Mach Learn (2016) 103:27–55
DOI 10.1007/s10994-015-5532-x

Lifted generative learning of Markov logic networks

Jan Van Haaren[1] · **Guy Van den Broeck**[1,2] · **Wannes Meert**[1] · **Jesse Davis**[1]

Received: 16 March 2014 / Accepted: 2 September 2015 / Published online: 28 October 2015

Abstract Markov logic networks (MLNs) are a well-known statistical relational learning formalism that combines Markov networks with first-order logic. MLNs attach weights to formulas in first-order logic. Learning MLNs from data is a challenging task as it requires searching through the huge space of possible theories. Additionally, evaluating a theory's likelihood requires learning the weight of all formulas in the theory. This in turn requires performing probabilistic inference, which, in general, is intractable in MLNs. Lifted inference speeds up probabilistic inference by exploiting symmetries in a model. We explore how to use lifted inference when learning MLNs. Specifically, we investigate generative learning where the goal is to maximize the likelihood of the model given the data. First, we provide a generic algorithm for learning maximum likelihood weights that works with any exact lifted inference approach. In contrast, most existing approaches optimize approximate measures such as the pseudo-likelihood. Second, we provide a concrete parameter learning algorithm based on first-order knowledge compilation. Third, we propose a structure learning algorithm that learns liftable MLNs, which is the first MLN structure learning algorithm that exactly optimizes the likelihood of the model. Finally, we perform an empirical evaluation on three real-world datasets. Our parameter learning algorithm results in more accurate models than several competing approximate approaches. It learns more accurate models in terms of test-set log-likelihood as well as prediction tasks. Furthermore, our tractable learner outperforms intractable models on prediction tasks suggesting that liftable models are a powerful hypothesis space, which may be sufficient for many standard learning problems.

Keywords Statistical relational learning · Lifted probabilistic inference · Markov logic networks · Parameter learning · Structure learning

Editors: Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo.

✉ Jan Van Haaren
jan.vanhaaren@cs.kuleuven.be

[1] Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium

[2] Computer Science Department, University of California, Los Angeles, Los Angeles, CA, USA

1 Introduction

Statistical relational learning (SRL) (Getoor and Taskar 2007) and probabilistic logic learning (De Raedt et al. 2008) seek to develop representations that combine the benefits of *probabilistic* models, such as Markov or Bayesian networks, with those of *relational* representations, such as first-order logic. Markov logic networks (MLNs), which combine first-order logic with Markov networks, are one of the most widely used SRL formalisms (Richardson and Domingos 2006). MLNs attach a weight to each first-order logic formula in a theory. Given a set of objects, they compactly specify how to construct a very large propositional Markov network.

Markov logic networks pose a great challenge for inference and learning: using classical algorithms, these tasks reduce to inference and learning in densely connected Markov networks with millions of random variables. The intractability of reasoning with SRL models motivated a new class of *lifted inference* algorithms (Poole 2003; Kersting 2012), which exploit the abundant *symmetries* in relational representations to speed up probabilistic inference. For large classes of liftable models and queries, these algorithms perform efficient inference without ever grounding to a propositional Markov network (Jaeger and Van den Broeck 2012). Whereas lifted inference deals with the intractability of reasoning, the intractability of *learning* is what motivates us in this paper. Moreover, we seek to efficiently learn models that themselves are tractable, in the sense that they permit lifted inference.

When learning MLNs from data, two tasks are considered. The *parameter learning* task is to learn the weights associated with each formula in a given theory. This is an analogous problem to that of learning feature weights in a log-linear propositional Markov network. For the *structure learning* task, one also learns the first-order logic formulas. In both cases, the data consist of a set of relational databases. The recent success of lifted inference algorithms raises important questions for the learning task:

1. Can lifted inference techniques improve *parameter learning* of liftable models, in terms of learning time or the quality of the learned model?
2. Can we *learn liftable structures* that guarantee tractability for certain queries?
3. How is the *quality of the learned model* affected when learning liftable structures? Does the reduced expressivity hurt, or is insisting on liftability and symmetry an effective regularization technique? Does support for lifted parameter learning outweigh the reduced expressivity of liftable structures?

This paper addresses these questions for *generative* learning, where the objective is to learn a model that maximizes the probability of observing the data. We focus on techniques from the *exact* lifted inference literature, where liftable model and query classes were defined. Moreover, this will allow us to compare the exact likelihoods of the learned MLNs, which is the natural evaluation measure for generative learning (Darwiche 2009; Koller and Friedman 2009; Murphy 2012).

Our first contribution is a *generic lifted parameter learning algorithm* that can use any lifted inference oracle to efficiently learn weights that maximize the exact training-set likelihood. This is in contrast with most existing learners, which resort to optimizing an approximate objective such as pseudo-likelihood (Besag 1975). Computing likelihoods requires running inference, which is often highly intractable without lifting. We analyze two properties of the algorithm that speed up learning. A key insight from lifting is the possibility of grouping indistinguishable objects that can be reasoned about as a whole. Grouping these together can significantly reduce the number of inferences needed to learn parameters.

Moreover, by using a lifted inference oracle, parameter learning can run in time polynomial in the size of each training database when classical methods run in exponential time.

Our second contribution is a *concrete lifted parameter learning algorithm* based on first-order knowledge compilation (Van den Broeck et al. 2011). This is a state-of-the-art lifted inference algorithm that can compile a large class of MLNs into a circuit language. The circuit guarantees that likelihood computations run in time polynomial in the number of entities in the databases. Its most appealing property for weight learning is that compilation only needs to be performed once per MLN structure. In each iteration of lifted parameter learning, the compiled circuit can be reused with updated weights to compute a new likelihood and its gradient.

Our third contribution is a *lifted structure learning* algorithm that learns liftable MLN theories. It uses our lifted parameter learning algorithm as a subroutine, and therefore also optimizes the exact training-set likelihood. This contrasts with existing MLN structure learners which resort to optimizing pseudo-likelihood. Moreover, the learned structures are guaranteed to support certain types of queries efficiently, including, for example, conditional probability queries with bounded Boolean rank [see Van den Broeck and Darwiche (2013) for details]. Our work thus follows in a long tradition of tractable structure learning algorithms for probabilistic graphical models (e.g., Chechetka and Guestrin 2007), and is among the first tractable learning algorithms for statistical relational representations.

Our fourth contribution is an *extensive empirical evaluation* of lifted parameter and structure learning on three standard real-world SRL datasets. We find that our lifted parameter learning algorithm learns models with better test-set likelihood than competing approaches (including approximate inference techniques), and scales well with the amount of available data. When learning tractable structures, our lifted learning algorithm outperforms existing learners in terms of likelihood, but also in terms of conditional likelihood and area under the precision-recall curve on prediction tasks. We even find that our tractable structure learner outperforms off-the-shelf intractable learners on prediction tasks, suggesting that liftable models are a powerful hypothesis space, which is sufficient for many standard learning problems.

2 Background

We first present the necessary background on relational representations, Markov logic networks, and their inference and learning algorithms.

2.1 First-order logic

We first introduce some standard concepts from function-free first-order logic, which has the following types of symbols: (uppercase) *constants*, (lowercase) *variables*, and *predicates*. Constant symbols represent objects in the domain (e.g., people: Alice, Bob, etc.). Variable symbols (x, y, etc.) range over the objects in the domain. Predicate symbols represent relations among objects in the domain (e.g., `Friends`) or attributes of objects (e.g., `Smokes`). A *term* is a variable or a constant. A predicate applied to a tuple of terms is an *atom*. A *literal* is an atom or its negation. A *formula* is constructed by connecting literals using logical connectives. Following the convention for lifted inference, we assume that all variables are free, that is, formulas have no quantifiers [see Van den Broeck et al. (2014) for the general case]. A *ground* atom or formula contains no variables, only constants. A *database* (i.e., a *possible world*) assigns a truth value to each ground atom. A *grounding substitution* θ of a formula F replaces all logical variables in F by constants, denoted by $F\theta$.

2.2 Markov logic networks

Markov networks are undirected probabilistic graphical models that represent a joint probability distribution over a set of random variables $X_1, \dots X_n$ (Della Pietra et al. 1997). Each clique of variables $\mathbf{X}_k$ in the graph has an associated potential function $\phi_k(\mathbf{X}_k)$. The probability of a possible world $\mathbf{x}$ represented by a Markov network is $\Pr(\mathbf{x}) = \frac{1}{Z} \prod_k \phi_k(\mathbf{x}_k)$, where $\mathbf{x}_k$ is the state of the kth clique (i.e., the state of the variables that appear in that clique), and Z is a normalization constant. Markov networks are often conveniently represented as *log-linear models*, where clique potentials are replaced by an exponentiated weighted sum of features of the state: $\Pr(\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(\mathbf{x})\right)$. A feature f_i may be any real-valued function of the state.

Markov logic networks (MLNs) (Richardson and Domingos 2006) combine Markov networks with first-order logic. MLNs soften logic by associating a weight with each formula. Worlds that violate formulas become less likely, but not impossible. Formally, an MLN is a set of pairs, (F_i, w_i), where F_i is a first-order formula and $w_i \in \mathbb{R}$. As w_i increases, so does the strength of the constraint F_i imposed on the world. Formulas with infinite weights represent pure logic formulas.

MLNs provide a template for constructing Markov networks. When given a finite set of constants (the domain), the MLN formulas define a Markov network. Nodes in the network, representing random variables, are the ground instances of the atoms in the formulas. Edges connect literals that appear in the same ground instance of a formula. An MLN induces the following probability distribution over relational databases db:

$$\Pr(db) = \frac{1}{Z} \exp\left(\sum_{i=1}^{|F|} w_i n_i(db)\right) \tag{1}$$

where F is the set of formulas in the MLN, w_i is the weight of the ith formula, and $n_i(db)$ is the number of true groundings of formula F_i in database db.

Example 1 Consider the following model:

$$w \quad \texttt{Smokes}(x) \wedge \texttt{Friends}(x, y) \Rightarrow \texttt{Smokes}(y)$$

The logical formula states that smokers are only friends with other smokers. By associating a high weight w with this formula, its meaning becomes that smokers are more likely to be friends with other smokers. Assuming a domain of two constants, Alice and Bob, this MLN induces a distribution over 6 random variables, including Smokes(Alice) and Friends(Alice, Bob), and 2^6 possible worlds. The MLN has a single first-order formula with four groundings:

$$w \quad \texttt{Smokes}(\mathsf{Alice}) \wedge \texttt{Friends}(\mathsf{Alice}, \mathsf{Alice}) \Rightarrow \texttt{Smokes}(\mathsf{Alice})$$
$$w \quad \texttt{Smokes}(\mathsf{Alice}) \wedge \texttt{Friends}(\mathsf{Alice}, \mathsf{Bob}) \Rightarrow \texttt{Smokes}(\mathsf{Bob})$$
$$w \quad \texttt{Smokes}(\mathsf{Bob}) \wedge \texttt{Friends}(\mathsf{Bob}, \mathsf{Alice}) \Rightarrow \texttt{Smokes}(\mathsf{Alice})$$
$$w \quad \texttt{Smokes}(\mathsf{Bob}) \wedge \texttt{Friends}(\mathsf{Bob}, \mathsf{Bob}) \Rightarrow \texttt{Smokes}(\mathsf{Bob})$$

When n of these groundings are true in a possible world, its probability is e^{wn}/Z.

