Machine Learning manuscript No.
(will be inserted by the editor)

Learning Graphical Models for Relational Data via Lattice Search

Oliver Schulte and Hassan Khosravi

Received: date / Accepted: date

Abstract Many machine learning applications that involve relational databases incorporate first-order logic and probability. Relational extensions of generative graphical models include Parametrized Bayes Net [44] and Markov Logic Networks (MLNs). Many of the current state-of-the-art algorithms for learning MLNs have focused on relatively small datasets with few descriptive attributes, where predicates are mostly binary and the main task is usually prediction of links between entities. This paper addresses what is in a sense a complementary problem: learning the structure of a graphical model that models the distribution of discrete descriptive attributes given the links between entities in a relational database. Descriptive attributes are usually nonbinary and can be very informative, but they increase the search space of possible candidate clauses. We present an efficient new algorithm for learning a Parametrized Bayes Net that performs a level-wise search through the table join lattice for relational dependencies. From the Bayes net we obtain an MLN structure via a standard moralization procedure for converting directed models to undirected models. Learning MLN structure by moralization is 200-1000 times faster and scores substantially higher in predictive accuracy than benchmark MLN algorithms on three relational databases.

Keywords Statistical-Relational Learning · Graphical Models · Markov Logic Networks · Bayes Nets

1 Introduction

Many databases store data in relational format, with different types of entities and information about links between the entities. The field of statistical-relational learning (SRL) has developed a number of new statistical models for relational

School of Computing Science
Simon Fraser University
Vancouver-Burnaby, B.C., V5A 1S6
Canada
E-mail: oschulte@cs.sfu.ca,hkhosrav@cs.sfu.ca
Tel.: +1-778-782-3390
Fax: +1-778-782-3045

databases [18]. Markov Logic Networks (MLNs) form one of the most prominent SRL model classes; they generalize both first-order logic and Markov network models [8]. MLNs have achieved impressive performance on a variety of SRL tasks. Because they are based on undirected graphical models, they avoid the difficulties with cycles that arise in directed SRL models [40, 8, 56]. An open-source benchmark system for MLNs is the Alchemy package [30]. Essentially, an MLN is a set of weighted first-order formulas that compactly defines a Markov network comprising ground instances of logical predicates. The formulas are the structure or qualitative component of the Markov network; they represent associations between ground facts. The weights are the parameters or quantative component; they assign a likelihood to a given relational database by using the log-linear formalism of Markov networks. This paper addresses structure learning for MLNs in relational schemas that feature a significant number of descriptive attributes, compared to the number of relationships. Previous MLN learning algorithms do not scale well with such datasets. We introduce a new *moralization approach* to learning MLNs: first we learn a directed Bayes net graphical model for relational data, then we convert the directed model to an undirected MLN model using the standard moralization procedure (marry spouses, omit edge directions). The main motivation for performing inference with an undirected model is that they do not suffer from the problem of cyclic dependencies in relational data [8, 56, 23]. Thus our approach combines the scalability and efficiency of directed model search, with the inference power and theoretical foundations of undirected relational models.

Approach. We present a new algorithm for learning a Bayes net from relational data, the *learn-and-join* algorithm. While our algorithm is applicable to learning directed relational models in general, we base it on the Parametrized Bayes Net formalism of Poole [44]. The learn-and-join algorithm performs a level-wise model search through the table join lattice associated with a relational database, where the results of learning on subjoins constrain learning on larger joins. The join tables in the lattice are (i) the original tables in the database, and (ii) joins of relationship tables, with information about descriptive entity attributes added by joining entity tables. A single-table Bayes net learner, which can be chosen by the user, is applied to each join table to learn a Bayes net for the dependencies represented in the table. For joins of relationship tables, this Bayes net represents dependencies among attributes conditional on the existence of the relationships represented by the relationship tables.

Single-table Bayes net learning is a well-studied problem with fast search algorithms. Moreover, the speed of single-table Bayes net learning is significantly increased by providing the Bayes net learner with constraints regarding which edges are required and which are prohibited. Key among these constraints are the *join constraints*: the Bayes net model for a larger table join inherits the presence or absence of edges, and their orientation, from the Bayes nets for subjoins. We present a theoretical analysis that shows that, even though the number of potential edges increases with the number of join tables, the join constraint reduces the Bayes net model search space to keep it roughly constant size throughout the join lattice. In addition to the join constraints, the relational structure leads to several others that reduce search complexity; we discuss the motivation for these constraints in detail. One of the constraints addresses recursive dependencies (relational autocorrelations of an attribute on itself) by restricting the set of nodes

that can have parents (i.e., indegree greater than 0) in a Parametrized Bayes Net. A normal form theorem shows that under mild conditions, this restriction involves no loss of expressive power.

Evaluation. We evaluated the structures obtained by moralization using one synthetic dataset and five public domain datasets. In our experiments on small datasets, the run-time of the learn-and-join algorithm is 200-1000 times faster than benchmark programs in the Alchemy framework [28] for learning MLN structure. On medium-size datasets, such as the MovieLens database, almost none of the Alchemy systems returns a result given our system resources, whereas the learn-and-join algorithm produces an MLN with parameters within 2 hours; most of this time (98%) is spent optimizing the parameters for the learned structure. To evaluate the predictive performance of the learned MLN structures, we used the parameter estimation routines in the Alchemy package. Using standard prediction metrics for MLNs, we found in empirical tests that the predictive accuracy of the moralized BN structures was substantially greater than that of the MLNs found by Alchemy. Our code and datasets are available for ftp download [1].

Limitations. The main limitation of our current algorithm is that it does not find associations between links, for instance that if a professor advises a student, then they are likely to be coauthors. In the terminology of Probabilistic Relational Models [16], our algorithm addresses attribute uncertainty, but not existence uncertainty (concerning the existence of links). The main ideas of this paper can also be applied to link prediction.

Another limitation is that we do not propose a new weight learning method, so we use standard Markov Logic Network methods for parameter learning after the structure has been learned. While these methods find good parameter settings, they are slow and constitute the main computational bottleneck for our approach.

Paper Organization. We review related work, then statistical-relational models, especially Parametrized Bayes nets and Markov Logic Networks. We define the table join lattice, and present the learn-and-join algorithm for Parametrized Bayes nets. We provide detailed discussion of the relational constraints used in the learn-and-join algorithm. For evaluation, we compare the moralization approach to standard MLN structure learning methods implemented in the Alchemy system, both in terms of processing speed and in terms of model predictive accuracy.

Contributions. The main contributions may be summarized as follows.

1. A new structure learning algorithm for Bayes nets that model the distribution of descriptive attributes given the link structure in a relational database. The algorithm is a level-wise lattice search through the space of join tables.
2. Discussion and justification for relational constraints that speed up structure learning.
3. A Markov logic network structure can be obtained by moralizing the Bayes net. We provide a comparison of the moralization approach with other MLN methods.

2 Additional Related Work.

A preliminary version of the Learn-and-Join Algorithm was presented by Khosravi *et al.* [23]. The previous version did not use the lattice search framework. Our new version adds constraints on the model search, makes all constraints explicit, and provides rationales and discussion of each. We have also added more comparison with other Markov Logic Network learning methods (e.g., BUSL, LSM) and a lesion study that assesses the effects of using only part of the components of our main algorithm. Our approach to autocorrelations (recursive dependencies) was presented by Schulte *et al.* [52]. The main idea is to use a restricted form of Bayes net that we call the main functor node format. This paper examines how the main functor node format can be used in the context of the overall relational structure learning algorithm.

The syntax of other directed SRL models, such as Probabilistic Relational Models (PRMs) [16], Bayes Logic Programs (BLPs) [22] and Logical Bayesian Networks [11], is similar to that of Parametrized Bayes Nets [44]. Our approach applies to directed SRL models generally.

Nonrelational structure learning methods. Schmidt *et al.* [49] compare and contrast structure learning algorithms in directed and undirected graphical methods for nonrelational data, and evaluate them for learning classifiers. Domke *et al.* provide a comparison of the two model classes in computer vision [9]. Tillman *et al.* [58] provide the ION algorithm for merging graph structures learned on different datasets with overlapping variables into a single partially oriented graph. It is similar to the learn-and-join algorithm in that it extends a generic single-table BN learner to produce a BN model for a set of data tables. One difference is that the ION algorithm is not tailored towards relational structures. Another is that the learn-and-join algorithm does not analyze different data tables completely independently and merge the result afterwards. Rather, it recursively constrains the BN search applied to join tables with the adjacencies found in BN search applied to the respective joined tables.

Lattice Search Methods. The idea of organizing model/pattern search through a partial order is widely used in data mining, for instance in the well-known Apriori algorithm, in statistical-relational learning [46] and in Inductive Logic Programming (ILP) [32]. Search in ILP is based on the θ-subsumption or specialization lattice over clauses. Basically, a clause c specializes another clause c' if c adds a condition or if c replaces a 1st-order variable by a constant. The main similarity to the lattice of relationship joins is that extending a chain of relationships by another relationship is a special case of clause specialization. The main differences are as follows. (1) Our algorithm uses only a lattice over chains of relationships, not over conditions that combine relationships with attributes. Statistical patterns that involve attributes are learned using Bayes net techniques, not by a lattice search. (2) ILP methods typically stop specializing a clause when local extensions do not improve classification/prediction accuracy. Our algorithm considers all points in the relationship lattice. This is feasible because there are usually only a small number of different relationship chains, due to foreign key constraints.

Since larger table joins correspond to larger relational neighborhoods, lattice search is related to iterative deepening methods for statistical-relational learning

[40, Sec.8.3.1], [6]. The main differences are as follows. (1) Current statistical-relational learning methods do not treat dependencies learned on smaller relational neighborhoods as constraining those learned on larger ones. Thus dependencies learned for smaller neighborhoods are revisited when considering larger neighborhoods. In principle, it appears that other statistical-relational learning methods could be adapted to use the relationship join lattice with inheritance constraints as in our approach. (2) To assess the relevance of information from linked entities, statistical-relational learning methods use aggregate functions (e.g., the average grade of a student in the courses they have taken), or combining rules (e.g., noisy-or) [22,39]. In Probablistic Relational Models, Bayes Logic Programs, and related models, the aggregate functions/combining rules add complexity to structure learning. In contrast, our statistical analysis is based on table joins rather than aggregation. Like Markov Logic Networks, our algorithm does not require aggregate functions or combining rules, although it can incorporate them if required.

MLN structure learning methods. Current methods [28,37,20,4] successfully learn MLN models for binary predicates (e.g., link prediction), but do not scale well to larger datasets with descriptive attributes that are numerous and/or have a large domain of values. Mihalkova and Mooney [37] distinguish between top-down approaches, that follow a generate-and-test strategy of evaluating clauses against the data, and bottom-up approaches that use the training data and relational pathfinding to construct candidate conjunctions for clauses. In principle, the BN learning module may follow a top-down or a bottom-up approach; in practice, most BN learners use a top-down approach. The BUSL algorithm [37] employs a single-table Markov network learner as a subroutine. The Markov network learner is applied once after a candidate set of conjunctions and a data matrix has been constructed. In contrast, we apply the single-table BN learner repeatedly, where results from earlier applications constrain results of later applications.

We briefly describe the key high-level differences between our algorithm and previous MLN structure learning methods, focusing on those that lead to highly efficient relational learning.

Search Space. Previous Markov Logic Network approaches have so far followed Inductive Logic Programming techniques that search the space of clauses. Clauses define connections between *atoms* (e.g. $intelligence = hi, gpa = low$). Descriptive attributes introduce a large number of atoms, one for each combination of attribute and value, and therefore define a large search space of clauses. We utilize Bayes net learning methods that search the space of links between *predicates/functions* (e.g., $intelligence, gpa$), rather than atoms. Associations between predicates constitute a smaller model space than clauses that can be searched more efficiently. The efficiency advantages of searching the predicate space rather than the clause space are discussed by Kersting and deRaedt [22, 10.7].

Constraints and the Lattice Structure. We employ a number of constraints that are motivated by the relational semantics of the data. These further reduce the search space, mainly by requiring or forbidding edges in the Bayes net model. A key type of constraint are based on the lattice of relationship chains: These state that edges learned when analyzing shorter chains are inherited by longer chains. This allows the learn-and-join algorithm to perform a local statistical analysis for

a single point in the relationship chain lattice, while connecting the results of the local analyses with each other.

Data Representation and Lifted Learning. The data format used by Markov Logic Networks is a list of ground atoms, whereas the learn-and-join algorithm analyzes data tables. This allows us to apply directly propositional Bayes net learners which take single tables as input. From a statistical point of view, the learn-and-join algorithm requires only the specification of the frequency of events in the database (the sufficient statistics in the database) [50]. The data tables provide these statistics. In the case of join tables the statistics are frequencies conditional on the existence of a relationship (e.g., the percentage of pairs of friends who both smoke). The learn-and-join algorithm can be seen as performing *lifted learning*, in analogy to lifted probabilistic inference [44]. Lifted inference uses as much as possible frequency information defined at the class level in terms of 1st-order variables, rather than facts about specific individuals. Likewise, the learn-and-join algorithm uses frequency information defined in terms of 1st-order variables (namely the number of satisfying groundings of a 1st-order formula).

3 Background and Notation

Our work combines concepts from relational databases, graphical models, and Markov Logic networks. As much as possible, we use standard notation in these different areas. Section 3.5 provides a set of examples illustrating the concepts.

3.1 Logic and Functors

Parametrized Bayes nets are a basic statistical-relational learning model; we follow the original presentation of Poole [44]. A **functor** is a function symbol or a predicate symbol. Each functor has a set of values (constants) called the **range** of the functor. A functor whose range is $\{T, F\}$ is a **predicate**, usually written with uppercase letters like P, R. A **functor random variable** is of the form $f(\tau_1, \ldots, \tau_k)$ where f is a functor and each term τ_i is a first-order variable or a constant. We also refer to functor random variables as **functor nodes**, or for short **fnodes**.[1] Unless the functor structure matters, we refer to a functor node simply as a node. If functor node $f(\boldsymbol{\tau})$ contains no variable, it is **ground**, or a **gnode**. An assignment of the form $f(\boldsymbol{\tau}) = a$, where a is a constant in the range of f, is an **atom**; if $f(\boldsymbol{\tau})$ is ground, the assignment is a **ground atom**. A **population** is a set of individuals, corresponding to a domain or type in logic. Each first-order variable X is associated with a population $\mathcal{P}_X$ of size $|\mathcal{P}_X|$; in the context of functor nodes, we refer to **population variables** [44]. An **instantiation** or **grounding** γ for a set of variables $X_1, \ldots, X_k$ assigns a constant $\gamma(X_i)$ from the population of X_i to each variable X_i.

