that the job market in academia can be treated as a Poisson process with individual propensities to migrate following a log-normal distribution over the researcher's career stage. That is, although jobs enter the market constantly, researchers are generally not "memoryless" but have to care greatly about their next move. The propensity to make $k > 1$ migrations, however, follows a gamma distribution suggesting that migration at later career stages is "memoryless". This aligns well but actually goes beyond scientometric models typically postulated based on small case studies. On a very large, transnational scale, we establish the first general regularities that should have major implications on strategies for education and research worldwide.

In general, unveiling, explaining, and ultimately predicting sociological processes remain key challenges in understanding the behavior of society and science all around the globe. Over the last years, many collective human activities have been shown to exhibit universal patterns, see e.g., [262, 142, 61, 10, 40, 92, 135, 76, 22, 136, 72, 240, 11] among others. However, the possibility of universal patterns across timing events of researcher migration — the event of transfer from one residential location to another by a researcher — has barely been explored at global scale and data is mostly collected on a national level. This is surprising since education and science is, and has always been, international. For instance, according to the UNESCO Institute for Statistics, the global number of foreign students pursuing tertiary education abroad

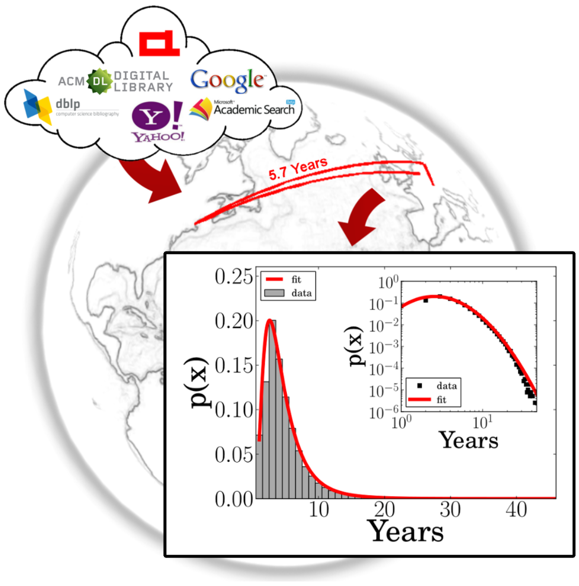


Figure 6.1: We will now infer migration patterns from our GeoDBLP database.

increased from 1.6 million in 1999 to 2.8 million in 2008[1]. As the UN notes [226], "there has been an expansion of arrangements whereby universities from high-income countries either partner with universities in developing countries or establish branch campuses there. Governments have supported or encouraged these arrangements, hoping to improve training opportunities for their citizens in the region and to attract qualified foreign students." Likewise, science thrives on the free exchange of findings and methods, and ultimately of the researchers themselves, as noted by the German Council of Science and Humanities [65]. The European Union even defined the free movement of knowledge in Europe as the "fifth fundamental freedom" [2]. Similarly, the US National Science Foundation argues that "international high-skill migration is likely to have a positive effect on global incentives for human capital investment. It increases the opportunities for highly skilled workers both by providing the option to search for a job across borders and by encouraging the growth of new knowledge" [187]. Generally, due to globalization and rapidly increasing international competition, today's scientific, social, and ecological challenges can only be met on a global scale both in education and science, and are accompanied by political and economic interests. Thus, research on scientist's migration and understanding it, play key roles in the future development of most computer science departments, research institutes, companies and, nations — especially if fertility continues to decline globally [133]. By analyzing our geo-augmented bibliographies, we want to provide decision makers and analysts with statistical regularities and patterns of migration to support policy formulation and funding in research.

On the first sight, reasons to migrate are manifold and complex: political stability and freedom of science, family influences such as long distance relationships and oversea relatives, and personal preferences such as exploration, climate, improved career, better working conditions,

[1]United Nations Education, Scientific and Cultural Organization, Data extract (Paris, 2011), accessed on 19 April 2011 at: `http://stats.uis.unesco.org/unesco/TableViewer`

[2]Council of the European Union (2008a), p. 5: "In order to become a truly modern and competitive economy, and building on the work carried out on the future of science and technology and on the modernization of universities, Member States and the EU must remove barriers to the free movement of knowledge by creating a 'fifth freedom' ..."

among others. Despite this complex web of interactions, we show in this chapter that the timing events of migration within different countries exhibit remarkable simple but strong and similar regularities. Specifically, we look at the distribution governing the data of researcher migration inferred from the Web for various reasons. Although, efforts to produce comparable and reliable statistics are underway, estimates of researcher flows are inexistent, outdated, or largely inconsistent, for most countries. Moreover, official (NSF, EU, DFG, etc.) and commercial (ISI, Springer, Google Scholar, AuthorMapper, ArnetMiner) records are often access restricted. On top of it, these information sources are often highly noisy. We have seen above that we can construct large-scale geo-tagged bibliographies by means of an AI machinery. We have seen that papers are written virtually everywhere in the scientific world, and the affiliations of authors tracked over time will be used as proxy for migration. GeoDBLP is the basis for our statistical analysis due to its size. However, we will also conduct a partial analysis on GeoAAN, to verify our findings on a dataset with more ground truth affiliation information. The main contribution of this chapter can be summarized as follows:

C6.1 We present the first statistical regularities and patterns for researcher migration in computer science based on geo-annotated online bibliographies introduced in Chapter 5.

The results were presented in [84] and can be further subdivided as follows:

- **Regularities:**
  - **(R1)** A specific researcher's propensity to migrate, i.e., to make the next move, follows a log-normal distribution. That is, researchers are generally not "memoryless" but have to care greatly about their next move. This is plausible due to the dominating early career researchers with non-permanent positions. This regularity of timing events is remarkably stable and similar within different continents and countries around the globe.
  - **(R2)** The propensity to make $k > 1$ migrations, however, follows a gamma distribution suggesting that migration at later career stages is "memoryless". That is, researchers have to care less about their next move since the majority of positions are permanent in later career stages.
  - Since jobs enter the market all the time, R1 and R2 together suggest that the job market can be treated as a Poisson-log-normal process.
  - **(R3)** The brain circulation, i.e., the time until a researcher returns to the country of their first publication, follows a gamma distribution. That is, returning is also memoryless. Researchers cannot plan to return but rather have to pick up opportunities as they arrive.
  - **(R4)** The inter-city migration frequency follows a power-law. That is, cities with a high exchange of researchers will exchange even more researchers in the future. So, investments into migration pay off.
- **Statistical patterns:**
  - **(SP1)** A link analysis of the author-migration graph discovers additional statistical patterns such as migration sinks, sources and incubators.
  - **(SP2)** We can also use the author-migration graph to find the hottest migration cities for researchers in computer science.

These results validate and go beyond migration models based on small case studies. Ultimately, they can provide forecasts of (re-)migration which can help decision makers who investigate actively the migration and the return of their researchers to reach better decisions regarding the timing of their efforts.

6.1.1 Related Work

Already Zipf [262] in 1946 investigated inter-city migration. He analyzed so called gravity models. These models incorporate terms measuring the masses of each origin and destination, and the distance between them. Afterwards, log-linear regression techniques are used to statistically calibrate the models. Over the years, several modifications and alternatives have been postulated, see e.g. [40, 205] and the references in there.

To characterize the productivity of scientists, Stewart [214] reviewed the Poisson-log-normal model for bibliometric/scientometric distributions. Our results actually suggest that this Poisson-log-normal model can also be used for migration and scales to a transnational, massive level. Sums of Poisson processes and other Poisson regression models, as well as ordinary least-squares, have actually a long tradition within migration research, see [215, 186] for recent overviews. All of these approaches, however, have considered small scale data only which is additionally often publicly undisclosed. Neither did these studies consider researcher migration in computer science explicitly.

To the best of our knowledge, only very few large-scale migration studies have been presented so far. For example, Zagheni and Weber [255] have recently analyzed a large-scale e-mail dataset to estimate international migration rates, but not specific to computer scientists, since the occupation was unobserved. Moreover, Zagheni and Weber have not presented any statistical regularities nor dealt with missing information. Using a large-scale, IP-address-based dataset, State et al. [213] also investigated mobility data and migration flows. While Sheldon et al. [201] presented a model for migration probabilities between countries, they also used access restricted data and did not aim at unveiling the sociology of computer science. In contrast, we present the first large-scale migration study for computer science inferred from publicly available data using AI and report statistical regularities in this dataset.

Our results connect to many other human activities that have been shown to exhibit patterns, see e.g. [262, 142, 40, 76] among others. Further prominent examples are mobility patterns drawn from communication [135, 94] and web services [173], as well as mining blog dynamics [72] and social ties [240]. These examples include several approaches of extracting large-scale sociological information from the Web which has attracted a lot interest lately. However, most of this work has focused on services like Twitter or Facebook, see e.g. [32, 176], and most importantly, the work was directly conducted on the raw data; no AI or Machine Learning technique has been used to fill in missing data. Moreover, these information sources are not open access and, hence, results are not reproducible. In contrast, our methods and findings complement these results by highlighting the value of using freely available information from the Web together with AI to deal with missing information. Our worldwide lens onto researcher migration enables analysts to develop global strategies for migration in research and to enrich the public debate. Following this discussion, the papers most related to this chapter are:

- Stewart [214] presents the Poisson-log-normal model which we will use to describe the job market in computer science.

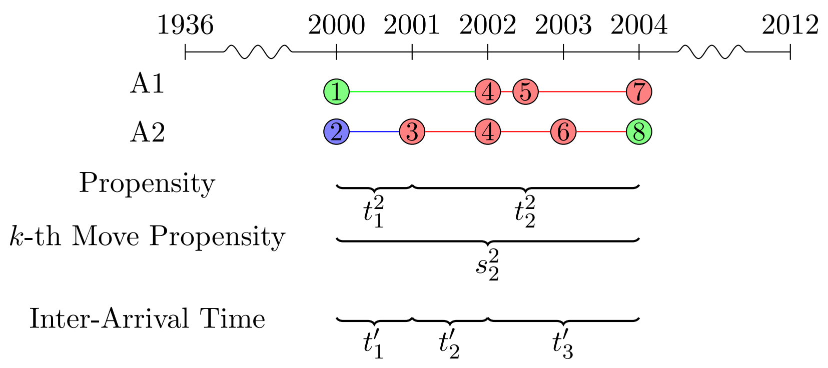


Figure 6.2: Individual propensities and (inter-)arrival times illustrated for our running example from Figure 5.1. Every node denotes a publication and the node colors indicate different affiliations, i.e., there are three affiliations here: blue, red, and green. From this, we can read off migration. For example, author A2 moves from blue to red and from red to green. A researcher's propensity is their likelihood of migrating. The k-th move propensity is their likelihood of making $k > 1$ moves. This should not be confused with the (inter-)arrival times of the job market, i.e., of the overall Poisson process, which counts for all hops across researchers.

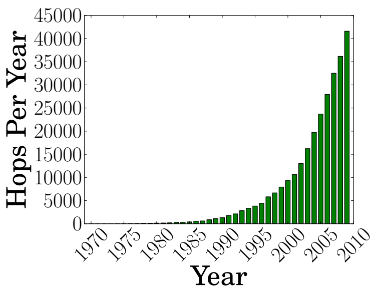
- Gonzales et al. [76] analyze human mobility patterns on an urban scale and serves as a representative for this area of work where regularities in human activities are found and appropriate models are developed.
- Cohen et al. [40] are also concerned with migration analysis and propose a Generalized Linear Model which is capable of predicting future number of migrants. However, Cohen et al. only analyze data of eleven countries while we look at a truly transnational dataset.

6.1.2 Sketching Migration

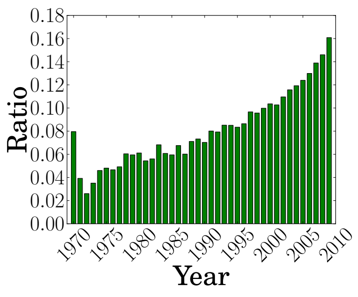
Unfortunately, we cannot directly observe the event of transfer from one residential location, respectively institution, to another one by a researcher. Instead, we use the affiliations mentioned in their publication record as a proxy. In the case of GeoDBLP, we have 4,318,206 author-paper-pairs with geo-tags after running Label Propagation. However, this list may still be noisy and does not provide the timing information easily. To illustrate this, an author may very well move to a new affiliation and publish a paper with their old affiliation because the work was done while being with the old affiliation. Therefore, we considered *migration sketches* only. Intuitively, a sketch captures only the main stations of a researcher career. More formally, we define a migration sketch as the set of the unique affiliations of an author ordered by the first appearance in the list of publications.

Example 6.1. *For instance, in our running example, we have* $[2000 : Aff_g, 2002 : Aff_r]$ *for author* A_1 *and* $[2000 : Aff_b, 2001 : Aff_r, 2004 : Aff_g]$ *for author* A_2*. That is, author* A_1 *has two different affiliations,* Aff_g *appearing in 2000 the first time and the first publication with* Aff_r *in 2002.*

Of course, this approach has the drawback that we can not capture if a person returns to an earlier affiliation after several years. Finally, we dropped implausible entries from the resulting sketch database. For instance, we dropped sketches with more than ten affiliations. It is very unlikely that a single person has moved more than ten times and these sketches should



(a) Number of Hops



(b) Ratio

Figure 6.3: Migration statistics over time in GeoDBLP. The ratio between hops and authors per year (b) does not grow as fast as the number of hops (a) or authors (Figure 5.5(right)).

rather be attributed to an insufficient entity disambiguation. Having the migration sketches at hand, we can now define a *migration* or *move* of a researcher as the event of transfer from one residential location to another one by a researcher in their migration sketch. Figure 6.2 shows the moves of authors A_1 and A_2 in our running example. In total, we found 310,282 migrations in GeoDBLP. The number of moves per year is displayed in Figure 6.3a and it shows that the number of moves increases over the years super-linearly. However, when we normalize the number of moves by the number of authors, we see roughly a linear slope (cf. Figure 6.3b). With this information at hand, we will now start to investigate the statistical properties of researcher migration.

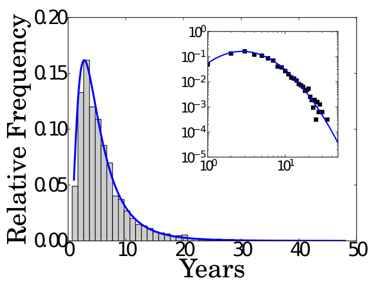
6.1.3 Regularities of Timing Events

As mentioned above, reasons to migrate are manifold. Despite this complex web of interactions, we now show that researcher migration shows remarkably simple but strong global regularities in the timing.

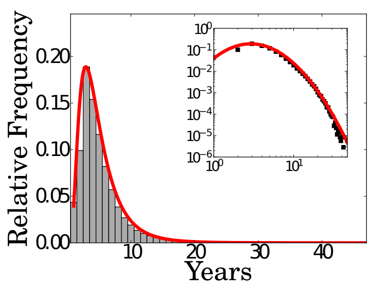
R1: Migration Propensity is Log-Normal

Given the migration sketches, we can now read off timing information. First, we estimate the propensity to transfer to a new residential location or institution across scientists. To do so, let T_i^j be the point in time when a researcher moves from one location to the next one. T_0^j is the first temporal reference we have for an author, i.e., the year of their first publication listed in the bibliography. Let t_i^j be the time between the T_{i-1}^j and T_i^j. We call t_i^j, i.e., the time between two moves, the *migration propensity* (see Figure 6.2). It reflects the bias of researchers to stay for a specific amount of time until moving on. Figure 6.3 shows that the number of hops per author has been increasing constantly over the past decades, indicating that researchers are more flexible and willing to move nowadays. We should note that reading off this information is only possible with our annotated database. DBLP alone does not provide such information.

Figure 6.4 shows the best fitting distribution for t_i^j in terms of log-likelihood and KL-divergence among various distributions such as log-normal, gamma, exponential, inverse-Gaussian, and power-law using maximum likelihood estimation for the parameters (see Table 6.1 for details).



(a) GeoAAN



(b) GeoDBLP

Figure 6.4: Migration propensity. Due to the many early stage careers, with non-permanent contracts, the individual migration propensity is best fitted by a log-normal distribution in both datasets. That is, although jobs enter the market all the time, researchers are generally not "memoryless" but have to care greatly about their next move, and this timing is a multiplicative function of many independently distributed factors.

	Global		Africa		Asia		Europe		North America		Oceania		South America	
	LL	KL	LL	KL	LL	KL	LL	KL	LL	KL	LL	KL	LL	KL
Log-Normal	**-2.33**	**0.01**	**-2.25**	**0.04**	**-2.20**	**0.01**	**-2.34**	**0.01**	**-2.42**	**0.01**	**-2.24**	**0.01**	**-2.20**	**0.01**
Gamma	-2.35	0.03	**-2.25**	**0.04**	-2.21	0.02	-2.35	0.03	-2.45	0.04	-2.25	0.02	**-2.20**	0.02
Inverse-Gauss	-2.34	**0.01**	-2.26	0.05	-2.21	0.02	**-2.34**	**0.01**	-2.43	**0.01**	-2.25	0.02	-2.21	0.02
Weibull	-2.40	0.08	-2.28	0.08	-2.26	0.07	-2.40	0.07	-2.50	0.09	-2.28	0.06	-2.25	0.06
Exponential	-2.59	0.27	-2.53	0.32	-2.50	0.31	-2.61	0.28	-2.67	0.25	-2.53	0.30	-2.54	0.35
Power-Law	-2.77	0.43	-2.68	0.44	-2.65	0.43	-2.80	0.45	-2.87	0.43	-2.68	0.43	-2.73	0.52

Table 6.1: Goodness-of-fit for individual migration propensities in GeoDBLP. Shown are the log-likelihood and the KL-divergence at global scale and continental scale for the individual propensities. Maximum values are bold. Looking at the log-likelihood, overall log-normal is fitting best (significant at $p = 0.05$ according to a paired McNemar test on the continental as well as national level).

Both fits, for GeoAAN and GeoDBLP, are log-normal distributed [5, 214]. That is, the log of the propensity is a Gaussian distribution with density

$$f(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} \ .$$

The parameters μ and $\sigma^2 > 0$ are the mean and the standard deviation of the variable's natural logarithm. The mean of the log-normal distribution is $E[X] = e^{\mu + \sigma^2/2}$ and the mode is equal to $e^{\mu - \sigma^2}$. This fit is a plausible model due to Gibrat's "law of proportionate effects" [69]. The underlying propensity to move is a multiplicative function of many independently distributed factors, such as motivation, open positions, short-term contracts, among others. That is, such factors do not add together but are multiplied together, as a weakness in any one factor reduces the effects of all the other factors. This leading to log-normality can be seen as follows. Recall that log-normal random variables are transformed to normal random variables by taking the logarithm and that by the law of large numbers, the sum of independent random variables becomes a normal distribution regardless of the distribution of the individuals. When random variables are multiplied, as the sample size increases, the distribution of the product becomes a

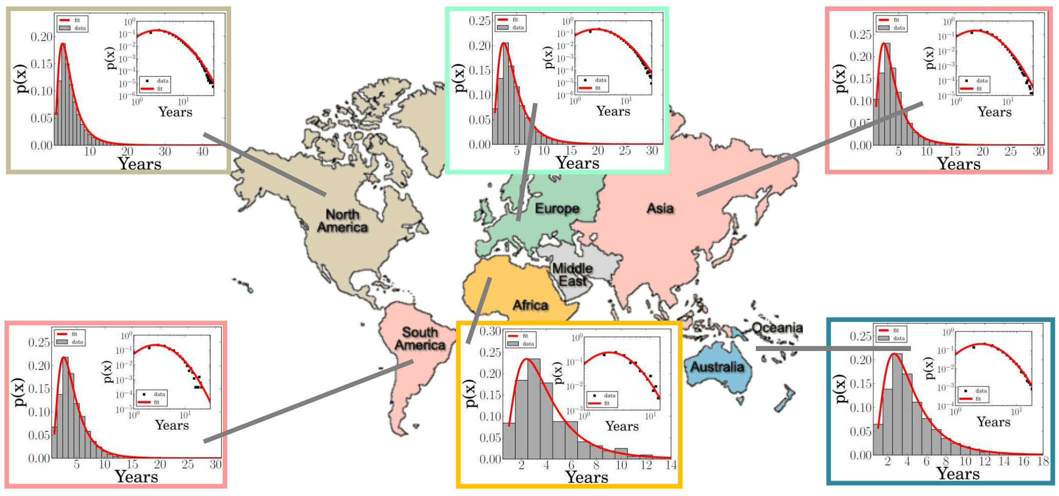


Figure 6.5: In GeoDBLP, migration propensities are remarkably similar across continents and again best fitted by a log-normal distribution. Thus, the timing of research careers has no cultural boundaries across continents.

log-normal distribution regardless of the distribution of the individuals. This might explain why the log-normal distribution is one of the most frequently observed distributions in nature and describes a large number of physical, biological and even sociological phenomena [139].

Moreover, although the overall job market can be seen as a Poisson process, as we will show later on (see Section 6.1.3), it is useful that the migration propensity is not exponential. It is precisely this non-Poisson that makes it possible to make predictions based on past observations. Since positions are occupied in a rather regularly way, upon taking a new position, it is very unlikely that a researcher will take up another position very soon. In the Poisson case, which is the dividing case between clustered and regular processes, a researcher should be indifferent to the time since the last move.

Even more importantly, the log-normality of the propensity can be found across continents and countries of the world, see Figure 6.5 and Figure 6.6, where we considered only moves originating from a continent, respectively country, in GeoDBLP. These results show that timing research careers has clearly no cultural boundaries. Due to the lack of datapoints, we have omitted this analysis on GeoAAN.

Although one can expect that the information we use is noisy and the aforementioned procedures in Section 5.2 and Section 6.1.2 may inject additional noise, a comparison of Figure 6.4a and Figure 6.4b shows that the shapes of both fits look quite similar and both distributions are from the same family. This gives confidence that our findings are reliable because the results on a mainly manually curated dataset do not differ hugely from the data that was inferred completely automatically. Nevertheless, let us describe some of the possible reasons for noise injection in GeoDBLP. The first possible source of noise is ACM DL. In the very first step it could happen that we match an ACM paper with a paper from DBLP incorrectly. On top of that, the affiliations provided by ACM are not necessarily always correct. In the next phase, we reduce the affiliation strings to their geographical locations. It is not always possible to uniquely identify a geo-tag for every affiliation and additionally we cannot assume that the response by Google's Geocoding API always returns an appropriate geo-information.

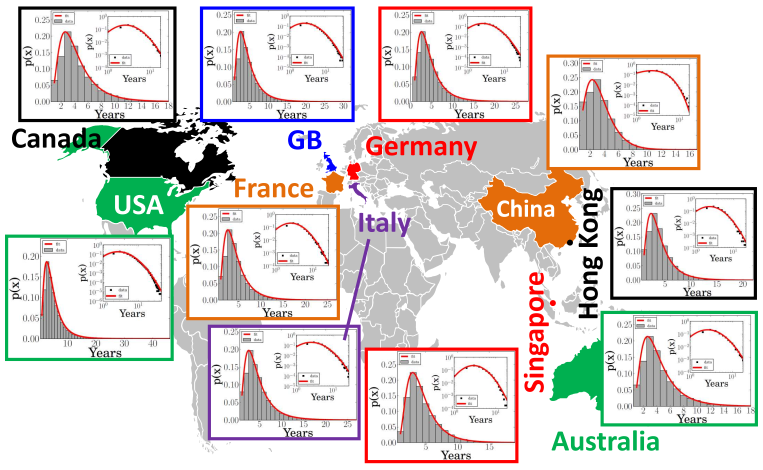


Figure 6.6: Zooming in on migration propensities in GeoDBLP: across the most productive countries in the world, migration propensities are best fitted by a log-normal distribution. The countries USA, China, Germany, UK, Australia, Singapore, Canada, France, Italy, and Hong Kong are shown as representatives. Except for China, all are best fitted by log-normal distributions. China's migration propensity follows a gamma distribution.

Hence, it might happen that we assign a wrong city to an author-paper-pair. We then run our relational Label Propagation to infer the missing labels and the algorithm will not label every unknown node correctly. Finally, creating a sketch from the list of publications can create incorrect resumés. For example, we currently cannot map researchers returning to a city where they have stayed previously. Nevertheless, the results in Section 5.2.3 have shown that our rule-based Label Propagation is capable of labeling nodes with high accuracy, convincing us to trust the data in GeoDBLP.

However, there are differences in the estimated parameters, and in turn of the estimated statistics such as mean and variance, for the two bibliographies. For instance, the difference in the obtained parameters can be easily described in terms on the mean values of the distributions. Calculating the means for both networks, we obtain about 5.7 years on the GeoDBLP dataset and 6.4 years on the GeoAAN dataset. I.e., in the AAN data, it takes a researcher about eight months longer to make the next move. This, however, can also be attributed to ACL being a sub-community and AAN being a much smaller sample. Additionally, one can see that the distributions reach their mode in the third year in both cases, i.e., most hops occur in the third year. Thus headhunters, for example, should start approaching young potentials in their fourth year, or even earlier. On the other hand, one should probably reconsider the common practice, e.g., in the EU and the US, of having projects lasting only three years to fill in the gap.