2.3 Lifted probabilistic inference and tractability

The advent of statistical relational languages such as Markov logic has motivated a new class of *lifted inference* algorithms (Poole 2003). SRL models with large domains lead to very large graphical models, causing inference to become intractable. Lifted algorithms mitigate this cost, by exploiting the high-level structure and symmetries of the first-order logic formulas to speed up inference (Kersting 2012). Surprisingly, they perform tractable inference even in the absence of conditional independencies (Niepert and Van den Broeck 2014). For instance, lifted inference algorithms exactly compute single marginal probabilities for the MLN in Example 1 in time linear in the size of the corresponding Markov network (Van den Broeck et al. 2011). They scale up to millions of random variables, whereas classical algorithms require exponential time.

Recently, efforts were made to understand lifted inference at a theoretical level, and to delineate the classes of MLNs and inference tasks[1] for which lifted inference is tractable. The intuition that lifted algorithms should efficiently deal with large domains is formalized by the notion of *domain-lifted inference*.

Definition 1 (Van den Broeck (2011)) An algorithm is domain-lifted when it runs in time polynomial in the domain size.

Domain-lifted algorithms are polynomial in the number of objects in the world, but can be exponential in other parameters, such as the number of formulas.

Based on this notion of tractability, several classes of MLNs and inference tasks were shown to support (domain-)lifted inference. Tractable *classes of MLNs* include two-variable MLNs, where all features (formulas) have at most two logical variables (Van den Broeck 2011; Taghipour et al. 2013). Any combination of universal and existential quantification is liftable (Van den Broeck et al. 2014). More tractable and intractable classes are discussed in Beame et al. (2015). This list of known tractable classes is far from exhaustive, and many more complex MLNs are liftable, including ones with more than two variables. The MLNs considered in this paper are generally not in the classes above, yet they are still liftable. Tractable *classes of inference tasks* include partition functions, single marginal probabilities (Van den Broeck 2011), and expected counts of MLN formulas (Van den Broeck et al. 2013). Conditional probability queries are liftable given unary evidence atoms (Van den Broeck and Davis 2012; Bui et al. 2012) and binary evidence atoms of bounded Boolean rank (Van den Broeck and Darwiche 2013).

2.4 Parameter learning

The weight learning task for MLNs (Singla and Domingos 2005; Richardson and Domingos 2006; Lowd and Domingos 2007; Huynh and Mooney 2009) uses data to automatically learn the weight associated with each feature (formula) by optimizing a given objective function. Ideally, each candidate model would be scored by its training-set (log-)likelihood. For MLNs, the log-likelihood is a convex function of the weights and learning can be solved via convex optimization. The derivative of the log-likelihood with respect to the jth feature is (Richardson and Domingos 2006):

$$\frac{\partial}{\partial w_j} \log \Pr_w(db) = n_j(db) - \mathbb{E}_w[n_j] \tag{2}$$

where $n_j(db)$ is the number of true groundings of F_j in the training data and $\mathbb{E}_w[n_j]$ is computed using the current weight vector. The jth component of the gradient is simply

[1] Note that tractability of probabilistic models is always w.r.t. a class of inference tasks. For example, polytrees, arithmetic circuits, and sum-product networks are generally considered to be tractable, but (partial) MAP inference in them is still NP-complete (Park 2002).

the difference between the empirical counts of the jth feature in the data and its expectation according to the current model. Thus, each iteration of weight learning must perform inference on the current model to compute the expectations. This is often computationally infeasible.

Currently, the default generative weight learning approach for MLNs is to optimize the pseudo-likelihood (Besag 1975), which is more efficient to compute. The pseudo-likelihood is defined as

$$\Pr_w^{\bullet}(\mathbf{x}) = \prod_{j=1}^{|\mathbf{X}|} \Pr_w(X_j = \mathbf{x}_j | MB_j = \mathbf{x}_{MB_j}),$$

where $|\mathbf{X}|$ is the number of random variables, $\mathbf{x}_j$ is the state of the jth variable in $\mathbf{x}$, MB_j is the Markov blanket of the jth variable, and $\mathbf{x}_{MB_j}$ is the state of that Markov blanket in $\mathbf{x}$. Maximum pseudo-likelihood weights can also be learned via convex optimization. The (pseudo-)likelihood for a set of training examples is the product of the (pseudo-)likelihoods for the individual examples.

For the purpose of discriminative learning, there has been work on optimizing the *conditional* likelihood and the most advanced work is by Lowd and Domingos (2007). They propose several approaches, the best of which is a second-order method called pre-conditioned scaled conjugate gradient (PCSG). They use MC-SAT (Poon and Domingos 2006), which is a slice-sampling Markov chain Monte Carlo method, to approximate the expected counts. Generative learning is a special case of discriminative learning where the query set contains all the variables in the domain and the evidence set is empty. Therefore this approach is suitable for learning maximum likelihood weights, although, to the best of our knowledge, this has yet to be attempted until this paper (cf. Sect. 6).

2.5 Structure learning

The structure learning task is to learn both the formulas and their associated weights from data. Structure learning is an incredibly challenging problem as there is a huge number of candidate clauses and an even larger space of candidate models. Typically, the structure of an MLN is learned by greedily adding one clause at a time to the MLN. While multiple MLN structure learning approaches exist, they can be broadly divided into two categories: top-down and bottom-up.

MSL (Kok and Domingos 2005) is a canonical example of a top-down approach. MSL begins with an MLN that only contains the unit clauses. MSL starts by constructing all clauses of length two. It then runs a beam search to find the current best clause and adds it to the network. In each iteration, MSL constructs new candidate clauses by adding literals to the best clauses in the beam. The search iterates until no clause improves the score of the MLN. To evaluate the merit of each clause, MSL uses weighted pseudo-log-likelihood (WPLL), which is an extension of pseudo-log-likelihood that diminishes the importance of predicates with a large number of groundings. It does this by normalizing a predicate's PLL by its number of possible groundings (Kok and Domingos 2005). To avoid overfitting, each clause receives a penalty term proportional to the number of literals that differ between the current clause and the initial clause.

A second category of structure learners adopts a bottom-up approach (e.g., Mihalkova and Mooney 2007; Kok and Domingos 2010), using the data to restrict the search space. BUSL (Mihalkova and Mooney 2007) is a two-step algorithm that follows this paradigm. In the first step, it constructs a template Markov network from a ground Markov network by discovering recurring paths of true atoms. In the second step, it transforms the template Markov network into candidate clauses. It greedily iterates through the set of candidate clauses. It adds the clause to the MLN that most improves the score of the model. The search terminates when no clause improves the model's score. BUSL also uses WPLL to evaluate the merit of a candidate clause.

3 Lifted generative weight learning

In this section, we describe a general algorithm that uses lifted inference techniques for the weight learning task, and only assumes a domain-lifted inference black box.

Even given a single formula F, learning its maximum likelihood weight is a challenging task. Computing the likelihood of a weight w requires computing the partition function Z, which is hard. Moreover, optimizing the likelihood requires computing the expectation $\mathbb{E}_w[n_F]$ to obtain the gradient (Eq. 2) at each iteration of a convex optimization algorithm, by summing the probabilities of each possible grounding of F:

$$\mathbb{E}_w[n_F] = \Pr(F\theta_1) + \cdots + \Pr(F\theta_m) \tag{3}$$

where $\theta_1, \ldots, \theta_m$ are the grounding substitutions of F. Answering these marginal probability queries simply requires calculating the probability that each formula is true according to the model, and does not involve conditioning on the data. The only way in which the probabilities depend on the data is the domain size of each variable, that is, the number of objects in the world. Nevertheless, computing these marginal probabilities is computationally expensive with traditional algorithms.

We will now show how to use lifted inference techniques to compute the likelihood of a set of weights and its gradient. This approach yields *two benefits*:

1. Leveraging insights from the lifted inference literature allows weight learning to compute a small number of marginals to compute the gradient.
2. Each query is computed more efficiently using lifted inference. Namely, it is polynomial in the size of the databases, that is, the number of objects in the databases, whereas propositional inference is in general exponential in this size.

3.1 First benefit: decreasing the number of inference tasks

We first review some techniques from the lifted inference literature. Second, we show how to use these techniques to efficiently compute $\mathbb{E}_w[n_F]$. Third, we analyze the extent to which this reduces the number of required inference tasks.

3.1.1 Equiprobable random variables

A set of random variables $\mathbf{X}$ is called *equiprobable* with respect to a given distribution Pr iff for all $X_1, X_2 \in \mathbf{X} : \Pr(X_1) = \Pr(X_2)$. In the absence of evidence, many of the queries in an MLN will be equiprobable, because of the symmetries imposed by the model. Lifted inference algorithms therefore excel at answering queries with no evidence. In fact, one of the key insights from lifted inference is that we can partition the set of random variables into equiprobable sets by purely syntactic operations on the first-order model, and reason about the sets as a whole.

Example 2 To illustrate this point, consider again the MLN from Example 1:

$$w \quad \texttt{Smokes}(x) \land \texttt{Friends}(x, y) \Rightarrow \texttt{Smokes}(y)$$

Assuming a domain of three constants, Alice, Bob, and Charlie, the random variables Friends(Alice, Bob) and Friends(Bob, Charlie) are equiprobable. The queries have identical probabilities because they are indistinguishable w.r.t. the MLN. Intuitively, this can be seen by looking at a permutation of the constants, mapping Alice into Bob and Bob into Charlie. This permutation turns Pr(Friends(Alice, Bob)) into Pr(Friends(Bob, Charlie)), whereas the MLN model is invariant under this permutation (it does not even explicitly mention the constants). Therefore, these atoms are indistinguishable and they must have the same marginal probability. The random variables Friends(Alice, Alice) and Friends(Bob, Charlie) are not equiprobable, since there is no permutation of the constants that turns one into the other.

