Learning Bayes Nets for Relational Data With Link Uncertainty

Zhensong Qian and Oliver Schulte *

School of Computing Science
Simon Fraser University
Vancouver-Burnaby, Canada
{zqian,oschulte}@sfu.ca
http://www.cs.sfu.ca/~oschulte/

Abstract. We present an algorithm for learning correlations among link types and node attributes in relational data that represent complex networks. The link correlations are represented in a Bayes net structure. This provides a succinct graphical way to display relational statistical patterns and support powerful probabilistic inferences. The current state of the art algorithm for learning relational Bayes nets captures only correlations among entity attributes *given* the existence of links among entities. The models described in this paper capture a wider class of correlations that involve uncertainty about the link structure. Our base line method learns a Bayes net from join tables directly. This is a statistically powerful procedure that finds many correlations, but does not scale well to larger datasets. We compare join table search with a hierarchical search strategy.

1 Introduction

Scalable link analysis for relational data with multiple link types is a challenging problem in network science. We describe a method for learning a Bayes net that captures simultaneously correlations between link types, link features, and attributes of nodes. Such a Bayes net provides a succinct graphical representation of complex statistical-relational patterns. A Bayes net model supports powerful probabilistic reasoning for answering "what-if" queries about the probabilities of uncertain outcomes conditional on observed events. Previous work on learning Bayes nets for relational data was restricted to correlations among attributes given the existence of links [16]. The larger class of correlations examined in our new algorithms includes two additional kinds:

1. Dependencies between different types of links.

* This research was supported by a Discovery Grant to Oliver Schulte from the Canadian Natural Sciences and Engineering Council. And Zhensong Qian was also supported by a grant from the China Scholarship Council. The preliminary version of this paper was presented in the IJCAI 2013 GKR workshop.

2. Dependencies among node attributes given the *absence* of a link between the nodes.

Discovering such dependencies is useful for several applications.

Knowledge Discovery Dependencies provide valuable insights in themselves. For instance, a web search manager may wish to know whether if a user searches for a video in Youtube for a product, they are also likely to search for it on the web.

Relevance Determination Once dependencies have been established, they can be used as a relevance filter for focusing further network analysis only on statistically significant associations. For example, the classification and clustering methods of Sun and Han [19] for heterogeneous networks assume that a set of "metapaths" have been found that connect link types that are associated with each other.

Query Optimization The Bayes net model can also be used to estimate relational statistics, the frequency with which statistical patterns occur in the database [17]. This kind of statistical model can be applied for database query optimization [4].

Approach We consider three approaches to multiple link analysis with Bayes nets.

Flat Search Applies a standard Bayes net learner to a single large join table. This table is formed as follows: (1) take the cross product of entity tables. (An entity table lists the set of nodes of a given type.) (2) For each tuple of entities, add a relationship indicator whose value "true" or "false" indicates whether the relationship holds among the entities.

Hierarchical Search Conducts bottom-up search through the lattice of table joins hierarchically. Dependencies (Bayes net edges) discovered on smaller joins are propagated to larger joins. The different table joins include information about the presence or absence of relationships as in the flat search above. This is an extension of the current state of the art Bayes net learning algorithm for relational data [16].

Evaluation. We compare the learned models using standard scores (e.g., Bayes Information Criterion, log-likelihood). These results indicate that both flat search and hierarchical search are effective at finding correlations among link types. Flat search can on some datasets achieve a higher score by exploiting attribute correlations that depend on the absence of relationships. Structure learning time results indicate that hierarchical search is substantially more scalable.

The main contribution of this paper is extending the current state-of-the-art Bayes net learner to model correlations among different types of links, with a comparison of a flat and a hierarchical search strategy.

Paper Organization We describe Bayes net models for relational data (Poole's Parametrized Bayes Nets). Then we present the learning algorithms, first flat search then hierarchical search. We compare the models on four databases from different domains.

2 Related Work

Approaches to structure learning for directed graphical models with link uncertainty have been previously described, such as [3]. However to our knowledge, no implementations of such structure learning algorithms for directed graphical models are available. Our system builds on the state-of-the-art Bayes net learner for relational data, whose code is available at [6]. Implementations exist for other types of graphical models, specifically Markov random fields (undirected models) [2] and dependency networks (directed edges with cycles allowed) [10]. Structure learning programs for Markov random fields are provided by Alchemy [2] and Khot et al [9]. Khot et al. use boosting to provide a state-of-the-art dependency network learner. None of these programs are able to return a result on half of our datasets because they are too large. For space reasons we restrict the scope of this paper to directed graphical models and do not go further into undirected model. For an extensive comparison of the learn-and-join Bayes net learning algorithm with Alchemy please see [16].