R2: k-th Migration Propensities are Gamma

Figure 6.7 shows the best fitting distribution in terms of log-likelihood and KL-divergence among various distributions such as log-normal, gamma, exponential, inverse-Gauss, and power-law using maximum likelihood estimation for the propensity to make $k > 1$ moves (see Table 6.2 for

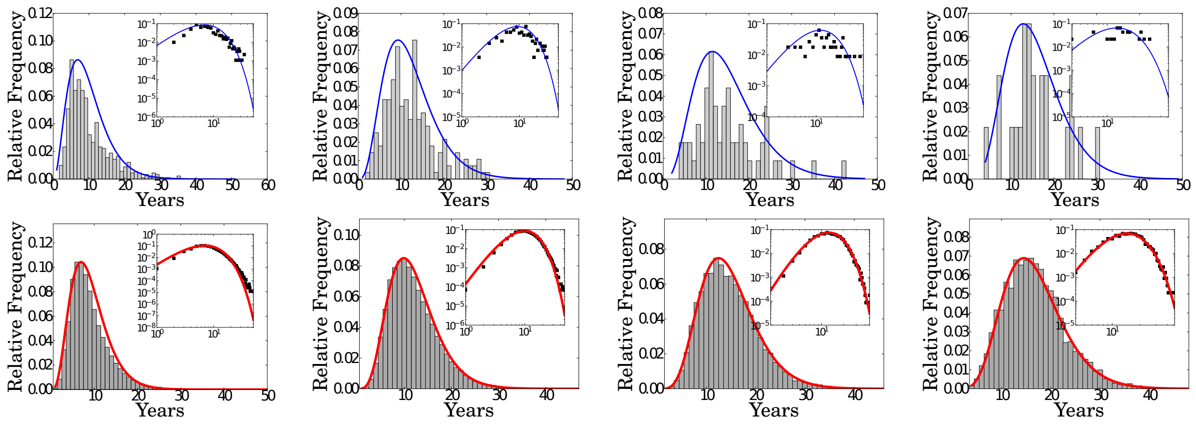


Figure 6.7: k-th migration propensities. (top) Regularities in GeoAAN. (bottom) Regularities in GeoDBLP. For larger numbers of moves, i.e., for senior scientists, the k-th move propensities (from left to right with $k = 2, 3, 4, 5$) turn into a gamma distribution due to permanent positions. This suggests that migration at later career stages is "memoryless", i.e., it follows an exponential distribution.

details). More precisely, the *k-th move propensity* for an author A_i is defined as $s^i_k = \sum_{j=1}^{k} t^i_j$ and most accurately described by a gamma distribution in both datasets,

$$f(x) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-\frac{x}{\beta}} \ ,$$

with shape $\alpha > 0$, scale $\beta > 0$, and $\Gamma(\alpha) = \int_0^\infty s^{\alpha-1} e^{-s} ds$. The mean value of the gamma distribution is given by $E[X] = \alpha \cdot \beta$. This finding suggests that migration at later career stages is "memoryless". This follows from the theory of Poisson processes. For Poisson processes, we know that the inter-arrival times are independent and obey an exponential form,

$$f(x) = \gamma e^{-\gamma x} \ ,$$

where $\gamma > 0$ is called the *intensity rate.* The inter-arrival time is the time between each pair of consecutive events in a Poisson process. The important consequence of this fact is that the distribution of x conditioned on $\{x > \hat{x}\}$ is again exponential. Hence, we can assume that the remaining time after we have not moved to a new position at time $\hat{x}$ has the same distribution as the original time x, i.e., it is "memoryless". The connection between the exponential inter-arrival times and the gamma distribution rises from the insight that the sum of exponential random variables is gamma distributed. This follows from two facts. First, the gamma distribution turns into an exponential distribution when setting $\alpha = 1$. Second, we can show by induction that the sum of gamma distributed variables is again gamma distributed. Moreover, we have shown that the time until the k-th move — the k-th move propensity — has a gamma distribution; it is the sum of the first k propensities of senior researchers. So, the propensities for the next move turn exponential for later career stages. This is plausible, since early career researchers have seldom taken many positions and, hence, we consider here rather senior researchers, which typically have permanent positions. These senior researchers do not have to care as much about their moves because they are often on a tenured position. Therefore, the longer a tenured researcher stays with an affiliation, the less likely it is that they

	k = 2		k = 3		k = 4		k = 5	
	LL	KL	LL	KL	LL	KL	LL	KL
Log-Normal	**-2.78**	**0.00**	**-2.98**	**0.00**	-3.12	0.01	-3.20	0.02
Gamma	**-2.78**	0.01	**-2.98**	**0.00**	**-3.11**	**0.00**	**-3.19**	**0.01**
Inverse-Gauss	**-2.78**	0.01	-2.99	0.01	-3.12	0.01	-3.20	0.02
Weibull	-2.84	0.07	-inf	inf	-3.14	0.04	-inf	inf
Exponential	-3.19	0.42	-3.49	0.51	-3.69	0.58	-3.82	0.64
Power-Law	-3.81	1.03	-4.27	1.29	-4.26	1.16	-4.06	0.88

Table 6.2: Goodness-of-fit for k-th move propensities in GeoDBLP. Results are depicted for $k = 2, 3, 4, 5$ at global scale and maximum values are bold. Here, gamma is the significantly best fitting function (significant at $p = 0.05$ according to a paired McNemar test on the continental as well as national levels).

move again. As a consequence, competing universities have to top the current position of a senior researcher if they want to hire them.

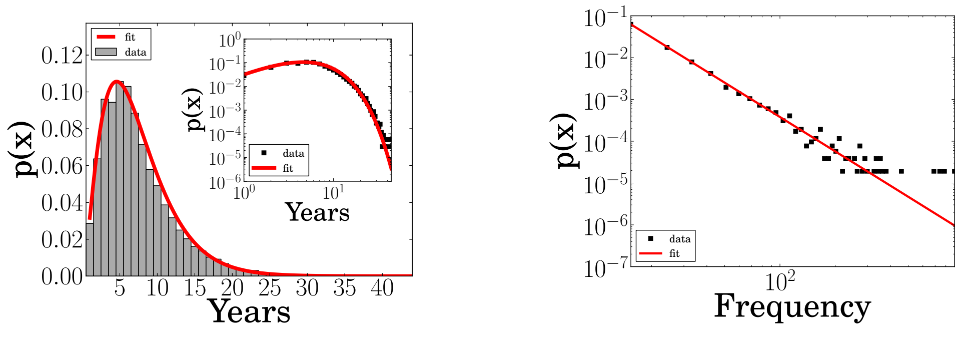
If we compare the regularities in GeoAAN and GeoDBLP in Figure 6.7, we see that the shapes are again similar and from the same family for $k > 2$, again suggesting that GeoDBLP has high quality labels. Similar to the case of the general propensity, we can compare the mean values of the gamma distributions, to give an intuitive description of the difference in the parameters. For example, if we look at the case of $k = 3$, we expect 12 years to pass until a researcher makes the third move in GeoDBLP. Looking at the mean value of GeoAAN's distribution, we obtain almost the same value, namely 12.1 years. However, due to the lower number of authors and papers in the AAN dataset, the data is sparse for higher values of k and the results become less reliable. In this small network, there are not sufficiently many researchers making that many moves such that a comparison for high values of k is not meaningful.

Job Market is Poisson-Log-Normal

So far, we have shown that the propensities, let us call them λ for now, to move to a new residential location, respectively institute, follow a log-normal distribution. We have also shown that the k-th move propensities follow a gamma distribution, suggesting that the propensities of senior researchers are exponential. The latter fact already points towards a Poisson model since a *Poisson process* is a renewal process with exponentially distributed inter-arrival times and gamma distributed waiting times. More precisely, we here postulate that the job market follows a Poisson-log-normal model [214]. That is, given a specific scientist's migration propensity λ, their probability of migrating follows a simple Poisson model:

$$p(x) = \frac{\lambda^x}{x!} e^{-\lambda} \ ,$$

for $x \in \mathbb{N}$. Thus λ, i.e., the rate of the Poisson process, is a function of the migration propensity. The number of migrations for all scientists having the same λ value will follow the same Poisson process. Moreover, since the sum of Poisson processes is again a Poisson process, we know that every finite sample of scientists with λs drawn from a log-normal is again following a Poisson process. Thus, assuming the job market to be a Poisson model is plausible, although we cannot directly observe the openings of new jobs in the market from our data. It actually tells us



(a) Brain Drain (b) Inter-City Migration

Figure 6.8: (a) The circulation of expertise, i.e., the time until a researcher returns to the country of their first publication follows a gamma distribution. Therefore, returning is also memoryless. (b) The inter-city migration frequency distribution follows a power-law. That is, cities with a high exchange of researchers will even exchange more researchers in the future.

that the arrival of job openings is memoryless. Open positions should always be announced as they come. On a global scale, there is no point in waiting to announce them. There are always researchers ready to take it. And, individual researchers can always look out for new job openings.

R3: Brain Circulation is gamma

Brain circulation, or more widely known as *brain drain*, is the term generically used to describe the mobility of high-level personnel. It is an emerging global phenomenon of significant proportion as it affects the socio-economic and socio-cultural progress of a society and a nation. Here, we defined it as the time until a researcher returns to the country of their first publication. Only $29,398$ out of $193,986$ (15%) mobile researchers, i.e., researchers that have moved at least once, and out of a total of $1,080,958$ (3%) researchers returned to their roots when examined in terms of publications. As to be expected from the statistical regularity for k-th move propensities, it also follows a gamma distribution, as shown in Figure 6.8a. Since a gamma distribution is the sum of exponential distributions, returning is memory less. Researchers cannot plan to return to their roots but rather have to pick up opportunities as they arrive.

6.1.4 Link Analysis of Migration

Link analysis techniques provide an interesting alternative view on our migration data. That is, we view migration as a graph where nodes are cities and directed edges are migration links between cities. More formally, the author-migration graph is a directed graph $G = (V, E)$ where each vertex $v \in V$ corresponds to a city in our database. There is an edge $e_{ij} \in E$ from vertex V_i to vertex V_j if there is at least one author who moved from an affiliation in city V_i to an affiliation in city V_j. We can also add weights to the edges by considering the flow, i.e., the total number of moves between two cities in one direction.

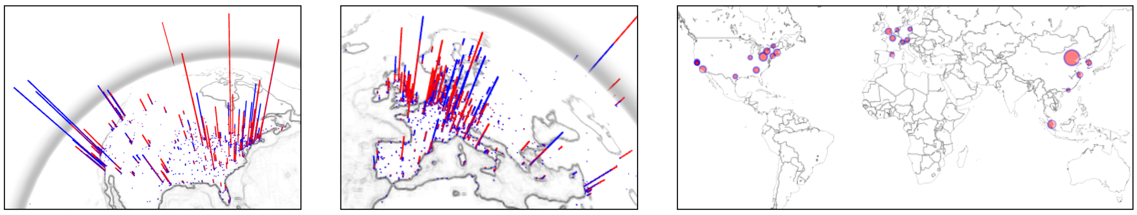


Figure 6.9: (left and middle) Running HITS on the directed author-migration graph reveals sending, receiving, and incubator cities. Shown are representative cities in North America (left) and Europe (middle). The size of spikes encodes the value of the "authority" (blue) and "hub" (red) scores. Incubator cities have well balanced scores. As one can see, the European cities rather send researchers. US cities at the east-cost are incubators, and west-cost cities receive researchers. (right) Top 25 migration cities ranked by PageRank.

R4: Inter-City Migration is Power-Law

Triggered by Zipf's early work and other recent work on inter-city migration [262, 40, 205], we investigated the frequency of inter-city researcher migration. The frequency of a connection between two cities can be seen as knowledge exchange rate between the cities. It is a kind of knowledge flow because one can assume that researchers take their acquired knowledge to the next affiliation. If one looks at the author-movement-graph as a traffic network, high frequent connections corresponds to highly used streets. Figure 6.8b shows the distribution of inter-city migration with a fitted power law using maximum likelihood estimation after removing low-frequency connections. A likelihood comparison to other distributions such as log-normal and gamma revealed that a power law fits the best. Thus, there are only few pairs of cities with frequent researcher exchange on the one hand and many low-frequent pairs on the other hand. However, as the power-law suggests, cities with a high exchange of researchers will exchange even more researchers in the future. Hence, early investments into migration pay off.

SP1: Migration Authorities and Hubs

Next, we were interested in mining the migration authorities and hubs. To do so, we use Kleinberg's HITS-algorithm [122] on the author-migration graph. The algorithm is an iterative power method and returns two scores for every node in the graph, which are known as *hubs* and *authorities*. This terminology arises from the Web where hubs and authorities represent websites. Hubs are pages with many outlinks and authorities are pages with many inlinks.

In our context, inlinks correspond to researchers arriving in a city — they pick up a new position — whereas an outlink corresponds to a researcher leaving a city — e.g. funding ends. Hubs can be seen as "sending" cities, i.e., they send out researcher across the world. On the other hand, authorities can either be cities where people want to stay and tenure positions are available or where people drop out of research, i.e., heading to industry. These are "receiving" cities. Moreover, if we make the assumption that only high-quality students and scientists get new positions, one may view sending cities as institutions producing high profile scientists but also cannot hold all of them due to restricted capacities or low attractiveness. In contrast, receiving cities might have the capacities and reputation to hold many migrating researchers or highly interesting industrial jobs are close by. Cities having generally high scores for both values are incubators: they attract a lot of migration but also send them to other places.

Figure 6.9 shows the sending and receiving scores for cities in the representative regions of the US and Europe[3]. The US clearly shows an east-coast/west-coast movement. The east coast aggregates many sending cities while receiving cities dominate the west coast. This is plausible. Not only are there many highly productive universities on the west-coast, but labor market for high-tech workers in, say, the Bay Area is the strongest in a decade. Thousands of new positions are being offered by small startups and established tech giants. However, one should view many of the east-coast cities as incubators since they have high overall scores. The scores of European cities are typically much smaller, see Figure 6.9(middle). Europe is dominated by sending cities. Few exceptions are Berlin, Munich, Stockholm, and Zurich. The largest receiving city in the world is Singapore. This is plausible as well. The city-state is known for its remarkable investment in research in recent years, as for example noted by a recent Nature Editorial [1][4]. In contrast, the largest sending city by far is Beijing. This is also plausible. There has been upsurge in Chinese emigration to Western countries since the mid-first decade of the 21st century [131]. In 2007, China became the biggest worldwide contributor of emigrants.

SP2: Moving Cities

Following up on HITS, we also computed PageRank [174] on the author-migration graph. Compared to HITS, PageRank produces only a single score: a page is informative or important if other important pages point to it. More formally, by converting a graph to an ergodic Markov chain, the PageRank of a node V_i is the (limit) stationary probability that a random walker is at V_i. In the context of migration, this has a natural and very appealing analogy. The PageRank computes the (limit) stationary probability that a random researcher is at a city.

To compute the PageRank of a city, we call it the *MigrationRank*, the author-migration graph is transformed into the PageRank-matrix on which a power method is applied to obtain the PageRank-vector, containing a score for every node in the graph. The transformed matrix also contains the stochastic adjustment identical to the random surfer in the original work. That is, a researcher can always migrate from one affiliation to another affiliation, even if no one else did so before. Figure 6.9(right) shows the top 25 cities in the world according to the MigrationRank. Compared to the productivity reported for different cities in Section 5.3, one can clearly see many similarities but although notable differences. The US is not only productive but thrives on migration. Vancouver is among the top 25 when it comes to migration but not when it comes to productivity. Generally, productivity does not imply a high migration rank. Beijing, however, is top in both when it comes to productivity and migration. Singapore is higher ranked for migration than for productivity. European cities seem to also thrive on migration more than on productivity. At least, there are much more cities in the top 25 than for productivity.

6.1.5 Summary

In this section, we presented statistical regularities and patterns on migration inferred from two different datasets, namely GeoAAN and GeoDBLP. Describing and understanding migration is of great interest and considerable meaning because international mobility among researchers not only benefits the individual development of scientists, but also creates opportunities for

[3]Rendered with WebGL Globe (see `www.chromeexperiments.com/globe`).

[4]The recent economic pressure mounting on research communities in Singapore and around the world is not well captured in our data, which lasts to roughly 2011 only.

intellectually productive encounters, enriching science in its entirety, preparing it for the global scientific challenges lying ahead. Moreover, mobile scientists act as ambassadors for their home country and, after their return, also for their former host country, giving mobility a cultural-political dimension. So far, no statistical regularities have been established for the researcher migration within computer science at global scale. One explanation might be that no transnational, registered dataset existed before, and is enabled through our data harvesting approach presented in Chapter 5. We analyzed migration behavior from different points of view and presented models for the general migration propensity and the k-th propensity among others. Additionally, GeoDBLP revealed that there are no cultural boundaries underlying these timing events. The patterns remain similar no matter what region one looks at. Thus, moving on to a new position is a common pattern in terms of timing across different countries and independent of geography. Indeed, people have had the suspicion of many of these regularities but we have shown that they go beyond folklore.

The fact that we find in both datasets distributions from the same families for the general propensity and the k-th propensity shows again that inferring missing information based on the methodology presented in Chapter 5 is a valid approach. The observed difference in parameters, i.e., different means in the general migration propensity, must not necessarily result from the multi-source data harvesting process but can also indicate that different sub-communities of computer science have different cultures and incentives to make the next move.

Naturally, one future goal of this analysis is the translation of the patterns and regularities into recommendable actions, in order to help decision makers. For example, our findings indicate that the average time to hop is longer than most projects and many PhD scholarships last. Therefore, funding agencies should reconsider and review the duration of many projects. This also touches upon the question of the time required to finish a PhD which is often discussed controversially. Estimates of this statistic can be read from our datasets as well and it would be interesting to compare our data to the dataset presented by Espenshade and Rodriguez [55]. The work compares the performance of US and foreign students by looking at the time required to complete a PhD for a small number of US universities.

To further improve the quality of our database, and hence increase the reliability of our findings, several steps can be taken. Improving the construction of migration sketches and operating on the affiliation level will certainly enable a finer level of analysis. However, in the case of the affiliations, this will significantly increase the computational costs and requires a high-quality entity resolution of the affiliation strings. Creating a set of ground-truth resumés for a small number of researchers can be of great benefit as well. These resumés cannot only be used for finding adequate weights in Label Propagation but also to further measure the quality of migration sketches. However, this process is very time consuming and requires a standard definition of a resumé. Additionally, one has to figure out a sample process over all researchers to obtain a dataset which covers all fields of computer science well.

Indeed, we have only started to look into migration through the “AI and the Web” lens, but we have demonstrated that AI can significantly broaden the scope of computational sociology since it is able to infer missing values. This is an important advancement in understanding and describing the migration behavior of computer scientists and should be extended beyond this particular field. In the future, other AI and Machine Learning techniques should be explored to reveal more migration patterns, and our results should be extended beyond computer science. This should also include an analysis over time and the distribution over distances traveled when migrating. In any case, our results are an encouraging sign that harvesting and inferring data from the Web at large-scale may give fresh impetus to demographic research.

6.2 Other Social and Behavioral Phenomena

Of course, migration data is by far not the only human activity we can observe online. The Web has become a part of everyday life for many people and is one of the main sources for media content nowadays. Besides text documents, images, and videos are consumed frequently. Services such as *reddit* and *YouTube* are amongst the top 25 websites visited each month worldwide and even higher ranked in several countries[5]. Therefore, these websites provide an interesting playground for observing human behavior online and analyzing the spread of content among people and the Web.

6.2.1 Internet Memes

In [14], we analyzed time series of search queries related to Internet memes, i.e., catch phrases, humorous pictures, or video clips that are distributed via websites such as *reddit* and then spread among Internet users. We used *Google Trends* to investigate the dynamics of growth and decline of such memes by looking at the search frequencies of such Internet phenomena over time in the Google search engine. Internet memes are of great interest to many people because their rapid growth makes the underlying process appealing to understand for all kinds of purposes. Describing and characterizing successful Internet memes may help public relation professionals to design viral marketing or political campaigns. One specific property of many Internet memes is the initial explosive growth. However, once the peak has been reached, memes start to quickly fade out again but not as rapidly as their initial growth. Interestingly, this typical growth and decline can be nicely captured by mathematical models. In particular, we showed that the Weibull, the Gompertz, and the Frechet distribution can capture this behavior well and also explain the dynamics in terms of simple growth equations. Although Internet memes have previously been described by log-normal distributions [13], it has been empirically shown that the aforementioned distributions return a better fit to the data accompanied by an plausible explanation of the model.

However, most remarkable is the fact that the Weibull, the Gompertz, and the Frechet distributions share mathematical characteristics that describe growth and decline in terms of growth equations. Growth equations model how entities grow and decline over time and we will now exemplify this behavior for the case of the Frechet distribution since we found it to fit best for the majority of the memes under consideration. The probability density function of the Frechet distribution is defined as follows:

$$f(t) = \frac{\alpha}{\beta}\left(\frac{t}{\beta}\right)^{-\alpha-1} e^{-(t/\beta)^{-\alpha}} \overset{\clubsuit}{=} b\alpha t^{-(\alpha+1)} e^{-bt^{-\alpha}} ,$$

with shape and scale parameters α and β. Step ($\clubsuit$) follows from setting $b = (1/\beta)^{-\alpha}$. We can now rewrite this equation as follows:

$$f(t) = \frac{F(t)}{b\alpha t^{\alpha+1}} , \tag{6.1}$$

with $F(t)$ being the cumulative distribution function of the Frechet distribution. In this context, the cumulative distribution function has a particular appealing interpretation. It can be viewed as the sum of total attention attracted so far by the meme. For more details of the derivation

[5]Statistics obtained from Alexa (`www.alexa.com`) in March 2015

and a proof, we refer to Lemma 3 in [14]. Equation (6.1) can be viewed as a growth equation based on division. $f(t)$ will grow as long as $F(t)$ grows faster than $b\alpha t^{\alpha+1}$ and the decline term is polynomial in t. By rewriting the original distribution, we have decoupled growth and decline, and can better understand the underlying process. This is another advantage over the log-normal distribution when fitting time series because the log-normal does not allow an expression as growth equation. In a similar fashion, the Weibull and the Gompertz distribution can be reformulated as well.

We used maximum likelihood estimation to fit more than 200 Internet memes from the period January 2004 to February 2014 to the Weibull, the Gompertz, and the Frechet distributions. This statistical model fitting revealed that those distributions give better fits than the log-normal distribution in terms of the KL-divergence. Additionally, we observed that the Frechet distribution gave the best fit for the majority of the memes. This empirical evaluation indicates that Internet memes may indeed be described by simple and interpretable growth equations that explain their rise and fall. Using *Google Trends* as a proxy to observe popularity is another example where the Web provides information about social behavior.

6.2.2 Viral Videos

While Internet memes in general can be of several different kinds of content, we will now have a look at one particular class of content on the Web. With the increase in bandwidth of broadband Internet connections, video platforms such as *YouTube* have become enormously popular. Many short video clips also show "viral" behavior and their spread among Internet users has rarely been analyzed nor understood. These videos can be of different kinds such as professional music videos but also video clips taken by digital cameras or smartphones. We have above already discussed how Internet memes can be described by mathematical models, in [15] we focused on the spread of videos from the point of view of mathematical epidemiology.

Mathematical models of epidemic processes play a crucial role in various disciplines and we adopt its usage to time series of Google search frequencies and additionally *YouTube* view counts. The process of an epidemic outbreak can be used to describe the information spread in a network where an initial information is propagated along the edges of the network. One can distinguish users in a network being *susceptible (S)* to this effect, others being already *infected (I)*, and lastly infected users may either stay infected or *recover (R)*, i.e., become *susceptible* again.

Typically, these epidemic dynamics are modeled using *compartment models* which assume the entire user base being divided in three groups as described in the previous paragraph and therefore are often referred to as *SIR models*. Transitions between each of the groups are specified in terms of systems of coupled, non-linear differential equations. However, this does not allow for a closed-form solution of the temporal behavior at a certain point in time. Instead, one has to resort to numerical methods, such as Euler's method, to obtain the distribution iteratively. In order to avoid such kind of difficulties, we introduced a simpler probabilistic model based on Markov processes.

More precisely, we consider a homogeneous, time discrete Markov chain as depicted in Figure 6.10. This conceptually simple Markov process is able to explain a wide range of infection dynamics and we will see that this includes the spread of video clips on the Web.

$$\begin{tikzcd} S \arrow[loop above, "1-i"] \arrow[r, "i"] & I \arrow[loop above, "1-r"] \arrow[r, "r"] & R \arrow[loop above, "1"] \end{tikzcd}$$

Figure 6.10: Markov process for SIR models. The parameter i is the infection rate and r amounts to the recovery rate.

Based on this Markov chain, we can now characterize the percentage of infected users $I(t)$ in terms of a probability density functions as follows:

$$f(t) = \int_0^t \alpha e^{-\alpha x} \beta e^{-\beta(t-x)} dx \ .$$

Most notable about this distribution is the fact that it is the convolution of two exponential distributions and has a closed form solution. The rate parameters of the two exponentials are given by $\alpha = -\ln(1-i)$ and $\beta = -\ln(1-r)$ and therefore lend themselves directly to the SIR model. This admits an intuitive and plausible interpretation of the results. For full details and a proof of the closed form solution, we refer to Proposition 1 in [15]. We can now use maximum likelihood estimation again to determine the parameters of the model based on the observed time series.

We considered two datasets in the empirical analysis of our proposed Markov SIR model. The first one, presented by Jiang et al. [104], consists of more than 700 *YouTube* videos with at least 500,000 views while the second one presents *Google Trends* time series for 50 search queries related to videos known to exhibit viral behavior. For a baseline comparison, we also fitted the data to the log-normal and the Weibull distributions akin to the experiments in Section 6.1 on migration and the analysis of the Internet memes in Section 6.2.1.

The experimental results revealed that Markov SIR models provide a reasonably accurate description of the data and are the overall best fitting model. Although the log-normal and Weibull distributions also fit the data well in many cases, they lack a concrete explanation of the behavior in terms of an epidemic process. Hence, the experiments suggest that it is reasonable to describe the attention dynamics of viral videos with help of Markov SIR models.

6.3 Interim Conclusion

In this chapter we have seen applications of Machine Learning applied to data describing human behavior. This included the migration behavior of researchers in computer science and the viral spreading of Internet memes which in particular covered the analysis of viral videos. The data harvesting process was utilizing various information sources, such as online bibliographies, search frequencies in search engines, and view counts on video platforms among others. In the case of the migration data, we have shown that Machine Learning cannot only be used for the data analysis, as it is commonly done, but also to fill in the missing datapoints. In all scenarios we have seen that probability distributions with only one or two parameters a capable of describing complex real-world interactions. This is in contrast to many other approaches that use various parameters and hence can easily overfit. Additionally, we were able to present plausible explanations for our findings which makes interpretation a lot more accessible.

We already discussed the fact that the data harvesting process may be noisy. Of course, the way we currently construct our migration sketches further approximates the true vita of a researcher. Replacing these sketches by more expressive resumés is beneficial for all kinds of purposes. We expect not only to obtain more precise results but can also present and visualize the results more concretely. However, this requires a precise formal definition of a resumé which has to account for several outliers and abnormalities. Once such a standard resumé has been defined, one can also construct a "gold" corpus of selected resumés that can be used for further evaluation.

The interest in using Machine Learning on large-scale human behavior data has been increasing recently and can have far-reaching impact on other fields such as sociology and psychology. By extending our research on virality beyond Internet memes and video clips, as it was already partially highlighted for technology trends in [14], one can further contribute to the field of computational sociology. Fads have frequently been observed in the "old" economy and can be expected to be found in virtual economies as well. Hence, providing researchers, politicians, and marketers new tools. Similarly, GeoDBLP can only be seen as the first step in Machine Learning-driven migration analysis based on online bibliographies. Transferring this approach to related database such as PubMed[6] will extend this study to other areas and enable a comparison not only across countries but also across different fields of research.

The analysis of geo-tagged online bibliographies showed regularities for the entire research market and migration in computer science was seen as a whole. However, there is also an interest in the behavior of the individual researchers. For example, it would be interesting to learn a model that predicts the number years required for a student to finish their PhD. Or, to predict the time required for an entire group of PhD students at a specific location conditioned on their field of research or their supervisor. In the univariate case, Cohen et al. [40] have already looked at the prediction of the number of migrants moving between countries. However, eventually we are interested in considering multivariate cases where we can make predictions for a group of people influencing each other. When using PGMs for this task, we require random variables defined over the natural numbers to model the counts. These kind of PGMs did not receive much attention so far and therefore we will present a new approach for PGMs over count data in the next chapter.