The question then is how to partition queries into equiprobable sets. This can be done based on syntactic properties of the MLN by a technique called *preemptive shattering* (Poole et al. 2011). It is a conceptually simpler version of the influential *shattering* algorithm proposed by Poole (2003) and de Salvo Braz et al. (2005) in the context of lifted inference. This technique can be applied to a model with an arbitrary number of formulas. If the model does not mention specific constants, the algorithm enumerates all ways in which the logical variables in the same formula can be equal or different. More formally, in the single formula case, $\Pr(F\theta_k) = \Pr(F\theta_l)$ when for all pairs of logical variables x_i, x_j in F, we have $x_i\theta_k = x_j\theta_k$ iff $x_i\theta_l = x_j\theta_l$. If the model itself mentions certain unique information about specific constants, we can still partition the queries into equiprobable sets, but finding such sets gets slightly more complicated. Poole et al. (2011), Van den Broeck et al. (2012), Van den Broeck (2013) contain the full details for this procedure.

3.1.2 Computing the gradient

To achieve the first benefit—a smaller number of inference tasks—we identify equiprobable sets of groundings of the MLN formulas. This allows us to reduce the number of queries in Eq. 3 that need to be answered during weight learning. For each equiprobable set, only one representative query needs to be answered, as each other query in the group will have the identical marginal probability.

Consider again the single-formula case. Let $\mathcal{P} = \{E_1, \dots E_q\}$ be the equiprobable partition found for formula F by preemptive shattering, and let $F\theta_{E_i}$ be any arbitrary ground formula in E_i. We can then simplify Eq. 3 to

$$\mathbb{E}_w[n_F] = |E_1| \cdot \Pr(F\theta_{E_1}) + \cdots + |E_q| \cdot \Pr(F\theta_{E_q}), \tag{4}$$

involving as many queries as there are equiprobable sets in the partition $\mathcal{P}$.

3.1.3 Analysis

In the multiple-database setting (for example, when performing cross-validation), it is necessary to compute Eq. 4 for each database separately, as the domain size can vary between individual databases. Both the size of each E_i, and the probability of individual $F\theta_{E_i}$ are affected by a change in domain size. The partition size $|\mathcal{P}|$, however, is not affected by domain size, and only depends on the structure of the MLN formulas. This allows us to state the following.

Proposition 1 *Given b databases and an equiprobable partition $\mathcal{P}$, evaluating the gradient of the likelihood of an MLN requires computing $b \cdot |\mathcal{P}|$ probabilities.*

In the special case of a single formula with n logical variables, the number of queries we need to pose is the Bell number B_n (Bell 1934; Rota 1964).

Definition 2 (*Bell Number*) Bell numbers are recursively defined as

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k \quad \text{with} \quad B_0 = B_1 = 1.$$

The Bell number B_n represents the number of partitions of n elements into non-empty sets. In our case, it is the number of equivalence relations on the n logical variables in the formula, which does not depend on the size of the domain or database. Assuming a domain size of D, that same formula will have D^n groundings and computing Eq. 2 without using these insights from lifted inference would require answering D^n queries.

Example 3 The formula of Example 1 has two logical variables and Bell number two. Assuming a domain of 1000 people, one equiprobable set has size 999,000, and the other has size 1000. Computing the gradient with Eq. 3 requires one million inference calls. Computing the gradient with Eq. 4 requires two calls.

When we more generally have multiple formulas that do not explicitly mention any constants from the database, the analysis is also easy, based on properties of the preemptive shattering algorithm.

Proposition 2 *An MLN with k formulas, containing respectively $n_1, \ldots, n_k$ logical variables and no constants has an equiprobable partition $\mathcal{P}$ of size $\sum_{i=1}^{k} B_{n_i}$.*

For this case, Eq. 3 requires computing $\sum_{i=1}^{k} D^{n_i}$ marginals per database, whereas lifted learning requires computing $\sum_{i=1}^{k} B_{n_i}$, essentially removing the dependency on the size of the database from the number of inference calls. This difference can be significant. Formulas typically have a bounded number of variables n_i, between two and four, which gives Bell numbers $B_2 = 2$, $B_3 = 5$ and $B_4 = 15$. SRL databases, on the other hand, typically describe relations between thousands of objects, resulting in models with millions of random variables. This is also true for the databases used in Sect. 6.

In the most general case, where the MLN being learned explicitly mentions certain constants, the analysis becomes more complex. Still, the size of the equiprobable partition will grow polynomially with the number of constants that appear in MLN formulas, and be independent of the number of constants in the databases.

3.2 Second benefit: lifting individual inference tasks

Our general lifted weight learning algorithm assumes access to a lifted inference oracle that can efficiently compute the partition function and marginal probability of any random variable (ground atom) in the MLN, even for large domain sizes. More specifically, we assume a black-box inference algorithm is *domain-lifted*.

The second benefit of lifted learning over its propositional counterpart is the *complexity* of inference for each query. A domain-lifted inference algorithm guarantees that the complexity of computing the partition function (needed to compute the likelihood), and the marginal probabilities in Eq. 4 (needed to compute the gradient) grows polynomially with the domain size, and therefore polynomially with the size of the training databases. On the other hand, when doing propositional inference to compute the same numbers, inference is in general exponential in the domain size. Treewidth is a polynomial of the domain size for most non-trivial MLNs, and propositional inference is often exponential in treewidth. Indeed, running the propositional variable elimination algorithm on the MLN of Example 1 with 1000 people would require building a table with 2^{1000} rows.

4 Lifted weight learning using first-order knowledge compilation

The lifted weight learning algorithm assumed the presence of a lifted inference oracle. One could use any exact lifted inference technique, such as PTP (Gogate and Domingos 2011) or FOVE (de Salvo Braz et al. 2005; Milch et al. 2008; Taghipour and Davis 2012). We will now look at the implications of choosing one particular algorithm, namely *weighted first-order model counting* by *first-order knowledge compilation* (WFOMC) (Van den Broeck et al. 2011; Van den Broeck 2013). WFOMC is chosen because it supports domain-lifted inference on a large variety of MLN structures. Moreover, its knowledge compilation approach offers circuit reuse, which will prove to be beneficial for learning. First, we give the necessary background on WFOMC and then describe its application to lifted learning.

4.1 WFOMC background

The WFOMC approach to lifted probabilistic inference consists of the following three steps: (i) convert the MLN to a weighted model counting problem, (ii) compile the weighted model counting problem into a First-Order d-DNNF (FO d-DNNF) circuit, and (iii) evaluate the circuit for a given set of weights and domain sizes to compute the partition function. Marginal probabilities can be obtained by taking ratios of partition functions of different circuits. We will give a brief overview here, and refer to Van den Broeck (2013) for details. Darwiche (2009) discusses weighted model counting for propositional probabilistic inference.

Weighted first-order model counting A WFOMC problem is similar to a Markov logic network. The difference is that in a WFOMC problem, weights can only be associated with predicates. For example, for the predicate Q, only weighted formulas of the form $(\mathrm{Q}(x_1, \ldots, x_n), w)$ are allowed. Formulas that are complex (containing logical connectives) must be *hard formulas*, with infinite weight. Any MLN can be transformed into a WFOMC problem by adding new predicates to the theory, representing the truth value of each complex MLN formula.

Example 4 Example 1 contains one weighted complex formula. Its WFOMC representation introduces the predicate F to carry the weight of the MLN formula, and consists of a single weighted atom together with a hard formula:

$$\begin{aligned} &w \quad \mathrm{F}(x, y) \\ &\infty \quad \mathrm{F}(x, y) \Leftrightarrow [\mathtt{Smokes}(x) \wedge \mathtt{Friends}(x, y) \Rightarrow \mathtt{Smokes}(y)]. \end{aligned} \tag{5}$$

First-order knowledge compilation The goal of first-order knowledge compilation is to take a theory in first-order logic, and compile it into a circuit representation that allows certain queries to be answered efficiently. In our case, the logical theory Δ consists of the hard clauses in our WFOMC representation of the MLN M, the query is computing the weighted model count WFOMC$(\Delta, \bar{w}, D)$ for any weight vector $\bar{w}$ and domain size D, corresponding to the partition function of M, and the target circuit language is called FO d-DNNF. The circuit is logically equivalent to the input theory, but has certain syntactic properties, called decomposability, determinism, and automorphism, that make it a tractable representation for WFOMC problems. Note that the FO d-DNNF circuit for a WFOMC problem is independent

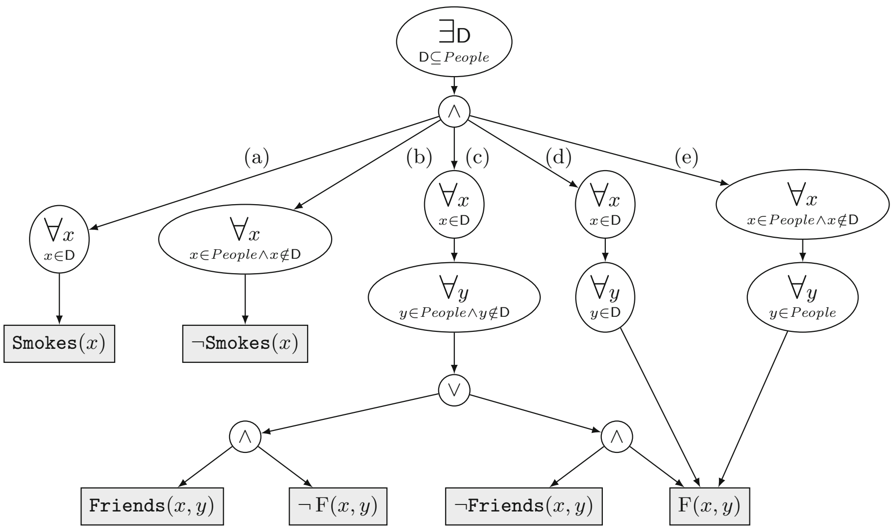

Fig. 1 A FO d-DNNF circuit for the friends and smokers problem (see Example 5)

of its domain size and weights. Once a FO d-DNNF is obtained, one can query the partition function for any domain size and weights efficiently, that is, polynomially in the domain size, by evaluating the circuit bottom-up.

Example 5 The FO d-DNNF for the WFOMC problem of Example 1 is shown in Fig. 1. Its syntax and semantics are formally defined in Van den Broeck (2013). Intuitively, the circuit root states that there exists a subset D of people in the domain. At the next layer, the circuit makes five assertions: (a) all people in D smoke, and (b) nobody else smokes. This defines D to be the set of smokers. Next, it encodes the truth value of the MLN formula: (c) when x is a smoker and y is a non-smoker, then the formula is satisfied (F is true) precisely when `Friends` is false. Moreover, when (d) x and y are both smokers, or (e) when x is a non-smoker, then the formula is always satisfied. These assertions make the circuit logically equivalent to Formula 5.