Getoor and Grant discuss the applications of function concepts for statistical-relational modelling in detail [17]. The functor formalism is rich enough to represent the constraints of an entity-relationship (ER) schema [59] via the follow-

[1] The term "functor" is used as in Prolog [5]. In Prolog, the equivalent of a functor random variable is called a "structure". Poole [44] refers to a functor random variable or fnode as a "parametrized random variable". We use the term fnode for brevity.

ing translation: Entity sets correspond to populations, descriptive attributes to functions, relationship tables to predicates, and foreign key constraints to type constraints on the arguments of relationship predicates. A **table join** of two or more tables contains the rows in the Cartesian products of the tables whose values match on common fields.

We assume that a database instance (interpretation) assigns a unique constant value to each gnode $f(\mathbf{a})$. The value of descriptive relationship attributes is well defined only for tuples that are linked by the relationship. For example, the value of $\mathit{grade}(\mathit{jack}, \mathit{101})$ is not well defined in a university database if $\mathit{Registered}(\mathit{jack}, \mathit{101})$ is false. In this case, we follow the approach of Schulte *et al.* [51] and assign the descriptive attribute the special value $\perp$ for "undefined". Thus the atom $\mathit{grade}(\mathit{jack}, \mathit{101}) = \perp$ is equivalent to the atom $\mathit{Registered}(\mathit{jack}, \mathit{101}) = F$. Fierens *et al.* [11] discuss other approaches to this issue. The results in this paper extend to functors built with nested functors, aggregate functions [25], and quantifiers; for the sake of notational simplicity we do not consider more complex functors explicitly.

3.2 Bayes Nets and Markov Nets for Relational Data and Markov Logic Networks

We employ notation and terminology from [43] for graphical models. Russell and Norvig provide a textbook introduction to many of the topics we review [48]. A **Bayes net structure** is a directed acyclic graph (DAG) G, whose nodes comprise a set of random variables denoted by V. In this paper we consider only discrete finite random variables. When discussing a Bayes net structure, we refer interchangeably to its nodes or its variables. A **family** in a Bayes net graph comprises a child node and the set of its parents. A Bayes net (BN) is a pair $\langle G, \boldsymbol{\theta}_G \rangle$ where $\boldsymbol{\theta}_G$ is a set of parameter values that specify the probability distributions of children conditional on assignments of values to their parents. The conditional probabilities are specified in a **conditional probability table**. For an assignment of values to all nodes in the Bayes net, the joint probability of the values is given by the product of the associated conditional probabilities. A Parametrized Bayes Net is a Bayes net whose nodes are functor nodes. In the remainder of this paper we follow [50] and use the term **Functor Bayes Net** or **FBN** instead of Parametrized Bayes Net, for the following reasons. (1) To emphasize the use of functor symbols. (2) To avoid confusion with the statistical meaning of "parametrized", namely that values have been assigned to the model parameters. We usually refer to FBNs simply as Bayes nets.

A **Markov net structure** is an undirected graph whose nodes comprise a set of random variables. For each clique C in the graph, a **clique potential function** Ψ_C specifies a nonnegative real number for each possible assignment of values to the clique. For an assignment of values to all nodes in the Markov net, the joint probability of the values is given by the product of the associated clique potentials, divided by a normalization constant. A **Functor Markov Net** is a Markov net whose nodes are functor nodes.

Bayes nets can be converted into Markov nets through the standard **moralization** method: connect all spouses that share a common child, and make all edges in the resulting graph undirected. Thus each family in the Bayes net becomes a clique in the moralized structure. For each state of each family clique, we define

the clique potential in the Markov net to be the conditional probability of the child given its parents. The resulting Markov net defines the same joint probability over assignments of values to the nodes as the original Bayes net.

3.3 Inference and Ground Models

In statistical-relational learning, the usual approach to inference for relational data is to use a ground graphical model for defining a joint distribution over the attributes and links of entities. This approach is known as *knowledge-based model construction* [42,31,61]. For a Functor Markov Net M, this leads to the notion of a **ground** Functor Markov net that is derived from M by instantiating the functor nodes in M in every possible way. Formally, there is an edge $f_1(\boldsymbol{a}_1) - f_2(\boldsymbol{a}_2)$ between two gnodes if and only if there is an edge $f_1(\boldsymbol{\tau}_1) - f_2(\boldsymbol{\tau}_2)$ in M and there is a grounding γ of $\boldsymbol{\tau}_1, \boldsymbol{\tau}_2$ such that $\gamma(\boldsymbol{\tau}_i) = \boldsymbol{a}_i$, for $i = 1, 2$.

Each clique among gnodes inherits the clique potential from the clique potential of the 1st-order model. A given database instance specifies a value for each node in the ground graph. Thus the likelihood of the Functor Markov net for the database can be defined as the likelihood assigned by the ground Markov net to the facts in the database following the usual product of all clique potentials involving ground nodes. Viewed on a log-scale, this is the sum of the log-potentials.

In the case where the Functor Markov net is obtained by moralizing a Functor Bayes net, the resulting log-likelihood is as follows: For each possible child-parent state in the Bayes net, multiply the logarithm of the corresponding conditional probability by the number of instantiations of the child-parent states in the database. This is similar to the standard single-table Bayes net log-likelihood, where for each possible child-parent state in the Bayes net, we multiply the logarithm of the corresponding conditional probability by the number of table rows that satisfy the given child-parent state.

The fact that the grounding semantics provides a conceptually straightforward way to define probabilistic inferences for relational data has been a major competitive advantage of undirected relational models [8,56]. Below, we discuss the difficulties that can arise in applying the grounding semantics with directed models, making reference to some examples, which we introduce in the next subsection.

3.4 Markov Logic Networks

A **Markov Logic Network** (MLN) is a finite set of 1st-order formulas or clauses $\{\phi_i\}$, where each formula ϕ_i is assigned a weight. A Markov Logic Network can be viewed as a specification of a Markov network using logical syntax [8]. Given an MLN and a database $\mathcal{D}$, let $n_i(\mathcal{D})$ be the number of groundings that satisfy ϕ_i in $\mathcal{D}$. An MLN assigns a log-likelihood to a database according to the equation

$$ln(P(\mathcal{D})) = \sum_i w_i n_i(\mathcal{D}) - ln(Z) \tag{1}$$

where Z is a normalization constant.

Thus the log-likelihood is a weighted sum of the number of groundings for each clause. Functor Markov Nets have a simple representation as Markov Logic

Networks as follows. For each assignment of values to a clique of functor nodes, add a conjunctive formula to the MLN that specifies that assignment. The weight of this formula is the logarithm of the clique potential. For any Functor Markov net, the MLN likelihood function defined by Equation (1) for the corresponding MLN is exactly the Markov net likelihood defined by grounding the Functor Markov net. *Therefore we can use MLN inference to carry out inference for Functor Markov Nets.*

3.5 Examples

We illustrate Functor Bayes Nets and Markov Logic Networks with two relational schemas.

Friendship Database. Figure 1 shows a simple database instance in the ER format, following [8]. Figure 2 illustrates Functor Bayes net concepts. An example of a family formula with child node $Smokes(Y)$ is

$$Smokes(Y) = T, Smokes(X) = T, Friend(X, Y) = T.$$

Figure 3 shows the MLN structure obtained by moralization and the corresponding ground Markov net for the database of Figure 1. For converting the Bayes net conditional probabilities to MLN clause weights, Domingos and Richardson suggest using the log of the conditional probabilities as the clause weight [8, 12.5.3], which is the standard conversion for propositional Bayes nets. Figure 3 illustrates moralization using log-probabilities as weights. In this paper we apply moralization only to the model structure, not to the model parameters. Table 1 shows how the unnormalized log-likelihood of the sample database is computed for the ground model.

People

Name	Smokes	Cancer
Anna	T	T
Bob	T	F

Friend

Name1	Name2
Anna	Bob
Bob	Anna

Fig. 1 A simple relational database instance.

University Database. Table 2 shows a university relational schema and Figure 4 a Functor Bayes net for this schema.

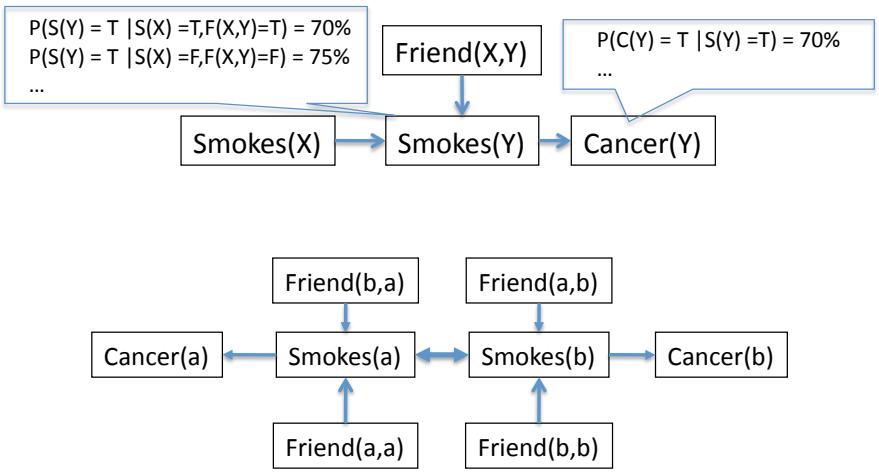

Fig. 2 A Functor Bayes Net and its grounding for the database of Figure 1. The double arrow $\leftrightarrow$ is equivalent to two directed edges. Conditional probability parameters are chosen arbitrarily for illustration.

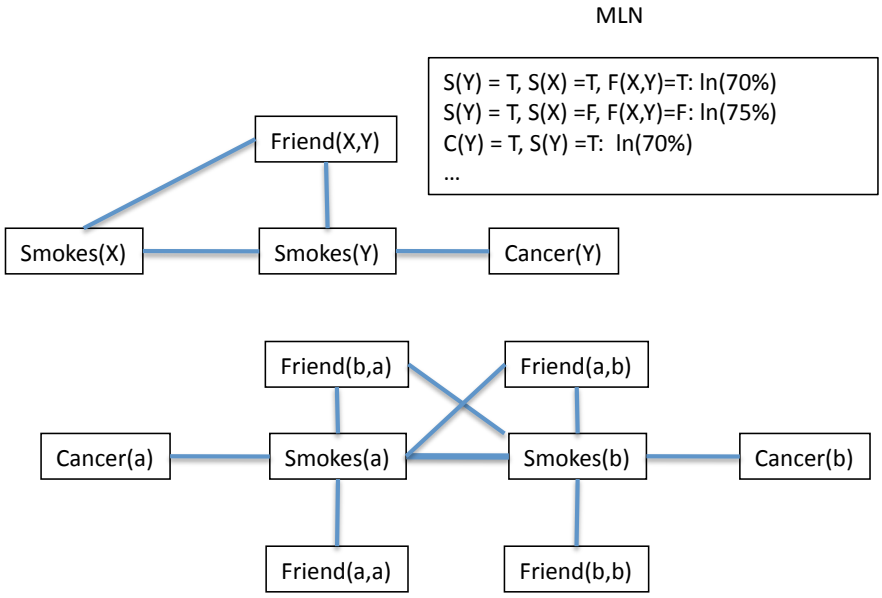

Fig. 3 Top: The moralized Functor Bayes net of Figure 2. On the right, it shows the clauses and the weights for the corresponding Markov Logic Network. The clauses specify the family states in the Functor Bayes net. Each clause weight is the logarithm of the conditional probability that corresponds to the clause. In graphical terms, these are the log-clique potentials for the clique comprising the nodes $Smokes(X), Friend(X, Y), Smokes(Y)$. Bottom: The ground Markov network for the database of Figure 1.

Table 1 The computation of the Markov Net log-likelihood for the database of Figure 1. For simplicity, we used uniform probabilities as probability parameters for the nodes $Friend(X, Y)$ and $Smokes(X)$.

Clique Formula	log-potential	#groundings	log-potential × groundings
Smokes(X)=T	ln(50%)	2	-1.39
Friend(X,Y)=T	ln(50%)	2	-1.39
Friend(X,Y)=F	ln(50%)	2	-1.39
Friend(X,Y)=T,Smokes(X)=T,Smokes(Y)=T	ln(70%)	2	-0.71
Friend(X,Y)=F,Smokes(X)=T,Smokes(Y)=T	ln(100%)	2	0
Cancer(Y)=T, Smokes(Y) = T	ln(70%)	1	-0.36
Cancer(Y)=F, Smokes(Y) = T	ln(30%)	1	-1.20
Total unnormalized log-likelihood			-6.43

$Student(\underline{student\_id}, intelligence, ranking)$
$Course(\underline{course\_id}, difficulty, rating)$
$Professor\ (\underline{professor\_id}, teaching\_ability, popularity)$
$Registered\ (\underline{student\_id}, \underline{course\_id}, grade, satisfaction)$
$Teaches(\underline{professor\_id}, \underline{course\_id})$
$RA\ (\underline{student\_id}, \underline{prof\_id}, salary, capability)$
$TA\ (\underline{course\_id}, \underline{student\_id}, capability)$

Table 2 A relational schema for a university domain. Key fields are underlined. The RA and TA relations are not used in all examples.

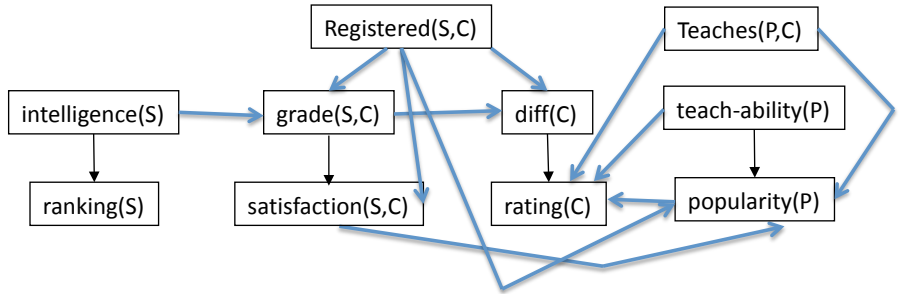

Fig. 4 A Functor Bayes net graph for the relational schema of Table 2 (without TA and RA relations).