[6]`www.ncbi.nlm.nih.gov/pubmed`

CHAPTER 7

Count Graphical Models

So far, we have seen inference and learning approaches for primarily binary models, as well as models with variables over discrete finite domains. The last chapter exemplified how we can use Label Propagation to handle random fields with variables over large label spaces. Although the range of each variable could take on several thousand values easily, it was still finite. Hence, if we were to predict the number of publications of an author in the next year or even subsequent years for example, or the productivity of an entire university or country measured in publications, we were required to artificially limit and bin the true range. This is just one particular example where the models described so far reach their limits. We have seen in Section 1.1 already another example where we highlighted the need for PGMs over count data. The case of traffic flow prediction naturally fits well into the domain of natural numbers and previous work of similar data approximated the congestion by discretization into different levels of traffic [178]. In fact, the world contains an unimaginably vast amount of information, and much of the data consists of counts, i.e., observations that can take only non-negative integer values. Without counting, we cannot know how many people are born and have died; how many men and women still live in poverty; how many children need education, and how many teachers to train or schools to build; the prevalence and incidence of diseases; how many customers arrive in a shop daily and whether demands for products are increasing, nor which team wins a sports match. Our scientific and digital lives also thrive on counts. Example data are publication and citation counts as seen in the previous chapter, bag-of-X representations of, e.g., collections of images or text documents, genomic sequencing data, user-ratings data, spatial incidence data, climate studies, and site visits, among others. Behavioral data of users visiting websites for example are tracked on a large scale. Visits or log-ins are counted and then used to enrich the user experience and to increase revenue. We have also seen in Chapter 6 how such kind of data can be used to discover interesting models for Internet memes and the spread of video clips online. Or consider computational social science, a field that leverages the capacity to collect and analyze data at a scale that may reveal patterns of individual and group behaviors. Here, for instance, the number of people living in a region and the number of people migrating from one city to another one, among others, are of great interest to politicians and also influence decisions in education and research. With the GeoDBLP corpus presented in the previous chapter, we have already seen a dataset that is suitable for such a discussion. Finally, counts also play an essential role in statistics. Consider for example an election analyst interested in the association of gender and voting intentions. Standing on a street corner for an hour recording data from anyone willing to talk to them, they build a contingency table, say, with the gender in the rows and and the voting intentions in the columns. In contrast to interviewing a fixed number of m individuals, there are no constraints on the row and column totals, and hence the cell entries follow a count distribution.

All these examples share a common attribute: they require a distribution over counts, i.e., a potentially skewed, discrete distribution over the natural numbers. Indeed, there are several count distributions. Here, we focus on one of the most widely used ones, namely the Poisson distribution. Among other observations, Poisson distributions are observed in sports data [107], such as goals in football matches, in different fields of the natural sciences [56] which includes the number of cells per region under a microscope for example, and Clarke [39] even observed that the points of impact of bombs in flying-bomb attacks are Poisson distributed. However, these classical studies have only considered uni- and bivariate cases even though in many situations count variables may directly influence each other and should not be considered independently. For instance, if “Neural” appears often in a document then it is also likely that “Network” appears in the same document. Indeed, one may consider to employ a probabilistic graphical model widely used for modeling distributions among several random variables. Unfortunately, research has so far mainly focused on graphical models over binary, multinomial and Gaussian random variables. These standard distributions, however, are often ill-suited for modeling count data since they either disregard the infinite range over the natural numbers or the potentially asymmetric shape of the distribution of count variables. Counts are neither binary nor continuous, they are discrete with a typically right skewed distribution over an infinite range. For example, if the mean λ of a Poisson distribution is less than five, it can be shown that its probability histogram is markedly asymmetrical, making a Gaussian-approximation inappropriate. However, if $\lambda > 1000$ its probability histogram is essentially symmetric and bell-shaped. Besides the shape, the Gaussian assumption of fixed variance is often violated for counts. The variance is typically not constant but increases with higher values. Here, the Poisson distribution accounts for this immediately because the mean correlates with the variance. Therefore, it is not surprising that extensions of the Poisson distribution to the multivariate setting have been proposed, see e.g. [19, 248, 249]. However, all existing extensions either model only negative conditional dependencies or neglect the prediction of counts, or do not scale well. Negative dependencies limit the expressiveness of the models because one can only represent that the mean of a variable decreases upon the observation of increasing neighboring counts.

To ease the modeling of multivariate count data, we therefore propose a novel family of Poisson graphical models, called *Poisson Dependency Networks (PDNs).* A PDN consists of a set of local conditional Poisson distributions — each representing the probability of a single count variable given the others — that naturally facilities a Gibbs sampling inference procedure. Moreover, the family admits simple training procedures induced by a functional gradient view on training. Specifically, triggered by the recent successes of functional gradient ascent for representing multinomial Dependency Networks [160], our first technical contribution is to show that PDNs can be represented as sums of regression models grown in a stage-wise optimization starting from a simple initial model. In fact, this first functional gradient approach to modeling multivariate Poisson distributions — as we will show empirically on several real world datasets — scales well and can model positive and negative dependencies among count variables while often outperforming state-of-the-art approaches. However, the main technical contribution of this chapter is the development of the first multiplicative functional gradient ascent, which is demonstrated empirically to be able to boost the performance even further, since it essentially implements an automated step size selection without incurring in computational overhead.

This chapter is structured as follows. We start off by discussing related work in more detail and give an first empirical result which indicates the strength of PDNs. We then formally introduce PDNs in Section 7.2 and describe different variants. Section 7.3 shows how to learn PDNs, in particular, using the multiplicative functional gradient approach, and in Section 7.4 we explain how to perform inference in PDNs. Section 7.5 presents our exhaustive experimental evaluation on different synthetic and real-world datasets demonstrating the effectiveness of this model family. Before summing up this chapter, Section 7.6 explains how relational PDNs can be implemented given a relational count dataset.

7.1 Related Work

Most of the existing AI and Machine Learning literature on graphical models is dedicated to binary, multinomial, or certain classes of continuous (e.g. Gaussian) random variables. So far, this thesis has also mainly considered binary or finite multinomial random variables. In Chapter 3, the focus was on binary models and Chapter 5 looked at multiclass labeling problems which also have an interpretation in terms of Gaussian MRFs. With few exceptions in Section 3.2, we assumed undirected graphical models so far. However, recall that Section 2.1 also briefly discussed *Dependency Networks (DNs)*, which are the focus of this chapter. We can summarize the papers that are most related to the work in this chapter as follows:

- Heckerman et al. [90]: This work leverages the advantages of directed and undirected graphical models and introduces Dependency Networks. We build upon this work and use Dependency Networks in the infinite Poisson case.
- Friedman [58]: This seminal work lays the foundation of gradient boosted approaches where functions are approximated via the sum of simpler components. We follow upon that work and use gradient tree boosting to lean our local models.
- Yang et al. [249]: Compared to early work on Poisson graphical models, such as Besag [19], this work introduces a modified Poisson graphical which is capable of modeling positive and negative dependencies without giving up consistency. Still, PDNs have less restrictions on the parameters and have lower run times.
- Allen and Liu [7]: This work presents a local Poisson model akin to our PDNs. However, these *Local Poisson Graphical Modelss (LPGMs)* solely use generalized linear models to define the mean of the local Poisson distributions.

DNs due to Heckerman et al. combine concepts from the directed and the undirected worlds. Specifically, like Bayesian networks, DNs have directed arcs but they allow for networks with cycles and bi-directional arcs, akin to MRFs. This makes DNs quite appealing because, if the data is fully observed, learning is done locally on the level of the conditional probability distributions for each variable. Hence, the learning is mixing directed and indirected as needed. Based on these local distributions, samples from the joint distribution are obtained via Gibbs sampling [90, 17]. Except for few cases that we will discuss below, surprisingly little attention has been paid to PGMs for multivariate count data within the AI and Machine Learning communities.

Indeed, when only one count variable is considered, the literature is vast. At lot of this can essentially be treated as a *Generalized Linear Model (GLM)*, see [143] for example. In GLMs, the response variable, i.e., the variable to predict, is related to a link function applied to a linear model. One specific instantiation of GLMs is the Poisson regression case where the link function is the logarithm. In this case, the mean of the Poisson distribution is defined by a log-linear model. Compared to ordinary least squares regression, an advantage of the GLM framework is the fact that non-Gaussian error structures are possible. However, when considering jointly two or more count variables, things become more complicated and there exists much less work.

For instance, one can define a multivariate Poisson distribution by modeling random variables as sums of independent Poisson variables, see e.g. [107, 68]. Since this is again Poisson, the marginals are Poisson as well. The resulting joint distribution, however, can only model positive correlations. There are also bivariate extension for specific models, e.g. [107]. In general, even calculating probabilities for these multivariate Poisson distributions is computationally challenging and hence their usage is often limited [224]. Hoff [93] proposed to use *Generalized Linear Mixed Models (GLMMs)* using Poisson regression and modeling the dependencies between variables using random effects. Training the resulting GLMMs, however, is computationally demanding because it requires the estimation of the unobserved mixed effects. Nevertheless, GLMMs are often used, for example in studies on ecology and evolution [23]. Using just GLMs, Yang et al. [248] have recently proposed an undirected Poisson model, called *GLM graphical model*, where — close in spirit to PDNs — each conditional node distribution is assumed to be from the exponential family with the Poisson distribution as one particular instantiation. In contrast to the functional gradient approach of PDNs, where interactions among variables are introduced only as needed and hence the learner does not explicitly consider the potentially immense parameter space, Yang et al. employ a sparsity constrained, parametrized conditional maximum likelihood estimation approach along the lines of [145]. It must be mentioned that Yang et al.'s GLM graphical models extend the seminal work by Besag [19]. More precisely, Besag's *Auto-Poisson model* can be seen as an instantiation of GLM graphical models, however, Yang et al. allow for higher-order cliques. In general, this line of work models the dependencies between the variables directly, instead of adding mixed effects. That is, the mean of each variable follows a GLM where all neighboring variables are used as explanatory variables. However, to guarantee a consistent joint probability distribution, the parameters are required to be negative, i.e., competitive relationships. In the case of arbitrary positive dependences, the joint distribution is not guaranteed to be normalizable anymore because the normalization constant becomes infinite. In contrast, PDNs drop the guarantee of consistency and stay local, allowing for negative and positive parameters, i.e., competitive and attractive relationships. Alternatively, Kaiser and Cressie [106] suggested the use of Winsorized Poisson distributions to remove the drawback of negative dependencies only. The Winsorized Poisson distribution is the univariate distribution obtained by truncating the Poisson variables at a finite constant. Doing so makes estimation considerably harder than for PDNs that are naturally equipped with a simple learning approach as we will demonstrate.

Probably closest in spirit to PDNs are the recent approaches due to Allen and Liu [7] and due to Yang et al. [249]. More precisely, Allen and Liu's LPGMs are also in the tradition of Besag's Auto-Poisson models and related to GLM graphical models. They assume that each variable conditioned on all other variables in the network follows a Poisson distribution. Similar

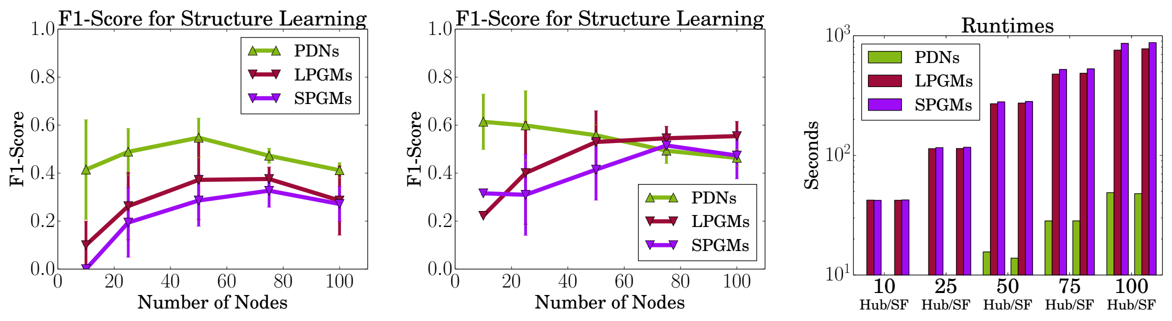
	Auto-Poisson [19]	TPGMs [249]	SPGMs [249]	LPGMs [7]	PDN
consistent joint distribution	✓	✓	✓	(✓)	(✓)
arbitrary parameters	-	✓	✓	✓	✓
unbounded variable range	✓	-	-	✓	✓
covers learning	✓	✓	✓	✓	✓
higher-order potentials	-	✓	✓	✓	✓
tree-structured potentials	-	-	-	-	✓
functional gradient	-	-	-	-	✓
investigates inference	-	-	-	-	✓

Table 7.1: A comparison of existing Poisson graphical models most similar to PDNs. As one can see, PDNs cover all important aspects of modeling count data while facing almost no restrictions. The "(✓)" denotes that this is the case given that an unordered Gibbs sampler on the model converges. See main text for more details.

to PDNs, they do not focus on a consistent joint distribution but consider local conditional probability models only. In contrast to PDNs, which employ a well scaling functional gradient ascent, the structure of LPGMs is learned with a computationally more demanding ℓ_1-norm penalized log-linear regression for neighborhood selection. Assuming the data is fully observed, the neighborhood for every node is learned separately in the spirit of Gaussian MRFs. Yang et al. [249] introduced *Truncated Poisson Graphical Models (TPGMs)*, *Quadratic Poisson Graphical Model (QPGMs)*, and *Sub-linear Poisson Graphical Models (SPGMs)*. TPGMs are a modification of Kaiser and Cressie's truncated models. QPGMs are a generalization of the previously introduced Poisson graphical models with quadratic base measure. QPGMs allow both positive and negative parameters under the restriction that the pairwise parameter matrix is negative-definite. Instead of changing the base measure, SPGMs use sub-linear sufficient statistics to ensure normalizability. All this makes learning harder compared to PDNs since parametric ℓ_1-norm penalized log-linear regression or proximal gradient ascent are used to find a single best model. Instead, PDNs are triggered by the intuition that finding many rough rules of thumb of how count variables interact can be a lot easier than finding a single, highly accurate Poisson model.

We have summarized the properties of the existing Poisson graphical models that are most similar to our work and how they compare to PDNs in Table 7.1. Note that the table lists the TPGMs from [249] instead of the Winsorized Poisson model from [106] because TPGMs provide a consistent joint distribution and present the state-of-the-art approach for truncated Poisson graphical models. The "(✓)" denotes that this feature does not generally hold. But as already mentioned, Bengio et al.'s recent result [17, Proposition 2] suggests the existence of a consistent distribution. More precisely, Bengio et al. showed that an unordered Gibbs sampler over a DN equipped with evidence induces a so-called *Generative Stochastic Network (GSN)* Markov chain. As long as this GSN Markov chain has a stationary distribution, i.e., the Gibbs sampler converges, the DN defines a joint distribution. However, this distribution does not have to be known in closed form. Now, since PDNs (and hence LPGMs) are DNs (with local Poisson distributions), this argument should carry over, but a formal analysis has not been presented yet.

Our first empirical results support this as illustrated in Figure 7.1. It summarizes the results of using the most recent approaches from Table 7.1 as well as PDNs for a network discovery task. More precisely, following Yang et al. [249], two network families were considered that are



(a) Scale-free networks (b) Hub networks (c) Run times in log-scale

Figure 7.1: Comparison of different count model approaches on synthetic data. Local models (LPGMs & PDNs) outperform global models (SPGMs). Moreover, PDNs can be an order of magnitude faster.

commonly used throughout genomics: hub and scale-free graph structures. As one can see, the local approaches, i.e., LPGMs (red) and PDNs (green), have competitive performances compared to the guaranteed consistent SPGMs (purple)[1]. The plots show the F1-scores (see Equation (2.14)) for structure recovery for varying problem sizes averaged over five runs. More details on the experimental setting can be found in Section 7.5.1. Moreover, the training of PDNs is considerably faster compared to the state-of-the-art approaches, often an order of magnitude (Figure 7.1c).

The speedup is likely due to the non-parametric nature of PDNs. In particular, we train PDNs using functional gradients. That is, we train them in a stage-wise manner at low and scalable costs following Friedman's Gradient Tree Boosting [58]. This boosting approach has been proven successful in a number of cases, see e.g. [189, 112, 50, 54, 160, 244, 164, 163], and since it estimates the parameters and the structure jointly, it is generally related to structure learning of graphical models. It is particularly akin to those approaches that use the local neighborhood of each variable to construct the entire graph. For example, covariance selection is used in Gaussian models where edges are added to the graph until a stopping criterion is met. Despite being greedy, the method is not practical for multivariate distributions with a large number of variables. A popular alternative is neighborhood selection via Lasso [145]. This has also been used for learning the structure of binary Ising models, see e.g. [185]. In the case of DNs, Heckerman et al. [90] originally did neighborhood selection implicitly by learning probabilistic decision trees for each variable. Also, undirected probabilistic relational models, such as MLNs, were learned with the help of (boosted) decision trees in [119, 140, 162].

7.2 Poisson Dependency Networks

The review of related work revealed that modeling multivariate Poisson distributions faces the following main challenges:

Unrestricted parameters: Some approaches only handle negative dependencies which is a serious limitation because positive dependencies are often present as well.

[1]We did not compare to TPGMs since they were compared to SPGMs already in [249].

Joint distribution: Given local Poisson probability distributions, formulate a joint probability distribution.

Learning: Recovery of the dependencies in the present data, so that an easy interpretation of the data is possible. A learned model should also quantify the dependencies properly to enable accurate inference on unlabeled instances.

Inference: Given a partially observed case, the goal of MAP inference is to predict the most likely assignment of the unobserved variables. Other probabilistic queries, such as marginal probabilities, are of great interest as well.

With this in mind, we now develop *Poisson Dependency Networks (PDNs)* that are different from existing Poisson graphical models in several ways. Considering Poisson distributions instead of the usual binary, multinomial, or Gaussian distributions in combination with our proposed learning algorithm is clearly “non-standard” and contributes in the following manner as previously submitted in [87]:

(C7.1) Positive and negative dependencies: PDNs are capable of modeling positive and negative dependencies. Furthermore, we do not have to restrict the weights to be symmetric or limited in any such way.

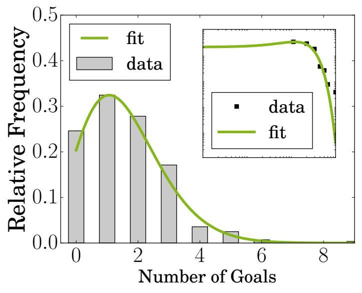
(C7.2) Non-parametric: PDNs are the first non-parametric Poisson graphical models, both at the level of local models, as well as at the global level of the overall model. The use of (Poisson) regression trees for the (initial) conditional distributions are more flexible than using parametric log-linear models for the mean in the Poisson distribution. Moreover, finding many rough rules of thumb of how count variables interact can potentially be a lot easier than finding a single, highly accurate local Poisson model.

(C7.3) Multiplicative GTB: We present the very first multiplicative functional gradient ascent. This implements an automated step size selection and hence avoids an expensive line search while improving convergence.

(C7.4) Flexible structure learning: The learning procedure estimates the structure and parameters simultaneously while scaling well. PDNs do not require fixing the missing symmetries in the structure to run inference. The structure learning of PDNs is competitive with consistent models, has much lower computational costs, and is readily parallelizable.

(C7.5) Predictions: PDNs naturally facilitate a simple Gibbs sampling inference. More precisely, we use a Gibbs sampler that provides us with samples from a joint distribution. These samples allow us to predict counts of unobserved instances easily instead of solely focusing on structure recovery as done for existing Poisson graphical models.

We will now introduce *Poisson Dependency Networks (PDNs)* more formally. Given a set of random variables $\mathbf{X} = (X_1, \ldots, X_n)$ where each variable is defined over the natural numbers, including 0, then a PDN is a pair (G, P). Here, $G = (V, E)$ is a directed, possibly cyclic, graph with V being a set of nodes where each node corresponds to a random variable in $\mathbf{X}$. As in the case of MRFs and BNs, we can use nodes in G and the random variables in $\mathbf{X}$ interchangeably. $E \subseteq V \times V$ is a set of directed edges where each edge models a dependency between variables, i.e., if there is no edge between two variables X_i and X_j, the variables are conditionally independent given the other variables $\mathbf{X}_{\setminus i,j}$ in the network. Here, $\mathbf{X}_{\setminus i,j}$ is



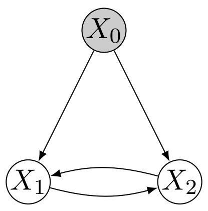


Figure 7.2: Illustration of a Poisson Dependency Network. (left) Plot of a fitted Poisson distribution. The number of goals scored in football matches is Poisson distributed. The plot shows the distribution of goals scored by the home team in the 2012/13 season of the German Bundesliga. Here, $\lambda = 1.59$, stating that the home team scored on average 1.59 goals per game. (right) Example PDN. The conditional distribution of each variable given its parents is Poisson distributed. Observed variables are denoted by gray shades. Similar to a Bayesian network, a PDN is directed, however, it may also contain cycles.

shorthand for $\mathbf{X} \setminus \{X_i, X_j\}$. As before, we refer to the nodes that have an edge pointing to X_i as the parents of X_i, denoted as pa_i. P is a set of conditional probability distributions for every variable in $\mathbf{X}$. For now, we assume that each variable X_i, given its parents pa_i, is Poisson distributed, i.e.,

$$p(x_i|\,\mathrm{pa}_i) = p(x_i|\mathbf{x}_{\setminus i}) = \frac{\lambda_i(\mathbf{x}_{\setminus i})^{x_i}}{x_i!} e^{-\lambda_i(\mathbf{x}_{\setminus i})} \,. \tag{7.1}$$

Here, $\lambda_i(\mathbf{x}_{\setminus i})$ highlights the fact that the mean can have a functional form that is dependent on X_i's neighbors where $\mathbf{x}_{\setminus i}$ depicts the set $\mathbf{x} \setminus \{x_i\}$. Often, we will refer to the mean only as λ_i. An example of such a local Poisson conditional probability distribution is illustrated in Figure 7.2(left). The construction of the local conditional probability distribution is similar to the (multinomial) BN case, however, in the case of PDNs, the graph is not necessarily acyclic and $p(x_i|\mathbf{x}_{\setminus i})$ has an infinite range and hence cannot be represented using a finite table of probability values. Finally, the full joint distribution is simply defined as the product of local distributions:

$$p(\mathbf{x}) = \prod\nolimits_{x_i \in \mathbf{x}} p(x_i|\mathbf{x}_{\setminus i}) = \prod\nolimits_{x_i \in \mathbf{x}} \frac{\lambda_i^{x_i}}{x_i!} e^{-\lambda_i} \,. \tag{7.2}$$

However, this only defines a consistent joint distribution under certain restrictions. As it is discussed for DNs in [90], there exists a one-to-one mapping between consistent DNs and MRFs, resulting in this factorization of the PDN. An example of a PDN with three variables is depicted in Figure 7.2(right). The gray shaded variable X_0 can be seen as an observational vector. In the spirit of other graphical models, for example Conditional Random Fields [130], we can define a set of observed variables in advance. It will not be necessary to learn a local model for these variables as they are always observed. Additionally, we have fewer constraints on these variables. They do not have to be count variables but instead can be arbitrary features. This allows us to easily incorporate features that are Gauss distributed for example, hence, paving the way for *hybrid models* with local models of all kind of local distributions. We call this new formalism *Poisson Dependency Networks* because they are inspired by Heckerman et al.'s Dependency Networks.

```
input  : A training dataset $[\mathbf{x}^{(j)} \in \mathbb{N}^n]_{j=1,\ldots,m}$
output : A set of optimal parameters $P$
score ← $-\inf$;
for $\alpha_i$ in HyperparameterSpace do
    parameterScore ← 0;
    for train, test in crossValidationSplit(data) do
        $P \leftarrow [\,]$;
        for $X_i$ in $\mathbf{X}$ do
            $P_i \leftarrow$ learnInitialCondProb($\alpha_i, X_i \sim \sum \mathbf{X}_{\setminus i}$);
            for $t$ in $[1, T]$ do
                gradientBoosting($P_i$);
            end
        end
        parameterScore ← parameterScore + ScoreFunction(test, $P$);
    end
    if parameterScore > score then
        optimalPDN ← $P$;
        score ← parameterScore;
    end
end
return optimalPDN;
```

Algorithm 9: Learning Algorithm for Poisson Dependency Networks

One crucial part of PDNs that we have not touched upon yet, is the encoding of $p(X_i|\mathbf{X}_{\setminus i})$ and of λ_i in particular. There are two sensible ways. First, one can follow a parametric approach as it has been done before in the literature. Second, as an alternative, we propose a non-parametric encoding. More precisely, we show that a PDN can be represented as sums, respectively products, of regression models grown in a stage-wise optimization starting from an initial constant value or a Poisson regression tree. Before doing so, let us touch upon the issue of consistency again. Our non-parametric approach to model the means can essentially be seen as linear functions with no restrictions on the parameters. In turn, we will generally not be able to formulate the underlying joint probability distribution of the PDN in closed-form by means of a standard (pairwise) Poisson graphical model. However, as we will show later, we can use sampling techniques to draw samples from the joint distribution because we have access to the conditional probabilities. If this sampling converges, there exists a consistent distribution, which however does not have to be known in closed form. We will discuss the question of consistency during this chapter in greater detail.

7.3 Learning Poisson Dependency Networks

Learning PDNs in the fully observed case amounts to determining the conditional probability distributions from a given set of m training instances over n count variables:

$$[\mathbf{x}^{(j)} \in \mathbb{N}^n]_{j=1,\ldots,m} \; .$$

For simplicity, we now ignore additional observed features that do not follow count distributions. Since we do not learn models for these features anyhow, this does not immediately affect the learning procedure. In turn, learning is equal to finding λ_i for each variable X_i because the Poisson distribution is completely determined by the mean. However, λ_i will possibly depend on all other variables in the network, and these dependencies define the structure of the network underlying the PDN. We will now develop a functional gradient ascent approach to learning PDNs, i.e., learning the λ_i. We will do so by starting from parametrized maximum likelihood estimation, extending it to additive local Poisson models grown in a stage-wise manner, and finally turning this into a multiplicative update.

Before going into details, let us summarize the resulting high-level approach in Algorithm 9, since it covers all the approaches. The mean λ_i of each variable is assumed to consist of a set of local models grown in a stage-wise manner. More precisely, initially the set is a singleton learned in Line 7. Here, the formula $X_i \sim \sum \mathbf{X}_{\setminus i}$ denotes that the mean for X_i can potentially have all other variables as features. Please note that the used models may include hyper-parameters, such as step sizes and pruning parameters in the case of regression trees which might be estimated using a grid search and a validation set. We will touch upon these parameters for each approach separately. After the initial learning, a pre-defined number of gradient steps (T) are made to further improve the model (Line 9). Depending on how we encode the initial models, as well as on how we grow the local models per variables, different learning approaches are realized which we will now discuss.