The FO d-DNNF circuit can be evaluated bottom-up to efficiently compute the partition function Z. For our running example, this computation has the form

$$Z = \sum_{d=0}^{n} \binom{n}{d} \cdot \underbrace{1^d}_{\text{(a)}} \cdot \underbrace{1^{n-d}}_{\text{(b)}} \cdot \underbrace{(1+e^w)^{d(n-d)}}_{\text{(c)}} \cdot \underbrace{(2e^w)^{d^2}}_{\text{(d)}} \cdot \underbrace{(2e^w)^{(n-d)n}}_{\text{(e)}}.$$

Here, n is the domain size (number of people), d is the size of D (number of smokers), and we associate each factor in the computation with a branch of Fig. 1.[2] Note that the complexity of this computation is polynomial in n.

The FO d-DNNF compilation algorithm cannot compile any arbitrary logical theory, but certain guarantees exist. For example, any MLN with up to two logical variables per formula can always be compiled. In our experiments, however, we will often be able to still compile MLN structures that are outside this class.

[2] Factors (a) and (b) are always 1 because our example has no weight for the `Smokes` relation.

4.2 Lifted weight learning

To illustrate the benefits of our knowledge compilation approach, we first consider computing the likelihood gradient for the single formula case, weights w and domain size D. Then, computing the expected number of true groundings of F requires estimating $\Pr(F\theta_{E_i})$ once for each equiprobable partition. WFOMC solves this by computing the ratio of weighted model counts

$$\frac{\mathrm{WFOMC}(F\theta_{E_i} \wedge \Delta, w, D)}{\mathrm{WFOMC}(\Delta, w, D)}$$

Notice that each ratio has the same denominator $\mathrm{WFOMC}(\Delta, w, D)$, which corresponds to the partition function of the original MLN. If we have q equiprobable partitions, evaluating the weighted model counts requires compiling $q + 1$ circuits: one for each equiprobable set, and one for the partition function. These circuits are *independent* of the weights and domains of the first-order model and can therefore be used to compute Eq. 4 for any database size and weight vector. Thus, each circuit can be *reused on each iteration of weight learning*. This exemplifies the idea behind the knowledge compilation approach to inference (Darwiche and Marquis 2002): transform the model from a representation where a certain task is hard to a representation where the task is easy, and reuse that representation to solve multiple problems.

Algorithm 1 outlines our Lifted Weight Learning (LWL) approach. It takes an MLN M and a set of databases DB as inputs and returns a weight vector $\bar{w}$. The algorithm works as follows. First, it builds all the circuits needed to compute the likelihood and its gradient. It compiles one circuit for M to compute the partition function. Then it preemptively shatters each weighted formula F in M to identify its equiprobable partition. It compiles one circuit for every equiprobable set in the partition. Second, it runs an iterative procedure to learn the weights. During each iteration i of convex optimization, it computes the gradient of the likelihood given the current weights $\bar{w}_i$. First it computes the partition function. Then, for each of the b databases, the expected counts for each formula are calculated by reevaluating the compiled circuit associated with every one of the formula's equiprobable partitions. Traditionally, this is the most challenging step in computing Eq. 2 (the gradient). The algorithm terminates when a stop condition is met (e.g., after a predefined number of iterations).

It follows from Proposition 1 that over t iterations of convex optimization, a vanilla lifted learning algorithm needs to answer $t \cdot b \cdot |\mathcal{P}|$ hard queries. Using WFOMC for lifted learning provides significant savings.

Proposition 3 *Lifted weight learning by first-order knowledge compilation performs* $1 + |\mathcal{P}|$ *hard compilation steps and reuses each circuit* $t \cdot b$ *times by reevaluating it for different weight vectors* $\bar{w}_i$ *and domain sizes* D.

An additional benefit of knowledge compilation is that it exploits local structure and context-specific independencies in the MLN. These are regularities that arise from using Boolean logic to compactly define potentials (Chavira and Darwiche 2008). They are not exploited by factor-based lifted inference algorithms, such as FOVE (de Salvo Braz et al. 2005).

For the special case of a single formula with n logical variables, by using both knowledge compilation and lifted inference, we went from answering $t \cdot b \cdot D^n$ queries whose complexity is exponential in the size of the databases to $1 + B_n$ compilations that are independent of the training databases, and $t \cdot b$ circuit evaluations whose complexity is polynomial in the size of the databases.

Algorithm 1 LIFTEDWEIGHTLEARNING(M, DB)

```
Input.
  M:    A set of MLN formulas with initial weights
  DB:   A set of training databases
Supporting functions.
  COMPILE:   Compile to FO d-DNNF circuit
  SHATTER:   Partition into equiprobable sets
  WFOMC:     Compute weighted FO model count
  LBFGS:     Optimization algorithm
Function.
 1: let $D^{db}$ be the domain sizes in database $db$
 2: let $n_F^{db}$ be the number of true groundings of formula $F$ in database $db$
 3: let $\bar{w}$ be the initial weight vector of $M$
 4: let $\Delta$ be the hard clauses in the WFOMC representation of $M$
 5: $C_Z \leftarrow$ COMPILE()
 6: for each $F \in M$ do
 7:   $\mathcal{P}_F \leftarrow$ SHATTER($M, F$)                         // Partition
 8:   for each $E \in \mathcal{P}_F$ do
 9:     for some $F\theta_E \in E$ do
10:       $C_E \leftarrow$ COMPILE($\Delta \wedge F\theta_E$)
11: repeat
12:   $\mathcal{L} \leftarrow 0$                                          // Log-likelihood
13:   $\nabla\mathcal{L} \leftarrow \bar{0}$                              // Log-likelihood gradient vector
14:   for each $db \in DB$ do
15:     $Z \leftarrow$ WFOMC($C_Z, \bar{w}, D^{db}$)
16:     $\mathcal{L} \leftarrow \mathcal{L} - \log(Z)$
17:     for each $F_i \in M$ do
18:       $\mathcal{L} \leftarrow \mathcal{L} + \bar{w}_i \cdot n_{F_i}^{db}$
19:       $\nabla\mathcal{L}_i \leftarrow \nabla\mathcal{L}_i + n_{F_i}^{db}$
20:       for each $E \in \mathcal{P}_{F_i}$ do
21:         $p \leftarrow$ WFOMC($C_E, \bar{w}, D^{db}$)$/Z$
22:         $\nabla\mathcal{L}_i \leftarrow \nabla\mathcal{L}_i - |E| \cdot p$
23:   $\bar{w} \leftarrow$ LBFGS($\mathcal{L}, \bar{w}, \nabla\mathcal{L}$)
24: until convergence
25: return $\bar{w}$
```

5 Lifted structure learning

In this section, we describe a general algorithm that uses lifted inference techniques for the structure learning task. To optimally benefit from the existing lifted inference algorithms and our lifted weight learning algorithm, we need a structure learning approach that learns liftable theories.

The ideal way to learn a liftable model is to design a search space that only contains liftable models. This is complicated by the fact that we still lack a full characterization of which models are liftable. We know that models where each formula contains at most two distinct logical variables are always liftable. However, this class of models may be too restrictive. Many models that contain more expressive formulas are also liftable. This process is complicated by the fact that two formulas, considered independently, may be liftable, but combining them into a single model results in a theory that is not liftable. Furthermore, even if a model is liftable, the circuit may be too big to fit in memory or too time-consuming to evaluate. The end result is that it is difficult to design a suitable search space.

Consequently, learning a liftable model can be achieved in two ways. The first way is to run an off-the-shelf structure learning algorithm, such as BUSL or MSL, using the default objective function of WPLL, and to perform parameter tuning in such a way that the final learned theory is compilable. Since we have only little understanding of which theories are compilable, the parameter tuning process can be tedious and time-consuming.

The second way is to integrate a check into the search procedure that verifies whether each candidate theory is compilable. Hence, unliftable candidate theories are discarded from the search space. Furthermore, a bias can be inserted into the search to avoid liftable theories that are too big to fit in memory or too complex to be evaluated in practice. An additional benefit is that the learner can directly optimize the exact likelihood instead of using an approximation such as the pseudo-likelihood. The end result is a theory that is guaranteed to be liftable.

Algorithm 2 outlines our Lifted Structure Learning (LSL) approach, which adopts the latter search strategy. LSL takes a set of MLN formulas and training databases as inputs and returns a theory of MLN formulas and their associated weights. The algorithm optimizes the training-set log-likelihood by iteratively adding formulas to an initially empty theory.

Algorithm 2 LIFTEDSTRUCTURELEARNING(*CFS*, *DB*)

Input.

CFS:	A set of candidate MLN formulas
DB:	A set of training databases

Supporting functions.

LIFTEDWEIGHTLEARNING	Compute formula weights
COMPUTELOGLIKELIHOOD	Compute training-set log-likelihood

Function.

```
1:  T ← ∅                                              // Initialize theory
2:  TLL ← 0                                            // Theory log-likelihood
3:  while |CFS| > 0 do
4:    BCT ← ∅                                          // Best candidate theory
5:    BCF ← ∅                                          // Best candidate formula
6:    BCTLL ← 0                                        // Best candidate theory log-likelihood
7:    for each CF ∈ CFS do
8:      CT ← T ∪ CF                                    // Compose candidate theory
9:      WCT ← LIFTEDWEIGHTLEARNING(CT,DB)
10:     WCTLL ← COMPUTELOGLIKELIHOOD(WCT,DB)
11:     if WCTLL > BCTLL then
12:       BCT ← WCT                                    // Update best candidate theory
13:       BCF ← CF                                     // Update best candidate formula
14:       BCTLL ← WCTLL                                // Update best candidate theory log-likelihood
15:   T ← BCT                                          // Replace theory by best candidate theory
16:   CFS ← CFS \ {BCF}                                // Remove best candidate formula from candidate set
17: return T
```

In each iteration, the algorithm performs three steps. First, the algorithm builds a set of candidate theories by adding each candidate formula to a distinct copy of the current theory. Second, the algorithm learns the associated formula weights and computes the training-set log-likelihood for each candidate theory. This step is allowed a user-specified amount of time to complete. Each candidate formula that cannot be compiled or whose weight cannot be learned within the allotted time, is discarded. By biasing the search process towards formulas that can be compiled, the final theory's liftability is ensured. Third, the algorithm replaces the current theory by the best candidate theory in terms of training-set log-likelihood if that theory yields a log-likelihood improvement. The algorithm ends when no more candidate formulas are available or none of the remaining candidate theories yields a log-likelihood improvement over the current best theory.