3.6 Directed Models and the Cyclicity Problem

Knowledge-based model construction with directed models is defined by instantiating the variables in the graph with all possible constants, as with undirected models [44], [48, Ch.14.6]. Two key issues arise.

(1) *The Combining Problem.* The directed model with population variables represents generic statistical relationships found in the database. For instance, a Bayes net may encode the probability that a student is highly intelligent given the properties of a single course they have taken. But the database may contain information about many courses that the student has taken, which needs to be combined. In terms of the ground graph, the problem is how to define the conditional probability of a child gnode given its set of multiple parents. For example, in the Bayes Net of Figure 2, the gnode $Smokes(a)$ will have a separate parent

$Smokes(y)$ for each person y instantiating the population variable Y. (Since our toy database contains only two people, there is only one parent gnode $Smokes(b)$.) Addressing the combining problem requires an aggregate function, as in PRMs, or a combining rule as in BLPs, or the log-linear formula of MLNs.

(2) *The Cyclicity Problem.* A directed model may face cyclic dependencies between the properties of individual entities. For example, if there is generally a correlation between the smoking habits of friends, then we may have a situation where the smoking of Jane predicts the smoking of Jack, which predicts the smoking of Cecile, which predicts the smoking of Jane, where Jack, Jane, and Cecile are all friends with each other. In the presence of such cycles, neither aggregate functions nor combining rules lead to well-defined probabilistic predictions. Figure 2 shows a cycle of length 2 between the two gnodes $Smokes(a)$ and $Smokes(b)$. This model also illustrates how cycles arise in the presence of relationships that relate entities of the same type, as *Friend* relates two people. Such relationships are called rings in Entity-Relationship models [59] and are called **self-relationships** by Heckerman, Koller, Meek [19]. Self-relationships typically give rise to **autocorrelations** where the value of an attribute for an entity depends on the value of the same attribute among related entities [41, 52]. For instance, in the ground Bayes net of Figure 2, the value of $Smokes(a)$ depends on the value of *Smokes* for other people.

Because cycles are not allowed in a valid Bayes net graph, grounding Functor Bayes nets that include self-relationships does not lead to a valid distribution for carrying out probabilistic inference. This cyclicity problem has been difficult to solve, which has led Neville and Jensen to conclude that "the acyclicity constraints of directed models severely limit their applicability to relational data" [40, p.241].

The approach of this paper is essentially a hybrid method that uses directed models for learning and undirected models for inference. The idea is to use scalable Bayes net algorithms to learn a Functor Bayes net, then convert the Bayes net to a Markov Logic network for inference using moralization. Converting the Bayes net to an undirected model avoids the cyclicity problem. Thus the approach of this paper combines advantages of both directed and undirected SRL models: Learning efficiency and interpretability from directed models on the one side, and on the other, the solutions to the combining and cyclicity problems together with the inference power of undirected models. The graph of Figure 5 summarizes the system architecture.

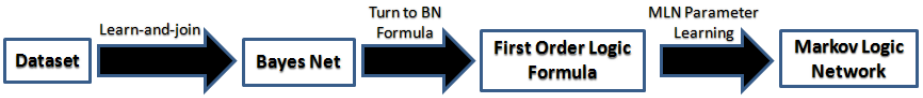

Fig. 5 System architecture for learning a Markov Logic Network from an input relational database.

4 Lattice Search for Attribute Dependencies

We describe the learn-and-join method for learning a Functor Bayes net that models correlations among descriptive attributes *given the link structure*. We begin with the data structures that the algorithm uses to represent relational objects. Then we give pseudocode for the algorithm and illustrate it with an example.

4.1 Overview

The components of the algorithm address the following main issues. (1) The representation of relationship sets. (2) Bounding the complexity of relational contexts. (3) Avoiding duplicate edges. (4) Propagating constraints from smaller relationship sets in the multinet lattice to larger ones.

Compared to the previous presentation of the learn-and-join algorithm by Khosravi *et al.* [23], we make two main changes. (i) We define and discuss the constraints on the graph structure that are used in the algorithm, separately from the description of the model search procedure. (ii) We describe the model search procedure as a lattice search, where the lattice points are chains of relationships. Conceptually, the lattice view makes the description simpler and more general without losing rigor. Computationally, the lattice diagram facilitates the implementation of the model search.

4.2 The Multinet Lattice

With each point in the relationship lattice, we associate a Bayes net model and a join data table. Thus the lattice structure defines a *multinet* rather than a single Bayes net. Multinets are a classic Bayes net formalism for modelling *context-sensitive* dependencies among variables. They have been applied for modelling diverse domains, such as sleep disorders, eye diseases, and turbines that generate electricity. Geiger and Heckerman contributed a standard reference article for the multinet formalism [15]. In their illustrative example, a building guard watches for three different types of people, visitors, spies, and workers. The existence of a dependency between the gender of a person and whether they wear an identification badge depends on the type of person. This scenario is modelled with three multinets, one for each type of person. The type of person is the context for the corresponding multinet.

Going back to the classic work of Ngo and Haddaway on context-sensitive dependencies in relational data [42], directed relational models usually include resources for representing the influence of context on dependencies [42, 11, 17, 39, 19, 48]. A common approach is to use a logical language for stating context conditions as well as dependencies between random variables. In this paper we employ multinets rather than context-sensitive rules for two reasons: (1) To stay close to standard graphical approaches for nonrelational Bayes nets. (2) To ensure that the dependencies found for a given context define a valid acyclic Bayes net structure.

In the learn-and-join algorithm, a context is defined by a chain of relationship functor nodes. Distinguishing these different contexts allows us to represent that the existence of certain dependencies among attributes of entities depend on which

kind of links exist between the entities. The final output of the learn-and-join algorithm is a single Bayes net derived from the multinet. As we discuss in Section 4.3, the output of the algorithm can also be converted to other formalisms, including those based on rules for context-sensitive dependencies.

4.2.1 Functor Nodes

Throughout the discussion we assume that a set of functor random variables F is fixed. The random variables F are partitioned into (i) functor nodes representing descriptive attributes of entities, (ii) functor nodes representing descriptive attributes of links, (iii) Boolean relationship functor nodes that indicate whether a relationship holds. Descriptive attribute functor nodes (i) take as arguments a single population variable, whereas the relational functor nodes (ii) and (iii) take as arguments two or more population variables. We make the following assumptions about the functor nodes F that appear in a Functor Bayes net.

1. A functor node contains variables only.
2. No functor node contains the same variable X twice.

These assumptions are met in typical SRL models. They do not actually involve a loss of modelling power because a functor node with a constant or a repeated variable can be rewritten using a new functor symbol (provided the functor node contains at least one variable). For instance, a functor node $\mathit{Friend}(X, \mathit{jack})$ can be replaced by introducing a new unary functor symbol $\mathit{Friend}_{\mathit{jack}}(X)$. Similarly, $\mathit{Friend}(X, X)$ can be replaced by the unary functor symbol $\mathit{Friend}_{\mathit{self}}(X)$. The functor node set F may be explicitly specified by the user or automaticallly generated from a relational database schema [23].

Examples. The nodes of the Bayes net of Figure 4 are the functor nodes generated from the schema of Table 2 with one population variable per entity type (e.g., S for *Student*). Self-relationships require two population variables of the same kind. This is illustrated in the Bayes net of Figure 2, which contains two population variables for the Person entity type X and Y for the self-relationship *Friend*. This allows the Bayes net to represent an autocorrelation involving the *Smokes* attribute: Given that person X is friends with person Y, the smoking habits of X predict those of Y.

4.2.2 Relationship Chains

A relationship set is a **chain** if it can be ordered as a list $[R_1(\boldsymbol{\tau}_1), \ldots, R_k(\boldsymbol{\tau}_k)]$ such that each functor $R_{i+1}(\boldsymbol{\tau}_{i+1})$ shares at least one population variable with the preceding terms $R_1(\boldsymbol{\tau}_1), \ldots, R_i(\boldsymbol{\tau}_i)$. All sets in the lattice are constrained to form a chain. For instance, in the University schema of Table 2, a chain is formed by the two relationship nodes

$$\mathit{RA}(P, S), \mathit{Registered}(S, C).$$

If relationship node $\mathit{TA}(C, S)$ is added, we may have a three-element chain

$$\mathit{RA}(P, S), \mathit{Registered}(S, C), \mathit{TA}(C, S).$$

The subset relation defines a lattice on relationship sets/chains. Figure 6 illustrates the lattice for the relationship nodes in the University schema of Figure 2. For reasons that we explain below, entity tables are also included in the lattice and linked to relationships that involve the entity in question.

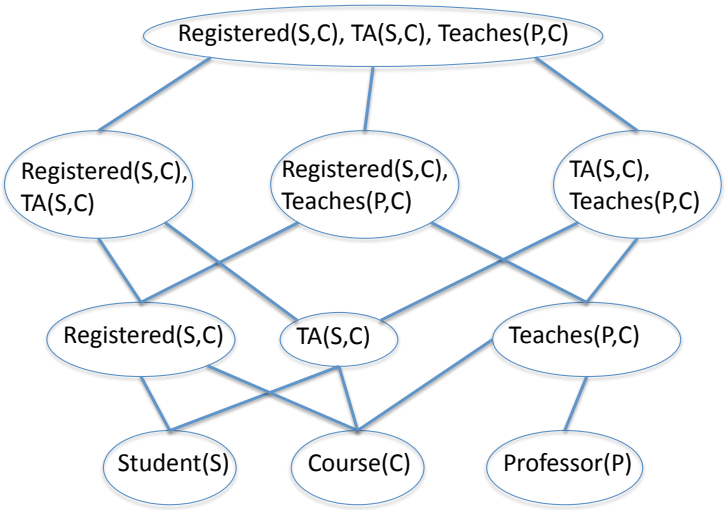

Fig. 6 A lattice of relationship sets for the University schema of Table 2 (without the RA relation). Links from entity tables to relationship tables correspond to foreign key pointers.

The concept of a relationship chain is related to but different from the notion of a slot chain as used in Probabilistic Relational Models [16]. The main difference is that a slot chain can connect entity tables as well as relationship tables. Thus a path from the Student table to the Registered table to the Course table constitutes a slot chain of length 3, but contains only a single relationship (relationship chain of length 1).

4.2.3 The Join Data Table

With each relationship set $\mathbf{R}$ is associated a join data table $\bowtie_{\mathbf{R}}$. The table represents the frequencies (sufficient statistics) with which combinations of attribute values occur, conditional on all relationships in $\mathbf{R}$ being true. Let R_i denote the data table associated with relationship functor $R_i(\boldsymbol{\tau}_i)$. For relationship functors $\mathbf{R} = \{R_1(\boldsymbol{\tau}_1), \ldots, R_k(\boldsymbol{\tau}_k)\}$ let $X_1, \ldots, X_l$ be the population variables that occur in the k relationship functors, and write E_j for the entity data table associated with the population variable X_j. Then the join table for the relationship set, or **relationship join**, is given by

$$\bowtie_{\mathbf{R}} \equiv \bowtie_{i=1}^{k} R_i \bowtie_{j=1}^{l} E_j.$$

If two or more variables are associated with the same population, then the same descriptive attribute will appear at least twice in the relationship join. In

this case we disambiguate the names of the descriptive attributes by adding the variable as their argument. Similarly, we add variables to disambiguate repeated occurrences of descriptive link attributes. Thus each column label in the relationship join corresponds to exactly one functor node. For each relationship set $\mathbf{R} = \{R_1(\boldsymbol{\tau}_1), \ldots, R_k(\boldsymbol{\tau}_k)\}$, the nodes in the associated Bayes net $B_{\mathbf{R}}$ are the column labels in $\bowtie_{\mathbf{R}}$, plus Boolean relationship indicator nodes $R_1(\boldsymbol{\tau}_1), \ldots, R_k(\boldsymbol{\tau}_k)$.

Examples. For the relationship chain $RA(P, S), Registered(S, C)$, the join data table is given by

$$RA \bowtie Registered \bowtie Professor \bowtie Student \bowtie Course.$$

The join data table associated with the relationship functor $Friend(X, Y)$—shown in Figure 4.2.3—is given by

$$Friend \bowtie People \bowtie People.$$

Name1 (=X)	Name2 (=Y)	S(X)	C(X)	S(Y)	C(Y)
Anna	Bob	T	T	T	F
Bob	Anna	T	F	T	T

Fig. 7 The join data table associated with the *Friend* relationship for the database instance of Figure 1.

4.3 Model Conversion

The output of the lattice search is the Bayes net associated with the largest relationship chain that forms the apex of the relationship lattice. The Bayes net of Figure 4 is associated with the relationship set $Registered(S, C), Teaches(P, C)$, which is the maximal conjunction of both relationship functors in this functor set. In Figure 6 the maximally large relationship set has three members. To obtain a Markov Logic Network, we convert the maximal Bayes net to an MLN using moralization. The Bayes net can similarly be converted to other clausal formalisms like BLPs and LBNs, since a Functor Bayes net defines a set of directed clauses of the form $child \leftarrow parents$ [22]. The existence of a link of a certain type can be taken as a context condition in rule languages where associations between random variables depend on a context.

5 The Learn-And-Join Algorithm

This section presents the Learn-and-Join Algorithm (LAJ) that takes as input a relational database and constructs a Bayes multinet for each relationship set in the multinet lattice. The algorithm enumerates relationship lists. This can be done using any standard technique, such as those developed for enumerating itemsets in association rule mining [2]. It proceeds level-wise by considering relationship sets of length $s = 1, 2, \ldots$. After Bayes nets have been learned for sets of length s, the learned edges are propagated to sets of length $s + 1$. In the initial case of single relationship tables where $s = 1$, the edges are propagated from Bayes nets learned for entity tables. In addition to the edge constraints, the algorithm enforces a number of constraints that are motivated by the relational structure of the functor nodes.

We next provide a compact description of the constraints used, including their definition, an intuitive interpretation and examples. Then we show by examples how the constraints operate. Finally, we summarize the algorithm with pseudocode as Algorithm 1. Later sections discuss the constraints in detail, including motivation, mathematical analysis and references to related concepts that have appeared in the literature.