7.3.1 Poisson Log-Linear Models

To grasp the idea of functional gradient ascent for training PDNs, it is helpful to first describe how to learn a single parametric Poisson model. Most commonly, as mentioned in the related work section, the mean is modeled using a log-linear model. In its most naive form, without further regularization, this presents a fully-connected PDN. More precisely, using a log-linear model, the mean of each variable is:

$$\mathbb{E}[X_i] = \lambda_i = \exp(w_i + \sum_{j \neq i} w_{ij} \cdot x_j) \tag{7.3}$$

In the univariate case, this approach is often referred to as *Poisson Regression.* The parameters w_i and w_{ij} determine the mean and can be learned via maximum likelihood estimation. That is, we assume that the training examples are i.i.d. and we seek for parameters that maximize the conditional log-likelihood

$$\mathrm{cll}(\mathbf{w}) = \ln \prod_{k=1}^{m} p(x_i^{(k)} | \mathbf{x}_{\setminus i}^{(k)}) = \sum_{k=1}^{m} \ln p(x_i^{(k)} | \mathbf{x}_{\setminus i}^{(k)}) \ .$$

So, we obtain the partial derivatives

$$\frac{\partial\, \mathrm{cll}}{\partial w_i} = \frac{\partial}{\partial w_i} \sum\nolimits_{k=1}^{m} x_i^{(k)} \cdot \ln(\lambda_i^{(k)}) - \ln(x_i^{(k)}!) - \lambda_i^{(k)} = \sum\nolimits_{k=1}^{m} x_i^{(k)} - \lambda_i^{(k)}$$

and

$$\frac{\partial\, \mathrm{cll}}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum\nolimits_{k=1}^{m} x_i^{(k)} \cdot \ln(\lambda_i^{(k)}) - \ln(x_i^{(k)}!) - \lambda_i^{(k)} = \sum\nolimits_{k=1}^{m} (x_i^{(k)} - \lambda_i^{(k)}) \cdot x_j^{(k)} \ .$$

Since there is no closed form solution for solving this problem, we typically employ an iterative approach to improve the conditional log-likelihood. For instance, using a simple gradient ascent, we can take a step in the direction of the gradient in each iteration until convergence:

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \eta \cdot \nabla \,\mathrm{cll} \tag{7.4}$$

where ∇ cll denotes the gradient consisting of all partial derivatives. The speed of convergence is influenced by the choice of the step size η which denotes one of the possible hyper-parameters in Algorithm 9. Besides the simple gradient ascent, algorithms such as (L-)BFGS are common for solving such optimization problems. However, in the case of Poisson Regression, iteratively-re-weighted least squares (IRLS) is probably the most popular technique. We refer to [143] for further details on IRLS.

Before moving to functional gradient ascent, let us make a further note on the consistency of PDNs. In general, we may not be consistent, however, using log-linear models, we can fit our PDN with log-linear means into the setting of Besag [19] and its follow up work by restricting the parameters and structure. As Besag showed for the pairwise lattice case, there exists an undirected graphical model with local Poisson conditional probability distributions if the parameters w_{ij} are non-positive. It is further assumed that the pairwise parameters are symmetric, i.e., $w_{ij} = w_{ji}$. This was further generalized for the non-pairwise case in [248]. If our parameters satisfy these conditions, we can specify the joint distribution for our model and hence we do have a consistent Poisson MRF as well.

However, we are interested in models with as few limitations as possible on the structure and the parameters. Hence, we do not rely on the existence of a closed form equation for the joint distribution or even insist on the existence of a consistent distribution. Instead, we introduce a more general class of Poisson graphical models. This is mainly motivated by computational concerns. For domains with a large number of count variables and many dependencies among them, learning a consistent Poisson MRF is often too costly. Moreover, not being able to model both, positive and negative, influences hurts the performance and limits the expressiveness in many cases. Finally, as the results of our first experimental comparison in Figure 7.1 already showed, approaches that are guaranteed to provide an explicit joint distribution are not necessarily better. Specifically, we follow [90, 17] and employ the machinery of Markov chains to generate pseudo samples of our joint distribution. The generated samples are used to answer complex probabilistic count queries. For more details on Gibbs sampling and a justification for its usage we refer to Section 7.4.

7.3.2 Non-Parametric Poisson Models via Gradient Boosting

Local log-linear models tend to estimate fully connected networks and overfit. Consequently, they typically do not provide major insights into the structure of the underlying data generation process. Hence, regularization or post-processing such as thresholding [7, 249] have to be employed to extract the true nature of the network. Moreover, as demonstrated in our experiments below, the PDNs based on log-linear models often overshoot at prediction time, leading to overflows, and in some cases prediction is not possible at all.

Therefore, we propose an alternative: instead of a single local log-linear model for λ_i, we grow a set of local models in stage-wise manner using gradient information (Line 9 of Algorithm 9). This way we avoid considering the large parameter space explicitly and include interactions among count variables only as needed. We here employ regression trees for this boosting approach and the resulting approach is called *Gradient Tree Boosting (GTB).* Using trees has two significant advantages over standard parametric and non-parametric regression approaches:

- By allowing the tree-structure to handle much of the overall model complexity, the models in each leaf can be kept at a low order and are more easily interpreted.
- Interactions among features are directly conveyed by the structure of the regression tree. As a result, interactions can be understood and interpreted more easily in qualitative terms; we will touch upon this later again.

To develop gradient tree boosting, recall that the parameters of a parametrized log-linear model can be viewed as a sum of gradients (cf. Equation (7.4)). Gradient boosting lifts this intuition to the function space. That is, using the log link function, the mean $\lambda_i(\mathbf{x}_{\setminus i})$ of a Poisson variable X_i is viewed as $\lambda_i(\mathbf{x}_{\setminus i}) = \exp\left(\psi_i(\mathbf{x}_{\setminus i})\right)$, i.e., as a function of some (feature-)function ψ_i. Now, we perform a gradient ascent in (log) function space for some iterations T in order to estimate the feature functions $\psi_i(\mathbf{x}_{\setminus i})$ for the count variables X_i. This corresponds to representing the local functions of a PDN as a weighted sum:

$$\lambda_i^t(\mathbf{x}_{\setminus i}) = \exp\left(\psi_i^t(\mathbf{x}_{\setminus i})\right) = \exp\left(\psi_i^{t-1}(\mathbf{x}_{\setminus i}) + \eta \cdot \nabla_i^t\right) \tag{7.5}$$

The initial $\psi_i^0(\mathbf{x}_{\setminus i})$ can be, e.g., a constant, the logarithm of the empirical mean of X_i, or a more complex model. The ∇_i^t are functional gradients:

$$\nabla_i^t = \mathbb{E}_{\mathbf{x}_{\setminus i}, x_i}\left[\frac{\partial}{\partial \psi_i^{t-1}(\mathbf{x}_{\setminus i})} \ln p(x_i | \mathbf{x}_{\setminus i}; \psi_i^{t-1}(\mathbf{x}_{\setminus i}))\right] , \tag{7.6}$$

indicating intuitively how we would like the mean model to change in order to increase the log-likelihood. The expectation $\mathbb{E}[\cdot]$ in Equation (7.6), however, requires access to the joint distribution, which we do not have.

Fortunately, we can approximate it with the help of the instances in our training dataset, which are sampled from the true joint distribution anyhow:

$$\nabla_i^t \approx \frac{1}{m} \sum\nolimits_j \frac{\partial}{\partial \psi_i^{t-1}(\mathbf{x}_{\setminus i}^{(j)})} \ln p(x_i^{(j)} | \mathbf{x}_{\setminus i}^{(j)}; \psi_i^{t-1}(\mathbf{x}_{\setminus i}^{(j)})) = \frac{1}{m} \sum\nolimits_j \nabla_i^t(x_i^{(j)}, \mathbf{x}_{\setminus i}^{(j)}) \; .$$

Intuitively, as long as we can estimate $\nabla_i^t(x_i^{(j)}, \mathbf{x}_{\setminus i}^{(j)})$, they give us point-wise regression examples

$$(\mathbf{x}_{\setminus i}^{(j)}, \nabla_i^t(x_i^{(j)}, \mathbf{x}_{\setminus i}^{(j)}))$$

of the functional gradient. Hence, it is a sensible idea to train a regression model using them approximating the true functional gradient. Following Dietterich et al. [50], we minimize the following objective

$$\sum\nolimits_j \left[h_i^t(\mathbf{x}_{\setminus i}^{(j)}) - \nabla_i^t(x_i^{(j)}, \mathbf{x}_{\setminus i}^{(j)})\right]^2 \tag{7.7}$$

using a regression tree h_i^t [28]. Here, j iterates over all instances in our training dataset where $\mathbf{x}^{(j)}$ is the jth training example. Although fitting a regression model to Equation (7.7) is not exactly the same as the desired ∇_i^t, it will point into the same direction if we have enough training examples. So, ascent in the direction of the regression model h_i^t — replacing ∇_i^t by h_i^t

in Equation (7.5) — will approximate the true functional gradient ascent Equation (7.5) and was empirically proven to be quite successful for training many probabilistic models.

We will now show how the point-wise gradient regression examples for PDNs look like. For the log link function $\lambda_i = \exp(\psi_i(\mathbf{x}_{\setminus i}))$ with some feature function ψ_i, where we have now omitted the iteration index t for the sake of simplicity, we get:

Theorem 7.1 (see also e.g. [189]). *Under the assumption of a log link function, the point-wise gradients of the log-probability can be generated using*

$$\frac{\partial \ln p(x_i|\mathbf{x}_{\setminus i};\psi_i(\mathbf{x}_{\setminus i}))}{\partial \psi_i(\mathbf{x}_{\setminus i})} = x_i - \lambda_i(\mathbf{x}_{\setminus i}) \ .$$

Proof. One can verify the correctness of this gradient as follows:

$$\begin{aligned}\frac{\partial \ln p(x_i|\mathbf{x}_{\setminus i};\psi_i(\mathbf{x}_{\setminus i}))}{\partial \psi_i(\mathbf{x}_{\setminus i})} &= \frac{\partial}{\partial \psi_i(\mathbf{x}_{\setminus i})} x \cdot \psi_i(\mathbf{x}_{\setminus i}) - \ln(x_i!) - \exp(\psi_i(\mathbf{x}_{\setminus i})) \\ &= x_i - \exp(\psi_i(\mathbf{x}_{\setminus i})) = x_i - \lambda_i(\mathbf{x}_{\setminus i}) \qquad \square\end{aligned}$$

These point-wise gradients are quite intuitive. We want to make the predicted means $\lambda_i(\mathbf{x}_{\setminus i})$ as similar to the observed count x_i as possible.

However, we are still left with the step size parameter η to compute the updated mean model in Equation (7.5). For other probabilistic models such as CRFs, performing a line search was reported to be too expensive [50]. Hence, it was suggested to rely on the "self-correcting" property of tree boosting to correct any overshoot or undershoot on the next iteration, i.e., to use a step size of $\eta = 1$. Unfortunately, we have observed in our experiments that using a fixed step size of $\eta = 1$ can lead to very slow convergence or failures due to overflows for large counts. Consequently, we now develop our main technical contribution, *a multiplicative gradient boosting approach* that handles these issues without incurring any computational overhead.

7.3.3 Multiplicative Gradient Boosting

For non-negative optimization problems, multiplicative update rules for the parameters have been shown to have much better convergence rates, see e.g., [132, 195, 199, 250], and we confirm this for PDNs in our experimental section. More importantly, since the tree-structure of the induced regression trees handles much of the overall PDN complexity, faster convergence implies sparser PDNs.

To derive a multiplicative update, we consider a simpler functional dependency between $\lambda_i(\mathbf{x}_{\setminus i})$ and the feature function ψ_i. More precisely, triggered by Chen et al. [35], who have proven the usage of the identity link function to be beneficial when using parametrized univariate Poisson distributions for large-scale behavioral targeting, we assume $\lambda_i = \psi_i$ (again omitting the iteration index t). For this link function, the point-wise gradients are the following:

Theorem 7.2. *The point-wise gradients of the log-probability assuming the identity link function are*

$$\frac{\partial \ln p(x_i|\mathbf{x}_{\setminus i};\psi_i(\mathbf{x}_{\setminus i}))}{\partial \psi_i(\mathbf{x}_{\setminus i})} = \frac{x_i}{\lambda_i(\mathbf{x}_{\setminus i})} - 1 \ ,$$

and can be used to realize an additive functional gradient ascent.

Proof. The correctness of the point-wise gradient can be seen as follows:

$$\begin{aligned}\frac{\partial \ln p(x_i|\mathbf{x}_{\setminus i};\psi_i(\mathbf{x}_{\setminus i}))}{\partial \psi_i(\mathbf{x}_{\setminus i})} &= \frac{\partial}{\partial \psi_i(\mathbf{x}_{\setminus i})} x_i \ln \psi_i(\mathbf{x}_{\setminus i}) - \ln(x_i!) - \psi_i(\mathbf{x}_{\setminus i}) \\ &= \frac{x_i}{\psi_i(\mathbf{x}_{\setminus i})} - 1 = \frac{x_i}{\lambda_i(\mathbf{x}_{\setminus i})} - 1 \end{aligned}$$ □

The multiplicative update now follows from Theorem 7.2 together with a functional step size.

Corollary 7.1. *The multiplicative functional gradient ascent using the identity link function is*

$$\psi^{t+1}(\mathbf{x}_{\setminus i}) = \psi^t(\mathbf{x}_{\setminus i}) \cdot \mathbb{E}_{\mathbf{x}_{\setminus i},x_i}\left[\frac{x_i}{\lambda_i(\mathbf{x}_{\setminus i})}\right], \tag{7.8}$$

and the training instances can be generated using

$$\left(\mathbf{x}^{(j)}_{\setminus i}, \frac{x_i}{\lambda_i(\mathbf{x}_{\setminus i})}\right) . \tag{7.9}$$

Proof. From Theorem 7.2 it follows that the point-wise gradients can be split into a negative part, namely 1, and a positive part, namely $x_i\lambda_i^{-1}(\mathbf{x}_{\setminus i})$. Since the functional gradient ∇_i^t in Equation (7.6) is an expectation, ∇_i^t itself can be split into a negative part and a positive part, $\nabla_i^{t-} = \mathbb{E}_{\mathbf{x}_{\setminus i},x_i}[1] = 1$ respectively $\nabla_i^{t+} = \mathbb{E}_{\mathbf{x}_{\setminus i},x_i}\left[\frac{x_i}{\psi_i(\mathbf{x}_{\setminus i})}\right]$ with $\nabla_i^t = \nabla_i^{t+} - \nabla_i^{t-}$. Setting now the step size to

$$\eta = \frac{\psi_i^t(\mathbf{x}_{\setminus i})}{\nabla_i^{t-}} ,$$

the additive gradient boosting Equation (7.5) can be rewritten as

$$\begin{aligned}\psi^{t+1}(\mathbf{x}_{\setminus i}) &= \psi_i^t(\mathbf{x}_{\setminus i}) + \eta \cdot \nabla_i^t = \psi^t(\mathbf{x}_{\setminus i}) + \eta \cdot (\nabla_i^{t+} - \nabla_i^{t-}) \\ &= \psi_i^t(\mathbf{x}_{\setminus i}) + \frac{\psi_i^t(\mathbf{x}_{\setminus i})}{\nabla_i^{t-}} \cdot (\nabla_i^{t+} - \nabla_i^{t-}) = \psi^t(\mathbf{x}_{\setminus i}) \cdot \frac{\nabla_i^{t+}}{\nabla_i^{t-}} = \psi^t(\mathbf{x}_{\setminus i}) \cdot \frac{\nabla_i^{t+}}{1} \\ &= \psi^t(\mathbf{x}_{\setminus i}) \cdot \mathbb{E}_{\mathbf{x}_{\setminus i},x_i}\left[\frac{x_i}{\psi_i(\mathbf{x}_{\setminus i})}\right] = \psi^t(\mathbf{x}_{\setminus i}) \cdot \mathbb{E}_{\mathbf{x}_{\setminus i},x_i}\left[\frac{x_i}{\lambda_i(\mathbf{x}_{\setminus i})}\right] .\end{aligned}$$

This proves the correctness of the multiplicative functional gradient in Equation (7.8). Moreover, following now the same steps as for the additive functional gradient ascent, this also proves the correctness of point-wise gradients in Equation (7.9). That is, as for the additive update, following the multiplicative functional gradient Equation (7.8), approximated using regression trees trained using Equation (7.9), will increase the log-likelihood assuming there are enough training examples. □

The point-wise multiplicative updates are quite intuitive, too. Instead of making the differences as small as possible, we want to make the *ratio* of observed counts x_i and predicted means $\lambda_i(\mathbf{x}_{\setminus i})$ as close to 1 as possible.

Finally, we comment on the choice of the identity link function. Using it with multiplicative updates, we make the assumption that the expected means are greater than zero. This is plausible, since a count variable should be observable and aligns well with the definition of the

#	X_1	X_2	X_3	∇_3^1	h_3^1	λ_3^1
1	0.5	4	0.5	$0.5 - 0.75$	0.67	$\exp(\ln(0.75) \cdot \frac{2}{3}) = 0.81$
2	1.5	3	0.5	$0.5 - 0.75$	0.67	$\exp(\ln(0.75) \cdot \frac{2}{3}) = 0.81$
3	3.5	1	1.25	$1.25 - 0.75$	1.67	$\exp(\ln(0.75) \cdot 1.67) = 0.79$

Figure 7.3: Example highlighting that multiplicative updates in the case of PDNs with log link mean can fail to optimize correctly.

Poisson distribution. Second, positive and negative dependencies can still be expressed in a single regression model induced over the iterations. Third, one can naturally realize Laplace smoothing by adding constants α_i and β_i to the point-wise gradients:

$$\frac{x_i + \alpha_i}{\lambda_i(\mathbf{x}_{\setminus i}) + \beta_i} .$$

Finally, the identity and log link functions are weakly connected. Following the same steps as in the proof of Corollary 7.1 but now using Theorem 7.1, we arrive at the following multiplicative functional gradient ascent for the log link function:

$$\psi^{t+1}(\mathbf{x}_{\setminus i}) = \psi^t(\mathbf{x}_{\setminus i}) \cdot \frac{\mathbb{E}_{\mathbf{x}_{\setminus i}, x_i}[x_i]}{\mathbb{E}_{\mathbf{x}_{\setminus i}, x_i}\left[\lambda_i(\mathbf{x}_{\setminus i})\right]} = \psi^t(\mathbf{x}_{\setminus i}) \cdot \mathbb{E}_{\mathbf{x}_{\setminus i}, x_i}\left[\frac{\bar{x}_i}{\lambda_i(\mathbf{x}_{\setminus i})}\right] , \tag{7.10}$$

since $\mathbb{E}_{\mathbf{x}_{\setminus i}, x_i}[x_i]$ is the empirical mean $\bar{x}_i$ of x_i. Nevertheless, since $\bar{x}_i$ is a constant for all iterations, we can move the expectation out of the ratio. Approximating the expectation using the training samples now yields $\mathbb{E}_{\mathbf{x}_{\setminus i}, x_i}[\bar{x}_i \cdot \lambda_i^{-1}(\mathbf{x}_{\setminus i})]$

$$\approx \frac{1}{m} \sum_j \left[\frac{\bar{x}}{\lambda_i(\mathbf{x}_{\setminus i}^{(j)})}\right] = \frac{1}{m^2} \sum_j \left[\frac{\sum_l x_i^{(l)}}{\lambda_i(\mathbf{x}_{\setminus i}^{(j)})}\right] \geq \frac{1}{m^2} \sum_j \left[\frac{x_i^{(j)}}{\lambda_i(\mathbf{x}_{\setminus i}^{(j)})}\right] \tag{7.11}$$

because $\bar{x}_i$ is the empirical mean of non-negative values and, hence, it is always larger than any particular x_i. Since the number of training examples m is a constant, this proves that the multiplicative point-wise gradients for the identity link function establish a lower bound for a multiplicative update for the log link function.

However, we must be careful using the multiplicative update in the case of the log link mean. When the mean of the response variable is below 1, its logarithm is below zero and the optimization does not work. Figure 7.3 shows a toy example highlighting this situation. In this example, the mean of X_3 is 0.75. Therefore, in the first GTB iteration, the mean of the first two instances should be decreased, while the mean of the third instance should be increased. The tree learning the quotient of ∇^+ and ∇^- learns this properly, i.e., is below 1 for instances 1 and 2, and above 1 for the third instance. Here, we assume that h learns the data perfectly. However, due to the negative ψ^0, only the mean of the third instance is adjusted properly.

7.3.4 PRTs, Model Compression, and Dependency Recovery

So far, we have specifically assumed two natural candidates for the initial mean model $\lambda_i^0(\mathbf{x}_{\setminus i})$, namely a constant or the empirical mean of X_i. Both models might not be very precise and,

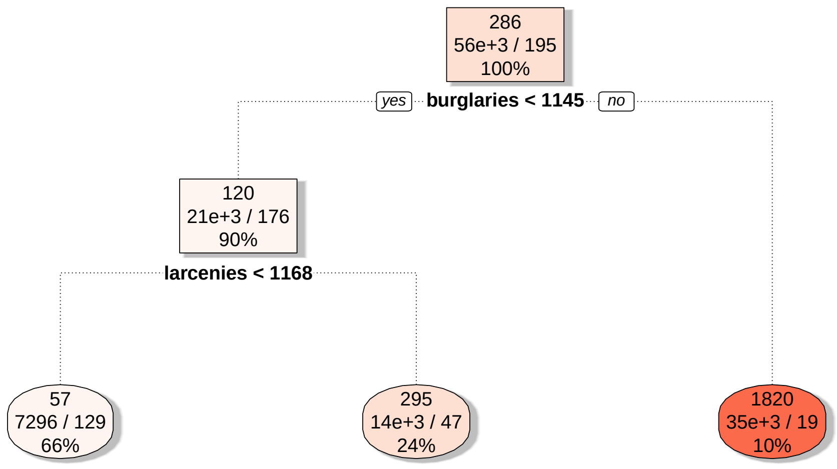


Figure 7.4: Example of a Poisson regression tree for the crime "auto theft" from the *Communities and Crime* dataset from our experimental evaluation. The root of the tree, denoted by a rectangle, represents the first splitting criterion. In this case, the root node splits the dataset according to the number of burglaries. Leafs are denoted by ellipses and contain the average mean of that partition in the first row of the label. Additionally, the color shade of a leaf node indicates the relative size of the mean (the darker, the higher).

in turn, may require many iterations of optimization, even when using adaptive step sizes via multiplicative updates, resulting in a large number of regression trees and a tendency to overfit.

Therefore, we propose another initial mean model that can potentially serve as a head start and comes with additional benefits as we will see. We are interested in a compact regression model for each node and since we are dealing with count data, *Poisson Regression Trees (PRTs)* [33] naturally lend themselves as a starting point. An example of a PRT is shown in Figure 7.4. To initialize our PDNs, we learn one PRT for each variable X_i, where we train $\text{tree}_i(X_i|\mathbf{X}_{\setminus i})$ and evaluate $\lambda_i^0(\mathbf{x}_{\setminus i}) = \text{tree}_i(\mathbf{x}_{\setminus i})$.

More precisely, a PRT partitions the training examples in the space of the dependent variables $\mathbf{X}_{\setminus i}$ in order to best fit the response variable X_i. It is a binary tree whose leaves represent the λ_i of that given partition, all other nodes represent a splitting criterion on a variable $X_j \in \mathbf{X}_{\setminus i}$. We use the PRT implementation of rpart [222] where the splitting criterion is given by the likelihood ratio test for two Poisson groups $D_{\text{parent}} - (D_{\text{left son}} + D_{\text{right son}})$ with the deviance given by

$$D = \sum \left[x_i \ln\left(\frac{x_i}{\bar{\lambda}}\right) - (x_i - \bar{\lambda}) \right] ,$$

where $\bar{\lambda}$ is the sample mean[2] of count variable X_i. This splitting is recursively applied until each subgroup reaches a minimum size or there are no improvements. To avoid overfitting, the depth of a tree is typically limited a priori, alternatively post-pruning can be used after the tree has been learned. In the rpart-implementation, the pre-pruning is controlled by a complexity parameter `cp`. This parameter, which we consider to be part of the hyper-parameter space of our algorithm (see Line 2 in Algorithm 9), controls the size of the initial tree. A secondary step is executed where the tree is pruned using cross validation after the initial tree has been

[2]For stability reasons the implementation uses a revised Bayes estimate of $\bar{\lambda}$.

```
input: A factor graph $G$
output: A list of generated samples $S$
S ← [ ];
s ← $\{0\}^n$;                                        /* Initialization */
for $k \leftarrow 0$ to $K$ do
    for $X_i \in V$ do
        s[$i$]← $X_i \sim P(X_i|\,\mathrm{nb}(X_i))$;       /* Sample new state of $X_i$ */
    end
    if $k >$ burn-in then
        S.append($s$);                            /* Append new sample to list */
    end
end
return S
```

Algorithm 10: Pseudo-code for the ordered Gibbs sampler.

learned. The height of the final tree will be the one that reduces the cross validation error. This learning approach generalizes well in our experiments as shown below in Section 7.5.

A potential head start is not the sole advantage of PRTs. We can also use PRTs for model compression [29] which is, next to Laplace estimates, an additional way to avoid overfitting. Here, we collapse the trained additive, respectively multiplicative model, into a single model. To do so, we evaluate it on the training set and learn a single PRT per count variable based on this evaluation. That is, the compressed PDN model consists only of a set of local PRTs that were learned based on the optimized GTB model.

Moreover, interactions among count variables are directly conveyed by the structure of the compressed PDN and, as a result, interactions can be understood and interpreted more easily in qualitative terms. More precisely, since after compression there is only one local tree model per count variable, we simply look at the count variables used in the inner split nodes of the trees; they indicate relevant features of the PDN. It is important to note that one tree can use a variable X_k multiple times with different splitting criteria. This indicates that the variable is important in the PDN. Also, count variables closer to the root of a tree are more important than a variable further down the tree. This is, e.g., captured by Breiman et al.'s notation of *relative importance* [28] saying how important the value of x_v is for predicting the value for x_u:

$$I^2(u|v;\lambda_u) = \sum_l i_l^2 \cdot \delta_{v(l)=u} \;, \tag{7.12}$$

where l iterates over the levels of the Poisson tree for λ_u of X_u, the value i_l^2 is the maximal estimated improvement over a constant fit over the entire region of the current node $v(l)$, and δ is an indicator function selecting all splits involving x_v. To summarize, in contrast to other Poisson graphical models, compressed PDNs return local models that are likely to be sparse and therefore easier to interpret.