The initial set of candidate formulas can be constructed either by running the candidate formula construction step of an off-the-shelf structure learning algorithm or by greedily enumerating all valid formulas satisfying certain constraints. In our experimental evaluation (see Sect. 6), we enumerate all formulas having at most three literals and at most three object variables. We only consider "connected" formulas for which a path via arguments exists between any two literals.

To compute the training-set log-likelihood for a candidate theory, we employ an internal cross-validation approach. We learn the formula weights on all-but-one training database and compute the log-likelihood on the left-out database. We repeat this procedure such that each database served as validation database once. To obtain the log-likelihood for a candidate theory, we simply average the log-likelihoods across the validation databases.

6 Empirical evaluation

In this section, we evaluate both our lifted weight learning (LWL) and lifted structure learning (LSL) approach.[3] We first present the experimental setup and then address five research questions; two questions related to weight learning and four questions related to structure learning.

Weight learning questions

– Q1: How well does lifted weight learning scale with respect to database size?
– Q2: Does exactly optimizing the likelihood during weight learning result in more accurate weights?

Structure learning questions

– Q3: How does lifted structure learning compare to the off-the-shelf structure learners in terms of log-likelihood when learning *tractable* models?
– Q4: How does lifted structure learning compare to the off-the-shelf structure learners in terms of AUC and CLL when learning *tractable* models?
– Q5: How does lifted structure learning compare to the off-the-shelf structure learners in terms of AUC and CLL when learning *intractable* models?
– Q6: What is the effect of our clause evaluation time-out on the complexity and quality of the models for each of the algorithms?

6.1 Experimental setup

We now introduce the datasets, explain how we learn the models, and discuss the inference setup.

6.1.1 Datasets

We use the following three real-world datasets and a synthetic dataset:

– The **IMDb** dataset comes from the IMDb.com website (Mihalkova and Mooney 2007). The dataset contains information about attributes (e.g., gender) and relationships among actors, directors, and movies. The data is divided into five different folds.

[3] LWL and LSL are available in the WFOMC package: http://dtai.cs.kuleuven.be/wfomc.

- The **UWCSE** dataset contains information about the University of Washington CSE Department (Richardson and Domingos 2006). The data contains information about students, professors and classes, and models relationships (e.g., teaching assistant and advisor) among these entities. The data consists of five folds, each one corresponding to a different group in the CSE Department.
- The **WebKB** dataset consists of Web pages from the computer science departments of four universities (Mihalkova and Mooney 2007). The data has information about labels of pages (e.g., student and course). There are four folds, one for each university.
- The **synthetic** dataset comes from the friends and smokers model from Example 1 (Singla and Domingos 2005).

In all domains, we perform cross-validation by holding out one fold as test set and learning a model on the remaining folds. Each fold serves as test set once.

6.1.2 Models

We compare tractable models learned by LSL with both tractable and intractable models learned by the bottom-up structure learner BUSL (Mihalkova and Mooney 2007) and the top-down structure learner MSL (Kok and Domingos 2005). We learned these models as follows:

- **Tractable LSL models:** LSL is run with all valid "connected" MLN formulas containing up to three literals and three distinct object variables as input. LSL discards any candidate theory for which the LWL subroutine fails to find weights within the allotted time limit of 5 min.
- **Tractable BUSL and MSL models:** To enforce tractability, we found it is sufficient to restrict BUSL and MSL to learn formulas containing up to four literals and three distinct object variables.
- **Intractable BUSL and MSL models:** BUSL and MSL are run with their default parameter settings, which allows them to learn formulas containing up to five literals and five distinct object variables.

6.1.3 Inference setup

In each domain, we predict the marginal probabilities of each predicate given evidence about all other predicates. Since lifted inference approaches cannot efficiently handle arbitrary binary evidence (Van den Broeck and Darwiche 2013), we use MC-SAT, which is part of the Alchemy package, to compute the probabilities. We use a burn-in of 10,000 samples and compute the probabilities with the following 100,000 samples. We measure the area under the precision-recall curve (AUC) and the test-set conditional log-likelihood (CLL) for the predicate of interest. AUC is insensitive to the large number of true negatives in the datasets, whereas CLL measures the quality of the probability estimates.

In our evaluation, we report the number of wins, losses, and ties for each algorithm. Since AUC and CLL are both skew-dependent metrics and the skew of a predicate varies across different predicates and different databases, simply averaging AUCs and CLLs, as has commonly been done in the past, is incorrect (Boyd et al. 2012).

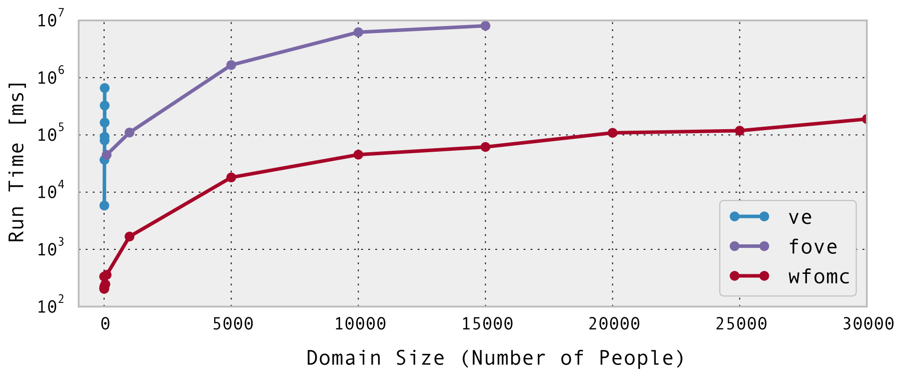

Fig. 2 Learning time for an increasing number of people in the friends–smokers dataset

6.2 Weight learning questions

6.2.1 Q1: How well does lifted weight learning scale with respect to database size?

The goal of this question is to explore how LWL scales with the database size. In this experiment, we use the synthetic dataset as a controlled environment. We vary the number of people in the domain from 100 to 30,000 and randomly generate a training database for each size. The largest database, for domain size 30,000, assigns truth values to $30,000^2 + 30,000$ or approximately 900 million ground atoms. Figure 2 shows the comparison of LWL with WFOMC and two other exact inference methods: variable elimination (VE) and FOVE.[4]

Lifted learning with VE benefits from a reduced number of inference tasks (Sect. 3.1), yet each query still runs in time exponential in the database size. Lifted learning with FOVE additionally benefits from domain-lifted inference (Sect. 3.2). Finally, lifted weight learning with WFOMC additionally benefits from knowledge compilation (Sect. 4) and outperforms the other approaches. The MLN structure is fully liftable, which means that the training time for FOVE and WFOMC is polynomial in the domain size. From the results for VE, it is clear that maximum-likelihood learning without lifting is highly intractable.

6.2.2 Q2: Does exactly optimizing the likelihood result in more accurate weights?

The goal of this question is to investigate whether exactly optimizing the likelihood yields better models than optimizing the approximated likelihood during weight learning. We use the tractable BUSL and MSL models to address this question. For each structure, we learn the weights with the following three algorithms:

- **PLL:** This approach optimizes the pseudo-likelihood of the weights via the limited-memory BFGS algorithm (Liu and Nocedal 1989). We use the implementation that is available in the Alchemy package (Kok et al. 2008).
- **PSCG:** The discriminative weight learning approach of Lowd and Domingos (2007) optimizes the log-likelihood of the data by making all the predicates query atoms and hence leaving the evidence set empty. We use the implementation that is available in the Alchemy package (Kok et al. 2008).

[4] We use the BLOG implementation (Milch et al. 2008): http://bayesianlogic.github.io.

Table 1 Test-set log-likelihoods for all three methods in all three domains for the models learned by BUSL

	IMDb			UWCSE			WebKB		
	PSCG	PLL	LWL	PSCG	PLL	LWL	PSCG	PLL	LWL
F1	−566	−548	**−378**	−1774	−1860	**−1524**	−863	−858	**−778**
F2	−548	−689	**−390**	−601	−594	**−535**	−1422	−1422	**−1331**
F3	−1223	−1157	**−851**	−1415	−1462	**−1245**	−717	−717	**−702**
F4	−425	−415	**−285**	−2781	−2820	**−2510**	−1224	−1224	**−1052**
F5	−423	−413	**−267**	−2634	−2763	**−2357**			

LWL consistently outperforms both PLL and PSCG for all 14 models. The best result for each fold in each domain is in bold

Table 2 Test-set log-likelihoods for all three methods in all three domains for the models learned by MSL

	IMDb			UWCSE			WebKB		
	PSCG	PLL	LWL	PSCG	PLL	LWL	PSCG	PLL	LWL
F1	−558	−831	**−440**	−1761	−1705	**−1469**	−869	−868	**−797**
F2	−561	−944	**−477**	−594	−574	**−509**	−1426	−1426	**−1324**
F3	−1336	−1576	**−909**	−1382	−1358	**−1198**	−711	−711	**−677**
F4	−442	−393	**−315**	−2745	−2758	**−2449**	−1207	−1207	**−1054**
F5	−443	−388	**−353**	−2616	−2582	**−2254**			

LWL consistently outperforms both PLL and PSCG for all 14 models. The best result for each fold in each domain is in bold

– **LWL:** This is the approach that is proposed in this paper. It uses the WFOMC package (Van den Broeck et al. 2011) for computing the gradient and the limited-memory BFGS algorithm that is part of the Breeze system.

First, each weight learning algorithm learns the weights for the given structures using the same data that produced each structure. Second, we compute the test-set log-likelihood for each model. Since we use WFOMC to compute the test-set log-likelihoods, the only difference among the three algorithms is *how* the formula weights are learned.

Table 1 reports the test-set log-likelihoods for all three methods in all three domains for the models learned by BUSL. LWL consistently outperforms both PLL and PSCG. Table 2 reports the test-set log-likelihoods for all three methods in all three domains for the models learned by MSL. LWL consistently outperforms both PLL and PSCG. These empirical results confirm our hypothesis that exactly optimizing the training-set log-likelihood results in more accurate formula weights.

For completeness, we adapted this experiment to also compare with learning by means of (generalized) belief propagation for factor graphs. We use the FastInf toolbox because it supports a relational factor graph representation (Jaimovich et al. 2010).[5] With a time-out of 24 hours, all 14 LWL tasks converged while only 3 out of 14 FastInf tasks converged. For each of the converged tasks, we compute the test-set log-likelihood of the parameters using

[5] Because FastInf learns each parameter in the factor graph, we cannot compare directly with learning the parameters of an MLN because one formula and associated parameter can represent multiple parameters in a factor. Therefore, the tractable MLNs learned by BUSL were transformed to have the same parameters (by adding weighted formulas). The resulting MLNs now have identical parameters to the equivalent factor graphs we give to FastInf.