5.1 Constraints Used in the Learn-And-Join Algorithm

The constraints fall into two types: relational constraints that capture the semantics of the relational schema, and lattice constraints on the presence/absence of edges that connect the results of learning from different points in the relationship set lattice.

The algorithm requires the specification of a main population variable for each entity type (e.g., Y for *People*). The intuitive interpretation is that the distribution of attributes for that entity type is modelled by functor nodes that involve the main variable, whereas other functor nodes play an auxilliary role (e.g., the distribution of the *Smokes* attribute is modelled by the functor node $Smokes(Y)$ rather than the functor node $Smokes(X)$).

5.1.1 Edge Inheritance In the Relationship Lattice

These constraints state that the presence or absence of edges in graphs associated with join tables lower in the lattice is inherited by graphs associated with join tables higher in the lattice. The intuition is that dependencies should be assessed in the most specific context possible. First edges from an entity table are inherited by relationship tables that involve the entity in question.

Constraint 1 *Let X be the main population variable for an entity type associated with entity table E. Let $\mathbf{R}$ be any relationship set that contains the variable X. Then the Bayes net associated with $\mathbf{R}$ contains an edge $f(X) \rightarrow g(X)$ connecting two descriptive attributes of X if and only if the Bayes net associated with E contains the edge $f(X) \rightarrow g(X)$.*

Example. If the *People* Bayes net contains an edge $Smokes(Y) \rightarrow Cancer(Y)$, then the Bayes net associated with the relationship *Friend* must also contain this

edge (see Figure 2). If the edge is absent in the *People* Bayes net, it must also be absent in the Bayes net associated with the relationship *Friend*.

The next constraint states that edges learned on smaller relationship sets are inherited by larger relationship sets. If the smaller sets are ambiguous with regard to the direction of an adjacency, the larger relationship set must contain the adjacency; the direction is then resolved by applying Bayes net learning to the larger relationship set.

Constraint 2 *Suppose that nodes* $f(\boldsymbol{\tau}), g(\boldsymbol{\tau}')$ *appear in the join table* $\bowtie_{\mathbf{R}}$*. Then*

1. *If* $f(\boldsymbol{\tau})$ *and* $g(\boldsymbol{\tau}')$ *are not adjacent in any graph* $B_{\mathbf{R}^*}$ *associated with a relationship subset* $\mathbf{R}^* \subset \mathbf{R}$*, then* $f(\boldsymbol{\tau})$ *and* $g(\boldsymbol{\tau}')$ *are not adjacent in the graph associated with the relationship set* $\mathbf{R}$*.*
2. *Else if all subset graphs agree on the orientation of the adjacency* $f(\boldsymbol{\tau}) - g(\boldsymbol{\tau}')$*, the graph associated with the relationship set* $\mathbf{R}$ *inherits this orientation.*
3. *Else the graph associated with the relationship set* $\mathbf{R}$ *must contain the edge* $f(\boldsymbol{\tau}) \rightarrow g(\boldsymbol{\tau}')$ *or the edge* $f(\boldsymbol{\tau}) \leftarrow g(\boldsymbol{\tau}')$*.*

Examples. Consider the lattice shown in Figure 6. Suppose that the graph associated with the relationship *Registered*(S, C) contains an edge

$$\textit{difficulty}(C) \rightarrow \textit{intelligence}(S),$$

and that the graph associated with the relationship *TA*(S, C) does *not* contain the edge *difficulty*$(C) \rightarrow$ *intelligence*(S). Then the edge *difficulty*$(C) \rightarrow$ *intelligence*(S) must be present in the Bayes net associated with the larger relationship set *Registered*(S, C), *TA*(S, C). If the edge is contained in neither of the graphs associated with *Registered*(S, C), and *TA*(S, C), it must not be present in the graph associated with the *Registered*(S, C), *TA*(S, C).

5.1.2 The Main Functor Node Format

Following Schulte *et al.* [52], the algorithm requires the specification of a main functor node for each functor (e.g., *Smokes*(Y) is the main functor node for the functor *Smokes*). Only main functor nodes are allowed to have edges pointing into them (i.e., indegree greater than 0). The intuition behind this constraint is that it suffices to model the conditional distribution of just one "copy" of the functor. For example, to model the conditional distribution of the *Smokes* attribute, it suffices to have parents only for the functor node *Smokes*(Y), rather than allow parents for both the functor node *Smokes*(Y) and *Smokes*(X).

Constraint 3 *For any Bayes net associated with a relationship set, if its graph contains an edge* $\rightarrow f(\boldsymbol{\tau})$ *pointing into a node* $f(\boldsymbol{\tau})$*, then* $f(\boldsymbol{\tau})$ *is the main functor node for* f*.*

Example. Suppose that *Smokes*(Y) is the main functor node for *Smokes*. Then the main functor constraint permits the edges *Smokes*$(X) \rightarrow$ *Smokes*(Y) and *Friend*$(X, Y) \rightarrow$ *Smokes*(Y), but rules out the edges *Smokes*$(Y) \rightarrow$ *Smokes*(X) and *Friend*$(X, Y) \rightarrow$ *Smokes*(X).

5.1.3 Population Variable Bound

We allow the user to specify a bound on the number of population variables that occur in a family (child + parent nodes). Intuitively, this bounds the number of distinct (generic) objects that can be considered in a single child-parent configuration. For instance, if the bound is 1, the family expresses patterns only about a single entity. With 2 population variables, patterns involving pairs can be expressed, with 3 triples can be modelled, etc.

Examples. For the node *Cancer*(Y) of Figure 2, its family contains a single population variable Y, so only patterns involving a generic person can be represented. For the node *Smokes*(Y), its family contains two population variables X and Y, so patterns involving pairs of people can be represented.

We emphasize that a variable number bound does *not* imply a bound on the size of families, or the length of clauses: even with a single population variable like S for students, we can have an arbitrary number of attributes of students in a single clause. Kok and Domingos [29] highlight the importance of learning long clauses for relational models.

5.1.4 Link Attributes

There is a deterministic dependency between a Boolean relationship indicator node and a descriptive attribute associated with the relationship: If the relationship does not exist between two entities, then the value of the descriptive attribute is undefined. In our representation, this means that the descriptive attribute takes on the value $\perp$ for undefined (cf. Section 3.1). This deterministic connection can be enforced given the following graphical constraint.

Constraint 4 *Suppose that f_R denotes a descriptive attribute of relationship R and that $f_R(\boldsymbol{\tau})$ is the main functor node for f_R and $R(\boldsymbol{\tau})$ is the main functor node for R. Then there is an edge $R(\boldsymbol{\tau}) \rightarrow f_R(\boldsymbol{\tau})$ in any Bayes net that contains $f_R(\boldsymbol{\tau})$.*

Examples. In the Bayes net of Figure 4, the functors *satisfaction* and *grade* denote descriptive attributes of the *Registered* relationship. So the Bayes net must contain edges *Registered*(S, C) $\rightarrow$ *satisfaction*(S, C) and *Registered*(S, C) $\rightarrow$ *grade*(S, C), which are the main nodes for their respective functors.

5.1.5 Relationship Parents

An edge $f(\boldsymbol{\tau}) \rightarrow g(\boldsymbol{\tau}')$ in a Bayes net $B_{\mathbf{R}}$ represents an influence that depends on the existence of the links in the relationship chain $\mathbf{R}$. To make this dependence explicit in the Bayes net graph, we add the members of $\mathbf{R}$ as parents to the child node $g(\boldsymbol{\tau}')$.

Constraint 5 *Suppose that an edge $f(\boldsymbol{\tau}) \rightarrow g(\boldsymbol{\tau}')$ appears in a graph $B_{\mathbf{R}}$ but not in any graph associated with a subset of $\mathbf{R}$, or in any graph associated with an entity table/population that occurs in $\mathbf{R}$. Then the graph $B_{\mathbf{R}}$ contains an edge $R(\boldsymbol{\tau}^*) \rightarrow g(\boldsymbol{\tau}')$ for each relationship indicator node $R(\boldsymbol{\tau}^*) \in \mathbf{R}$.*

Example. Consider the lattice shown in Figure 6. Suppose that the graph associated with the relationship $Registered(S, C)$ contains an edge

$$difficulty(C) \rightarrow intelligence(S).$$

Since this edge is not contained in either of the graphs associated with the entity tables *Student* or *Course*, the constraint requires an edge

$$Registered(S, C) \rightarrow intelligence(S).$$

Constraint 5 is not essential to the learning performance. Rather, adding the edges originating in relationship nodes has the effect of changing the representation of context-sensitivity from the multi-net format to a single-net format, which is the target output of the learn-and-join algorithm. If the target is another output model format, this constraint may be replaced by another formalism for representing context-sensitive associations (cf. Sections 4.2 and 4.3).

To complete the description of the constraints, we present examples of how the constraints operate in the learn-and-join algorithm. Later sections provide further discussion and analysis.

5.2 Examples.

We illustrate the learn-and-join algorithm on the example database of Figure 1; see Figure 8. The *TA* and *RA* relations are omitted for simplicity.

1. Applying the single-table Bayes net learner to the *People* table may produce a single-edge graph $Smokes(Y) \rightarrow Cancer(Y)$.
2. Then form the join data table

   $$J = Friend \bowtie People \bowtie People$$

   shown in Figure 4.2.3. The Bayes net learner is applied to J, with the following constraints.
   (a) From the *People* Bayes net, there must be an edge $Smokes(Y) \rightarrow Cancer(Y)$, where Y is the main population variable associated with *People*. (Constraint 1).
   (b) No edges may point into $Smokes(X)$ or $Cancer(X)$, since these are not the main functor nodes for the functors *Smokes* and *Cancer*. (Constraint 3).

The Bayes net learner applied to the join table J then may find an edge $Smokes(X) \rightarrow Smokes(Y)$. Since the dependency represented by this edge is valid only for pairs of people that are friends (i.e., conditional on $Friend(X, Y) = T$), the algorithm adds an edge $Friend(X, Y) \rightarrow Smokes(Y)$ (Constraint 5), whose associated Bayes net is shown in Figure 8.

Figure 9 shows the multinet for the University schema up to level 1. Continuing the construction up to the highest level 2 produces a single Bayes net for the maximal relationship set $Registered(S, C)$, $Teaches(P, C)$ that is shown in Figure 4.

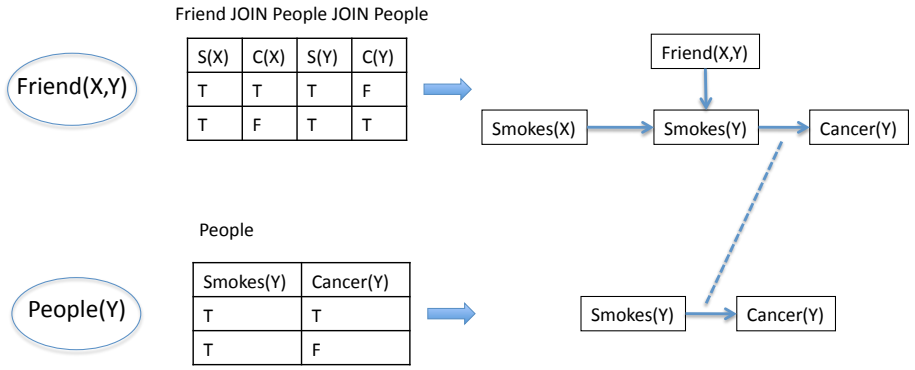

Fig. 8 The 2-net lattice associated with the DB instance of Figure 1. The figure shows the data tables associated with the only entity table *People* and the only relationship table *Friend*. The block arrow indicates that the output of a single-table Bayes net learner on the data table is the Bayes net shown. The dashed line that connects the two edges $Smokes(Y) \rightarrow Cancer(Y)$ indicates that this edge is propagated from the lower-level Bayes net to the higher-level Bayes net.

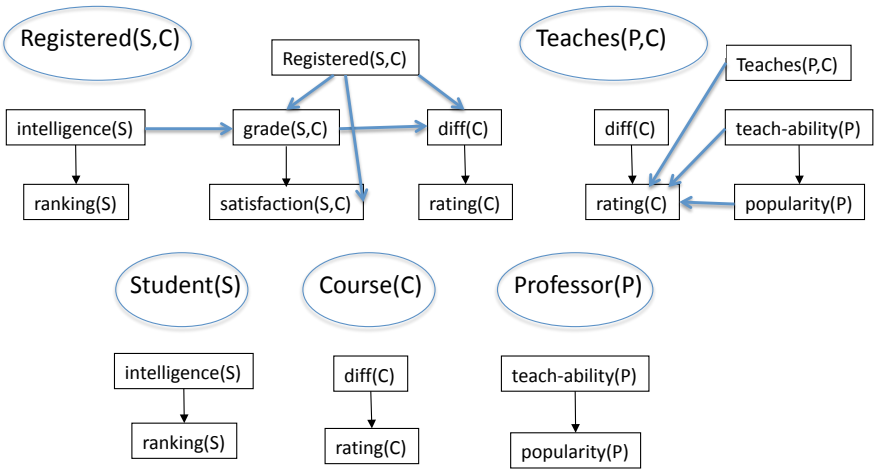

Fig. 9 The multinet lattice for the University Schema, restricted to entity and relationship functors. Join data tables are not shown. We omit the *TA* and *RA* relationships for simplicity.

5.3 Pseudocode

Algorithm 1 combines all the algorithm's components in pseudocode. We next discuss the constraints in more detail. The discussion can be skipped without loss of continuity.