3 Background and Notation

Poole introduced the Parametrized Bayes net (PBN) formalism that combines Bayes nets with logical syntax for expressing relational concepts [12]. We adopt the PBN formalism, following Poole's presentation.

3.1 Bayes Nets for Relational Data

A **population** is a set of individuals. Individuals are denoted by lower case expressions (e.g., *bob*). A **population variable** is capitalized. A **functor** represents a mapping $f : \mathcal{P}_1, \ldots, \mathcal{P}_a \rightarrow V_f$ where f is the name of the functor, each $\mathcal{P}_i$ is a population, and V_f is the output type or **range** of the functor. In this paper we consider only functors with a finite range, disjoint from all populations. If $V_f = \{T, F\}$, the functor f is a (Boolean) **predicate**. A predicate with more than one argument is called a **relationship**; other functors are called **attributes**. We use uppercase for predicates and lowercase for other functors.

A **Bayes Net (BN)** is a directed acyclic graph (DAG) whose nodes comprise a set of random variables and conditional probability parameters. For each assignment of values to the nodes, the joint probability is specified by the product of the conditional probabilities, $P(child|parent\_values)$. A **Parametrized random variable** is of the form $f(X_1, \ldots, X_a)$, where the populations associated with the variables are of the appropriate type for the functor. A **Parametrized Bayes Net** (PBN) is a Bayes net whose nodes are Parametrized random variables [12]. If a Parametrized random variable appears in a Bayes net, we often refer to it simply as a node.

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

Table 1. A relational schema for a university domain. Key fields are underlined. An instance for this schema is given in Figure 1.

3.2 Databases and Table Joins

We begin with a standard **relational schema** containing a set of tables, each with key fields, descriptive attributes, and possibly foreign key pointers. A **database instance** specifies the tuples contained in the tables of a given database schema. We assume that tables in the relational schema can be divided into *entity tables* and *relationship tables*. This is the case whenever a relational schema is derived from an entity-relationship model (ER model) [21, Ch.2.2]. The functor formalism is rich enough to represent the constraints of an ER schema by the following translation: Entity sets correspond to types, descriptive attributes to functions, relationship tables to predicates, and foreign key constraints to type constraints on the arguments of relationship predicates. Assuming an ER design, a relational structure can be visualized as a complex network [13, Ch.8.2.1]: individuals are nodes, attributes of individuals are node labels, relationships correspond to (hyper)edges, and attributes of relationships are edge labels. Conversely, a complex network can be represented using a relational database schema.

Table 1 shows a relational schema for a database related to a university. In this example, there are two entity tables: a *Student* table and a *Course* table. There is one relationship table *Registered* with foreign key pointers to the *Student* and *Course* tables whose tuples indicate which students have registered in which courses. Figure 1 displays a small database instance for this schema together with a Parametrized Bayes Net (omitting the *Teaches* relationship for simplicity.)

The **natural table join**, or simply join, of two or more tables contains the rows in the Cartesian products of the tables whose values match on common fields. In logical terms, a join corresponds to a conjunction [21].

4 Bayes Net Learning With Link Correlation Analysis

We outline the two methods we compare in this paper, flat search and hierarchical search.

4.1 Flat Search

The basic idea for flat search is to apply a standard propositional or single-table Bayes net learner to a single large join table. To learn correlations between link