7.4 Making Predictions using Poisson Dependency Networks

Similar to the experiments we have seen in the previous chapters, there exist also many applications for count models where we obtain only partially observed instances and we want to use probabilistic inference to predict the values of the missing variables. To be more precise, assume that $\mathbf{X} = \mathbf{Y} \cup \mathbf{E}$, where $\mathbf{Y}$ and $\mathbf{E}$ are disjoint. $\mathbf{Y}$ amounts to the unobserved variables and $\mathbf{E}$ describes the evidence. Then we also want to answer queries of the form $p(\mathbf{y}|\mathbf{e})$, $p(y_i|\mathbf{e})$, or $\arg\max_{\mathbf{y}} p(\mathbf{Y} = \mathbf{y}|\mathbf{e})$ in situations where $\mathbf{y}$ corresponds to one or several count random variables. In an univariate Poisson model, MAP inference consists of just reading off the mode of the Poisson distribution which is equal to $\lfloor \lambda_i \rfloor$. Other marginal probabilities, e.g. $p(y_i = k)$, can also be read off the distribution because the Poisson distribution is completely defined by λ_i. The same holds for a variable Y_i in a PDN if all neighbors of this variable are observed. However, PDNs with unobserved variables that also have further unobserved neighbors require an inference machinery to account for the dependencies. We resort to Gibbs sampling [64] to do so. Chapter 2 already referred to MCMC techniques and to the Gibbs sampler as one particular simple instance of a sampling algorithm. This chapter heavily relies on Gibbs sampling and thus we will also give the pseudo-code of an ordered Gibbs sampler in Algorithm 10.

Because we do not know the underlying joint distribution explicitly, the Gibbs sampler provides us with pseudo samples of a joint distribution, and hence it is also called *Pseudo Gibbs* sampler [90] due to the potential inconsistencies arising from conflicting local and joint distributions. Specifically, cf. Algorithm 10, the pseudo Gibbs sampler starts with an arbitrary initialization of the unobserved variables and then iterates over each variable for a previously defined number of sweeps. Each sweep produces a new sample by first calculating the conditional probability distribution for every variable X_i. That is, calculating the mean λ_i based on the current states of its parent variables pa_i. Based on the Poisson distribution parameterized by λ_i, a new state for X_i is sampled. This is depicted in Algorithm 10 in Line 5 for the general case of a variable X_i with neighbors $\text{nb}(X_j)$. This procedure will sample from the joint distribution after an adequate burn-in phase which removes early samples because they are heavily biased by the initial values.

Technically, this algorithm does not distinguish itself from a standard Gibbs sampler. However, in the case of an inconsistent set of local probability distributions, it is referred to as a “pseudo” Gibbs sampler because we do not know a closed form joint distribution for this PDN. From the collected samples we can then compute an approximate marginal distribution or MAP assignment. However, the order of the Gibbs does matter in the case of PDNs due to conflicting w_{ij} and w_{ji}, which also includes conflicting trees in the case of GTB or PRTs. Consequently, we follow Bengio et al. [17] and use an unordered pseudo Gibbs sampler which randomly chooses an X_i in each step. Bengio et al. showed that this unordered pseudo Gibbs sampler induces a so-called Generative Stochastic Network (GSN) Markov chain. As long as this GSN Markov chain has a stationary distribution, the DN defines a joint distribution, which, however, does not have to be known in closed form. From this perspective, PDNs aim to estimate the generating distribution of multivariate count data indirectly, by boosting the transition operator of a Markov chain rather directly. As long as the chain is ergodic, we will arrive at a consistent estimator of the associated joint distribution.

7.5 Experimental Evaluation

Our intention here is to investigate the usefulness of PDNs and to show their benefits in comparison to other Poisson graphical models. To this aim, we investigated the following questions:

Q7.1 Can PDNs learn both positive and negative dependencies?

Q7.2 Can Gibbs sampling predict good counts?

Q7.3 Can PDNs outperform existing Poisson graphical models?

Q7.4 Can PDNs be easy to interpret?

Q7.5 Can gradient tree boosting improve the quality of an initial model?

Q7.6 Can multiplicative updates speed up training without sacrificing performance?

If all questions can be answered affirmatively, PDNs have the potential to be a valid alternative to existing Poisson graphical models.

We implemented PDNs using a combination of Python and R, and conducted several experiments on synthetic and real-world datasets on a standard single-core Linux machine with 32GB RAM.

To assess the quality of the trained models, we used different measures. We have seen in previous chapters likelihood based scores already. For PDNs, PLL_SCORE is slightly adapted compared to Equation (2.11) in such away that it is normalized and averaged over a set of instances. Given a training dataset $(\mathbf{x}^{(j)})_{j=1,\dots,m}$, the pseudo log-likelihood score on the training data is defined as follows:

$$\text{PLL\_SCORE}(\mathbf{x}^{(1)},\dots,\mathbf{x}^{(m)}|\boldsymbol{\theta}) = -\frac{1}{m\cdot n}\sum_{j=1}^{m}\sum_{i=1}^{n}\ln p(x_i^{(j)}|\mathbf{x}^{(j)}{}_{\setminus i};\theta_i)\ . \tag{7.13}$$

Here, θ_i amounts to the set of parameters that define the conditional probability distributions. This can be the $\mathbf{w}$ in the case of log-linear models or the set of learned trees in the case of boosted models. This also subsumes the initial model, for example the initializing Poisson regression tree. The score on the test data can be calculated accordingly. However, sometimes it is more informative to consider the scores separately for each count variable x_j, i.e.,

$$\text{PLL\_SCORE}_j(\mathbf{x}^{(1)},\dots,\mathbf{x}^{(m)}|\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\ln p(x_j^{(i)}|\mathbf{x}^{(i)}{}_{\setminus j};\theta_j)\ .$$

Both measures can also be used to measure the performance of the predicted counts from the Gibbs sampler. Assuming that $\hat{x}_i^{(j)}$ denotes the predicted value of variable X_i for the data case j and $x_i^{(j)}$ is the true observation of variable X_i, we can calculate the predictive pseudo log-likelihood score:

$$\text{PLL\_SCORE}(\mathbf{x}^{(1)},\dots,\mathbf{x}^{(m)};\hat{\mathbf{x}}^{(1)},\dots,\hat{\mathbf{x}}^{(m)}|\boldsymbol{\theta}) = -\frac{1}{m\cdot n}\sum_{j=1}^{m}\sum_{i=1}^{n}\ln p(x_i^{(j)}|\hat{\mathbf{x}}_{\setminus i}^{(j)};\theta_i)\ .$$

Following convention, we also consider a performance score based on the averaged RMSE. For PDNs this score differs marginally from the one we have seen in Equation (2.13):

$$\text{RMSE} = \sqrt{\frac{1}{m\cdot n}\sum_{j=1}^{m}\sum_{i=1}^{n}\left(\hat{x}_i^{(j)} - x_i^{(j)}\right)^2}\ ,$$

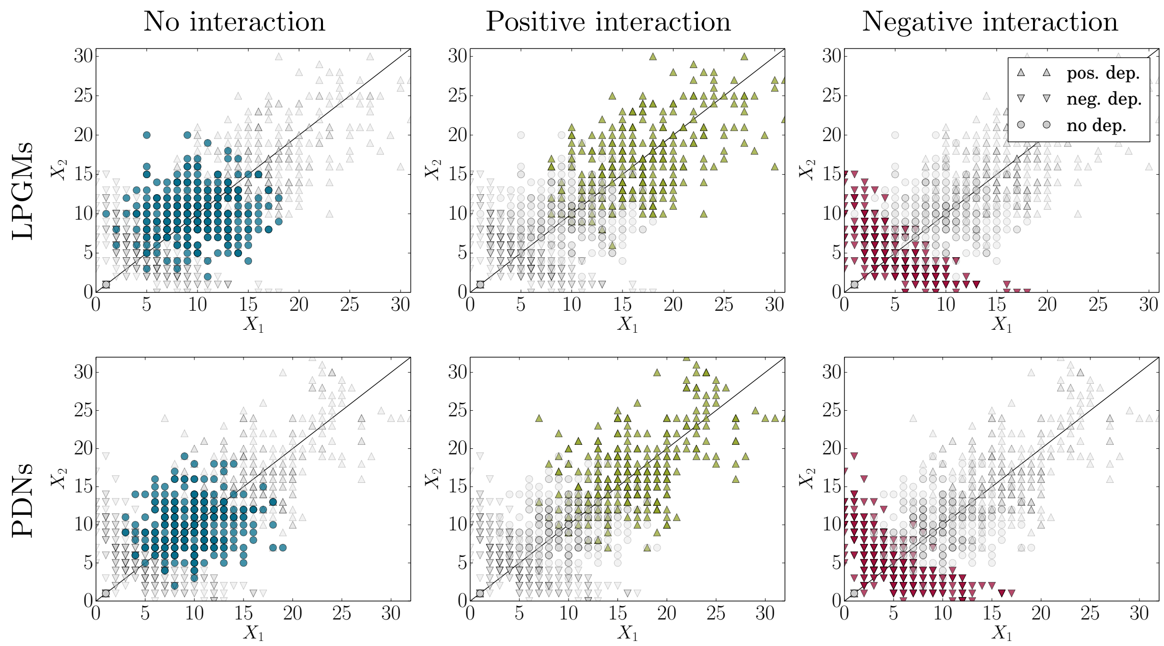


Figure 7.5: (top) A log-linear Poisson model. (a) Model without dependencies between X_1 and X_2. The data points (blue circles) are close to $w_0 = \ln(10)$. (b) Model with positive dependencies. The data points (green triangles) are now shifted towards the upper right corner. (c) Model with negative dependencies. The data points (red triangles) are pushed to the lower corner. (bottom) A boosted PDN. We re-learned a boosted PDN from the samples generated by the log-linear model. One can see that all three groups are well separated by the tree-based model as well.

where $\hat{x}_i^{(j)}$ again denotes the predicted value of variable X_i for the data case j and $x_i^{(j)}$ is the true observation of variable X_i. Compared to the univariate measure in Equation (2.13), we now calculate the error over all variables and all datacases.

However, especially in the case of Poisson distributed variables with small values, the RMSE is not a good measurement because it assumes a symmetrically shaped error. Moreover, it takes only point predictions into account. A slightly better loss for our task is the normalized RMSE (NRMSE):

$$\mathrm{NRMSE} = \frac{1}{n} \sum_i^n \frac{\left(\frac{1}{m} \sum_j^m (\hat{x}_i^{(j)} - x_i^{(j)})^2\right)^{\frac{1}{2}}}{x_i^{\max} - x_i^{\min}} .$$

7.5.1 Network Discovery from Simulated Data

In the first set of experiments we focus on **Q7.1**, **Q7.2**, and **Q7.3**. In order to investigate whether PDNs can model positive and negative dependencies we used the simple PDN that was shown already earlier in Figure 7.2. This PDN consists of three variables where one variable, namely X_0, only acts as a dummy variable for the constant parameter. The means of X_1 and X_2 are described by log-linear models, e.g., $\lambda_1 = \exp(w_1 + w_{12}X_2)$. Because X_0 corresponds to the constant feature with value 1, we can omit it from the sum. In turn, this PDN is an LPGM

	Hub					Scale-free				
	10	25	50	75	100	10	25	50	75	100
PDNs	**0.614**	**0.599**	**0.558**	0.493	0.463	**0.415**	**0.489**	**0.548**	**0.472**	**0.412**
LPGMs	(0.222)	0.400	0.530	**0.545**	**0.554**	(0.100)	0.262	0.372	0.376	0.286
SPGMs	(0.316)	0.310	0.415	0.515	0.473	(0.000)	0.194	0.286	0.327	0.271

Table 7.2: Comparison of network discovery performance based on F1-scores (the higher, the better). Results are averaged over five graphs for each size. LPGMs are the locally learned models from [7] and SPGMs are the consistent Poisson graphical models introduced in [249]. The F1-scores in brackets "(·)" indicate that SPGMs and LPGMs failed to converge on several graphs and hence did not return results for every experiment.

as introduced by Allen and Liu. We chose $w_i = \ln(10)$, which means that we expect an average value of 10 in the completely independent case. For different choices of the pairwise parameters, we use this PDN to sample 200 instances from the joint distribution after a burn-in phase. We consider three different parameter choices of the PDN in total:

i the independent case (blue circles)

ii cooperative interactions (green triangles)

iii competitive interactions (red triangles).

For (ii) and (iii) we assumed $w_{12} = w_{21}$. The obtained samples are plotted in the top row of Figure 7.5. As it can be observed, the samples drawn with the different parameter settings are well separated and all three types of dependencies are captured well. Now we estimated boosted PDNs on this data using additive updates. We then used this model to generate 200 new samples. The results are summarized in the bottom row of Figure 7.5. Again, it can be observed that the three groups are well separated, highlighting that boosted PDN model both, positive and negative, dependencies well. This clearly answers **Q7.1** affirmatively. In addition, this also highlights the fact that the (pseudo) Gibbs sampler does generate good samples for boosted PDN models. This already provides an affirmative answer to question **Q7.2**, but we will additionally use the experiments in the next subsection to answer this question using real world datasets.

Before doing so, we will return to the experiments already shown in Section 7.1. For this structure recovery task, we used the code provided by Yang et al. [249] for both, the data generation and the algorithmic comparison. We employed the "PGM"[3] R-package which contains several implementations of the models described in [249]. This includes SPGMs and LPGMs, and additionally also provides code to sample instances from Winsorized Poisson graphical models.

The datasets were generated as follows. First, with the help of the R-package "huge"[4] a graph with $n = 10, 25, 50, 75, 100$ nodes was generated. This data generation process is based on multivariate normal distributions for different graph structures; we created "hub" and "scale-free" graphs. The adjacency matrix of the resulting graph was used to construct the neighborhoods in the Poisson graphical model. Together with an unary ($w_i = 0$) and a

[3]`http://github.com/zhandong/XFam/tree/master/PGMs/Package_Trial/T2`
[4]`http://cran.r-project.org/web/packages/huge/`

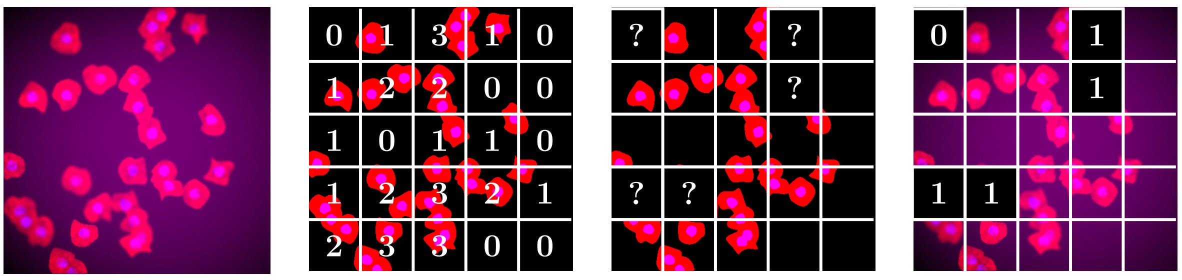


Figure 7.6: Prediction of cell counts. We partitioned the original image (a) into 25 patches of equal size. We then counted the number of cells in each patch as depicted in (b). To measure the predictive power of our PDNs, we removed the counts of 5 random patches in the images (c) and ran the Gibbs sampler of the partially observed image. We then read off the most likely counts of each missing patch as shown in (d).

pairwise ($w_{ij} = 0.1$) parameter, we then obtained the Poisson graphical model. We used the Gibbs Simulator for Winsorized Poisson graphical models contained in the "PGM" package to generate $m = 1000$ samples. This resulted in an $m \times n$ matrix with count values used to learn a model for the original neighborhood graph. More precisely, we used the relative importance to construct an adjacency matrix from our learned models by setting a cell to 1 whenever $I^2(u|v; \lambda_u) > 0$. Similarly, an adjacency matrix can be read off from the parameter matrix of LPGMs and SPGMs. The results in terms of F1-scores are shown in Table 7.2 and correspond to Figure 7.1. As it can be seen, PDNs achieve competitive results on the hub graphs and outperform LPGMs and SPGMs on the scale-free networks. In the latter case, LPGMs and SPGMs may not even converge in some experiments. Moreover, as already shown in Figure 7.1c, PDNs can be an order of a magnitude faster.

Because of this, we only compared PDNs to LPGMs in the remaining experiments. In fact, we did ran SPGMs, but on some of the more complex datasets they were not only an order of magnitude slower but did not terminate after days of running; while learning a PDN took less than thirty minutes. Moreover, we note that in the following, LPGMs refer to local Poisson models with GLM mean models learned without regularization. To cover regularization, we also present results for PDNs with Poisson regression trees as mean models, which can be seen as a regularized model.

7.5.2 Cell Counts and Bibliography Data

The next set of experiments is dedicated to **Q7.2**, **Q7.3**, and **Q7.5**. To investigate the quality of predicted counts returned by the Gibbs sampler, we used two different datasets, namely cell counts and counts of publications.

Cell Counts. The analysis of microscope images has become a popular application combining biology and computer vision. One particular application is counting the number of cells in an image. To do so, we used the SIMCEP-tool[5], which is described in [134], to generate 100 microscope images showing cells. An example image is depicted in Figure 7.6a. We split each image into 25 patches of equal size, and the cells within each patch were counted (see Figure 7.6b). This gives us a dataset with 25 count values per image. We employed 10-fold cross

[5] `www.cs.tut.fi/sgn/csb/simcep/tool.html`

validation. More precisely, we used 90 images to train the PDNs in various ways over these counts and used the remaining 10 images to do predictions based on the learned model. We are primarily interested in constructing a simple use case for our PDNs and showing that reasonable predictions are possible. Therefore, we randomly removed the counts of five patches in each image for prediction, as depicted in Figure 7.6c, and used the trained PDNs to infer the cell counts for the missing patches. We ran the Gibbs sampler for 50,000 iterations and determined the MAP configurations of the missing patches from the samples produced, cf. Figure 7.6d.

The results are summarized in Table 7.3(top rows) and Table 7.4(top rows). As it can be seen, additively boosted PDNs — denoted by "A" — produced the lowest *Normalized RMSE (NRMSE)*, followed by multiplicatively boosted PDNs — denoted by "M". Due to the structure of the problem, an identical step size was chosen for all variables in PDN-A. It must be mentioned that we determined a step size of 0.25 by systematic search. Nevertheless, we also present experimental results for a step size of 1.0. It can be observed that simply setting $\eta = 1.0$ leads to worse performance and the learning algorithms is not capable of improving over more than two iterations as too large steps are taken. We also tested Laplace smoothing with $\alpha_i = 0.1$ and $\beta_i = 0.2$ in the cases of multiplicative updates. It can be seen that this smoothing improves the quality of the predictions in terms of the predictive log-score. Although PDNs with a log-linear mean model, i.e., LPGMs, produced the best log-score on the test data (LOO LL), they were not able to make meaningful predictions due to overflows. This is mainly due to the fact that patches are often correlated with strong positive weights. Using the Gibbs sampler quickly produces (too) large counts which are meaningless. In comparison, using Poisson regression trees — PDN$^{\text{tree}}$ — helps accommodating for highly varying counts and in turn to limit the predicted counts.

Publication Data. We have already shown in the previous chapter that bibliographic datasets are interesting for Machine Learning applications. In this section, we focus on the prediction of publications counts and use the AAN corpus as discussed above (see Section 5.3.1). The goal is to predict the number of publications of a researcher in future years given an initial observation period. We observe the first six years of a researchers' publication record and predict the number of publications for the following four years. We take only active researchers into account, i.e., researchers who had publications in the first three years of their career.

The 10-fold cross-validated results for the training and test likelihoods are summarized in Table 7.3. Here, boosted PDNs with additive updates achieve the best results. However, one should note that these results were obtained by optimizing the step size for each class separately. Instead of a simple grid search, we used ParamILS [95] to find the best step sizes. In our setting, we obtained step sizes of 0.5, 0.075, 0.05, and 0.075 for the four years to predict. Although Allen and Liu's LPGMs — PDNs with log-linear mean-models — do a full maximum likelihood optimization for the parameters, they are not able to outperform the boosted approaches. Most importantly, with only a few iterations of optimization and no ParamILS, multiplicative updates achieved a better train likelihood than LPGMs, without sacrificing the test likelihood much.

Even more interesting are the collective prediction performances as summarized in Table 7.4. As one can clearly see, PDN$^{\text{tree}}$, i.e., PDNs with Poisson Regression Trees, do the best job. On the first sight, it is striking that boosted PDNs with initial tree models and multiplicative updates perform worse than standard PDN$^{\text{tree}}$. We attribute this to the rather small validation set used during boosting and in turn to overfitting. This is also confirmed by the boosting results using Laplace smoothing with $\alpha_i = 0.1$ and $\beta_i = 0.2$. It significantly reduced the error

	Boost. Iters.	Train LL	LOO LL	
PDN-A$^{\text{const,log}}$ (w/ $\eta = 1$)	1.91	1.026	1.541	cells
PDN-A$^{\text{const,log}}$ (w/ selected η)	5.60	1.046	1.428$^{\bullet}$	
PDN-M$^{\text{const,id}}$	1.26	1.167	1.547	
PDN-M$^{\text{const,id}}$ (Laplace)	1.75	1.074	1.511	
PDN-M$^{\text{tree,id}}$	**0.85**	1.134	1.655	
PDN-M$^{\text{tree,id}}$ (Laplace)	1.18	1.049	1.590	
PDN$^{\text{tree}}$	-	1.219	1.525	
LPGM [7, Allen *et al.*]	-	**0.963**	**1.117**$^{\circ}$	
PDN-A$^{\text{const,log}}$ (w/ $\eta = 1$)	x	x	x	publications
PDN-A$^{\text{const,log}}$ (w/ ParamILS η)	31.50	0.980	**1.182***	
PDN-M$^{\text{const,id}}$	1.95	1.069	1.248	
PDN-M$^{\text{const,id}}$ (Laplace)	2.68	0.994	1.228	
PDN-M$^{\text{tree,id}}$	**0.23**	0.998	1.232	
PDN-M$^{\text{tree,id}}$ (Laplace)	0.83	**0.955**	1.235	
PDN$^{\text{tree}}$	-	1.004	1.212	
LPGM [7, Allen *et al.*]	-	1.137	1.233	

Table 7.3: 10-fold cross validation comparison of the log-scores (the lower, the better) on training and test data for the cell count images (top) and publication data (bottom). The "x" indicates that training did not finish properly due to oscillation or overflows. For LOO LL, a "$\bullet$" denotes that boosted PDNs are significantly better then non-boosted PDNs, a "$\circ$" denotes that LPGMs are significantly better than PDNs, and a "$\star$" denotes that PDNs are significantly better than LPGMs.

for PDN-M$^{\text{const,id}}$. In any case, LPGMs — the only model not developed in the present work — are not capable of predicting all test instances due to overflows in the Gibbs sampling.

To summarize, both experiments answer questions **Q7.2** and **Q7.3** affirmatively. Moreover, they already indicate that **Q7.5** may also be answered affirmatively.

7.5.3 Bag-of-Word PDNs

To investigate whether PDNs can easily be interpreted (**Q7.4**), we trained a PDN on a text corpus. Specifically, we used the NIPS bag-of-words dataset from the UCI repository[6] containing 1,500 documents with a vocabulary above 12k words. We considered the 100 most frequent words in the corpus only and trained an additively boosted PDN on this data, i.e., PDN-A$^{\text{const,log}}$. Figure 7.7 shows the dependency structure among the words extracted from the trained PDN using relative importance I^2 (cf. Equation (7.12)). The size of a node is relative to the occurrence of the word in the corpus. As one can see, the word dependencies are rather sparse and reflect natural groupings of words. Words such as "neural", "net", "network", "weight", "input", and "unit" respectively "learning", "algorithm","loss" and "function" are inter-related as indicated by the strength of the edges. Generally, the interactions are asymmetric. For

[6] `https://archive.ics.uci.edu/ml/datasets/Bag+of+Words`

	Pred. LL	NRMSE	
PDN-A$^{\text{const,log}}$ (w/ selected η)	**1.429**$^\bullet$	**0.396**$^\bullet$	
PDN-M$^{\text{const,id}}$	1.570	0.408	
PDN-M$^{\text{const,id}}$ (Laplace)	1.503	0.410	
PDN-M$^{\text{tree,id}}$	1.619	0.455	cells
PDN-M$^{\text{tree,id}}$ (Laplace)	1.576	0.448	
PDN$^{\text{tree}}$	1.511	0.439	
LPGM [7, Allen *et al.*]	-	-	
PDN-A$^{\text{const,log}}$ (w/ PARAMILSη)	**0.446**$^\bullet$	0.262	
PDN-M$^{\text{const,id}}$	0.609	0.320	
PDN-M$^{\text{const,id}}$ (Laplace)	0.610	0.268	
PDN-M$^{\text{tree,id}}$	0.525	0.256	publications
PDN-M$^{\text{tree,id}}$ (Laplace)	0.550	0.254	
PDN$^{\text{tree}}$	0.508	**0.249**	
LPGM [7, Allen *et al.*]	-	-	

Table 7.4: 10-fold cross validation comparison of collective prediction performances (the lower, the better) made by the Gibbs sampler on cell count images (top) and publication data (bottom). A "-" indicates that PDNs with log-linear mean were not able to predict the counts for all instances due to overflows. A "$\bullet$" denotes that boosted PDNs are significantly better than non-boosted PDNs.

instance, the frequency of "training" has an impact on how often we read "error". However, words such as "model" and "data" are connected by edges of similar weights in both directions. This suggest that they often co-occur. Overall, the extracted dependency structure shows that PDNs can be easily interpreted and used to gain non-trivial insights; an affirmative answer to **Q7.4**.

7.5.4 Communities & Crime and Click-Stream Data

Finally, to investigate the overall performance of PDNs (**Q7.5**), as well as to compare additive and multiplicative updates (**Q7.6**), we used two real-world datasets, namely the Communities & Crime dataset and a click-stream dataset.