Algorithm 1: Pseudocode for structure learning with lattice search

Input: Database $\mathcal{D}$ with entity tables $E_1, ..E_e$; functors F; #variable bound $maxVar$.
Output: MLN formulas for $\mathcal{D}$; a Bayes multi-net $B_{\mathbf{R}}$ for relationship subsets $\mathbf{R}$ of F.
Calls BNL: Any propositional Bayes net learner that accepts edge constraints and a single table of cases as input.
Notation: $BNL(T, Econstraints)$ is the output DAG of the Bayes net learner given data table T and constraints $Econstraints$.

```
Econstraints := Forbidden + Required Edges from Constraint 3 and Constraint 4.
for i ← 1 to e do
    B_{E_i} := BNL(E_i, Econstraints).
end for
Econstraints += Lattice Constraints from entity tables (Constraint 1).
for list size s ← 1, 2 ... do
    Enumerate relationship sets R_1, ..., R_{s_k} of size s, such that for each i:
    (1) R_i is a chain.
    (2) the number of population variables in R_i is no greater than maxVar.
    if there is no such list of size s then
        terminate computation
    else
        for i ← 1 to s_k do
            B_{R_i} := BNL(⋈_{R_i}, Econstraints) + edges originating in the R_i functors
            (Constraint 5).
        end for
    end if
    Econstraints += Lattice Constraints from relationship joins of size s (Constraint 2).
end for
Let B_max be the Bayes net associated with the maximal relationship set.
Add all family formulas of B_max to MLN.
```

6 Discussion: Lattice Constraints

A key feature of the learn-and-join algorithm are the lattice inheritance constraints 1 and 2 that a Bayes net for a table join must respect the links found for the joined tables. We describe a computational and a statistical motivation for them.

Computational Efficiency. The edge-inheritance constraint reduces the search complexity considerably. To illustrate, consider the impact of Constraint 1 for two entity tables that contain k descriptive attributes each. In an unconstrained join with a relationship table, the search space of possible adjacencies has size $\binom{2k}{2}$, whereas with the constraint, the search space size is $k^2/2$, which is smaller than $\binom{2k}{2}$ because the quadratic k^2 factor has a smaller coefficient. For example, with $k = 6$, we have $\binom{2k}{2} = 66$ and $k^2/2 = 18$. For the learn-and-join algorithm, the main computational challenge in scaling to larger table joins is therefore not the increasing number of columns (attributes) in the join, but only the increasing number of rows (tuples).

Statistical Motivation. In addition to efficiency, a statistical motivation for the edge-inheritance constraint 1 is that the marginal distribution of descriptive attributes may be different in an entity table than in a relationship table. For instance, if a highly intelligent student s has taken 10 courses, there will be at least ten satisfying groundings of the conjunction $Registered(S, C), intelligence(S) = hi$.

If highly intelligent students tend to take more courses than less intelligent ones, then in the *Registered* table, the frequency of tuples with intelligent students is higher than in the general student population. In general, the distribution of database frequencies conditional on a relationship being true may be different from its unconditional distribution. The edge inheritance constraint ensures that the subgraph of the final Bayes net whose nodes correspond to the attributes of the E table is exactly the same as the graph that the single-table Bayes net learner constructs for the E table.

The motivation for Constraint 2 is similar: a dependency ought to be evaluated on a minimal context. For instance, the presence of an edge $intelligence(S) \rightarrow difficulty(C)$ given that $Registered(S, C) = T$ ought to depend only on the *Registered* relationship and not on a relationship that involves another object, such as a TA for the course (i.e., the edge is inherited in the larger context $Registered(S, C) = T, TA(C, G)$).

A further statistical foundation is provided by a plausible pseudo-likelihood function that measures the fit of a Bayes Nets to a given input database [50]. The relational pseudo log-likelihood is defined just like the regular single-table log-likelihood for a Bayes net, with the database frequency of a parent-child state replacing the number of rows that feature the parent-child state. Schulte shows that the learn-and-join algorithm optimizes the pseudo-likelihood function [50].

7 Discussion: The Main Functor Node Format

The functor concept allows different nodes in a Functor Bayes net to be associated with the same attribute or relationship, where the difference between the nodes is in their variable arguments only. This expressive power is essential to represent recursive dependencies where instances of an attribute/relationship depend on other instances of the same attribute/relationship. However, it causes additional complexity in learning if each functor is treated as a separate random variables. Consider for example the Bayes net shown in Figure 10.

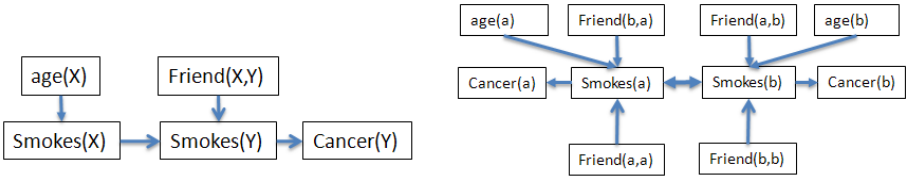

Fig. 10 A Bayes net with different predictors for $Smokes(X)$ and $Smokes(Y)$, and its grounding for two individuals a and b. The Bayes net is not in main functor node form.

If we treat $Smokes(X)$ and $Smokes(Y)$ as entirely separate variables, learning needs to consider additional edges similar to those already in the Bayes net, like $Smokes(X) \rightarrow Cancer(X)$ and $age(Y) \rightarrow Smokes(Y)$. However, such edges are redundant because the population variables X and Y are interchangeable as they refer to the same entity set. In terms of ground instances, the two exchanges connect exactly the same ground instances. Redundant edges can be avoided if

we restrict the model class to the main functor format, where for each function symbol f (including relationships), there is a *main functor node* $f(\boldsymbol{\tau})$ such that all other functor nodes $f(\boldsymbol{\tau}')$ associated with the same functor are sources in the graph, that is, they have no parents. The term "main functor node" expresses that the main node is the main instance of functor f from the point of view of modelling the conditional distribution of f.

Example. The Bayes net of Figure 2, reproduced in Figure 11, is in main functor form. The Bayes net of Figure 10 is not in main functor node form because we have two functor nodes for *Smokes* with nonzero indegree.

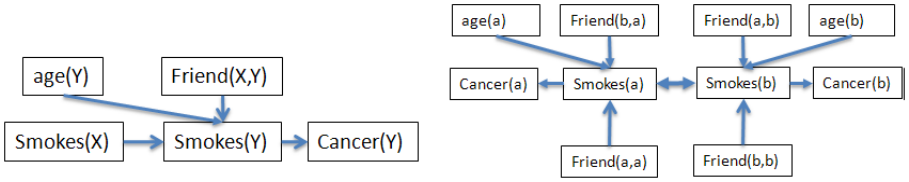

Fig. 11 An Bayes net in main functor node form where $Smokes(Y)$ is the main functor for $Smokes(X)$. The ground Bayes net is the same as the ground Bayes net for the graph of Figure 10.

Schulte *et al.* [52] provide a theoretical justification for the main functor node format: under a mild assumption, every Bayes net B can be transformed into an equivalent Bayes net B' that is in main functor node form. Equivalence here means that both Bayes nets have the same ground graph for any database. A Functor Bayes Net is **stratified** if there is an ordering of the *functors* such that for every edge $f(\boldsymbol{\tau}) \rightarrow g(\boldsymbol{\tau}')$ in the Bayes net, either the functor symbol f precedes the functor symbol g in the ordering, or f and g are the same. Both Bayes nets in Figures 10 and 11 are stratified given the functor ordering $age > Friend > Smokes > Cancer$.

Proposition 1 *Let B be a stratified Bayes net. Then there is a Bayes net B' in main functor node form such that for every database $\mathcal{D}$, the ground graph of B is the same as the ground graph of B'.*

For the proof see [52]. Figures 10 and 11 illustrate the proposition. Stratification is a widely imposed condition on logic programs, because it increases the tractability of reasoning with a relatively small loss of expressive power [33, Sec.3.5],[3]. Related ordering constraints have also appeared in previous statistical-relational models [10,14].

While the transformed Bayes nets have the same groundings, they are not equivalent at the variable or class level. For instance, in the model of Figure 10 the maximum indegree is 2, whereas in the model of Figure 2 the maximum indegree is 3. In effect, the main functor node format moves one or more parents from the auxilliary functor nodes to the main functor node, which produces larger families. In terms of Markov Logic Network clauses that result from moralization, the main functor format therefore leads to longer rules. Our experiments below provide empirical confirmation of this theoretical expectation.

8 Discussion: Population Variable Bound.

As both the statistical and the computational complexity of Functor Bayes nets can be high, it is desirable to allow a user to specify a complexity bound to control the trade-off between expressive power and computational difficulty. A key issue is the length of the relationship chains that the algorithm needs to consider. The number of relationship chains grows exponentially with this parameter. We expect that more distantly related entities carry less information, so many SRL algorithms assume a small bound on the length of possible slot chains, on the order of 3 or so. A less restrictive way to bound the complexity of relationship chains is to allow the user to specify a bound on the number of 1st-order or population variables that occur in a family (child + parent nodes), following a proposal of Vardi [60]. Intuitively, this bounds the number of distinct (generic) objects that can be considered in a single child-parent configuration. The computational complexity of computing sufficient statistics for a relationship set depends on the number of population variables as well: with no bound, the problem is #P-complete [8, Prop.12.4]. With a constant bound, sufficient statistics can be computed in polynomial time [60, 24].

The next section presents empirical evidence about the performance of the learn-and-join algorithm on benchmark datasets.

9 Evaluation: Experimental Design

All experiments were done on a QUAD CPU Q6700 with a 2.66GHz CPU and 8GB of RAM. Our code and datasets are available on the world-wide web [1]. We made use of the following existing implementations.

Single Table Bayes Net Search GES search [7] with the BDeu score as implemented in version 4.3.9-0 of CMU's Tetrad package (structure prior uniform, ESS=10; [57]).

MLN Parameter Learning The default weight training procedure [35] of the Alchemy package [30], Version 30.

MLN Inference The MC-SAT inference algorithm [45] to compute a probability estimate for each possible value of a descriptive attribute for a given object or tuple of objects.

Join Data Tables The join data tables for a given relationship chain are computed using a straightforward SQL inner join over the required relationship and entity tables. Our database management system is MySQL Version 1.2.15.

The computation of the join data tables could be optimized in a number of ways. For instance, like most Bayes net scores, the BDeu score requires only the sufficient statistics, or database frequencies of events. Rather than materializing the entire join data table, we could use indices or the SQL *Count* aggregate function to compute these summary statistics only. We did not include optimizations of the database computations because data management is not the focus of our paper, and we already achieve very fast learning without them. Thus the database is used only to find the set of tuples that satisfy a given join condition (e.g., find the set of (x, y) pairs such that $\mathit{Friend}(X, Y) = T, \mathit{Smokes}(X) = T, \mathit{Smokes}(Y) = F$).

9.1 Datasets

We used two synthetic and five benchmark real-world datasets for our evaluation.

University Database. We manually created a small dataset, based on the schema given in Table 2. The entity tables contain 38 students, 10 courses, and 6 Professors. The *Registered* table has 92 rows and the *RA* table has 25 rows.

University+ Database. To test learning algorithms with autocorrelations, we extended this database with a self-relationship *Friend* between Students and several descriptive attributes of students, such that the ranking and coffee drinking habits of a student are strongly correlated with the ranking and coffee drinking habits of her friends, respectively.

MovieLens Database. The MovieLens dataset is from the UC Irvine machine learning repository. It contains two entity tables: *User* with 941 tuples and *Item* with 1,682 tuples, and one relationship table *Rated* with 80,000 ratings. The *User* table has 3 descriptive attributes *age*, *gender*, *occupation*. We discretized the attribute age into three bins with equal frequency. The table *Item* represents information about the movies. It has 17 Boolean attributes that indicate the genres of a given movie. We performed a preliminary data analysis and omitted genres that have only weak correlations with the rating or user attributes, leaving a total of three genres.

Mutagenesis Database. This dataset is widely used in ILP research [55]. Mutagenesis has two entity tables, *Atom* with 3 descriptive attributes, and *Mole*, with 5 descriptive attributes, including two attributes that are discretized into ten values each (logp and lumo). It features two relationships *MoleAtom* indicating which atoms are parts of which molecules, and *Bond* which relates two atoms and has 1 descriptive attribute.

Hepatitis Database. This dataset is a modified version of the PKDD02 Discovery Challenge database. We adopted the modifications of Frank *et al.* [13], which includes removing tests with null values. It contains data on the laboratory examinations of hepatitis B and C infected patients. The examinations were realized between 1982 and 2001 on 771 patients. The data are organized in 7 tables (4 entity tables, 3 relationship tables and 16 descriptive attributes). They contain basic information about the patients, results of biopsy, information on interferon therapy, results of out-hospital examinations, results of in-hospital examinations.

Mondial Database. This dataset contains data from multiple geographical web data sources [36]. We follow the modification of She *et al.* [54], and use a subset of the tables and features. Our dataset includes a self-relationship table *Borders* that relates two countries.

Dataset	#tuples	#Ground atoms
University	171	513
Movielens	82623	170143
Mutagenesis	15218	35973
Hepatitis	12447	71597
Mondial	814	3366
UW-CSE	2099	3380

Table 3 Size of datasets in total number of table tuples and ground atoms. Each descriptive attribute is represented as a separate function, so the number of ground atoms is larger than that of tuples.

Dataset	#tuples	#Ground atoms
MovieLens1 (subsample)	1468	3485
MovieLens2 (subsample)	12714	27134
Mutagenesis1 (subsample)	3375	5868
Mutagenesis2 (subsample)	5675	9058
Hepatitis1	6162	41335

Table 4 Size of subdatasets in total number of table tuples and ground atoms. Each descriptive attribute is represented as a separate function, so the number of ground atoms is larger than that of tuples.

UW-CSE database. This dataset lists facts about the Department of Computer Science and Engineering at the University of Washington (UW-CSE), such as entities (e.g., Student, Professor) and their relationships (i.e. AdvisedBy, Publication) [8]. The dataset was obtained by crawling pages in the department's Web site (www.cs.washington.edu). Publications and AuthorOf relations were extracted from the BibServ database (www.bibserv.org).

Table 3 lists the databases and their sizes in terms of total number of tuples and number of ground atoms, which is the input format for Alchemy. To convert attribute information from the relational database to ground atoms, we used a key-value representation that introduces a binary predicate for each attribute of entities. The first argument is the id of the entity and the second is the value of the predicate for the entity. For example if the value of attribute *gender* for person Bob is *male*, the input to the Markov Logic Network contains the ground atom $gender(Bob, male)$. Similarly, we introduce a ternary predicate for each attribute of a link.[2]

Because several of the Alchemy systems returned no result on some of the real-world datasets, we formed two subdatabases using standard subgraph subsampling [12]. First, draw a random sample of entities from each entity table. Then restrict the relationship tuples in each subdatabase to those that involve only the selected entities. Table 4 lists the resulting subdatabases and their sizes in terms of total number of tuples and number of ground atoms.