Table 3 Test-set log-likelihoods for all methods when learning tractable models

	IMDb			UWCSE			WebKB		
	BUSL	MSL	LSL	BUSL	MSL	LSL	BUSL	MSL	LSL
F1	−378	−440	**−274**	−1524	−1469	**−1407**	−778	−797	**−777**
F2	−390	−477	**−311**	−535	**−509**	−543	−1331	**−1324**	−1341
F3	−851	−909	**−737**	−1245	−1198	**−1157**	−702	−677	**−662**
F4	−285	−315	**−222**	−2510	−2449	**−2409**	−1052	−1054	**−1049**
F5	−267	−353	**−220**	−2357	−2254	**−2089**			

LSL outperforms both BUSL and MSL in terms of test-set log-likelihood in 12 of the 14 settings, doing only marginally worse than MSL on the second fold of the UWCSE and WebKB dataset. The best result for each fold in each domain is in bold

WFOMC. The test-set log-likelihood of the LWL parameters was consistently 100 times better than the log-likelihoods of the FastInf parameters.

6.3 Structure learning questions

6.3.1 Q3: How does lifted structure learning compare to the off-the-shelf structure learners in terms of log-likelihood when learning tractable models?

The goal of this question is to investigate whether models learned by LSL yield better test-set log-likelihoods than tractable models learned by the off-the-shelf structure learning algorithms. We compare the tractable LSL models to the tractable BUSL and MSL models to address this question. We use LWL to learn the weights for the BUSL and MSL models since this approach outperforms the traditional weight learning algorithms (see Q1).

Table 3 reports the test-set log-likelihoods for tractable models learned by BUSL, MSL, and LSL. LSL outperforms both BUSL and MSL in terms of test-set log-likelihood in 12 of the 14 settings, doing only marginally worse than MSL on the second fold of the UWCSE and WebKB dataset. These results show there is no reason to prefer an off-the-shelf structure learner to our lifted structure learning approach for learning tractable models when optimizing the test-set log-likelihood.

6.3.2 Q4: How does lifted structure learning compare to the off-the-shelf structure learners in terms of AUC and CLL when learning tractable models?

The goal of this question is to investigate whether models learned by LSL are better at answering queries than tractable models learned by the off-the-shelf structure learning algorithms. We compare the tractable LSL models to the *tractable* BUSL and MSL models to address this question. We use LWL to learn the weights for the BUSL and MSL models since this approach outperforms the traditional weight learning algorithms (see Q1).

Table 4 reports the number of times LSL wins, loses, and ties against the off-the-shelf structure learners BUSL and MSL in terms of both AUC and CLL. In terms of AUC, LSL beats BUSL in 61 of the 95 settings, ties in 6 settings, and loses in 28 settings. In terms of CLL, LSL beats BUSL in 72 of the 95 settings and loses in 23 settings. In terms of AUC, LSL beats MSL in 50 of the 95 settings, ties in 13 settings, and loses in 32 settings. In terms of CLL, LSL beats MSL in 70 of the 95 settings and loses in 25 settings.

Table 4 Comparison of AUC and CLL results when learning tractable models in all three domains

		IMDb			UWCSE			WebKB		
		Win	Loss	Tie	Win	Loss	Tie	Win	Loss	Tie
LSL versus BUSL	AUC	**20**	5	5	**14**	6	0	**27**	17	1
LSL versus MSL	AUC	10	8	**12**	**12**	8	0	**28**	16	1
LSL versus BUSL	CLL	**25**	5	0	**15**	5	0	**32**	13	0
LSL versus MSL	CLL	**23**	7	0	**15**	5	0	**32**	13	0

In comparison to BUSL, LSL wins in 133 of the 190 settings, ties in 6 settings, and loses in 51 settings. In comparison to MSL, LSL wins in 120 of the 190 settings, ties in 13 settings, and loses in 57 settings. The most frequent result for each comparison is in bold

Table 5 Comparison of AUC and CLL results when learning intractable models in all three domains

		IMDb			UWCSE			WebKB		
		Win	Loss	Tie	Win	Loss	Tie	Win	Loss	Tie
LSL versus BUSL	AUC	**12**	7	11	**12**	8	0	**26**	18	1
LSL versus MSL	AUC	7	11	**12**	9	**11**	0	**31**	14	0
LSL versus BUSL	CLL	**17**	12	1	**15**	5	0	**33**	12	0
LSL versus MSL	CLL	**16**	14	0	**11**	9	0	**28**	17	0

In comparison to BUSL, LSL wins in 115 of the 190 settings, ties in 13 settings, and loses in 62 settings. In comparison to MSL, LSL wins in 102 of the 190 settings, ties in 12 settings, and loses in 76 settings. The most frequent result for each comparison is in bold

LSL is consistently leading to better models: it achieves more wins than both BUSL and MSL for both metrics. Although we used LWL (and thus exact likelihood) to relearn the weights for the BUSL and MSL models, during structure learning these algorithms initially optimize WPLL, which has a very similar objective to CLL, and thus should be to their advantage on that metric. This makes it surprising that LSL does particularly well at CLL compared to BUSL and MSL.

"Appendix 1" provides an extensive overview of per-predicate results.

6.3.3 Q5: How does lifted structure learning compare to the off-the-shelf structure learners in terms of AUC and CLL when learning intractable models?

The goal of this question is to investigate whether models learned by LSL are better at answering queries than models learned by the off-the-shelf structure learning algorithms. We compare the tractable LSL models to the *intractable* BUSL and MSL models to address this question. We use PLL to relearn the weights for the BUSL and MSL models. We cannot use LWL to relearn the weights since these models cannot be compiled.

Table 5 reports the number of times LSL wins, loses, and ties against the off-the-shelf structure learners BUSL and MSL in terms of both AUC and CLL. In terms of AUC, LSL beats BUSL in 50 of the 95 settings, ties in 12 settings, and loses in 33 settings. In terms of CLL, LSL beats BUSL in 65 of the 95 settings, ties in 1 setting, and loses in 29 settings. In terms of AUC, LSL beats MSL in 47 of the 94 settings, ties in 12 settings, and loses in 36 settings. In terms of CLL, LSL beats MSL in 55 of the 95 settings and loses in 40 settings.

Table 6 Test-set log-likelihoods for LSL with three different time-out values: 1, 2, and 5 min

	IMDb			UWCSE			WebKB		
	1 min	2 min	5 min	1 min	2 min	5 min	1 min	2 min	5 min
F1	−276	−275	**−274**	−1454	−1421	**−1407**	**−777**	−794	**−777**
F2	**−309**	−310	−311	−594	−562	**−543**	**−1341**	**−1341**	**−1341**
F3	−739	−739	**−737**	−1209	−1162	**−1157**	−664	−663	**−662**
F4	**−222**	**−222**	**−222**	−2434	**−2406**	−2409	−1049	**−1043**	−1049
F5	**−219**	−220	−220	**−2089**	−2142	**−2089**			

In most settings, the time-out value has either a small impact or no impact at all on the test-set log-likelihood. The best result for each fold in each domain is in bold

Surprisingly, BUSL and MSL do only slightly better at answering queries when they are no longer bound to learning tractable models. Their performance remains roughly the same. These results show that learning longer, more complex formulas does not necessarily lead to much better inference results. A possible explanation is that more complex models may fit the data better but also lead to more complicated inference tasks, which in turn leads to a decreased predictive performance.

"Appendix 2" provides an extensive overview of per-predicate results.

6.3.4 *Q6: What is the effect of our clause evaluation time-out on the complexity and quality of the models for each of the algorithms?*

The goal of this question is to investigate whether a restriction on the clause evaluation time leads to simpler models. To answer this question, we modify the off-the-shelf structure learners such that they can only spend a specified amount of time to evaluate each candidate clause. We run all three structure learning algorithms with three different clause evaluation time-outs: 1, 2, and 5 min. Keeping all other parameter values unchanged, we learn both tractable and intractable models with the off-the-shelf structure learners.

Table 6 reports the test-set log-likelihoods for the models learned by LSL for all three clause evaluation time-outs. In most settings, the time-out has either a small impact or no impact at all on the test-set log-likelihood. The only domain that seems to benefit from a longer run time is UWCSE, which is the most complicated domain in terms of number of predicates and facts. These results show that LSL is robust to the clause evaluation time-out value and confirm our observation that most structures can be compiled either relatively quickly or not at all. Hence, restricting the clause evaluation time for LSL does not lead to simpler models.

Furthermore, the clause evaluation time-out also does not have an impact on the complexity of the models learned by the off-the-shelf structure learners. Both BUSL and MSL learn the same structures for all three time-out values. Since these algorithms do not need to run inference to evaluate a candidate clause, they require only little time per clause evaluation. Hence, restricting the clause evaluation time for the off-the-shelf structure learners does not lead to simpler models either.

Tables 7 and 8 report the average number of clauses in a learned theory and the average clause length for all learning methods in each domain. We compute these metrics because they are indicative of model complexity and allow us to further explore if LSL has a bias towards simpler models. Note that both BUSL and MSL include a complexity penalty based

Table 7 The average number of clauses in a theory for each algorithm in each domain

	IMDb	UWCSE	WebKB
LSL (5-min time-out)	5.40 ± 1.14	10.20 ± 0.45	4.75 ± 1.50
BUSL tractable	1.40 ± 0.55	7.60 ± 3.21	3.25 ± 0.50
MSL tractable	3.00 ± 0.00	6.80 ± 0.45	4.00 ± 1.15
BUSL intractable	6.60 ± 3.36	18.60 ± 10.55	8.50 ± 4.20
MSL intractable	4.20 ± 0.84	4.60 ± 1.95	4.00 ± 1.15

Table 8 The average length of the clauses for each algorithm in each domain

	IMDb	UWCSE	WebKB
LSL (5-min time-out)	2.69 ± 0.14	2.69 ± 0.09	2.85 ± 0.17
BUSL tractable	3.00 ± 0.35	2.18 ± 0.25	2.35 ± 0.31
MSL tractable	2.67 ± 0.00	2.65 ± 0.07	2.53 ± 0.22
BUSL intractable	3.12 ± 0.51	2.71 ± 0.22	3.02 ± 0.37
MSL intractable	4.23 ± 0.33	3.27 ± 0.48	2.53 ± 0.22

on clause length. The results show that all approaches tend to learn similarly sized theories. The exception is that BUSL, in the intractable setting, learns more clauses than the other approaches, and its clauses tend to be slightly longer on average.

These results give some evidence that LSL does not offer better performance simply because it has a preference for simpler models. Instead, regularization by liftability and support for maximum-likelihood learning can explain this success.