Communities & Crime. This dataset from the UCI repository[7] was obtained from 2,215 communities in the United States, reporting different crime statistics. It contains 125 observational features presenting different demographics such as the population of a community but also statistics such as the unemployment rate. There are also eight different target statistics per community and we focus on the count values specifying crimes such as the number of robberies, burglaries, and others. The dataset contains missing values for some of the features and for some target variables. For our evaluation, we removed communities with incomplete

[7]`archive.ics.uci.edu/ml/datasets/Communities+and+Crime+Unnormalized`

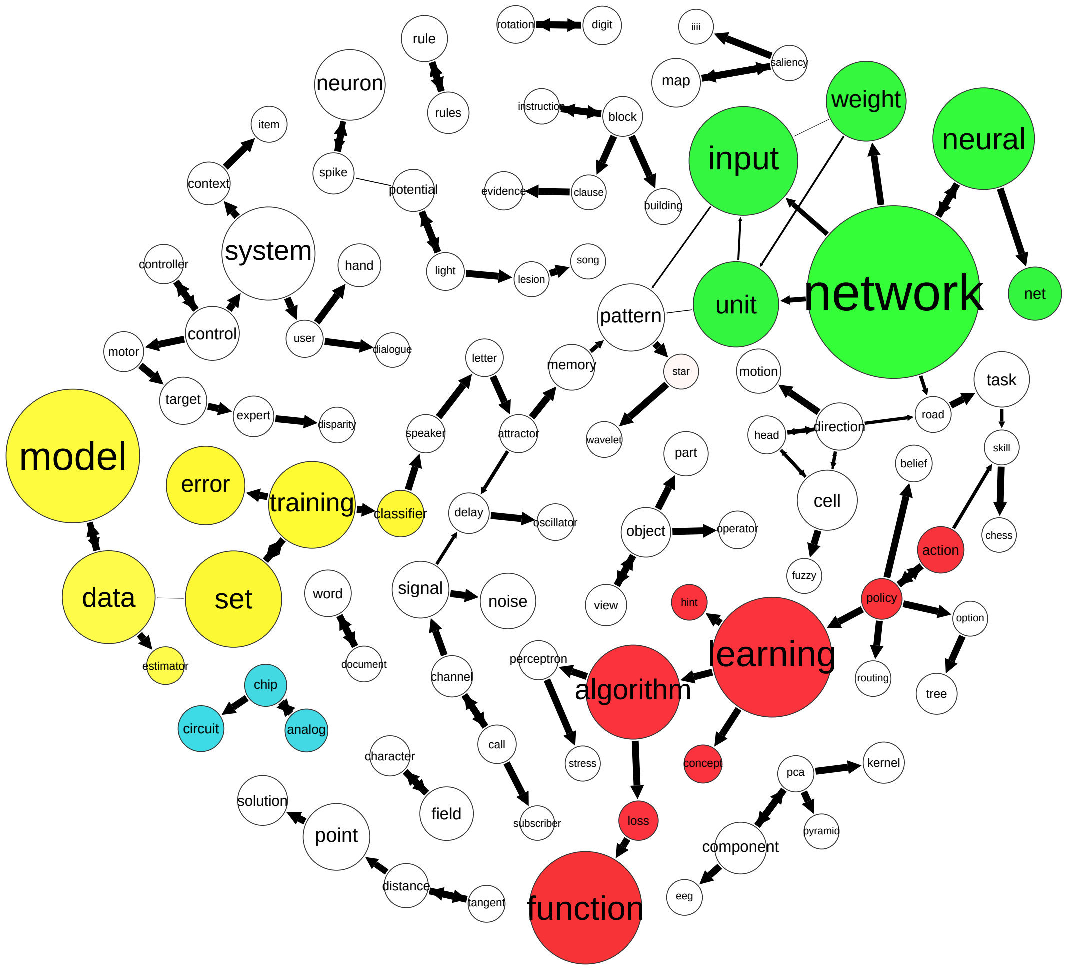


Figure 7.7: Dependency structure among the top 100 words of the NIPS corpus extracted from a PDN using relative importance. The dependencies reflect natural groupings of words as illustrated for some cases by the colors.

data as well as features that are not available for all communities[8]. Our cleaned data dataset contains 1,902 communities with 101 features.

For a comparison of the additive and multiplicative updates for PDNs, we used a 10-fold cross validation based on the folds defined in the dataset. The plots in Figure 7.8 show the learning curves for the eight different crimes. The plots are averaged over the ten folds using a maximum of $T = 50$ iterations and measure the effectiveness of the learning rate in terms of the log-score per class. For the additive models, PDN-A, we used a step size of $\eta = 0.01$ in case of the identity link function (red) and the log link case used $\eta = 1^{-5}$ (purple). Both step sizes were found based on a systematic search. We started with $\eta = 1.0$ and decreased the step size until no more oscillation was observed during the training. In particular, when using a constant mean-model for initialization, one can see that the multiplicative update (green) outperforms

[8] As one further exception, we also removed New York City from the data as it presents an extreme outlier in terms of size.

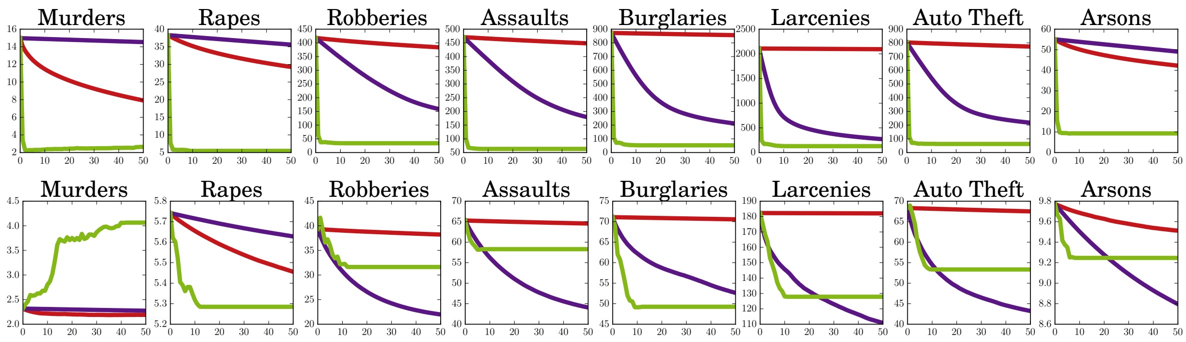


Figure 7.8: Comparison of gradient tree boosting PDNs on the C&C-dataset with initial mean or Poisson regression tree models. The plots show the learning curves in terms of the LL_SCORE$_i$ for the eight different crime types in the dataset. The top plots show the curves for initial models based on the mean. The curves highlight that multiplicative updates (green) learn much faster than the additive updates (red and purple). The bottom plots show the learning curves for initial regression tree models and indicate that the trees can give an head start.

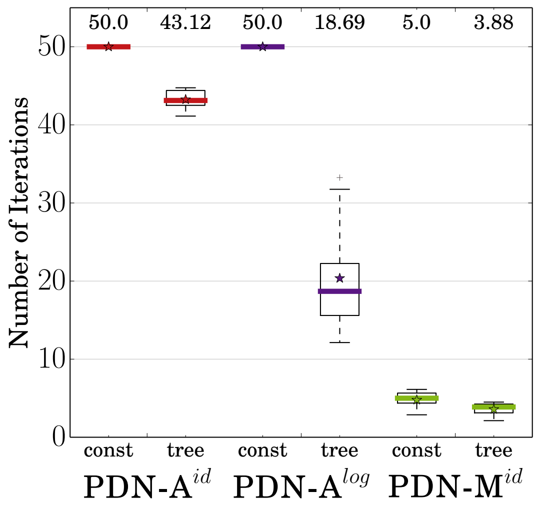
the additive ones: it is much faster and achieves better test performance. It is worth mentioning that this was achieved without a time consuming selection of an adequate step size at all. In case of the additive updates, the log link function learns faster in cases of crimes with high counts such as “larcenies” with data mean around 2,000 per community. For means with small values, such as “murders” having an empirical mean of around 6.4, however, we see that additive updates using the identity link function learned faster. Moreover, as shown in Figure 7.9(left), the additive updates always required the maximum of 50 iterations to obtain the best value. We determined the optimal iteration based on a validation set during the learning. On the other hand, multiplicative updates required only a fraction of the iterations, while not sacrificing predictive performance (see Figure 7.9(right)). Averaged over all experiments and classes, multiplicative updates require far less than 10 iterations and the log likelihood score on the test data is significantly better than for the additive case with mean initialization. This clearly shows the advantage of multiplicative updates.

What about using Poisson regression trees for initialization? Looking at the learning curves for PDNs with Poisson regression trees (second set of eight plots in Figure 7.8), we first see that the initial value of the log-score is drastically lower compared to the initialization with a constant mean. This is not surprising as the tree can represent the data initially already much better than a single mean value. The number of iterations required drops as well. This is also emphasized in the boxplot that compares the number of iterations (Figure 7.9, left). On average, the multiplicative updates, PDN-M, add only less than four trees to the initial tree and achieve comparable performance.

To summarize, boosting improves the initial model and multiplicative updates outperform additive updates. These are affirmative answers to our initial questions (**Q7.5**, **Q7.6**).

Click-Stream-Data. To reaffirm the results of the last experiment, we created a count dataset based on the *MSNBC.com* dataset from the UCI repository[9]. The data gives sequences corresponding to a user's page views for an entire day which are grouped into 17 categories. Instead of the original dataset, we used the post-processed version from the *SMPF* library[10]

[9]`archive.ics.uci.edu/ml/datasets/MSNBC.com+Anonymous+Web+Data`
[10]`www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php`



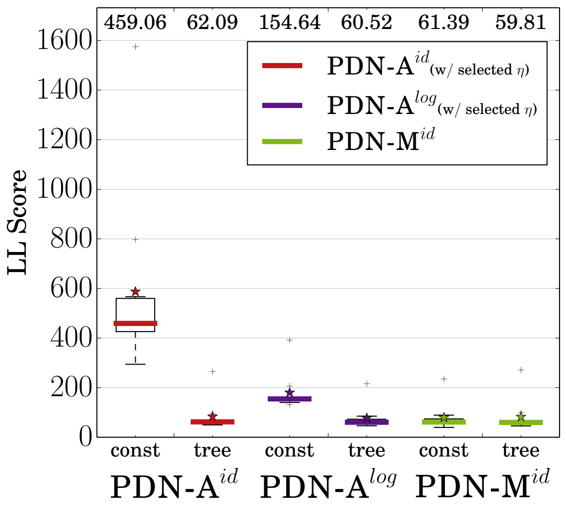


Figure 7.9: Comparison of gradient tree boosting PDNs on the C&C-dataset with initial mean or Poisson regression tree models. (left) The boxplot shows the average best iteration found by cross validation. For easier comparison, each column shows the mean in the top area of the plot. (right) The boxplot shows the average PLL_SCORE of the best iteration on the test dataset. The results show that the multiplicative updates, PDN-M, require only very few iterations to achieve comparable, or even better, results than the additive updates, PDN-A. This holds for both types of initializations, mean and tree models, even though the step sizes were carefully chosen for PDN-A.

which removed very short click sequences. In total, this dataset contains information of about 31,790 users. We discard the sequence information and instead analyzed solely the frequencies of the visited categories. In contrast to the C&C dataset, the means of the 17 categories have all low mean values. To be more precise, the category “frontpage” has the highest mean (3.62) and the category “msn-news” has the lowest mean (0.03). We also note that the variance in the data is much lower than in the C&C-dataset.

First, we considered boosted PDNs with the empirical mean as initial model again. Since the learning curves were qualitatively identical to the C&C-experiment, we do not show them here again. Due to the low means and variance of the categories, the simple initial models themselves achieve already a good average log-score of 1.22. Still, boosting was able to improve the model with both, additive and multiplicative, updates. Additive updates achieved log-scores of 1.14 (PDN-A$^{\text{const,id}}$) and 1.08 (PDN-A$^{\text{const,log}}$), however, requiring more than 40 iterations. With less than 4 iterations on average, multiplicative updates achieved an average log-score of 1.09. Again, multiplicative updates are an order of magnitude faster, without sacrificing predictive performance. Initialization using PRTs gives a head start for the additive updates but not for the multiplicative ones. These results further confirm the answers to questions **Q7.5** and **Q7.6**.

Taking all results together, the experimental evaluation clearly shows that all six questions (**Q7.1** to **Q7.6**) can be answered affirmatively and indicate that PDNs have the potential to be a fast alternative to existing Poisson graphical models.

patch		
ID	image ID	count
1	1	2
2	1	4
3	1	0
⋮	⋮	⋮
i	2	1
⋮	⋮	⋮
k	2	?
⋮	⋮	⋮

neighbor		
ID	patch ID-1	patch ID-2
1	1	2
2	2	3
3	1	4
⋮	⋮	⋮
j	i	k
⋮	⋮	⋮

Figure 7.10: Illustration of Relational Count Databases

7.6 Relational Count Models

Often regression problems are inherently relational which asks for relational regression algorithms. As for the propositional case, not much research has been conducted on relational count models. There exist a few approaches beyond the binary case, again mostly for the Gaussian case. For example, Choi and Amir [37] introduce *Relational Continuous Models (RCMs)* where relational atoms define attributes of objects. In the ground case, RCMs amount to pairwise Gaussian random fields. Choi and Amir also propose a lifted inference algorithm for RCMs which is based on ideas from lifted Variable Elimination [180, 46]. There have also been approaches that extend MLNs to handle continuous variables. In particular, Wang and Domingos [242] present hybrid MLNs which add numeric terms to the MLN-syntax.

Relational PDNs are of great interest because many of the problems involving counts provide relational information. For example, in the cell count experiments above (Section 7.5.2), we learned one separate model for each patch. Resulting in 25 models for an image with 5×5 patches. However, one can argue that all patches behave fairly similar and can be treated as an observation from the same type with the same properties. Therefore, we outline in this section how a relational PDN approach can be realized for such count models.

This can be exemplified most easily by means of a relational database. Figure 7.10 shows two tables of a possible relational representation of the cell count images. The table "patch" corresponds to a single patch in one of the images. The table "neighbor" models the neighborhood structure of the images, e.g., the typical 4-neighbor grid. Again, it is possible that some of the counts are not observed, indicated by "?" in the "count" column. This is a fairly simple example because it contains only a single type of counts. One can also think of various cases where more than one type is present. For example, the images could show different kinds of cells. Then, we would distinguish between "count-type-A" and "count-type-B". Both possibly influencing each other by attraction or repulsion.

If we now want to learn a single model for the cell count of a patch, we need to make use of aggregators because cells have a different number of neighbors and we want to make statements on groups of neighbors. Such aggregators are typically operators defined over the neighbors of variable X_i, such as $a_{\max}(\mathrm{pa}_i)$, returning the maximum count of X_i's neighbors. Typically, aggregation is done only across variables of the same type of variables. Other aggregators are the minimum or sum over the neighboring counts. Additionally, aggregators do not necessarily

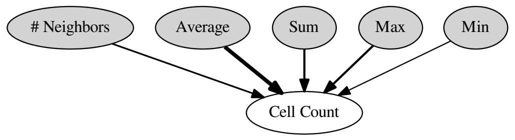


Figure 7.11: A model learned for a relational PDN. The width of the edges indicates relative importance of feature.

have to be Poisson, e.g., a Boolean aggregator like $a_{\sum>5}(\mathrm{pa}_i)$, firing when the sum over the neighbors is larger than 5. We can also learns logic rules to construct more complex features, e.g., $\exists X_j \in \mathrm{pa}_i : X_j > 5 \wedge \max(\mathrm{pa}_i) < 10$.

At learning time, we ground instances for each patch in the database. Applying the aggregators to the data results in a propositionalized dataset. Each training instance corresponds to a count of a variable and the values of the aggregators. Learning such a model results in a PDN as shown in Figure 7.11. Here, we can see that the count of a patch is influenced the most by the average of its neighbors, indicated by the edge strength. During the Gibbs sampling based inference, we have to update the aggregators associated with neighboring patches after sampling a particular variable.

However, the strength of probabilistic relational methods is to learn and reason about observations jointly. Therefore, we briefly explain how to extend the model for patches to the pairwise case, i.e., we lean models for $p(X_i, X_j)$ where X_i and X_j correspond to the counts of two patches. To represent the joint model $p(\mathbf{X})$, we now require combining rules [159], because X_i appears in several pairwise models and hence we obtain multiple conditional distributions for X_i. Natarajan et al. [159] refer to this issue as the “multiple-parent problem”.

Eventually, the goal is to prevent the full grounding and run inference on the lifted level. Therefore, lifted sampling approaches such as the ones described in Section 2.4 should be adapted to work with count data as well. Depending on the evidence and the latent patches, we are interested in finding a model that is as compact as possible. We can either adapt top-down or bottom-up approaches to find a representation that permits as much compression as possible. However, it is not immediately clear, how to adapt for example the Color Passing algorithm to an inconsistent GTB-PDN and this remains ongoing work. Generally, the grouping will depend on the conditional probabilities, i.e., the λ_i, but also has to take inconsistencies into account during the lifting procedure.

The way we learn our PDNs also influences the lifting potential. The way we group the variables induces a parameter tying and hence leads to a compact model, preparing the ground for lifting. However, one can also imagine to prune or augment the non-parametric PDNs afterwards to increase lifting capabilities. Similarly, log-linear PDNs can be regularized to obtain a structure more suitable for lifting. Additionally, it can be reasonable to cluster variables that are almost identical, i.e., leading to an approximate lifting as discussed in Section 2.4.2. For example, in the tree case, one could cut the tree at a certain level and cluster variables if they are equal up to a certain height.

7.7 Interim Conclusion

Count data is increasingly ubiquitous in Machine Learning settings. Example data are bag-of-X representations of, e.g., collections of images or text documents, genomic sequencing data, user-ratings data, spatial incidence data, climate studies, and site visits, among others. Unfortunately, standard graphical models, such as multinomial or Gaussian ones, are often ill-suited for modeling this data. We have therefore introduced PDNs, a new graphical model for multivariate count data. While its representation naturally facilitates a Gibbs sampler, this model in addition allows for a very simple training procedure: starting from a simple initial model, which in the simplest case can be a constant value or in more advanced cases a Poisson regression tree, PDNs are represented as sums, respectively products, of regression models grown in a stage-wise optimization. We have empirically demonstrated on several real-world datasets that PDNs are competitive multivariate count models, both in terms of efficiency and predictive performance.

PDNs suggest several interesting avenues for future work. In Section 7.3.4, we have already discussed how PRTs can be used for model compression. One should further investigate this idea and compare the results of the compressed models to the full model. Furthermore, one should explore PDNs within other Machine Learning tasks such as characterizing neural dependencies [18], training topic models [63], also capturing word dependencies within each topic [99, 98], and recommendation [78]. Additionally, predicting user behavior such as retention and churn [85] in mobile games or smartphone apps is an attractive avenue for future work.

In particular the latter application asks for relational count models. It has already been shown that relational approaches are well suited for modeling such kind of user behavior. For example, Dierkes et al. [49] have used MLNs to model churn and buying behavior in the mobile phone market. However, many aspects of such problems are non-binary and count models seem to be more appropriate. For example, it is of greater value to predict the precise number of future purchases of an user than solely an indicator if they are likely to buy any virtual good at all. Relational PDNs also lend themselves to approaches such as the work by Kazemi et al. [109] where relational logistic regression is used as an aggregator in relational models with varying population sizes. This also suggests to look into Poisson regression as aggregator. The problem of varying population sizes also matters in relational Poisson models. For example, we would like to build models which work for images of different sizes. It should not matter how many patches we have if they all share the same properties.

In many situations, we cannot assume that the training data is completely observed. Therefore, one should extend the learning procedure to handle partially observed datacases. This requires to run inference iteratively, and hence further asks for efficient inference algorithms beyond Gibbs sampling. In the case of MAP inference, we can add artificial temperature annealing to the Gibbs sampler in order to obtain improved MAP configurations. One should also explore message-passing-like inference for PDNs as we have seen in Chapter 3. For example, understanding how max-product BP or Likelihood Maximization looks in the case of PDNs, or generally Poisson graphical models, is appealing. Due to the infinite ranges of the variables, a compact representation of the messages has to be found. In the case of Gaussian random fields, Gaussian BP is only required to pass around values for the mean and the variance of each variable. Developing a similar approach for Poisson graphical models is left for future work. Message passing based inference also suggests to look into lifting of these algorithms. We have already touched upon this idea in Section 7.6. The local structure of PDNs should be in favor of lifting and paves the way for additional approximate lifting.

Another problem not well studied yet, is the idea of Label Propagation with labels over infinite ranges. The Label Propagation algorithm that was discussed in Chapter 5 assumed a finite set of labels. However, one can also imagine tasks where nodes receive a natural number as label. When solving such problems with Label Propagation, we need to compactly describe the distribution over the labels. Making the Poisson assumption can possibly be one step in achieving this goal.

Another interesting avenue for future work is to exploit functional gradients for learning hybrid multivariate models. Along the way, one should investigate Dependency Networks for the complete family of GLMs. For instance, Dobra [51] has shown that hybrid Dependency Networks among Gaussian and logistic variables perform well for discovering genetic networks, and Guo and Gu [80] have shown logistic (conditional) Dependency Networks to perform well for multi-label classification. Upgrading the resulting non-parametric, hybrid Dependency Networks to relational domains may provide novel structure learning approaches for BLOG [148] and probabilistic programming languages [77], which feature Poisson distributions. This would also complement relational Gaussian models [208, 37, 3] as well as relational copulas as proposed by Xiang and Neville [247] for relational collective classification. In general, copula models for multivariate count data [91] are an interesting option, in particular extending them to the relational case based on Xiang and Neville. All this could lead to better methods for mining Web-populated knowledge bases such as TextRunner, NELL, YAGO, and KnowledgeGraph. In such open information retrieval tasks, one cannot easily assume a fixed number of "entities" and in turn use existing probabilistic relational models such as MLNs.

CHAPTER 8

Conclusion

Probabilistic Graphical Models (PGMs) belong to the most important formalisms in AI and Machine Learning, with lifting being a prominent approach to reduce run time of learning and inference algorithms in recent years. This thesis contributed in different ways to the PGM and lifted inference literature. By considering graphical models and lifted inference in “non-standard” settings, we proposed approaches to lifted decimation, lifted labeling, and networks of count data, among others. By doing so, we have shown that lifting, which originated in the *Statistical Relational Learning (SRL)* community, does not only apply to probabilistic inference in relational PGMs but also decreases run time in other settings likewise. We have also seen that many applications produce count data and various observations are tracked as counts which are not handled appropriately by standard PGMs. This lead to the development of a novel class of PGMs specifically tailored towards random variables over the natural numbers. We will now summarize these contributions again in greater detail, depict the lessons learned during the work on this thesis, and present an outlook on possible lines of futures research.

8.1 Summary

After reviewing the basics of probabilistic inference and a discussion of existing lifted inference algorithms in Chapter 2, we showed in Chapter 3 how the idea of lifted message passing can be extended to message passing algorithms for SAT problems. This was achieved by lifting well known approaches to solving satisfiability problems based on message passing in factor graphs, namely Warning Propagation and Survey Propagation. Applying naive lifting to factor graphs representing CNFs indicated lifting potential but also revealed the requirement for more sophisticated lifting algorithms. This holds in particular, when the CNF is simplified based upon clamping in an iterative procedure. This resulted in novel lifting algorithms that handle incremental changes in the evidence efficiently and fit well into the concept of decimation. Decimation approaches turn a complex inference problem into a sequence of simpler tasks where we want to exploit lifting for each subtask. This general paradigm is not only suitable for propositional satisfiability, but can also be used for sampling joint configurations in MRFs. We then showed how Shortest-Path-Sequences-lifting is capable of handling incrementally growing evidence in such decimation procedures. We have shown that the resulting lifted decimation framework greatly reduced run times of SAT solving and BP-guided sampling. We have further shown that an algorithm for MAP inference based on likelihood maximization is liftable by similar means, yet again benefiting from adjusted treatment of evidence. In particular, we introduced Pseudo Evidence which can be seen as soft evidence turning into hard evidence during the course of a single run of iterative probabilistic inference. This additional evidence can be further exploited as it introduces new independencies. In a nutshell, Chapter 3 presented an unifying framework for lifted message passing algorithms with several extension when evidence is being set sequentially.

To motivate the research on non-binary PGMs along with (lifted) inference approaches not based on message passing, we described labeling problems and their probabilistic interpretation in Chapter 4. Although Color Passing and its extensions are not limited to the binary case, using classic multinomial MRFs with BP is not always practical. For example, in labeling problems, where each node in a graph has to be assigned a label from a predefined alphabet, an adapted problem formulation is better suited and additionally comes along with a simple algorithm solving the problem exactly. We explained this particular algorithm, namely Label Propagation, and described how it is solving multiclass labeling problems based on simple matrix-multiplications. This also facilitated a different view on lifting. Instead of the computational view based on Color Passing in Chapter 3, lifting was described with help of graph theoretic concepts and algebraic operations.

However, a Gaussian similarity function, which is typically used in Label Propagation, is suboptimal as it produces dense graphs which are not tractable for large-scale problems. Instead, we proposed a similarity function based on logical rules in Chapter 5 which is akin to the idea of Markov Logic Networks. This relational Label Propagation allowed us to label graphs with over five million nodes. In particular, we presented a data processing pipeline where online bibliographies are enriched with an initial set of affiliations and geo-tags. Label Propagation was then run on a graph where each node corresponded to an author-paper-pair and the labels, i.e., geo-tags, of the missing nodes were inferred. Running Label Propagation on problems of such size naturally asks for efficient implementations and any further reduction in run time is most welcome. Therefore, we also developed a lifted variant of Label Propagation where the lifting is based on graph theoretic concepts and its correctness was proven by means of algebraic operations. This novel Lifted Label Propagation does not require any modifications of the ground algorithm and amounts to standard Label Propagation on lifted matrices. Hence, implementing the lifting reduces to a preprocessing step and any efficient existing Label Propagation implementation can be used on top of it.

The advancements in Chapter 5 prepared the ground for an extensive analysis of migration behavior of researchers in computer science by compiling geo-enriched online bibliographies. Based upon GeoDBLP, and further resources from the Web, we investigated in Chapter 6 statistical regularities and patterns underlying human behavior for different social phenomena. The GeoDBLP dataset provides over 300k migration hops of roughly one million researchers in total. Fitting this data to various statistical models revealed that migration patterns are fairly regular. In particular, we found that the migration propensity follows a log-normal distribution and the overall job market in computer science academia is Poisson-log-normal. Although these are not the first results on such distributions in observational data, up to recent years, data was mostly collected in the real world based on manually prepared studies. We solely focused on data freely available on the Web and used Machine Learning techniques to fill in the missing datapoints. Besides publication and migration data, the Web provides vast amounts of data describing other human behavior and therefore asking for similar studies. By analyzing time series for Internet memes and online videos, we were also able to find descriptive mathematical models describing this data precisely in terms of distributions with only few parameters.

All datasets in Chapter 6 had in common that we were looking at count data. We counted the number of hops per year, the number of search queries for Internet memes, or the number of views of a video clip. Surprisingly, very little work has been focusing on PGMs based on random variables defined over count distributions though. With few exceptions, models with variables over infinite ranges are mostly based on Gaussian distributed random variables. Still, many problems are most accurately defined over the natural numbers, i.e., discrete, infinite ranges, and can only be squeezed into finite or Gaussian models by binning or other approximations. By introducing *Poisson Dependency Networks (PDNs)*, we have presented a new class of PGMs for count data where each variable is defined over the natural numbers. More precisely, the conditional distribution of each random variable given its neighbors in the graph is Poisson distributed. Although PDNs follow a local approach, which does not necessarily define a joint distribution in closed form, one can query the model based on Gibbs sampling which returns samples from a joint distribution. We have shown that PDNs can be learned efficiently by means of Gradient Tree Boosting and we contributed to the existing literature on gradient boosted models by introducing multiplicative updates to obtain faster convergence. Several experiments showed how PDNs can be used on various count datasets and highlighted that their performance in terms of quality is comparable while training and inference is done very efficiently.