[2] Another data format was explored by Kok and Domingos [27], which introduces a single unary predicate for each possible value of the attribute. To illustrate, for instance the attribute *gender* with values (male, female), is represented with two unary predicates $gender_{male}(Person)$, $Salary_{female}(Person)$. The effect is that the arguments to all predicates are primary keys. Khosravi *et al.* [23] compared MLN learning with the two different data formats and found comparable performance.

9.2 Graph Structures Learned by the Learn-and-Join algorithm

Figure 12 shows the learned Bayes nets for the University, Hepatitis, MovieLens, and Mondial datasets. The graphs illustrate how the learn-and-join algorithm learns models with complex dependency patterns.

10 Moralization vs. Other Structure Learning Methods: Basic Comparisons

We begin with a set of comparisons on standard benchmarks that follows the design and performance metrics of previous MLN structure learning studies [23, 37, 28]. Then we focus on experiments that assess specific components of the learn-and-join algorithm.

10.1 Comparison Systems and Performance Metrics

We compared the learn-and-join algorithm with 4 previous Markov Logic Network structure learning methods implemented in different versions of Alchemy.

MBN uses the learn-and-join algorithm to learn a Functor Bayes net. To perform inference, the Functor Bayes net is converted to a Markov Logic Network using moralization, which we refer to as the Moralized Bayes Net (see Section 3). Weights for the moralized Bayes net are learned using Alchemy's weight learning routine.

MSL uses beam search which begins by adding unit clauses to an Markov Logic Network. MSL then considers all clauses of length two and always maintains the n highest-scoring ones in the set. MSL terminates when it cannot find a clause that improves upon the current Markov Logic Network's score [26].

LHL Lifted Hypergraph Learning [28] uses relational path finding to induce a more compact representation of data, in the form of a hypergraph over clusters of constants. Clauses represent associations among the clusters.

BUSL Applies relational path finding to ground atoms and variabilizes each ground atom in a path. It then constructs a Markov network for the nodes in the path, computes a single data table for the path, and learns edges between the nodes using a single-table Markov network learner [37].

LSM Learning Structural Motifs [29] uses random walks to identify densely connected objects in data, and groups them and their associated relations into a motif.

We use 3 performance metrics: Runtime, Accuracy, and Conditional log likelihood. These measures have been used in previous studies of Markov Logic Network learning [37, 28, 23].

Runtime The time taken to learn the model from the training dataset.

Accuracy (ACC) To define accuracy, we apply MLN inference to predict the probability of an attribute value, and score the prediction as correct if the most probable value is the true one. For example, to predict the gender of person Bob, we apply MLN inference to the atoms $gender(Bob, male)$ and

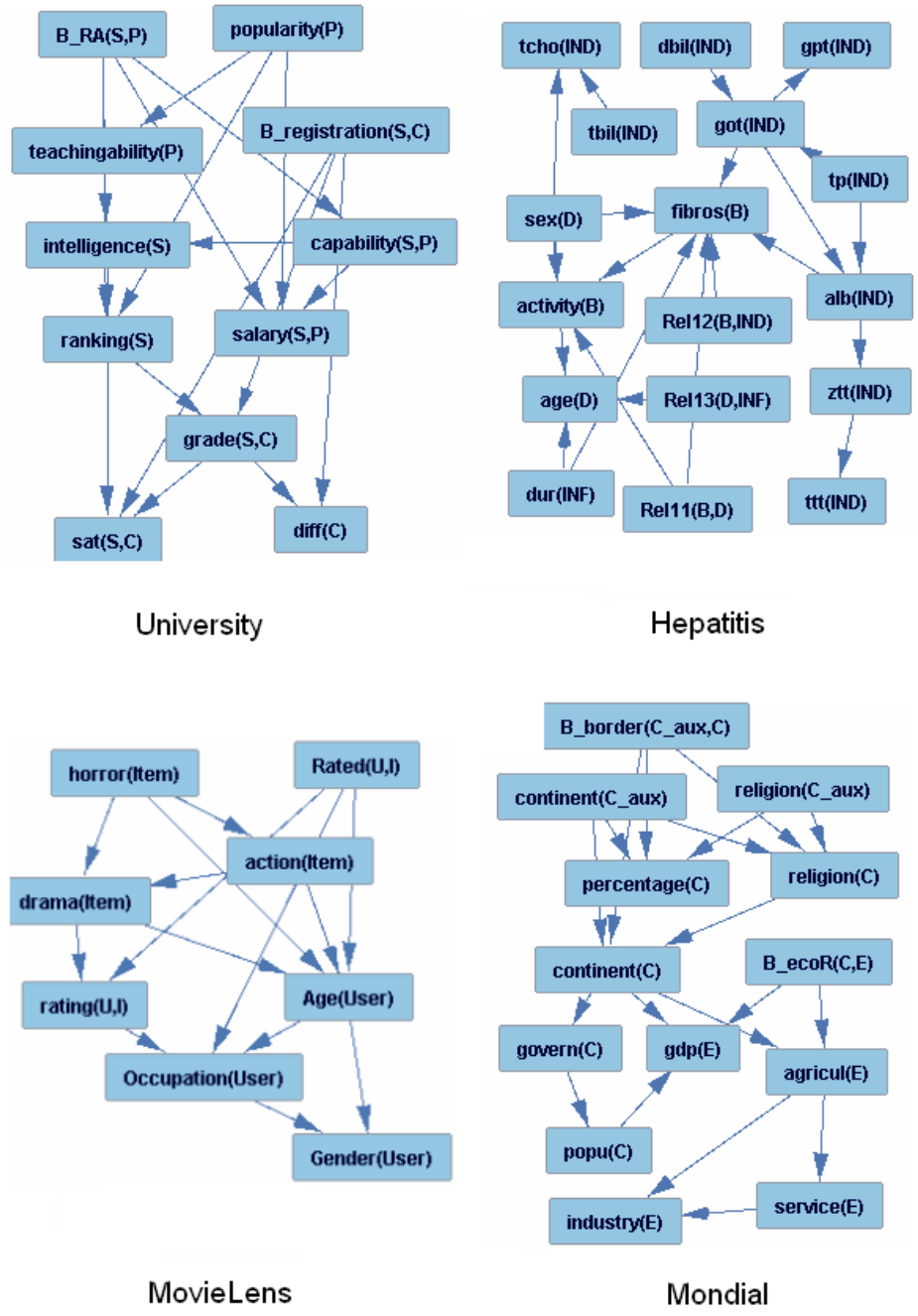

Fig. 12 The Bayes net structures learned by the learn-and-join algorithm for 4 datasets.

$gender(Bob, female)$ (cf. Section 9.1). The result is correct if the predicted probability of $gender(Bob, male)$ is greater than that of $gender(Bob, female)$.

Conditional Log-Likelihood (CLL) The conditional log-likelihood of a ground atom in a database $\mathcal{D}$ given an Markov Logic Network is its log-probability given the Markov Logic Network and $\mathcal{D}$. The CLL directly measures how precise the estimated probabilities are.

For ACC and CLL the values we report are averages over all attribute predicates. Khosravi *et al.* also report the AUC (area-under-curve) for predicates that correspond to descriptive attributes with binary values (e.g. gender), for the databases MovieLens and Mutagenesis [23]. As there are only few binary descriptive attributes, we omit AUC from this study. For the existing binary predicates, the AUC improvement of the MBN approach over previous Markov Logic Network methods is similar to that for ACC and CLL. Mainly to study autocorrelations, we report additional measures for the databases University+, Mondial, and UW-CSE, so our first set of simulations reports results for the remaining databases.

10.2 Runtime Comparison

Table 5 shows the time taken in minutes for learning in each dataset. The Alchemy times include both structure and parameter learning. For the MBN approach, we report both the Bayes net structure learning time and the time required for the subsequent parameter learning carried out by Alchemy.

	Alchemy Methods				LAJ
Dataset	**MSL**	**LHL**	**BUSL**	**LSM**	**MBN**
University	5.02	3.54	0.5	**0.01**	**0.03 + 0.032**
MovieLens	NT	NT	NT	**0.45**	**1.2 +120**
MovieLens1	44	34.52	50	**0.03**	**0.05 + 0.33**
MovieLens2	2760	3349	NT	**0.06**	**0.12 + 5.10**
Mutagenesis	NT	NT	NT	**0.53**	0.5 + 106
Mutagenesis1	3360	3960	150	**0.12**	**0.1 + 5**
Mutagenesis2	NT	NT	NT	**0.17**	**0.2 +12**
Hepatitis	NT	NT	NT	**0.15**	**0.4 + 95.6**
Hepatitis1	NT	NT	NT	**0.1**	**0.2 + 4.8**

Table 5 Runtime to produce a parametrized Markov Logic Network, in minutes. The MBN column shows structure learning time + weight learning time. NT indicates non-termination.

Structure learning is very fast for both the MBN and the LSM method, orders of magnitude faster than for the other methods. The total runtime for MBN is dominated by the time required by Alchemy to find a parametrization for the moralized Bayes net. On the smaller databases, this takes between 5-12 minutes. On MovieLens, parametrization takes two hours, and on Mutagenesis, over 1.5 hours. While finding optimal parameters for the MBN structures remains challenging, the combined structure+weight learning system is much faster than the overall structure + weight learning time of most of the Alchemy methods: They do not scale to the complete datasets, and for the subdatabases, the MBN approach is faster by a factor ranging from 200 to 1000.

These results are strong evidence that the MBN approach leverages the scalability of Bayes net learning to achieve scalable Markov Logic Network learning on databases of realistic size. The LSM method is very fast for all datasets. Inspection of the learned clauses by LSM shows that the rules are mostly just the unit clauses that model marginal probabilities. This indicates underfitting the data, as the following measurements of accuracy and conditional log-likelihood confirm.

10.3 Predictive Accuracy and Data Fit

Previous work on Markov Logic Network evaluation has used a "leave-one-out" approach that learns Markov Logic Networks for a number of subdatabases with a small subset omitted [37]. This is not feasible in our setting because even on a training set of size about 15% of the original, finding an Markov Logic Network structure using the slower Alchemy methods is barely possible. Given these computational constraints, we investigated the predictive performance by learning an Markov Logic Network on one randomly selected 2/3 of the subdatabases as a training set, testing predictions on the remaining 1/3. While this does not provide strong evidence about the generalization performance in absolute terms, it gives information about the relative performance of the methods. In the next section we give further cross validation results using the fastest Alchemy methods LSM and LHL. Tables 6 and 7 report the average accuracy and conditional log-likelihood of each real-world dataset. (The synthetic dataset University was too small for learning on a subset). Higher numbers indicate better performance and NT indicates that the system was not able to return an Markov Logic Network for the dataset, either crashing or timing out after 4 days of running. *MBN achieved substantially better predictions on all test sets, in the range of 10-20% for accuracy.*

The CLL performance of LSM is acceptable overall. The parameter estimates are biased towards uniform values, which leads to predictions whose magnitudes are not extreme. Because the average accuracy is low, this means that when mistaken predictions are made, they are not made with great confidence. The LSM pattern of low accuracy and acceptable log-likelihood is found in our other datasets as well.

Accuracy	Alchemy Methods				LAJ
Dataset	**MSL**	**LHL**	**BUSL**	**LSM**	**MBN**
Movielens11	0.40	0.42	0.34	0.37	**0.63**
Movielens12	0.41	0.44	NT	0.49	**0.62**
Movielens	NT	NT	NT	0.30	**0.69**
Mutagenesis1	0.34	0.31	0.37	0.30	**0.69**
Mutagenesis2	NT	0.35	NT	0.28	**0.65**
Hepatitis	NT	NT	NT	0.32	**0.54**
Hepatitis1	NT	NT	NT	0.29	**0.53**

Table 6 The table compares accuracy performance of the moralization approach (MBN) vs. previous Markov Logic Network learning methods. The data are obtained by training on 2/3 of the database and testing on the other 1/3. ACC is reported as an average over all attribute predicates of the datasets.

Conditional Log-likelihood	Alchemy Methods				LAJ
Dataset	**MSL**	**LHL**	**BUSL**	**LSM**	**MBN**
Movielens11	-4.22	-4.60	-2.80	-1.21	**-1.15**
Movielens12	-3.55	-3.38	NT	**-1.06**	-1.33
Movielens	NT	NT	NT	-1.1	**-0.98**
Mutagenesis1	-4.45	-4.33	-2.54	**-1.12**	-1.85
Mutagenesis2	NT	NT	NT	**-1.18**	-1.48
Hepatitis	NT	NT	NT	-1.26	**-1.18**
Hepatitis1	NT	NT	NT	-1.34	**-1.09**

Table 7 The table compares conditional log likelihood performance of the moralization approach (MBN) vs. previous Markov Logic Network learning methods. The data are obtained by training on 2/3 of the database and testing on the other 1/3. CLL is reported as an average over all attribute predicates of the datasets.

Where the learning methods return a result on a database, we also measured the predictions of the different Markov Logic Network models for the facts in the training database. This indicates how well the Markov Logic Network summarizes the statistical patterns in the data. These measurements test the log-linear equation (1) as a solution to the combining problem for inference (see Section 3). While a small improvement in predictive accuracy may be due to overfitting, the very large improvements we observe are evidence that the Markov Logic Network models produced by the Alchemy methods underfit and fail to represent statistically significant dependencies in the data. Tables 8 and 9 show the results for Accuracy and Conditional Log-likelihood. *MBN achieved substantially better predictions on all test sets, at least 20% for accuracy.*

Accuracy	Alchemy Methods				LAJ
Dataset	**MSL**	**LHL**	**BUSL**	**LSM**	**MBN**
University	0.37	0.37	0.38	0.40	**0.84**
Movielens11	0.43	0.42	0.36	0.39	**0.69**
Movielens12	0.42	0.48	NT	0.53	**0.68**
Movielens	NT	NT	NT	0.34	**0.74**
Mutagenesis1	0.36	0.33	0.37	0.33	**0.80**
Mutagenesis2	NT	NT	NT	0.31	**0.65**
Hepatitis	NT	NT	NT	0.33	**0.57**
Hepatitis1	NT	NT	NT	0.30	**0.57**

Table 8 The table compares accuracy performance of the moralization approach (MBN) vs. previous Markov Logic Network learning methods. The data report the training error where inference is performed over the training dataset. ACC is reported as an average over all attribute predicates of the datasets.