7 Related work

Jaimovich et al. (2007) describe a formalism for relational Markov random fields and propose a form of lifted belief propagation for generative parameter learning in that language. This is related to Markov logic in that logic variables are used to express templates that generate the underlying factor graph. Markov logic, however, uses first-order logic, or constraints, to define the weights in the factors whereas relational Markov random fields require all weights in a factor template to be given explicitly. Overall, our work provides a much more detailed treatment of the subject. We looked at using *exact* lifted inference for learning, as opposed to approximate, which furthermore guarantees liftability of the learned models. Our experiments provide some initial evidence that generative parameter learning with belief propagation can provide sub-optimal results (Sect. 6.2).

Ahmadi et al. (2012) recently proposed using lifted belief propagation (Singla and Domingos 2008; Kersting et al. 2009), in a stochastic gradient optimization approach to piecewise *discriminative* weight learning. They show that the lifted learning approach reaches the same quality solution that can be achieved by propositional, approximate parameter learning with belief propagation, but converges to a solution over an order of magnitude faster.

Learning tractable probabilistic models, that is, models that always permit efficient inference for certain queries, is an emerging area of research. The largest body of work restricts the structure of the learned models (e.g., Chechetka and Guestrin 2007). One way to do this is to only consider models with a low tree-width (Narasimhan and Bilmes 2004; Chechetka and Guestrin 2007). Another body of work looks at simultaneously learning either a Bayesian network or a Markov network as well as an alternative representation (typically an arithmetic circuit) of the model that permits efficient inference. Then the model is penalized by the cost of inference, which can be calculated based on well-defined properties of the representation. By penalizing the circuit size of the associated model, it is possible to bias the learning algorithm towards models where efficient inference is possible (Lowd and Domingos 2008; Lowd and Rooshenas 2013). Our work fits within this framework since we have proposed tractable structure learning towards models that allow lifted inference. This guarantees tractable inference for certain types of queries (see Sect. 2.3). While tractable statistical relational languages have been investigated before (Domingos and Webb 2012), we believe our work is among the first to consider the problem of learning such tractable representations.

8 Conclusions

We investigated the effect of lifted inference for parameter and structure learning in the statistical relational learning setting. Specifically, we investigate generative learning, where the goal is to maximize the probability of observing the data, in the context of Markov logic networks (MLNs). We present four contributions. Our first contribution is a *generic lifted parameter learning algorithm* that can use exact lifted inference approaches to efficiently learn weights that maximize the exact training-set likelihood. We employ the concept of equiprobable random variables to characterize the reduced cost of computing the gradient given a lifted inference oracle. Our second contribution is a *concrete lifted parameter learning algorithm* based on first-order knowledge compilation. Its most appealing property for parameter learning is that compilation only needs to be performed once per MLN structure. Our third contribution is a *lifted structure learning* algorithm that learns liftable MLN theories. In contrast to existing MLN structure learners, which resort to optimizing pseudo-likelihood, it optimizes the exact likelihood using our lifted parameter learning algorithm as a subroutine. Our fourth contribution is an *extensive empirical evaluation* of lifted parameter and structure learning on three real-world SRL data sets. We find that our lifted parameter learning algorithm learns models with better test-set likelihood than competing approaches and scales well with the amount of training data. Our lifted structure learning algorithm outperforms existing learners in terms of likelihood as well as conditional likelihood and area under the precision-recall curve on prediction tasks. More surprisingly, we found that liftable models outperformed unliftable ones on prediction tasks. This provides some evidence that liftable models are a powerful hypothesis space that are suitable for modeling commonly used domains.

Acknowledgments Jan Van Haaren is supported by the Agency for Innovation by Science and Technology in Flanders (IWT). Guy Van den Broeck is supported by the Research Foundation-Flanders (FWO-Vlaanderen). Jesse Davis is partially supported by the Research Fund KU Leuven (OT/11/051), EU FP7 Marie Curie Career Integration Grant (#294068) and FWO-Vlaanderen (G.0356.12).

Appendix 1: Detailed experimental results for research question Q4

This section provides per-predicate inference results for research question Q4, where we investigate how lifted structure learning compares to off-the-shelf structure learners BUSL and MSL in terms of AUC and CLL when learning tractable models (Tables 9, 10, 11, 12, 13, 14).

Table 9 AUC results for learning tractable IMDb models

In comparison to BUSL, LSL wins in 20 of the 30 settings, loses in 5 settings, and ties in 5 settings. In comparison to MSL, LSL wins in 10 of the 30 settings, loses in 8 settings, and ties in 12 settings

The most frequent outcome is in bold

	BUSL			MSL		
	Win	Loss	Tie	Win	Loss	Tie
Actor	5	0	0	0	0	5
Director	0	0	5	0	0	5
Genre	5	0	0	2	1	2
Male	3	2	0	2	3	0
Movie	2	3	0	3	2	0
WorkedUnder	5	0	0	3	2	0
Total	**20**	5	5	10	8	**12**

Table 10 CLL results for learning tractable IMDb models

In comparison to BUSL, LSL wins in 25 of the 30 settings, loses in 5 settings, and ties in 0 settings. In comparison to MSL, LSL wins in 23 of the 30 settings, loses in 7 settings, and ties in 0 settings

The most frequent outcome is in bold

	BUSL			MSL		
	Win	Loss	Tie	Win	Loss	Tie
Actor	5	0	0	4	1	0
Director	5	0	0	5	0	0
Genre	5	0	0	5	0	0
Male	5	0	0	4	1	0
Movie	0	5	0	0	5	0
WorkedUnder	5	0	0	5	0	0
Total	**25**	5	0	**23**	7	0

Table 11 AUC results for learning tractable WebKB models

In comparison to BUSL, LSL wins in 14 of the 20 settings, loses in 6 settings, and ties in 0 settings. In comparison to MSL, LSL wins in 12 of the 20 settings, loses in 8 settings, and ties in 0 settings

The most frequent outcome is in bold

	BUSL			MSL		
	Win	Loss	Tie	Win	Loss	Tie
CourseProf	1	3	0	3	1	0
CourseTA	4	0	0	3	1	0
Faculty	1	3	0	1	3	0
Project	4	0	0	3	1	0
Student	4	0	0	2	2	0
Total	**14**	6	0	**12**	8	0

Table 12 CLL results for learning tractable WebKB models

In comparison to BUSL, LSL wins in 15 of the 20 settings, loses in 5 settings, and ties in 0 settings. In comparison to MSL, LSL wins in 15 of the 20 settings, loses in 5 settings, and ties in 0 settings

The most frequent outcome is in bold

	BUSL			MSL		
	Win	Loss	Tie	Win	Loss	Tie
CourseProf	4	0	0	4	0	0
CourseTA	4	0	0	4	0	0
Faculty	1	3	0	1	3	0
Project	4	0	0	4	0	0
Student	2	2	0	2	2	0
Total	**15**	5	0	**15**	5	0

Table 13 AUC results for learning tractable UWCSE models

In comparison to BUSL, LSL wins in 27 of the 45 settings, loses in 17 settings, and ties in 1 setting. In comparison to MSL, LSL wins in 28 of the 45 settings, loses in 16 settings, and ties in 1 setting

The most frequent outcome is in bold

	BUSL			MSL		
	Win	Loss	Tie	Win	Loss	Tie
Advisedby	5	0	0	3	2	0
Courselevel	2	3	0	3	2	0
Phase	1	4	0	3	2	0
Professor	2	2	1	2	2	1
Publication	5	0	0	4	1	0
Ta	2	3	0	4	1	0
Taughtby	3	2	0	4	1	0
Tempadvisedby	5	0	0	5	0	0
Yearsinprogram	2	3	0	0	5	0
Total	**27**	17	1	**28**	16	1

Table 14 CLL results for learning tractable UWCSE models

In comparison to BUSL, LSL wins in 32 of the 45 settings, loses in 13 settings, and ties in 0 settings. In comparison to MSL, LSL wins in 32 of the 45 settings, loses in 13 settings, and ties in 0 settings

The most frequent outcome is in bold

	BUSL			MSL		
	Win	Loss	Tie	Win	Loss	Tie
Advisedby	5	0	0	5	0	0
Courselevel	4	1	0	4	1	0
Phase	0	5	0	1	4	0
Professor	1	4	0	1	4	0
Publication	5	0	0	5	0	0
Ta	4	1	0	4	1	0
Taughtby	5	0	0	5	0	0
Tempadvisedby	5	0	0	5	0	0
Yearsinprogram	3	2	0	2	3	0
Total	**32**	13	0	**32**	13	0

Appendix 2: Detailed experimental results for research question Q5

This section provides per-predicate inference results for research question Q5, where we investigate how lifted structure learning compares to off-the-shelf structure learners BUSL and MSL in terms of AUC and CLL when learning intractable models (Tables 15, 16, 17, 18, 19, 20).

Table 15 AUC results for learning IMDb models

In comparison to BUSL, LSL wins in 12 of the 30 settings, loses in 7 settings, and ties in 11 settings. In comparison to MSL, LSL wins in 7 of the 30 settings, loses in 11 settings, and ties in 12 settings
The most frequent outcome is in bold

	BUSL			MSL		
	Win	Loss	Tie	Win	Loss	Tie
Actor	1	0	4	0	0	5
Director	0	0	5	0	0	5
Genre	3	0	2	2	1	2
Male	1	4	0	1	4	0
Movie	4	1	0	2	3	0
WorkedUnder	3	2	0	2	3	0
Total	**12**	7	11	7	11	**12**

Table 16 CLL results for learning IMDb models

In comparison to BUSL, LSL wins in 17 of the 30 settings, loses in 12 settings, and ties in 1 setting. In comparison to MSL, LSL wins in 16 of the 30 settings, loses in 14 settings, and ties in 0 settings
The most frequent outcome is in bold

	BUSL			MSL		
	Win	Loss	Tie	Win	Loss	Tie
Actor	2	2	1	3	2	0
Director	5	0	0	3	2	0
Genre	5	0	0	3	2	0
Male	3	2	0	2	3	0
Movie	0	5	0	2	3	0
WorkedUnder	2	3	0	3	2	0
Total	**17**	12	1	**16**	14	0

Table 17 AUC results for learning WebKB models

In comparison to BUSL, LSL wins in 12 of the 20 settings, loses in 8 settings, and ties in 0 settings. In comparison to MSL, LSL wins in 9 of the 20 settings, loses in 11 settings, and ties in 0 settings
The most frequent outcome is in bold

	BUSL			MSL		
	Win	Loss	Tie	Win	Loss	Tie
CourseProf	2	2	0	2	2	0
CourseTA	2	2	0	3	1	0
Faculty	1	3	0	0	4	0
Project	4	0	0	3	1	0
Student	3	1	0	1	3	0
Total	**12**	8	0	9	**11**	0