8.2 Lessons Learned

During the research over the last years, we have encountered several challenges and we can distill some of the observations made into general lessons learned that are likely to apply beyond the cases in this thesis.

Applying and extending several different lifted inference approaches has shown that considering lifting independent of the inference algorithm in the first instance is beneficial. The universal notion of symmetries is very flexible and provides a general view on lifting which does not only cover lifted message passing. With recent work on lifted linear programs by Mladenov et al. [151] and the reparameterization of MRFs by Mladenov et al. [152], the adaption of the ground inference algorithm can be avoided elegantly. Considering lifting as a general preprocessing step, as it has been done in the case of Label Propagation, enables us to use any efficient problem solver. This also detaches the inference algorithm from the lifting procedure itself and helps to develop independent views and solutions to each problem. The knowledge of the algebraic view on lifting and coarsest equitable partitions can also present a novel view on lifted satisfiability where even more efficient approaches are thinkable. Although an abstract view on symmetries is advantageous in many cases, one should also notice that not every problem can be easily reparameterized to work with an arbitrary inference algorithm of choice.

Establishing trust in Machine Learning algorithms and applications is one of the hardest challenges even if the results seem to be self-evident. It is often difficult to convince readers of the models — even in cases where they are physically and socially plausible. This holds inside the community but becomes additionally challenging when people outside the community have to be convinced of the learned models and the impact of the results. Therefore, finding simple models that describe complex interactions well does not only follow Occam's razor and avoids overfitting, but also allows an interpretation by non-experts. Although less complex models seem to be less expressive on the first sight, this is often proven wrong if the problem at hand is taken into account. Examples of such simple models are not only the distributions shown in Chapter 6, but also the idea of small interpretable trees in Gradient Tree Boosting instead of a fully parameterized model. We have shown examples where distributions were decomposed into a growth and decline term, both giving a meaningful interpretation of the observations. Similarly, the set of small trees in Gradient Tree Boosting can be seen as a set of few general rules of thumb that describe dependencies between variables. Not only are the trees interpretable, but they also provide a good intuition. Using such simple and interpretable models will hopefully further push the advances in the fields of AI and Machine Learning because we can explain the statistical models in natural language.

8.3 Outlook

Looking at the algorithmic aspects of this thesis, and in particular at the lifting approaches, the work focused on exact lifting. However, many real-world applications do not result in perfectly symmetric problem structures. Even in the case of relational models, that tend to produce symmetries in the underlying PGM, evidence often breaks symmetries and hence cancels the benefits of lifted inference. However, the introduced algorithms could all benefit from approximate lifting. A possible decrease in performance is often negligible because approximate inference is used anyhow. This just adds another level of approximation that can even increase performance of message passing algorithms in some cases. We have seen a similar behavior by introducing Pseudo Evidence which transforms soft evidence into hard evidence without significantly harming the quality of the obtained MAP assignment. Approximate lifted inference approaches by Kersting et al. [115], as well as more recent approaches by Singla et al. [211] and Van den Broeck and Niepert [229], can replace exact Color Passing or SAUCY by approximate lifting. Alternatively, one can combine both types of lifting.

The models considered in this thesis contained only latent variables of the same type. However, many problems exist where different types of latent variables depend on each other. For example, one can easily imagine a PGM for count data where some of the variables are Poisson distributed while others are better fitted by a geometric distribution. Hence, extending Poisson Dependency Networks to something like hybrid Dependency Networks. In general, designing additional types of PGMs with random variables of mixed distributions, e.g., multinomial, Poisson, Gaussian, among others, would extend the capabilities of PGMs greatly. This is not trivial though because it requires to model dependencies between distributions of different types. For example, a Gaussian and a Poisson distribution where the means of each distribution dependent on each other. And on top of that, the variance of the Gaussian distribution has to be fitted as well, taking the other parameters into account.

Furthermore, integrating continuous or count random variables into SRL models is quite appealing. This will help to avoid artificial binning of the data or the usage of inapplicable distributions. For example, when SRL methods are applied in the medical domain, many of the features are continuous and incomplete [161]. Therefore missing values have to be inferred which requires non-binary features to be discretized which is only possible with good domain expert knowledge. Likewise, count data is currently often binned and converted to several binary predicates before being used in formalisms such as MLNs. There exist approaches such as Hybrid MLNs [242], that add Gaussian variables to MLNs, but we have sufficiently discussed in the previous chapters why one should be interested in non-Gaussian distributions as well. Hence, we should further investigate relational hybrid, or mixed, models that extend Poisson Dependency Networks to the relational setting and beyond Poisson distributions.

Ideas akin to lifting have been used in other fields as well. In Chapter 3, we already referred to techniques such as symmetry breaking for satisfiability. Additional examples of grouping variables to reduce the run time of inference can be found in computer vision [120] or natural language processing [138], among others. Several of these approaches are tailored towards specific applications and can exploit certain properties of the underlying problem structure, e.g., the lattice structure of an image or a document consisting of sentences. Although not all algorithms might be equally well applicable for any kind of problem, one should further connect them to lifted inference in SRL. Generally, SRL and PGMs are quite popular in these communities and also lifted inference has been applied in the past already. Besides the example we have seen in Section 3.3.4, online lifted inference approaches have also been applied to video streams [165] for example. Hence, this further supports the idea to investigate sequential or bootstrapped lifting approaches in combination with approximate lifting for these applications. Presenting an unifying view on lifting and symmetries in general across fields will further contribute to the understanding of lifted inference. Additionally, this can result in revised lifted inference algorithms that incorporate specialized subroutines for certain problem structures. Akin to the idea of lifted satisfiability, a notion of *lifted computer vision* or *lifted natural language processing* could be established.

We have mentioned in the introduction already that relational databases have been used predominantly for mass storage and we have discussed approaches that combined such databases with MRFs to obtain relational PGMs. However, a trend towards NoSQL databases has been observed lately in the context of Big Data. Following the DB-Engines Ranking[1], MongoDB has reached the fifth position in the database popularity ranking in January 2015. Therefore, one should also look at augmenting such databases with the notion of uncertainty and connect them to PGMs. NoSQL databases such as MongoDB, are frequently used in analytics applications tracking user behavior. On the one hand, it is possible to store millions of users events in the database without specifying precise schema. On the other hand, the retrieval process now becomes more challenging. Often, developers resort again to relational databases and build a second user database based upon aggregated information on the user events. This relational database is then used to build Machine Learning models to answer uncertainty and to make predictions. Examples of such tasks are user churn prediction or buyer classification [192, 85]. Hence, it is desirable, to integrate such concepts of uncertain properties directly into NoSQL databases. First approaches that involve statistical methods have been employed recently. For example, by offering approaches such as HyperLogLog [57] for cardinality estimation in NoSQL databases. The relationship between SQL and NoSQL databases is akin to the concepts of top-down and bottom-up lifting. Top-down lifting also assumes a schema while the bottom-up lifting tries to find symmetries in propositional data. The benefits and strength of bottom-up lifting has been shown throughout this thesis and one should further connect these ideas to the database domain.

In general, PGMs and SRL based approaches, have been used in various domains and find an increasing number of applications in different fields. SRL formalisms do not only have the benefit of being easily specified based on logic, but the resulting models are also often easy to interpret, even for non-experts. This is especially important in interdisciplinary work

[1] `db-engines.com/en/ranking`

where communication with external domain experts is inevitable and the learned model is supposed to be used by non machine learners. For example, SRL has been used in the medical domain to analyze electronic health records to predict myocardial infarctions [244, 245] or to diagnose coronary heart diseases [160]. In these cases, SRL methods have several benefits. First, being able to interpret a learned model is a key demand in order to reach helpful conclusions. Relational models based on logical formulas can often be easily reformulated into natural language text by Machine Learning experts. Second, incorporation of domain expert knowledge is possible and it would be wasteful to abstain from decades of knowledge in medical research. Instead, this information should be used to improve and guide learning algorithms. Since building electronic health records and similar databases has just begun in recent years, and is nowadays accompanied by small and medium sized businesses collecting data at massive scale, the opportunities for SRL methods and algorithms will hugely grow over the next years.

Still, the applicability of SRL methods is limited due to the complexity of the corresponding learning and inference algorithms. This thesis aimed at circumventing this drawback by presenting new efficient approaches to inference and learning in PGMs. However, much remains to be done, to increase efficiency and feasibility of applications, and to eventually bring AI and Machine Learning to all areas of our everyday life, including small devices with limited resources, such as smartphones. Efforts in this area have recently become popular, for example to save energy consumption when using MRFs on small devices [179].

Additionally, one should not neglect that SRL approaches combine methodologies from different fields which are not limited to logic and PGMs. Besides the computationally complexity, this also imposes a hurdle to enter this field. This holds in general for many algorithms and approaches in the field of AI and Machine Learning, hence, often preventing a wider usage. Therefore, the democratization of AI and Machine Learning becomes a long term vision to give a broad range of developers and engineers access to these technologies. This has to be further pursued by extending “Machine Learning in the Cloud” and by providing algorithms specialized for specific applications. In this context, declarative programming languages help to specify problems more easily and allow to apply state-of-the art techniques without knowing all the little details. Hence, the methods and techniques would be of greater use to a wider audience if the problem formulation was more straightforward. With more people applying Machine Learning, maybe even as standard subroutines, this field will observe the next boost and help AI to take the next step in reaching the goals we all are envisioning.

Appendix

1 Selected Publications

- **F. Hadiji**, A. Molina, S. Natarajan, and K. Kersting. Poisson Dependency Networks: Gradient Boosted Models for Multivariate Count Data. *Machine Learning*, 2015.

  I have contributed to the ideas and the writing of the paper. The ideas and the content have been discussed amongst all authors. Together with Alejandro Molina, I have developed the code for learning and inference in this new class of probabilistic graphical models. I also ran most of the experiments.

- **F. Hadiji**, M. Mladenov, C. Bauckhage, and K. Kersting. Computer Science on the Move: Inferring Migration Regularities from the Web via Compressed Label Propagation. *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI)*, 2015.

  I have contributed to the ideas and the writing of this paper. I developed the idea of Lifted Label Propagation and worked out the proof of its correctness together with Martin Mladenov. I have implemented most parts of the underlying code and also used Christian Bauckhage's code for fitting different probability distributions. I also ran the experiments of this publication. The statistical regularities and patterns have been discussed amongst all authors.

- C. Bauckhage, K. Kersting, and **F. Hadiji**. How Viral are Viral Videos? In *Proceedings of the 9th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2015.

  I conducted experiments on the YouTube view counts and I was involved in the discussion and revision of the paper.

- R. Sifa, **F. Hadiji**, J. Runge, A. Drachen, K. Kersting, and C. Bauckhage. Predicting purchase decisions in free to play mobile games. *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, 2015. Received best paper award.

  I was mainly responsible for casting the task as a regression problem. I implemented the Poisson regression approach, did the corresponding experiments, and did the writing on this part of the paper. I was also involved in the preparation and discussion of the paper.

- C. Ide, **F. Hadiji**, L. Habel, A. Molina, T. Zaksek, M. Schreckenberg, K. Kersting and C. Wietfeld. LTE Connectivity and Vehicular Traffic Prediction based on Machine Learning Approaches. *IEEE Vehicular Technology Conference (VTC-Fall)*, 2015.

  I was involved in the discussions leading to this paper and I also contributed to the writing. I have mainly dealt with the machine learnings aspects in the paper. The ideas and concepts were discussed amongst all authors in the SFB-876-project B4. Additionally, I was also involved in the implementation and I conducted experiments.

- **F. Hadiji**, R. Sifa, A. Drachen, C. Thurau, K. Kersting, and C. Bauckhage. Predicting Player Churn in the Wild. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, 2014.

  This paper was a collaborative effort based on the joint research of all authors. Several of the participants, including myself, were affiliated with GameAnalytics at the time the paper was written and therefore, parts of the paper were motivated upon research questions at GameAnalytics. I was involved in developing the ideas of this paper and also the writing. I wrote source code for the algorithms that were applied to the data provided by GameAnalytics. I ran the experiments together with Rafet Sifa.

- **F. Hadiji** and K. Kersting. Reduce and Re-Lift: Bootstrapped Lifted Likelihood Maximization for MAP. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, 2013.

  I was involved in developing the ideas of this paper and writing the paper. The ideas and content of the paper were discussed among both authors of the paper. I implemented the algorithms used in this paper and I also ran the experiments.

- **F. Hadiji**, K. Kersting, C. Bauckhage, and B. Ahmadi. GeoDBLP: Geo-Tagging DBLP for Mining the Sociology of Computer Science. *arXiv preprint arXiv:1304.7984*, 2013.

  I have contributed to the ideas and the writing of this paper. The ideas and the content of this paper were discussed amongst all authors. I followed up upon Babak Ahamdi's work on the rule based Label Propagation and implemented a new version based on LAMA which enables large-scale experiments. I implemented the code for harvesting the data and fitting the probability distribution to the data. The fitting code was build upon Christian Bauckhage's code. The discussion of the results and findings was done amongst all authors.

- C. Bauckhage, K. Kersting, and **F. Hadiji**. Mathematical Models of Fads Explain the Temporal Dynamics of Internet Memes. In *Proceedings of the 7th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2013.

  I was primarily involved in discussions of the paper.

- **F. Hadiji**, B. Ahmadi, and K. Kersting. Efficient Sequential Clamping for Lifted Message Passing. In *Proceedings of the 34th Annual German Conference on Artificial Intelligence (KI)*, 2011.

  I contributed to the ideas of this paper and its writing. The ideas and the content of this paper were discussed amongst all authors. I implemented the code for the lifted decimation and used Babak Ahmadi's code for SPS lifting to implement the code for lifted satisfiability. I ran the experiments for the lifted decimation.

- **F. Hadiji**, K. Kersting, and B. Ahmadi. Lifted message passing for satisfiability. In *Working Notes of the AAAI-10 Workshop on Statistical Relational AI (StarAI)*, 2010.

  I was involved in the ideas of this paper and its writing. The ideas and the content of this paper were discussed amongst all authors. I implemented the necessary source code to run the lifted message passing algorithms such as Lifted WP and Lifted SP. I also ran the experiments.

- K. Kersting, Y. El Massaoudi, B. Ahmadi, and **F. Hadiji**. Informed Lifting for Message-Passing. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

  I was involved in the development of the ideas for this paper and also participated in the writing of the paper.

- B. Ahmadi, K. Kersting, and **F. Hadiji**. Lifted Belief Propagation: Pairwise Marginals and Beyond. In *Proceedings of the 5th European Workshop on Probabilistic Graphical Models (PGM)*, 2010.

  I was involved in the development of the ideas and ongoing discussions.

References

[1] Singapore's salad days are over. *Nature*, 468:731–731, 2010.

[2] B. Ahmadi, K. Kersting, and F. Hadiji. Lifted belief propagation: Pairwise marginals and beyond. In *Proceedings of the 5th European Workshop on Probabilistic Graphical Models (PGM)*, 2010.

[3] B. Ahmadi, K. Kersting, and S. Sanner. Multi–evidence lifted message passing, with application to pagerank and the kalman filter. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1152–1158, 2011.

[4] B. Ahmadi, K. Kersting, M. Mladenov, and S. Natarajan. Exploiting symmetries for scaling loopy belief propagation and relational training. *Machine Learning*, 92(1):91–132, 2013.

[5] J. Aitchison and J. Brown. *The Lognormal Distribution.* Cambridge University Press, 1957.

[6] A. Alexandrescu and K. Kirchhoff. Graph-based learning for phonetic classification. In *Proceedings of Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 359–364, 2007.

[7] G. Allen and Z. Liu. A local poisson graphical model for inferring networks from sequencing data. *IEEE Transactions on Nanobioscience*, 12(3):189–198, 2013.

[8] F. A. Aloul, K. A. Sakallah, and I. L. Markov. Efficient symmetry breaking for boolean satisfiability. *IEEE Transactions on Computers*, 55(5):549–558, 2006.

[9] U. Apsel and R. I. Brafman. Exploiting uniform assignments in first-order MPE. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 74–83, 2012.

[10] A. Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435: 207–211, 2005.

[11] A. Barabasi, C. Song, and D. Wang. Handful of papers dominates citation. *Nature*, 491: 40, 2012.

[12] D. Battaglia, M. Kolář, and R. Zecchina. Minimizing energy below the glass thresholds. *Physical Review E*, 70:036107, 2004.

[13] C. Bauckhage. Insights into Internet memes. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 42–49, 2011.

[14] C. Bauckhage, K. Kersting, and F. Hadiji. Mathematical models of fads explain the temporal dynamics of Internet memes. In *Proceedings of the 7th International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 22–30, 2013.

[15] C. Bauckhage, K. Kersting, and F. Hadiji. How viral are viral videos? In *Proceedings of the 9th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2015.

[16] Y. Bengio, O. Delalleau, and N. Le Roux. Label propagation and quadratic criterion. In *Semi-Supervised Learning*, pages 193–216. MIT Press, 2006.

[17] Y. Bengio, É. Thibodeau-Laufer, G. Alain, and J. Yosinski. Deep generative stochastic networks trainable by backprop. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 226–234, 2014.

[18] P. Berkes, F. Wood, and J. Pillow. Characterizing neural dependencies with copula models. In *Advances in Neural Information Processing Systems 21 (NIPS)*, pages 129–136, 2009.

[19] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36:192–236, 1974.

[20] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, 1975.

[21] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[22] J. Bohorquez, S. Gourley, A. Dixon, M. Spagat, and N. Johnson. Common ecology quantifies human insurgency. *Nature*, 462:911–914, 2009.

[23] B. M. Bolker, M. E. Brooks, C. J. Clark, S. W. Geange, J. R. Poulsen, M. H. H. Stevens, and J.-S. S. White. Generalized linear mixed models: a practical guide for ecology and evolution. *Trends in Ecology & Evolution*, 24:127–135, 2009.

[24] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–655, 1998.

[25] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(11): 1222–1239, 2001.

[26] A. Braunstein and R. Zecchina. Survey propagation as local equilibrium equations. *Journal of Statistical Mechanics: Theory and Experiment*, P06007:812–815, 2004.

[27] A. Braunstein, M. Mzard, and R. Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures and Algorithms*, 27(2):201–226, 2005.

[28] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Wadsworth, 1984.

[29] C. Bucila, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACMSIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 535–541, 2006.

[30] H. Bui, T. N. Huynh, and S. Riedel. Automorphism groups of graphical models and lifted variational inference. In *Proceedings of the 29th Annual Conference on Uncertainty in AI (UAI)*, pages 132–141, 2013.

[31] H. B. Bui, T. N. Huynh, and R. de Salvo Braz. Exact lifted inference with distinct soft evidence on every object. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI)*, pages 1875–1881, 2012.

[32] M. Burke, L. Adamic, and K. Marciniak. Families on facebook. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, pages 41–50, 2013.

[33] P. Chaudhuri, W.-D. Lo, W.-Y. Loh, and C.-C. Yang. Generalized regression trees. *Statistica Sinica*, 5:641–666, 1995.

[34] M. Chavira and A. Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6):772–799, 2008.

[35] Y. Chen, D. Pavlov, and J. F. Canny. Large-scale behavioral targeting. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (CIKM)*, pages 209–218, 2009.

[36] H. Chieu and W. Lee. Relaxed survey propagation for the weighted maximum satisfiability problem. *Journal of Artificial Intelligence Research (JAIR)*, 36:229–266, 2009.

[37] J. Choi and E. Amir. Lifted inference for relational continuous models. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 126–134, 2010.

[38] J. Choi and E. Amir. Lifted relational variational inference. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 196–206, 2012.

[39] R. D. Clarke. An application of the poisson distribution. *Journal of the Institute of Actuaries*, 72, 1946.

[40] J. Cohen, M. Roig, D. Reuman, and C. GoGwilt. International migration beyond gravity: A statistical model for use in population projections. *PNAS*, 105(40):15269–15274, 2008.

[41] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[42] J. M. Crawford. A theoretical analysis of reasoning by symmetry in first-order logic. In *AAAI Workshop on Tractable Reasoning*, 1992.

[43] J. M. Crawford and L. D. Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81(1):31–57, 1996.

[44] L. De Raedt. *Logical and relational learning.* Cognitive Technologies. Springer, 2008.

[45] L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26(2-3):99–146, 1997.

[46] R. de Salvo Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1319–1325, 2005.

[47] R. de Salvo Braz, S. Natarajan, H. Bui, J. Shavlik, and S. Russell. Anytime lifted belief propagation. In *Workshop on Statistical Relational Learning (SRL)*, 2009.

[48] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[49] T. Dierkes, M. Bichler, and R. Krishnan. Estimating the effect of word of mouth on churn and cross-buying in the mobile phone market with Markov logic networks. *Decision Support Systems*, 51(3):361–371, 2011.

[50] T. G. Dietterich, G. Hao, and A. Ashenfelter. Gradient tree boosting for training conditional random fields. *Journal of Machine Learning Research (JMLR)*, 9(10):2113–2139, 2008.

[51] A. Dobra. Variable selection and dependency networks for genomewide data. *Biostatistics*, 19(4):621–639, 2009.

[52] J. Duchi, D. Tarlow, G. Elidan, and D. Koller. Using combinatorial optimization within max-product belief propagation. In *Advances in Neural Information Processing Systems 19 (NIPS)*, pages 369–376, 2007.

[53] F. Eaton and Z. Ghahramani. Choosing a variable to clamp. In *International Conference on Artificial Intelligence and Statistics*, pages 145–152, 2009.

[54] J. Elith, J. Leathwick, and T. Hastie. A working guide to boosted regression trees. *Journal of Animal Ecology*, 77(4):802–813, 2008.

[55] T. J. Espenshade and G. Rodriguez. Completing the Ph. D.: Comparative performances of US and foreign students. *Social Science Quarterly*, 78(2):593–605, 1997.

[56] W. Feller. *An Introduction to Probability Theory and Its Applications.* Wiley, 1968.

[57] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Proceedings of the Conference on Analysis of Algorithms (AofA)*, pages 127–146, 2007.

[58] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[59] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1300–1309, 1999.

[60] Y. Fujiwara and G. Irie. Efficient label propagation. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 784–792, 2014.

[61] X. Gabaix, G. Parameswaran, V. Plerou, and H. Stanley. A theory of power law distributions in financial market fluctuations. *Nature*, 423:267–270, 2003.

[62] R. Garnett, Y. Krishnamurthy, X. Xiong, J. Schneider, and R. Mann. Bayesian optimal active search and surveying. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1239–1246, 2012.

[63] P. Gehler, A. Holub, and M. Welling. The rate adapting Poisson model for information retrieval and object recognition. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 337–344, 2006.

[64] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 6(6):721–741, 1984.

[65] German Council of Science and Humanities. Recommendations on German science policy in the European research area. Technical Report Drs. 9866-10, Berlin 02 07 2010, 2010.

[66] L. Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning.* The MIT Press, 2007.

[67] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Relational data mining*, pages 307–335. Springer, 2001.

[68] M. Ghitany, D. Karlis, D. Al-Mutairi, and F. Al-Awadhi. An EM algorithm for multivariate poisson regression models and its application. *Applied Mathematical Sciences*, 6(137): 6843–6856, 2012.

[69] R. Gibrat. Une loi des repartitions economiques: Leffet proportionelle. *Bulletin de Statistique General*, 19:469–514, 1930.

[70] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 553–560, 2008.

[71] C. Godsil and G. Royle. *Algebraic Graph Theory.* Springer Verlag, 2001.

[72] M. Goetz, J. Leskovec, M. McGlohon, and C. Faloutsos. Modeling blog dynamics. In *Proceedings of the Third International Conference on Weblogs and Social Media (ICWSM)*, pages 26–33, 2009.

[73] V. Gogate and P. Domingos. Probabilistic theorem proving. In *Proceedings of the 27th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 256–265, 2011.

[74] V. Gogate, A. K. Jha, and D. Venugopal. Advances in lifted importance sampling. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1910–1916, 2012.

[75] C. P. Gomes, J. Hoffmann, A. Sabharwal, and B. Selman. From sampling to model counting. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2293–2299, 2007.

[76] M. Gonzales, C. Hidalgo, and A. Barabasi. Understanding individual human mobility patterns. *Nature*, 453:779–782, 2008.

[77] N. Goodman. The principles and practice of probabilistic programming. In *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 399–402, 2013.

[78] P. Gopalan, L. Charlin, and D. Blei. Content-based recommendations with Poisson factorization. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 3176–3184, 2014.

[79] M. Grohe, K. Kersting, M. Mladenov, and E. Selman. Dimension reduction via colour refinement. In *Algorithms - ESA 2014*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 505–516, 2014.

[80] Y. Guo and S. Gu. Multi-label classification using conditional dependency networks. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2113–2139, 2011.

[81] F. Hadiji and K. Kersting. Reduce and re-lift: Bootstrapped lifted likelihood maximization for MAP. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, pages 394–400, 2013.

[82] F. Hadiji, K. Kersting, and B. Ahmadi. Lifted message passing for satisfiability. In *AAAI Workshop on Statistical Relational AI (StarAI)*, 2010.

[83] F. Hadiji, B. Ahmadi, and K. Kersting. Efficient sequential clamping for lifted message passing. In *Proceedings of the 34th Annual German Conference on AI (KI)*, 2011.

[84] F. Hadiji, K. Kersting, C. Bauckhage, and B. Ahmadi. GeoDBLP: Geo-tagging DBLP for mining the sociology of computer science. *arXiv preprint arXiv:1304.7984*, 2013.

[85] F. Hadiji, R. Sifa, A. Drachen, C. Thurau, K. Kersting, and C. Bauckhage. Predicting player churn in the wild. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, pages 131–139, 2014.

[86] F. Hadiji, M. Mladenov, C. Bauckhage, and K. Kersting. Computer science on the move: Inferring migration regularities from the web via compressed label propagation. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI)*, pages 171–177, 2015.

[87] F. Hadiji, A. Molina, S. Natarajan, and K. Kersting. Poisson dependency networks. *Machine Learning*, 100(2-3):477–507, 2015.

[88] G. Hahn and G. Sabidussi. *Graph Symmetry: Algebraic Methods and Applications*. Springer, 1997.

[89] J. M. Hammersley and P. Clifford. Markov field on finite graphs and lattices. 1971.

[90] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for density estimation, collaborative filtering, and data visualization. *Journal of Machine Learning Research (JMLR)*, 1:49–76, 2000.