10.4 UW-CSE Dataset

The UW-CSE dataset is naturally divided into 5 folds according to the subarea of computer science, so learning studies have used a cross-validation approach [28, 37], which we follow. This dataset has been used extensively in previous Markov Logic Network experiments, and it differs from the others in that it features a relatively small set of 4 attributes relative to the set of 5 relationships. We therefore

Conditional Log-likelihood	Alchemy Methods				LAJ
Dataset	**MSL**	**LHL**	**BUSL**	**LSM**	**MBN**
University	-5.79	-5.91	-2.92	-0.82	**-0.47**
Movielens11	-4.09	-4.09	-2.44	-1.18	**-1.06**
Movielens12	-3.55	-3.38	NT	-1.10	**-1.09**
Movielens	NT	NT	NT	-1.02	**-0.6**
Mutagenesis1	-4.33	-4.33	-2.54	-1.17	**-0.62**
Mutagenesis2	NT	-4.65	NT	-1.15	**-0.7**
Hepatitis	NT	NT	NT	-1.22	**-1**
Hepatitis1	NT	NT	NT	-1.28	**-1.03**

Table 9 The table compares conditional log-likelihood performance of the moralization approach (MBN) vs. previous Markov Logic Network learning methods. The data report the training error where inference is performed over the training dataset. CLL is reported as an average over all attribute predicates of the datasets.

provide a more detailed set of measurements that compare predictive performance for each attribute separately. As with the other datasets, the speed and predictive accuracy of the learn-and-join algorithm is a substantive improvement. The breakdown by attribute shows that while the extent of the improvement varies with the predicates, the moralized Bayes net approach performs uniformly well on all predicates.

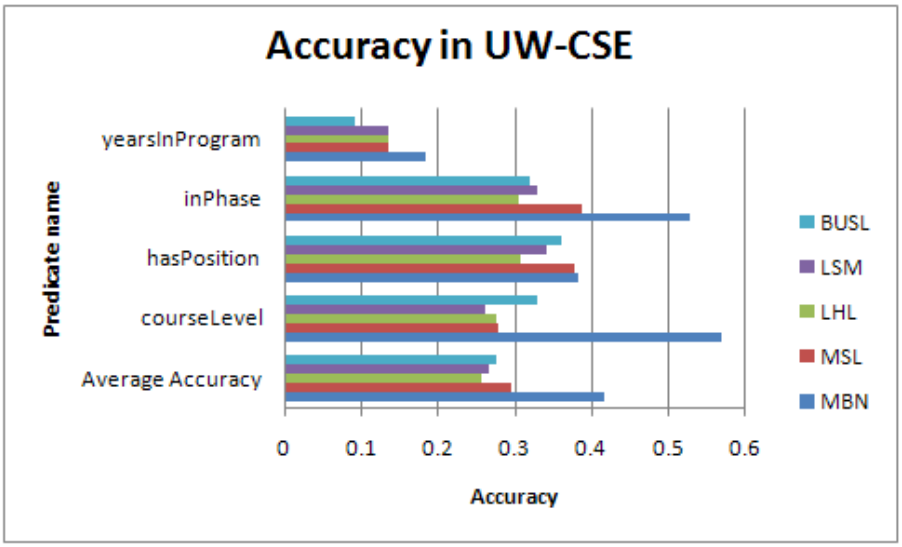

Fig. 13 Predictive Accuracy by attribute, measured by 5-fold cross-validation. The methods are ordered as shown, with MBN at the bottom.

Our results so far indicate that the two most recent Markov Logic Network structure learning methods—Lifted Hypergraph Learning and Learning Structural Motifs—show the best performance. This is confirmed in independent empirical

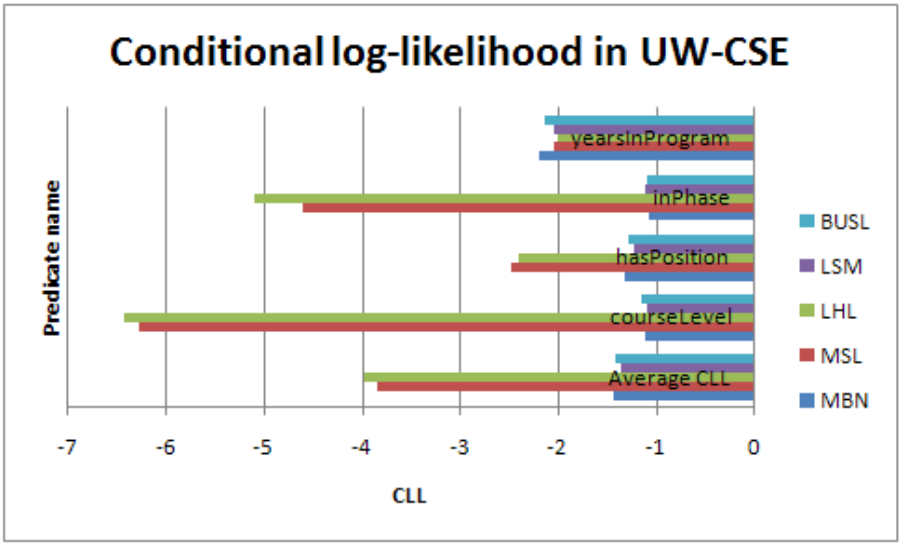

Fig. 14 Conditional Log-likelihood by attribute, measured by 5-fold cross-validation. The methods are ordered as shown, with MBN at the bottom.

Table 10 Cross-validation averages for the UW-CSE dataset

UW-CSE	**MSL**	**LHL**	**LSM**	**BUSL**	**MBN**
Time(min)	2160	2413	2	276	**0.6**
Accuracy	0.29	0.25	0.26	0.27	**0.41**
Conditional log-likelihood	-3.85	-3.98	**-1.37**	-1.42	-1.43

studies by other researchers [29]. The remaining sections of the paper therefore focus on comparing LHL and LSM with the moralization approach.

10.5 Comparison with Inductive Logic Programming on Mutagenesis

We compare the performance of the learn-and-join algorithm for a classification task, predicting the mutagenicity of chemical compounds. This problem has been extensively studied in Inductive Logic Programming (ILP). The purpose of this comparison is to benchmark the predictive performance of the moralization approach against discriminative learning by methods that are different from Markov Logic Network learners.

The class attribute is the mutagenicity (log p). Compounds recorded as having positive mutagenicity are labeled active (positive examples) and compounds recoreded as having 0 or negative mutagenicity are labeled inactive (negative examples). The database contains a total of 188 compounds. Whereas Inductive Logic Programming methods are discriminative, the moralization method performs generative learning over all attributes, a significantly more difficult task. We compare the predictive accuracy of the Moralized Bayes net with well-known ILP methods. Table 11 presents the results of Lodhi and Muggleton [34]. For the STILL system,

we followed the creators' evaluation methodology of using a randomly chosen training and test set. The other systems are evaluated using 10-fold cross-validation. The table shows that the classification performance of the generative Moralized Bayes net model matches that of the discriminative Inductive Logic Programming models.

Method	Evaluation	Accuracy	Reference
MBN	10-fold	0.87	
P-progol	10-fold	0.88	[38]
FOIL	10-fold	0.867	[47]
STILL	90%train-10%test	0.936	[53]
MBN	90%train-10%test	0.944	

Table 11 A comparison of the Moralized Bayes net method with standard Inductive Logic Programming systems trained to predict mutagenicity. Although Bayes net learning produces a generative model, its performance is competitive with discriminative learners.

11 Learning Autocorrelations

In this section we focus on databases that feature self-relationships between entities of the same type. Such schemas potentially give rise to autocorrelations where the value of an attribute for an entity can be predicted by the value of the same attribute for related entities. While recursive dependencies are a key statistical phenomenon in relational data, discovering valid autocorrelations has been a challenge for statistical-relational methods [21, 41]. We investigate how well our approach using the main functor form discovers autocorrelations compared to general Markov Logic structure learning methods. Our benchmark databases are the synthetic University+ dataset and the real-world Mondial database. Table 12 shows the recursive dependencies discovered by the learn-and-join algorithm on each database. We use clausal notation of the form $child \leftarrow \{parents\}$. *Neither of the Markov Logic methods LHL nor LSM discovered any recursive dependencies.*

Table 13 and 14 show the runtime and average predictive performance. The time reported for MBN includes both structure and parameter learning. To achieve a high resolution, results on Mondial are based on 5-fold cross validation. The University dataset is small, so we test and train on the same dataset. As in the previous experiments, both MBN and LSM are fast. The predictive accuracy using Markov Logic Network inference was much better in the moralized Bayes net model (average accuracy improved by 25% or more). This indicates that the discovered recursive dependencies are important for improving predictions. For further discussion, please see [52].

12 Lesion Studies

In this section we study the effects of relaxing different constraints used in the learn-and-join algorithm. We focus on the main functor constraint for learning

Database	Recursive Dependency Discovered
University	$gpa(X) \leftarrow ranking(X), grade(X,Y), registered(X,Y), friend(X,Z), gpa(Z)$
University	$coffee(X) \leftarrow coffee(Y), friend(X,Y)$
Mondial	$religion(X) \leftarrow continent(X), border(X,Y), religion(Y)$
Mondial	$continent(X) \leftarrow border(X,Y), continent(Y), gdp(X), religion(Y)$

Table 12 Autocorrelations discovered by the learn-and-join algorithm using the main functor constraints.

	MBN	LSM	LHL
Time (seconds)	12	**1**	2941
Accuracy	**0.85**	0.44	0.47
CLL	**-0.8**	-2.21	-4.68

Table 13 Results on the University+ database.

	MBN	LSM	LHL
Time (seconds)	50	**2**	15323
Accuracy	**0.50**	0.26	26
CLL	**-1.05**	-1.43	-3.69

Table 14 Results on the Mondial database.

autocorrelations, and on the lattice constraints. The other constraints simply reflect the semantics of the relational schema rather than learning principles. All results are based on 5-fold cross validation. We report a number of quantities for comparing the learned structures.

SLtime(s) Structure learning time in seconds
Numrules Number of clauses in the Markov Logic Network excluding rules with weight 0.
AvgLength The average number of atoms per clause.
AvgAbWt The average absolute weight value.

Because constraints curtail the search space, we expect constrained methods to have the following effects compared with unconstrained methods.

1. Faster run-time.
2. A simpler model with fewer rules.
3. If the constraints are chosen well, they should allow the system to identify important rules and not impair predictive performance.

12.1 Main Functor Constraints

We remove the constraints of specifying a main functor node to study its importance. This means that we introduce a copy of an entity table that is potentially involved in an autocorrelation (e.g., in the University+ database, there are two tables $Student_1, Student_2$). This duplication approach is used in other relational learning systems (e.g., [63,62]). The unconstrained method applies the learn-and-join algorithm in the same way to all entity tables, including the copies. We investigated the following main hypotheses about the effect of the main functor constraint.

1. There should be a tendency towards longer clauses associated with the main functor node (see Section 7).
2. Since Proposition 1 implies that the ground models with and without the constraint are the same, predictive accuracy should be similar.

3. The duplicate edges should lead to duplicate clauses without improvement in predictive performance since the ground models are the same.

Table 15 shows the results for University and Table 16 shows the results for Mondial dataset. *Constraint* is the learn-and-join algorithm with the main functor constraint, whereas *Duplicate* is the learn-and-join algorithm applied naively to the duplicate tables without the constraint. As expected, the constraint speeds up structure learning, appreciably in the case of the larger Mondial dataset. The number of clauses is significantly less (50-60), while on average clauses are longer. The size of the weights indicates that the main functor constraint focuses the algorithm on the important rules.

University+	Constraint	Duplicate
SLtime(s)	**3.1**	3.2
# Rules	**289**	350
AvgAbWt	**2.08**	1.86
AvgLength	4.26	4.11
ACC	**0.86**	**0.86**
CLL	**-0.89**	**-0.89**

Table 15 Comparison to study the effects of removing Main Functor Constraints on University+ dataset.

Mondial	Constraint	Duplicate
SLtime(s)	**8.9**	13.1
# Rules	**739**	798
AvgAbWt	0.22	**0.23**
AvgLength	3.98	3.8
ACC	**0.43**	**0.43**
CLL	**-1.39**	**-1.39**

Table 16 Comparison to study the effects of removing Main Functor Constraints on Mondial dataset.

We also report the number of clauses of a given chain length. Since a clause contains conditions on both attributes and links, we use the maximal slot chain length: The chain length of a rule is computed as the maximal length of a sequence of predicates appearing in the rule such that the database tables corresponding to the predicates are related by foreign key pointers [16]. The measurements show that the algorithm can find long chains, although informative long chains are rare.

12.2 Lattice Constraints

A key feature of the learn-and-join algorithm is the use of lattice constraints. In terms of the join lattice, Bayes nets for join tables higher in the lattice inherit the edges from Bayes nets for join tables lower in the lattice. We remove this constraint to assess its effect on learning. If the Bayes nets learned for different join tables are no longer constrained to be consistent with each other, the

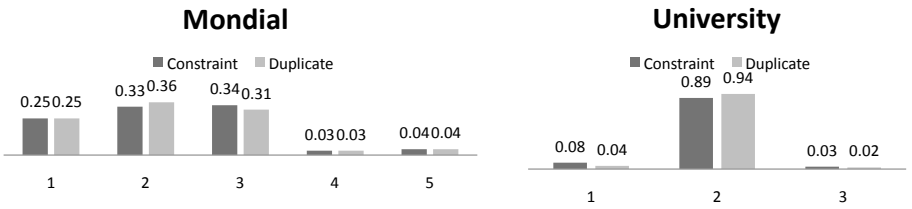

Fig. 15 The percentage of rules of a given chain length for Mondial and University+ dataset in the autocorrrelation lesion study.

question arises how to merge them into a single Bayes net. One possibility is to learn a Bayes net for the maximum join table (e.g., for the relationship set $\mathit{Registered}(S, C)$, $\mathit{Teaches}(P, C)$ in Figure 4). However, in experiments we found that for our benchmark datasets, the maximum join table is too small to yield meaningful results, because too few tuples meet the join conditions (e.g., not many students are RAs). We therefore investigated an intermediate approach where different Bayes nets are learned for single relationship tables joined with the associated entity tables (e.g., $\mathit{Registered}(S, C) \bowtie \mathit{Student} \bowtie \mathit{Course}$). In our benchmark datasets, there were very few conflicts between the edges in different Bayes nets, which we broke randomly. So we could obtain a single acyclic graph for the database by merging the graphs of the intermediate joins; we refer to this approach as the *intermediate* method.