Table 18 CLL results for learning WebKB models

	BUSL			MSL		
	Win	Loss	Tie	Win	Loss	Tie
CourseProf	4	0	0	3	1	0
CourseTA	4	0	0	4	0	0
Faculty	2	2	0	0	4	0
Project	3	1	0	3	1	0
Student	2	2	0	1	3	0
Total	**15**	5	0	**11**	9	0

In comparison to BUSL, LSL wins in 15 of the 20 settings, loses in 5 settings, and ties in 0 settings. In comparison to MSL, LSL wins in 11 of the 20 settings, loses in 9 settings, and ties in 0 settings
The most frequent outcome is in bold

Table 19 AUC results for learning UWCSE models

	BUSL			MSL		
	Win	Loss	Tie	Win	Loss	Tie
Advisedby	3	2	0	4	1	0
Courselevel	1	4	0	3	2	0
Phase	2	3	0	5	0	0
Professor	0	4	1	1	4	0
Publication	5	0	0	4	1	0
Ta	4	1	0	3	2	0
Taughtby	2	3	0	3	2	0
Tempadvisedby	4	1	0	5	0	0
Yearsinprogram	5	0	0	3	2	0
Total	**26**	18	1	**31**	14	0

In comparison to BUSL, LSL wins in 26 of the 45 settings, loses in 18 settings, and ties in 1 setting. In comparison to MSL, LSL wins in 31 of the 45 settings, loses in 14 settings, and ties in 0 settings
The most frequent outcome is in bold

Table 20 CLL results for learning UWCSE models

	BUSL			MSL		
	Win	Loss	Tie	Win	Loss	Tie
Advisedby	5	0	0	5	0	0
Courselevel	5	0	0	4	1	0
Phase	0	5	0	1	4	0
Professor	1	4	0	1	4	0
Publication	4	1	0	4	1	0
Ta	5	0	0	4	1	0
Taughtby	5	0	0	2	3	0
Tempadvisedby	5	0	0	5	0	0
Yearsinprogram	3	2	0	2	3	0
Total	**33**	12	0	**28**	17	0

In comparison to BUSL, LSL wins in 33 of the 45 settings, loses in 12 settings, and ties in 0 settings. In comparison to MSL, LSL wins in 28 of the 45 settings, loses in 17 settings, and ties in 0 settings
The most frequent outcome is in bold

References

Ahmadi, B., Kersting, K., & Natarajan, S. (2012). Lifted online training of relational models with stochastic gradient methods. In *Proceedings of the 2012 European conference on machine learning and principles and practice of knowledge discovery in databases*.

Beame, P., Van den Broeck, G., Gribkoff, E., & Suciu, D. (2015). Symmetric weighted first-order model counting. In *Proceedings of the 34th symposium on principles of database systems*.

Bell, E. (1934). Exponential numbers. *American Mathematical Monthly*, *41*, 411–419.

Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician*, *24*, 179–195.

Boyd, K., Santos Costa, V., Davis, J., & Page, D. (2012). Unachievable region in precision-recall space and its effect on empirical evaluation. In *Proceedings of the 29th international conference on machine learning*.

Bui, H. B., Huynh T. N., & de Salvo Braz, R. (2012). Exact lifted inference with distinct soft evidence on every object. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*.

Chavira, M., & Darwiche, A. (2008). On probabilistic inference by weighted model counting. *Artificial Intelligence*, *172*(6–7), 772–799.

Chechetka, A., & Guestrin, C. (2007). Efficient principled learning of thin junction trees. *Advances in Neural Information Processing Systems*, *20*, 273–280.

Darwiche, A. (2009). *Modeling and reasoning with bayesian networks*. Cambridge: Cambridge University Press.

Darwiche, A., & Marquis, P. (2002). A knowledge compilation map. *Journal of Artificial Intelligence Research*, *17*(1), 229–264.

De Raedt, L., Frasconi, P., Kersting, K., & Muggleton, S. (2008). *Probabilistic inductive logic programming: Theory and applications*. Berlin: Springer.

de Salvo Braz, R., Amir, E., & Roth, D. (2005). Lifted first-order probabilistic inference. In *Proceedings of the 19th international joint conference on artificial intelligence* (pp. 1319–1325).

Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*, 380–392.

Domingos, P., & Webb, W. A. (2012). Tractable Markov logic. In *Proceedings of the 26th AAAI conference on artificial intelligence*.

Getoor, L., & Taskar, B. (Eds.). (2007). *An introduction to statistical relational learning*. Cambridge, MA: MIT Press.

Gogate, V., & Domingos, P. (2011). Probabilistic theorem proving. In *Proceedings of the 27th conference on uncertainty in artificial intelligence*.

Huynh, T. N., & Mooney, R. J. (2009). Max-margin weight learning for Markov logic networks. In *Proceedings of the 2009 European conference on machine learning and principles and practice of knowledge discovery in databases* (pp. 564–579).

Jaeger, M., & Van den Broeck, G. (2012). Liftability of probabilistic inference: upper and lower bounds. In *Proceedings of the 2nd international workshop on statistical relational AI*.

Jaimovich, A., Meshi, O., & Friedman, N. (2007). Template based inference in symmetric relational markov random fields. In *Proceedings of the 23rd conference on uncertainty in artificial intelligence* (pp. 191–199).

Jaimovich, A., Meshi, O., McGraw, I., & Elidan, G. (2010). FastInf: An efficient approximate inference library. *Journal of Machine Learning Research*, *11*, 1733–1736.

Kersting, K. (2012). Lifted probabilistic inference. In *Proceedings of the 20th European conference on artificial intelligence*.

Kersting, K., Ahmadi, B., & Natarajan, S. (2009). Counting belief propagation. In *Proceedings of the 25th conference on uncertainty in artificial intelligence* (pp. 277–284).

Kok, S., & Domingos, P. (2005). Learning the structure of Markov logic networks. In *Proceedings of the 22nd international conference on machine learning* (pp. 441–448).

Kok, S., & Domingos, P. (2010). Learning Markov logic networks using structural motifs. In *Proceedings of the 27th international conference on machine learning* (pp. 551–558).

Kok, S., Sumner, M., Richardson, M., Singla, P., Poon, H., Lowd, D., & Domingos, P. (2008). *The alchemy system for statistical relational AI*. Tech. Rep. Seattle, WA: Department of Computer Science and Engineering, University of Washington, http://alchemy.cs.washington.edu

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. Cambridge, MA: MIT Press.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, *45*(3), 503–528.

Lowd, D., & Domingos, P. (2007). Efficient weight learning for Markov logic networks. In *Proceedings of the 11th conference on the practice of knowledge discovery in databases* (pp. 200–211).

Lowd, D., & Domingos, P. (2008). Learning arithmetic circuits. In *Proceedings of the 24th conference on uncertainty in artificial intelligence*.

Lowd, D., & Rooshenas, A. (2013). Learning Markov networks with arithmetic circuits. In *Proceedings of the 16th international conference on artificial intelligence and statistics* (pp. 406–414).

Mihalkova, L., & Mooney, R. J. (2007). Bottom-up learning of Markov logic network structure. In *Proceedings of the 24th international conference on machine learning* (pp. 625–632).

Milch, B., Zettlemoyer, L. S., Kersting, K., Haimes, M., & Kaelbling, L. P. (2008). Lifted probabilistic inference with counting formulas. In *Proceedings of the 23rd AAAI conference on artificial intelligence* (pp. 1062–1068).

Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. Cambridge, MA: MIT Press.

Narasimhan, M., & Bilmes, J. (2004). PAC-learning bounded tree-width graphical models. In *Proceedings of the 20th conference on uncertainty in artificial intelligence* (pp. 410–417).

Niepert, M., & Van den Broeck, G. (2014). *Tractability through exchangeability: A new perspective on efficient probabilistic inference*. arXiv:1401.1247.

Park, J. D. (2002). MAP complexity results and approximation methods. In *Proceedings of the 18th conference on uncertainty in artificial intelligence* (pp. 388–396).

Poole, D. (2003). First-order probabilistic inference. In *Proceedings of the 18th international joint conference on artificial intelligence* (pp. 985–991).

Poole, D., Bacchus, F., & Kisynski, J. (2011). *Towards completely lifted search-based probabilistic inference*. arXiv:1107.4035.

Poon, H., & Domingos, P. (2006). Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the 21st national conference on artificial intelligence* (pp. 458–463).

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*(1), 107–136.

Rota, G. C. (1964). The number of partitions of a set. *The American Mathematical Monthly*, *71*(5), 498–504.

Singla, P., & Domingos, P. (2005). Discriminative training of Markov logic networks. In *Proceedings of the 20th national conference on artificial intelligence* (pp. 868–873).

Singla, P., & Domingos, P. (2008). Lifted first-order belief propagation. In *Proceedings of the 23rd AAAI conference on artificial intelligence* (pp. 1094–1099).

Taghipour, N., & Davis, J. (2012). Generalized counting for lifted variable elimination. In *Proceedings of the 2nd international workshop on statistical relational AI*.

Taghipour, N., Fierens, D., Van den Broeck, G., Davis, J., & Blockeel, H. (2013). Completeness results for lifted variable elimination. In *Proceedings of the 16th international conference on artificial intelligence and statistics* (pp. 572–580).

Van den Broeck, G. (2011). On the completeness of first-order knowledge compilation for lifted probabilistic inference. *Advances in Neural Information Processing Systems*, *24*, 1386–1394.

Van den Broeck, G. (2013). *Lifted inference and learning in statistical relational models*. PhD thesis, KU Leuven.

Van den Broeck, G., & Darwiche, A. (2013). On the complexity and approximation of binary evidence in lifted inference. *Advances in Neural Information Processing Systems*, *26*, 2868–2876.

Van den Broeck, G., Davis, J. (2012). Conditioning in first-order knowledge compilation and lifted probabilistic inference. In *Proceedings of the 26th AAAI conference on artificial intelligence*.

Van den Broeck, G., Taghipour, N., Meert, W., Davis, J., & De Raedt, L. (2011). Lifted probabilistic inference by first-order knowledge compilation. In *Proceedings of the 22nd international joint conference on artificial intelligence* (pp. 2178–2185).

Van den Broeck, G., Choi, A., & Darwiche, A. (2012). Lifted relax, compensate and then recover: From approximate to exact lifted probabilistic inference. In *Proceedings of the 28th conference on uncertainty in artificial intelligence*.

Van den Broeck, G., Meert, W., & Davis, J. (2013). Lifted Generative Parameter Learning. In *Proceedings of the 3rd International Workshop on Statistical Relational AI*.

Van den Broeck, G., Meert, W., & Darwiche, A. (2014). Skolemization for weighted first-order model counting. In *Proceedings of the 14th international conference on principles of knowledge representation and reasoning*.