[91] E. Hee Lee. *Copula Analysis of Correlated Counts*, chapter 11, pages 325–348. Emerals Group Publishing, 2014.

[92] C. Hidalgo and C. Rodriguez-Sickert. The dynamics of a mobile phone network. *Physica A*, 287(12):3017–3024, 2008.

[93] P. Hoff. Random effects models for network data. In *Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers*, 2003.

[94] P. Hui, R. Mortier, M. Piorkowski, T. Henderson, and J. Crowcoft. Planet-scale human mobility measurement. In *Proceedings of the 2nd ACM International Workshop on Hot Topics on Planet-Scale Measurements*, 2010.

[95] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research (JAIR)*, 36:267–306, 2009.

[96] C. Ide, F. Hadiji, L. Habel, A. Molina, T. Zaksek, M. Schreckenberg, K. Kersting, and C. Wietfeld. LTE connectivity and vehicular traffic prediction based on machine learning approaches. In *IEEE Vehicular Technology Conference (VTC-Fall)*, pages 1–5, 2015.

[97] A. Ihler, J. Fisher III, and A. Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research (JMLR)*, 6:905–936, 2005.

[98] D. Inouye, P. Ravikumar, and I. Dhillon. Admixture of Poisson MRFs: A topic model with word dependencies. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 226–234, 2014.

[99] D. Inouye, P. Ravikumar, and I. Dhillon. Capturing semantically meaningful word dependencies with an admixture of Poisson MRFs. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 3158–3166, 2014.

[100] M. Jaeger. Relational Bayesian networks. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 266–273, 1997.

[101] A. Jaimovich, O. Meshi, and N. Friedman. Template based inference in symmetric relational Markov random fields. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 191–199, 2007.

[102] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 259–266, 2002.

[103] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 593–598, 2004.

[104] L. Jiang, Y. Miao, Y. Yang, Z. Lan, and A. G. Hauptmann. Viral video style: a closer look at viral videos on YouTube. In *Proceedings of International Conference on Multimedia Retrieval (ICMR)*, pages 193–200, 2014.

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

[106] M. S. Kaiser and N. Cressie. Modeling Poisson variables with positive spatial dependence. *Statistics & Probability Letters*, 35(4):423–432, 1997.

[107] D. Karlis and I. Ntzoufras. Analysis of sports data by using bivariate Poisson models. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 52(3):381–393, 2003.

[108] H. Katebi, K. A. Sakallah, and I. L. Markov. Graph symmetry detection and canonical labeling: Differences and synergies. In *Turing-100*, pages 181–195, 2012.

[109] S. M. Kazemi, D. Buchman, K. Kersting, S. Natarajan, and D. Poole. Relational logistic regression. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2014.

[110] K. Kersting. Lifted probabilistic inference. In *Proceedings of 20th European Conference on Artificial Intelligence (ECAI)*, pages 33–38, 2012.

[111] K. Kersting and L. De Raedt. Towards combining inductive logic programming with Bayesian networks. In *International Conference on Inductive Logic Programming (ILP)*, pages 118–131, 2001.

[112] K. Kersting and K. Driessens. Non-parametric policy gradients: a unified treatment of propositional and relational domains. In *Proceedings of the 25th International Conference on Machine learning (ICML)*, pages 456–463, 2008.

[113] K. Kersting, L. De Raedt, and S. Kramer. Interpreting Bayesian logic programs. In *Proceedings of the AAAI Workshop on Learning Statistical Models from Relational Data*, pages 29–35, 2000.

[114] K. Kersting, B. Ahmadi, and S. Natarajan. Counting belief propagation. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 277–284, 2009.

[115] K. Kersting, Y. E. Massaoudi, B. Ahmadi, and F. Hadiji. Informed lifting for message–passing. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1181–1186, 2010.

[116] K. Kersting, S. Natarajan, and D. Poole. Statistical relational AI: Logic, probability and computation. 2011.

[117] K. Kersting, M. Mladenov, R. Garnett, and M. Grohe. Power iterated color refinement. In *Proceedings of th 28th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1904–1910, 2014.

[118] H. Khosravi, O. Schulte, J. Hu, and T. Gao. Learning compact Markov logic networks with decision trees. *Machine learning*, 89(3):257–277, 2012.

[119] T. Khot, S. Natarajan, K. Kersting, and J. Shavlik. Learning Markov logic networks via functional gradient boosting. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM)*, pages 320–329, 2011.

[120] T. Kim, S. Nowozin, P. Kohli, and C. D. Yoo. Variable grouping for energy minimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1913–1920, 2011.

[121] A. Kimmig, L. Mihalkova, and L. Getoor. Lifted graphical models: A survey. *Machine Learning*, 99(1):1–45, 2014.

[122] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[123] S. Kok and P. Domingos. Learning the structure of Markov logic networks. In *Proceedings of the 22nd International Conference on Machine learning (ICML)*, pages 441–448, 2005.

[124] D. Koller and N. Friedman. *Probabilistic Graphical Models*. The MIT Press, 2009.

[125] L. Kroc, A. Sabharwal, and B. Selman. Survey propagation revisited. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 217–226, 2007.

[126] L. Kroc, A. Sabharwal, and B. Selman. Messagepassing and local heuristics as decimation strategies for satisfiability. In *Proceedings of the ACM Symposium on Applied Computing (SAC)*, pages 1408–1414, 2009.

[127] L. Kroc, A. Sabharwal, and B. Selman. Leveraging belief propagation, backtrack search, and statistics for model counting. *Annals of Operations Research*, 184(1):209–231, 2011.

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

[129] A. Kumar and S. Zilberstein. MAP estimation for graphical models by likelihood maximization. In *Advances in Neural Information Processing Systems 23 (NIPS)*, pages 1180–1188, 2010.

[130] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

[131] W. Lam. China's brain drain dilemma: Elite emigration. *The Jamestown Foundation: China Brief Volume*, 10(16), 2010.

[132] D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13 (NIPS)*, pages 556–562, 2001.

[133] R. Lee. The outlook for population growth. *Science*, 333(6042):569–573, 2011.

[134] A. Lehmussola, P. Ruusuvuori, J. Selinummi, H. Huttunen, and O. Yli-Harja. Computational framework for simulating fluorescence microscope images with cell populations. *IEEE Transactions on Medical Imaging*, 26(7):1010–1016, 2007.

[135] J. Leskovec and E. Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceedings of the 17th International World Wide Web Conference (WWW)*, pages 915–924, 2008.

[136] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceeding of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 497–506, 2009.

[137] M. Ley. DBLP: some lessons learned. In *Proceedings of VLDB Endowment*, pages 1493–1500, 2009.

[138] P. Liang, M. I. Jordan, and D. Klein. Type-based MCMC. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 573–581, 2010.

[139] E. Limpert, W. Stahel, and M. Abbt. Log-normal distributions across the sciences: Keys and clues. *BioScience*, 51(5):341–352, 2001.

[140] D. Lowd and J. Davis. Improving Markov network structure learning using decision trees. *Journal of Machine Learning Research (JMLR)*, 15(1):501–532, 2014.

[141] E. Maneva, E. Mossel, and M. Wainwright. A new look at survey propagation and its generalizations. *Journal of the ACM (JACM)*, 54:2–41, 2007.

[142] R. Mantegna and H. Stanley. Scaling behaviour in the dynamics of an economic index. *Nature*, 376:46–49, 1995.

[143] P. McCullagh and J. Nelder. *Generalized Linear Models.* Chapman and Hall, 1989.

[144] R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530, 2005.

[145] N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3):1436–1462, 2006.

[146] M. Mézard and A. Montanari. *Information, Physics, and Computation.* Oxford University Press, 2009.

[147] M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297:812–815, 2002.

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

[149] B. Milch, L. Zettlemoyer, K. Kersting, M. Haimes, and L. P. Kaelbling. Lifted probabilistic inference with counting formulas. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1062–1068, 2008.

[150] T. P. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 362–369, 2001.

[151] M. Mladenov, B. Ahmadi, and K. Kersting. Lifted linear programming. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 788–797, 2012.

[152] M. Mladenov, A. Globerson, and K. Kersting. Lifted message passing as reparametrization of graphical models. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 603–612, 2014.

[153] A. Montanari, F. Ricci-Tersenghi, and G. Semerjian. Solving constraint satisfaction problems through belief propagation-guided decimation. In *Proceedings of the 45th Allerton Conference on Communications, Control and Computing*, pages 352–359, 2007.

[154] J. M. Mooij. *Understanding and Improving Belief Propagation.* PhD thesis, Radboud University Nijmegen, 2008.

[155] J. M. 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.

[156] K. Morik and J. Kietz. A bootstrapping approach to concept clustering. In *Proceedings of the 6th International Workshop on Machine Learning (ML)*, pages 503–504, 1989.

[157] K. Morik and H. Köpcke. Analysing customer churn in insurance data: A case study. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 325–336, 2004.

[158] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 467–475, 1999.

[159] S. Natarajan, P. Tadepalli, T. G. Dietterich, and A. Fern. Learning first-order probabilistic models with combining rules. *Annals of Mathematics and Artificial Intelligence*, 54(1-3): 223–256, 2009.

[160] S. Natarajan, T. Khot, K. Kersting, B. Gutmann, and J. Shavlik. Gradient–based boosting for statistical relational learning: The relational dependency network case. *Machine Learning*, 86(1):25–56, 2012.

[161] S. Natarajan, K. Kersting, E. Ip, D. R. Jacobs, and J. Carr. Early prediction of coronary artery calcification levels using machine learning. In *Proceedings of the 25th Innovative Applications of Artificial Intelligence Conference (IAAI)*, pages 1557–1562, 2013.

[162] S. Natarajan, K. Kersting, T. Khot, and J. Shavlik. *Boosted Statistical Relational Learners: From Benchmarks to Data-Driven Medicine.* SpringerBriefs in Computer Science. Springer, 2014.

[163] S. Natarajan, J. M. P. Leiva, T. Khot, K. Kersting, C. Re, and J. Shavlik. Effectively creating weakly labeled training examples via approximate domain knowledge. In *Proceedings of the International Conference on Inductive Logic Programming (ILP)*, pages 92–107, 2014.

[164] S. Natarajan, B. N. Saha, S. Joshi, A. Edwards, E. Moody, T. Khot, K. Kersting, C. T. Whitlow, and J. A. Maldjian. Relational learning helps in three-way classification of Alzheimer patients from structural magnetic resonance images of the brain. *International Journal of Machine Learning and Cybernetics (IJMLC)*, 5(5):659–669, 2014.

[165] A. Nath and P. Domingos. Efficient lifting for online probabilistic inference. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1193–1198, 2010.

[166] M. Neumann, B. Ahmadi, and K. Kersting. Markov logic sets: Towards lifted information retrieval using PageRank and label propagation. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI)*, pages 447–452, 2011.

[167] J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research (JMLR)*, 8:653–692, 2007.

[168] J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 625–630, 2003.

[169] J. Neville, D. Jensen, and B. Gallagher. Simple estimators for relational Bayesian classifiers. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM)*, pages 609–612, 2003.

[170] M. Niepert. Lifted probabilistic inference: An MCMC perspective. In *Proceedings of the 2nd International Workshop on Statistical Relational AI (StarAI)*, 2012.

[171] M. Niepert. Markov chains on orbits of permutation groups. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 624–633, 2012.

[172] J. Noessner, M. Niepert, and H. Stuckenschmidt. Rockit: Exploiting parallelism and symmetry for MAP inference in statistical relational models. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, pages 739–745, 2013.

[173] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil. An empirical study of geographic user activity patterns in Foursquare. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 570–573, 2011.

[174] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999.

[175] G. Papandreou and A. L. Yuille. Perturb-and-MAP random fields: Using discrete optimization to learn and sample from energy models. In *IEEE International Conference on Computer Vision (ICCV)*, pages 193–200, 2011.

[176] J. Park, V. Barash, C. Fink, and M. Cha. Emoticon style: Interpreting differences in emoticons across cultures. In *Proceedings of the 7th International Conference on Weblogs and Social Media (ICWSM)*, pages 466–475, 2013.

[177] J. Pearl. *Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 2. edition, 1991.

[178] N. Piatkowski, S. Lee, and K. Morik. Spatio-temporal random fields: compressible representation and distributed estimation. *Machine learning*, 93(1):115–139, 2013.

[179] N. Piatkowski, L. Sangkyun, and K. Morik. The integer approximation of undirected graphical models. In *International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, pages 296–304, 2014.

[180] D. Poole. First-order probabilistic inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 985–991, 2003.

[181] H. Poon and P. Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 458–463, 2006.

[182] H. Poon, P. Domingos, and M. Summer. A general method for reducing the complexity of relational inference and its application to MCMC. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1075–1080, 2008.

[183] R. B. Potts. Some generalized order-disorder transformations. In *Mathematical Proceedings of the Cambridge Philosophical Society*, pages 106–109, 1952.

[184] D. Radev, P. Muthukrishnan, and V. Qazvinian. The ACL anthology network corpus. In *ACL Workshop on Natural Language Processing and Information Retrieval for Digital Libraries*, 2009.

[185] P. Ravikumar, M. J. Wainwright, and J. D. Lafferty. High-dimensional Ising model selection using a L1-regularized logistic regression. *The Annals of Statistics*, 38(3): 1287–1936, 2010.

[186] P. Rees, J. Stillwell, P. Boden, and A. Dennett. A review of migration statistics literature. In *UK Statistics Authority, Migration Statistics: the Way Ahead, Report 4, Part 2*. UK Statistics Authority, London, 2009.

[187] M. Regets. Research issues in the international migration of highly skilled workers: A perspective with data from the United States. Technical Report SRS 07-203, Division of Science Resources Statistics, National Science Foundation, 2010.

[188] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2): 107–136, 2006.

[189] G. Ridgeway. Generalized boosted models: A guide to the GBM package. *R vignette*, 2006.

[190] S. Riedel. Improving the accuracy and efficiency of MAP inference for Markov logic. In *Proceedings of the 24th Annual Conference on Uncertainty in AI (UAI)*, pages 468–475, 2008.

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

[192] J. Runge, P. Gao, F. Garcin, and B. Faltings. Churn prediction for high-value players in casual social games. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, 2014.

[193] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009.

[194] S. Sarkhel, D. Venugopal, P. Singla, and V. G. Gogate. An integer polynomial programming based framework for lifted MAP inference. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 3302–3310, 2014.

[195] L. Saul and D. Lee. Multiplicative updates for classification by mixture models. In *Advances in Neural Information Processing Systems 14 (NIPS)*, pages 897–904, 2001.

[196] B. Selman, H. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 521–532, 1995.

[197] P. Sen, A. Deshpande, and L. Getoor. Exploiting shared correlations in probabilistic databases. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 809–820, 2008.

[198] P. Sen, A. Deshpande, and L. Getoor. Bisimulation-based approximate lifted inference. In *Proceedings of the 25th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 496–505, 2009.

[199] F. Sha, L. K. Saul, and D. D. Lee. Multiplicative updates for large margin classifiers. In *Proceedings of the 16th Annual Conference on Computational Learning Theory (COLT)*, pages 188–202, 2003.

[200] J. W. Shavlik and S. Natarajan. Speeding up inference in Markov logic networks by preprocessing to reduce the size of the resulting grounded network. In *Proceedings of the 21st International Joint Conference in Artificial Intelligence (IJCAI)*, pages 1951–1956, 2009.

[201] D. Sheldon, T. Sun, A. Kumar, and T. G. Dietterich. Approximate inference in collective graphical models. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1004–1012, 2013.

[202] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt:. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research (JMLR)*, 12: 2539–2561, 2011.

[203] S. E. Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68 (2):399–410, 1994.

[204] R. Sifa, F. Hadiji, J. Runge, A. Drachen, K. Kersting, and C. Bauckhage. Predicting purchase decisions in free to play mobile games. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, pages 79–85, 2015. Best Paper Award.

[205] F. Simini, M. Gonzalez, A. Maritan, and A. Barabasi. A universal model for mobility and migration patterns. *Nature*, 484:96–100, 2012.

[206] P. Singla and P. Domingos. Entity resolution with Markov logic. In *Proceedings of the 6th International Conference on Data Mining (ICDM)*, pages 572–582, 2006.

[207] P. Singla and P. Domingos. Memory-efficient inference in relational domains. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 488–493, 2006.

[208] P. Singla and P. Domingos. Markov logic in infinite domains. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 368–375, 2007.

[209] P. Singla and P. Domingos. Lifted first-order belief propagation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1094–1099, 2008.

[210] P. Singla, A. Nath, and P. Domingos. Approximate lifted belief propagation. In *AAAI Workshop on Statistical Relational AI (StarAI)*, 2010.

[211] P. Singla, A. Nath, and P. Domingos. Approximate lifting techniques for belief propagation. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*, pages 2497–2504, 2014.

[212] D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*, pages 219–254. MIT Press, 2011.

[213] B. State, I. Weber, and E. Zagheni. Studying inter-national mobility through IP geolocation. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 265–274, 2013.

[214] J. Stewart. The Poisson-lognormal model for bibliometric/scientometric distributions. *Information Processing and Management*, 30(2):239–251, 1994.

[215] J. Stillwell. Inter-regional migration modelling: A review and assessment. In *Proceedings of the 45th Congress of the European Regional Science Association*, 2005.

[216] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases.* Morgan & Claypool Publishers, 2011.

[217] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning.* MIT Press, 2007.

[218] C. Sutton and A. McCallum. Piecewise training for structured prediction. *Machine learning*, 77(2-3):165–194, 2009.

[219] N. Taghipour, D. Fierens, J. Davis, and H. Blockeel. Lifted variable elimination: Decoupling the operators from the constraint language. *Journal of Artificial Intelligence Research (JAIR)*, 47:393–439, 2013.

[220] M. F. Tappen and W. T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV)*, pages 900–906, 2003.

[221] B. Taskar, A. Pieter, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 485–49, 2002.

[222] T. M. Therneau, B. Atkinson, and B. Ripley. *rpart: Recursive Partitioning*, 2011.

[223] M. Toussaint, L. Charlin, and P. Poupart. Hierarchical POMDP controller optimization by likelihood maximization. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 562–570, 2008.

[224] P. Tsiamyrtzis and D. Karlis. Strategies for efficient computation of multivariate Poisson probabilities. *Communications in Statistics, Simulation and Computation*, 33:271–292, 2004.

[225] J. Ugander and L. Backstrom. Balanced label propagation for partitioning massive graphs. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 507–516, 2013.

[226] UN System Task Team on the Post-2015 UN Development Agenda. Migration and human mobility. Technical report, 2012.

[227] G. Van den Broeck and A. Darwiche. On the complexity and approximation of binary evidence in lifted inference. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 2868–2876, 2013.

[228] G. Van den Broeck and J. Davis. Conditioning in first-order knowledge compilation and lifted probabilistic inference. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1961–1967, 2012.

[229] G. Van den Broeck and M. Niepert. Lifted probabilistic inference for asymmetric graphical models. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

[230] G. Van den Broeck, N. Taghipour, W. Meert, J. Davis, and L. De Raedt. Lifted probabilistic inference by first-order knowledge compilation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2178–2185, 2011.

[231] G. Van den Broeck, A. Choi, and A. Darwiche. Lifted relax, compensate and then recover: From approximate to exact lifted probabilistic inference. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 131–141, 2012.

[232] G. Van den Broeck, W. Meert, and J. Davis. Lifted generative parameter learning. In *AAAI Workshop on Statistical Relational AI (StarAI)*, 2013.

[233] J. Van Haaren, G. van den Broeck, W. Meert, and J. Davis. Tractable learning of liftable Markov logic networks. In *ICML Workshop on Learning Tractable Probabilistic Models*, 2014.

[234] D. Venugopal and V. Gogate. On lifting the Gibbs sampling algorithm. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 1655–1663, 2012.

[235] D. Venugopal and V. Gogate. Evidence-based clustering for scalable inference in Markov logic. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases ECMLPKDD*, pages 258–273, 2014.

[236] D. Venugopal and V. G. Gogate. Scaling-up importance sampling for Markov logic networks. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 2978–2986, 2014.

[237] 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.

[238] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. In *Workshop on Artificial Intelligence and Statistics*, 2003.

[239] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. MAP estimation via agreement on (hyper)trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2005.

[240] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A. Barabasi. Human mobility, social ties, and link prediction. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1100–1108, 2011.

[241] F. Wang, X. Wang, and T. Li. Efficient label propagation for interactive image segmentation. In *Proceedings of the 6th International Conference on Machine Learning and Applications (ICMLA)*, pages 136–141, 2007.

[242] J. Wang and P. Domingos. Hybrid Markov logic networks. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1106–1111, 2008.

[243] W. Wei, J. Erenrich, and B. Selman. Towards efficient sampling: Exploiting random walk strategies. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI)*, pages 670–676, 2004.

[244] J. Weiss, S. Natarajan, P. Peissig, C. McCarty, and D. Page. Statistical relational learning to predict primary myocardial infarction from electronic health records. In *Proceedings of the 24th Annual Conference on Innovative Applications of Artificial Intelligence (IAAI)*, pages 2341–2347, 2012.

[245] J. C. Weiss, S. Natarajan, P. L. Peissig, C. A. McCarty, and D. Page. Machine learning for personalized medicine: predicting primary myocardial infarction from electronic health records. *AI Magazine*, 33(4):33–45, 2012.

[246] Y. Weiss and W. T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural computation*, 13(10):2173–2200, 2001.

[247] R. Xiang and J. Neville. Collective inference for network data with copula latent Markov networks. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 647–656, 2013.

[248] E. Yang, P. Ravikumar, G. Allen, and Z. Liu. Graphical models via generalized linear models. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 1367–1375, 2012.

[249] E. Yang, P. Ravikumar, G. I. Allen, and Z. Liu. On Poisson graphical models. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 1718–1726, 2013.

[250] Z. Yang and J. Laaksonen. Multiplicative updates for non-negative projections. *Neurocomputing*, 71(1-3):363–373, 2007.

[251] C. Yanover, T. Meltzer, Y. Weiss, P. Bennett, and E. Parrado-Hernndez. Linear programming relaxations and belief propagation: An empirical study. *Journal of Machine Learning Research (JMLR)*, 7:1887–1907, 2006.

[252] J. Yarkony, C. Fowlkes, and A. Ihler. Covering trees and lower bounds on quadratic assignment. In *Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[253] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems 13 (NIPS)*, pages 689–695, 2001.

[254] J. S. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51 (7):2282–2312, 2005.

[255] E. Zagheni and I. Weber. You are where you e-mail: using e-mail data to estimate international migration rates. In *Proceedings of the ACM Conference on Web Science (WebSci)*, pages 348–351, 2012.

[256] H. Zhang and M. E. Stickel. An efficient algorithm for unit propagation. In *Proceedings of the 4th International Symposium on Artificial Intelligence and Mathematics (AI-MATH)*, pages 166–169, 1996.

[257] N. L. Zhang and D. Poole. A simple approach to Bayesian network computations. In *Proceedings of the 10th Biennial Canadian Artificial Intelligence Conference*, 1994.

[258] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16 (NIPS)*, pages 321–328, 2004.

[259] X. Zhu. *Semi-Supervised Learning with Graphs.* PhD thesis, School of Computer Science, Carnegie Mellon University, 2005.

[260] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

[261] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 912–919, 2003.

[262] G. Zipf. The p_1p_2/d hypothesis: On the inter-city movement of persons. *American Sociological Review*, 11(6):677–686, 1946.

Index

Nomenclature

E	Set of edges in a graph G
G	Graph
V	Set of nodes in a graph G
X_i	Random variable
$\mathcal{E}$	Energy function
$\mathbb{E}[\cdot]$	Expected value
$\mathfrak{G}$	Lifted (factor) graph
$\mathfrak{X}(X_i)$	Function returning color of X_i
$\mathfrak{X}_i$	Clusternode
$\mathfrak{f}(f_a)$	Function returning color of f_a
$\mathfrak{f}_a$	Clusterfactor
$\mathbf{X}$	Vector of random variables
$\mathbf{x}$	Instantiation of a vector of random variables
$\mathcal{X}_i$	Range of a random variable
$\mu_{a\to i,p}(\mathfrak{x}_i)$	Lifted message from factor to variable
$\mu_{i\to a,p}(\mathfrak{x}_i)$	Lifted message from variable to factor
d	Number of states of a random variable
x_i	Instantiation of a random variable
η	Step size, i.e., learning rate
fc	Factor count

$\mathbf{X}_{\setminus i,j}$	$\mathbf{X} \setminus \{X_i, X_j\}$
$\mathbf{x}_{\setminus i}$	$\mathbf{x} \setminus \{x_i\}$
$\mu_{a \to i}(x_i)$	Message from factor to variable
$\mu_{i \to a}(x_i)$	Message from variable to factor
$\mathrm{nb}(\cdot)$	Neighborhood function returning neighbors of a node in a graph
vc	Variable count
$b(x_i)$	Believe of a variable X_i
I^2	Relative importance
l	Number of labeled nodes
s_k^i	k-th move propensity of author i
s_i	State of a random variable
T	Number of gradient steps
t_i^j	Migration propensity of author i
u	Number of unlabeled nodes
w_a	Weight of a formula

List of Abbreviations

AI	Artificial Intelligence
BN	Bayesian network
BLM	Bootstrapped LM
BLLM	Bootstrapped Lifted LM
BP	Belief Propagation
CCT	colored computation tree
CEP	coarsest equitable partition
CNF	conjunctive normal form
CP	Color Passing
CT	computation tree
DMLN	Dynamic MLN
DN	Dependency Network
EACP	Evidence Aware Color Passing
EM	Expectation Maximization
FOL	first-order logic
GaBP	Gaussian BP
GLM	Generalized Linear Model
GLMM	Generalized Linear Mixed Model
GTB	Gradient Tree Boosting
iLBP	informed Lifted Belief Propagation
JT	Junction Tree

LEACP	Lifted EACP
LBP	Lifted Belief Propagation
LSP	Lifted Survey Propagation
LWP	Lifted Warning Propagation
LL	Log-Likelihood
LLM	Lifted Likelihood Maximization
LLP	Lifted Label Propagation
LM	Likelihood Maximization
LP	Label Propagation
LPGM	Local Poisson Graphical Models
MAP	maximum a posteriori
ML	Machine Learning
MLE	maximum likelihood estimation
MLN	Markov Logic Network
MCMC	Markov chain Monte Carlo
MRF	Markov Random Field
NRMSE	Normalized RMSE
PDN	Poisson Dependency Network
PE	Pseudo Evidence
PGM	Probabilistic Graphical Model
PRT	Poisson Regression Tree
RMSE	Root-Mean-Square Error
SAT	Boolean satisfiability problem
SP	Survey Propagation
SPS	Shortest-Path-Sequences
SRL	Statistical Relational Learning
StarAI	Statistical Relational AI
SPGM	Sub-linear Poisson Graphical Model
TPGM	Truncated Poisson Graphical Model

VE	Variable Elimination
WP	Warning Propagation