Tables 17, 18, and 19 shows the results for the University+, Hepatitis, and Mondial datasets. The numbers are averages from 5-fold cross validation. Folds are formed by randomly selecting entities as described in Section 9.1. The lattice constraints speed up the learning time spent on join tables, which is the dominant factor. The constrained model features fewer rules with comparatively higher weights. The average predictive accuracy was somewhat higher than with the unconstrained model, whereas the conditional log-likelihood performance was very similar. This is evidence that the constraints helped identify predictively relevant clauses.

University+	Constraint	Intermediate
SLtime(s)	**3.1**	3.3
# Rules	**289**	443
AvgAbWt	**2.08**	1.86
AvgLength	4.26	4.94
ACC	**0.86**	0.84
CLL	**-0.89**	-0.9

Table 17 Comparison to study the effects of removing Lattice Constraints on the University+ dataset.

Hepatitis	Constraint	Intermediate
SLtime(s)	**5.2**	6.1
# Rules	**818**	848
AvgAbWt	**1.68**	1.53
AvgLength	4.15	4.11
ACC	**0.47**	0.41
CLL	**-1.31**	-1.34

Table 18 Comparison to study the effects of removing Lattice Constraints on the Hepatitis dataset.

Mondial	Constraint	Intermediate
SLtime(s)	**8.9**	13.2
# Rules	**739**	1053
AvgAbWt	**0.22**	0.24
AvgLength	3.98	4.66
ACC	**0.43**	0.37
CLL	**-1.39**	**-1.39**

Table 19 Comparison to study the effects of removing Lattice Constraints on Mondial dataset

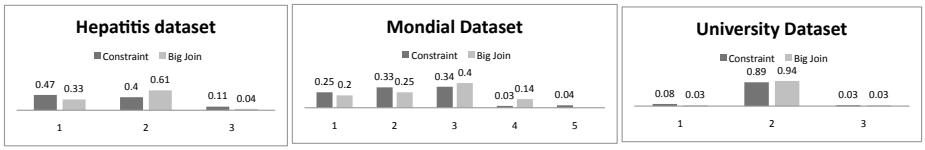

Fig. 16 The percentage of rules of a given chain length for Hepatitis, Mondial, and University dataset in the Lattice constraint lesion study.

13 Conclusion and Future Work

This paper considered the task of building a statistical-relational model for databases with many descriptive attributes. We combined Bayes net learning, one of the most successful machine learning techniques, with Markov Logic networks, one of the most successful statistical-relational formalisms. The main algorithmic contribution is an efficient new structure learning algorithm for a relational Bayes net that models the joint frequency distribution over attributes in the database, given the links between entities. Moralizing the Bayes net leads to a Markov Logic Network structure. Our evaluation on benchmark databases with descriptive attributes shows that compared to current Markov Logic Network structure learning methods, the approach using moralization improves the scalability and run-time performance by orders of magnitude. With standard parameter estimation algorithms and prediction metrics, the moralized Markov Logic Network structures make substantially more accurate predictions. We discuss future work for addressing the limitations of our current system.

Parameter Estimation. In this work we used generic Markov Logic Network algorithms for parameter estimation implemented in the Alchemy package. While the parameter estimation routines of Alchemy run much faster than the structure learning routines, on some datasets we found that parameter estimation can still take a long time. As Markov Logic Networks obtained by moralization have a

special structure, it may be possible to design fast parameter estimation routines for them.

Link Prediction. The learn-and-join algorithm learns a model of the distribution over descriptive attributes conditional on the relationship information in the database. An important project is to extend the learn-and-join algorithm so that it can learn not only dependencies among attributes, but also among relationships (e.g. $Daughter(X, Y)$ implies $Parent(X, Y)$). Since previous Markov Logic Network algorithms have performed well in link modelling, an interesting approach would be a hybrid system that uses the output of the learn-and-join algorithm as a starting point for an Alchemy-based structure learning system. In principle, the key ideas of the learn-and-join algorithm such as the lattice and main functor node constraints are also applicable to link prediction problems.

Acknowledgements This research was supported by a Discovery Grant from the Natural Sciences and Engineering Council of Canada (NSERC). Preliminary results were presented at the 2010 AAAI conference, the 2011 ILP conference, to the AI group at the University of British Columbia, and the IJCAI-STRUCK and IJCAI-GKR workshops (2009). We are grateful to the audiences for helpful questions and comments, especially Lise Getoor.

References

1. Learn and join algorithm code. URL = `http://www.cs.sfu.ca/~oschulte/jbn/`.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc International Conference on Very Large Databases*, pages 478–499, Santiage, Chile, 1994. Morgan Kaufmann, Los Altos, CA.
3. Krzysztof R. Apt and Marc Bezem. Acyclic programs. *New Generation Comput.*, 9(3/4):335–364, 1991.
4. Marenglen Biba, Stefano Ferilli, and Floriana Esposito. Structure learning of Markov logic networks through iterated local search. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikolaos M. Avouris, editors, *ECAI*, pages 361–365, 2008.
5. Ivan Bratko. *Prolog (3rd ed.): programming for artificial intelligence*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.
6. Hailiang Chen, Hongyan Liu, Jiawei Han, and Xiaoxin Yin. Exploring optimization of semantic relationship graph for multi-relational Bayesian classification. *Decision Support Systems*, July 2009.
7. D. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2003.
8. Pedro Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. In *Introduction to Statistical Relational Learning* [18].
9. Justin Domke, Alap Karapurkar, and Yiannis Aloimonos. Who killed the directed model? In *CVPR*, pages 1–8, 2008.
10. Daan Fierens. On the relationship between logical bayesian networks and probabilistic logic programming based on the distribution semantics. In Luc De Raedt, editor, *ILP*, volume 5989 of *Lecture Notes in Computer Science*, pages 17–24. Springer, 2009.
11. Daan Fierens, Hendrik Blockeel, Maurice Bruynooghe, and Jan Ramon. Logical bayesian networks and their relation to other probabilistic logical models. In Stefan Kramer and Bernhard Pfahringer, editors, *ILP*, volume 3625 of *Lecture Notes in Computer Science*, pages 121–135. Springer, 2005.
12. O. Frank. Estimation of graph totals. *Scandinavian Journal of Statistics*, pages 81–89, 1977.
13. Richard Frank, Flavia Moser, and Martin Ester. A method for multi-relational classification using single and multi-feature aggregation functions. In *proceeding of Pkdd*, 2007.
14. Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *In IJCAI*, pages 1300–1309. Springer-Verlag, 1999.
15. Dan Geiger and David Heckerman. Knowledge representation and inference in similarity networks and bayesian multinets. *Artif. Intell.*, 82(1-2):45–74, 1996.

16. Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. Probabilistic relational models. In *Introduction to Statistical Relational Learning* [18], chapter 5, pages 129–173.
17. Lise Getoor and John Grant. Prl: A probabilistic relational language. *Machine Learning*, 62(1-2):7–31, 2006.
18. Lise Getoor and Ben Tasker. *Introduction to statistical relational learning.* MIT Press, 2007.
19. D. Heckerman, C. Meek, and D. Koller. Probabilistic entity-relationship models, PRMs, and plate models. In Getoor and Taskar [18].
20. Tuyen N. Huynh and Raymond J. Mooney. Discriminative structure and parameter learning for markov logic networks. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *ICML*, pages 416–423. ACM, 2008.
21. David Jensen and Jennifer Neville. Linkage and autocorrelation cause feature selection bias in relational learning (2002). In *In Proceedings of the 19th International Conference on Machine Learning*, 2002.
22. Kristian Kersting and Luc de Raedt. Bayesian logic programming: Theory and tool. In *Introduction to Statistical Relational Learning* [18], chapter 10, pages 291–318.
23. Hassan Khosravi, Oliver Schulte andTong Man, Xiaoyuan Xu, and Bahareh Bina. Structure learning for Markov logic networks with many descriptive attributes. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI).*, pages 487–493, 2010.
24. Hassan Khosravi, Oliver Schulte, and Bahareh Bina. Virtual joins with nonexistent links. 19th Conference on Inductive Logic Programming (ILP), 2009. URL = `http://www.cs.kuleuven.be/~dtai/ilp-mlg-srl/papers/ILP09-39.pdf`.
25. Anthony C. Klug. Equivalence of relational algebra and relational calculus query languages having aggregate functions. *J. ACM*, 29(3):699–717, 1982.
26. Stanley Kok and Pedro Domingos. Learning the structure of Markov logic networks. In Luc De Raedt and Stefan Wrobel, editors, *ICML*, pages 441–448. ACM, 2005.
27. Stanley Kok and Pedro Domingos. Statistical predicate invention. In *ICML*, pages 433–440. ACM, 2007.
28. Stanley Kok and Pedro Domingos. Learning markov logic network structure via hypergraph lifting. In Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, *ICML*, pages 64–71. ACM, 2009.
29. Stanley Kok and Pedro Domingos. Learning markov logic networks using structural motifs. In *ICML'10*, pages 551–558. 2010.
30. Stanley Kok, M. Summer, Matthew Richardson, Parag Singla, H. Poon, D. Lowd, J. Wang, and Pedro Domingos. The Alchemy system for statistical relational AI. Technical report, University of Washington., 2009.
31. Daphne Koller and Avi Pfeffer. Learning probabilities for noisy first-order rules. In *IJCAI*, pages 1316–1323, 1997.
32. Wim Van Laer and Luc de Raedt. How to upgrade propositional learners to first-order logic: A case study. In *Relational Data Mining.* Springer Verlag, 2001.
33. Vladimir Lifschitz. Foundations of logic programming. Principles of Knowledge Representation, CSLI Publications, 1996.
34. Huma Lodhi and Stephen Muggleton. Is mutagenesis still challenging? In *Inductive Logic Programming*, pages 35,40, 2005.
35. Daniel Lowd and Pedro Domingos. Efficient weight learning for Markov logic networks. In *PKDD*, pages 200–211, 2007.
36. Wolfgang May. Information extraction and integration: The mondial case study. Technical report, Universitat Freiburg, Institut für Informatik, 1999.
37. Lilyana Mihalkova and Raymond J. Mooney. Bottom-up learning of Markov logic network structure. In *ICML*, pages 625–632. ACM, 2007.
38. Ashwin Srinivasan Muggleton, Ashwin Srinivasan, S. H. Muggleton, M. J. E. Sternberg, and R. D. King. Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, 85:277–299, 1996.
39. Sriraam Natarajan, Prasad Tadepalli, Thomas G. Dietterich, and Alan Fern. Learning first-order probabilistic models with combining rules. *Annals of Mathematics and Artifical Intelligence*, 54(1-3):223–256, 2008.
40. Jennifer Neville and David Jensen. Relational dependency networks. In *An Introduction to Statistical Relational Learning* [18], chapter 8.
41. Jennifer Neville and David Jensen. Relational dependency networks. *J. Mach. Learn. Res.*, 8:653–692, 2007.

42. Liem Ngo and Peter Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theor. Comput. Sci.*, 171(1-2):147–177, 1997.
43. J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
44. David Poole. First-order probabilistic inference. In Georg Gottlob and Toby Walsh, editors, *IJCAI*, pages 985–991. Morgan Kaufmann, 2003.
45. Hoifung Poon and Pedro Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*. AAAI Press, 2006.
46. Alexandrin Popescul and Lyle Ungar. Feature generation and selection in multi-relational learning. In *An Introduction to Statistical Relational Learning* [18], chapter 8.
47. J. Quinlan. Boosting first-order learning. In *Algorithmic Learning Theory*, pages 143–155. Springer, 1996.
48. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.
49. Mark Schmidt, Kevin Murphy, Glenn Fung, and Rómer Rosales. Structure learning in random fields for heart motion abnormality detection. In *CVPR*, 2008.
50. Oliver Schulte. A tractable pseudo-likelihood function for Bayes Nets applied to relational datasets. In *SIAM SDM*, pages 462–473, 2011.
51. Oliver Schulte, Hassan Khosravi, and Bahareh Bina. Bayes nets for combining logical and probabilistic structure. In *Proceedings STRUCK Workshop on Learning Structural Knowledge From Observations*. IJCAI-09, 2009.
52. Oliver Schulte, Hassan Khosravi, Tong Man, and Tianxiang Gao. Learning directed relational models with recursive dependencies. In *Inductive Logic Programming*, 2011.
53. Michele Sebag and Celine Rouveirol. Tractable induction and classification in first order logic via stochastic matching, 1997.
54. Rong She, Ke Wang, and Yabo Xu. Pushing feature selection ahead of join. 2005.
55. A. Srinivasan, SH Muggleton, MJE Sternberg, and RD King. Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, 85(1-2):277–299, 1996.
56. Benjamin Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In Adnan Darwiche and Nir Friedman, editors, *UAI*, pages 485–492. Morgan Kaufmann, 2002.
57. CMU The Tetrad Group, Department of Philosophy. The Tetrad project: Causal models and statistical data, 2008. http://www.phil.cmu.edu/projects/tetrad/.
58. Robert E. Tillman, David Danks, and Clark Glymour. Integrating locally learned causal structures with overlapping variables. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *NIPS*, pages 1665–1672. MIT Press, 2008.
59. J. D. Ullman. *Principles of database systems*. 2. Computer Science Press, 1982.
60. Moshe Y. Vardi. On the complexity of bounded-variable queries. In *PODS*, pages 266–276. ACM Press, 1995.
61. M.P. Wellman, J.S. Breese, and R.P. Goldman. From knowledge bases to decision models. *Knowledge Engineering Review*, 7:35–53, 1992.
62. Xiaoxin Yin and Jiawei Han. Exploring the power of heuristics and links in multi-relational data mining. In *ISMIS'08: Proceedings of the 17th international conference on Foundations of intelligent systems*, pages 17–27, Berlin, Heidelberg, 2008. Springer-Verlag.
63. Xiaoxin Yin, Jiawei Han, Jiong Yang, and Philip S. Yu. Crossmine: Efficient classification across multiple database relations. In *Constraint-Based Mining and Inductive Databases*, pages 172–195, 2004.