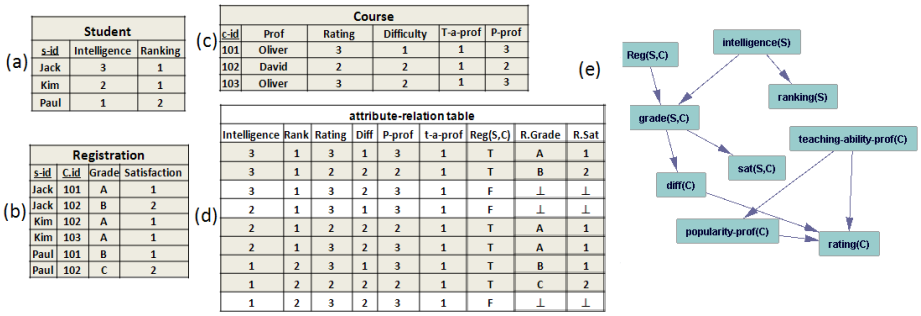

Fig. 1. Database Table Instances: (a) *Student* (b) *Registered* (c) *Course*. To simplify, we added the information about professors to the courses that they teach. (d) The attribute-relation table *Registered*$^+$ derived from *Registered*, which lists for each pair of entities their descriptive attributes, whether they are linked by *Registered*, and the attributes of a link if it exists. (e) A Parametrized Bayes Net for the university schema.

types, we need to provide the Bayes net with data about when links are present *and* when they are absent. To accomplish this, we add to each relationship table a **link indicator column**. This columns contains T if the link is present between two entities, and F if the link is absent. (The entities are specified in the primary key fields.) We add rows for all pairs of entities of the right type for the link, and enter T or F in the link indicator column depending on whether a link exists or not. We refer to relationship tables with a link indicator column as **extended** tables. Extended tables are readily computed using SQL queries. If we omit the entity Ids from an extended table, we obtain the **attribute-relation** table that lists (1) all attributes for the entities involved, (2) whether a relationship exists and (3) the attributes of the relationship if it exists. If the attribute-relation table is derived from a relationship R, we refer to it as R^+.

The attribute-relation table is readily defined for a set of relationships: take the cross-product of all populations involved, and add a link indicator column for each relationship in the set. For instance, if we wanted to examine correlations that involve both the *Registered* and the *Teaches* relationships, we would form the cross-product of the entity types *Student*, *Course*, *Professor* and build an attribute-relation table that contains two link indicator columns *Registered*(S, C) and *Teaches*(P, C). The **full join table** is the attribute-relation table for all relationships in the database.

The **flat search Bayes net learner** takes a standard Bayes net learner and applies it to the full join table to obtain a single Parametrized Bayes net. The results of [14] can be used to provide a theoretical justification for this procedure; we outline two key points.

1. The full join table correctly represents the *sufficient statistics*[5, 14] of the database: using the full join table to compute the frequency of a joint value assignment for Parametrized Random Variables is equivalent to the frequency with which this assignment holds in the database.

2. Maximizing a standard single-table likelihood score from the full join table is equivalent to maximizing the *random selection pseudo likelihood.* The random selection pseudo log-likelihood is the expected log-likelihood assigned by a Parametrized Bayes net when we randomly select individuals from each population and instantiate the Bayes net with attribute values and relationships associated with the selected individuals.

4.2 Hierarchical Search

Khosravi *et al.* [16] present the learn-and-join structure learning algorithm. The algorithm upgrades a single-table Bayes net learner for relational learning. We describe the fundamental ideas of the algorithm; for further details please see [16].

The key idea is to build a Bayes net for the entire database by level-wise search through the *table join lattice.* The user chooses a single-table Bayes net learner. The learner is applied to table joins of size 1, that is, regular data tables. Then the learner is applied to table joins of size $s, s+1, \ldots$, with the constraint that larger join tables inherit the absence or presence of learned edges from smaller join tables. These edge constraints are implemented by keeping a global cache of forbidden and required edges. Algorithm 1 provides pseudocode for the previous learn-and-join algorithm (LAJ) [15].

To extend the learn-and-join algorithm for multiple link analysis, we replace the natural join in line 7 by the extended join (more precisely, by the attribute-relation tables derived from the extended join). The natural join contains only tuples that appear in all relationship tables. Compared to the extended join, this corresponds to considering only rows where the link indicator columns have the value T. When the propositional Bayes net learner is applied to such a table, the link indicator variable appears like a constant. Therefore the BN learner cannot find any correlations between the link indicator variable and other nodes, nor can it find correlations among attributes conditional on the link indicator variable being F. Thus the previous LAJ algorithm finds only correlations between entity attributes conditional on the existence of a relationship. In sum, hierarchical search with link correlations can be described as follows.

1. Run the previous LAJ algorithm (Algorithm 1) using natural joins.
2. Starting with the constraints from step 1, run the LAJ algorithm where extended joins replace natural joins. That is, for each relationship set shown in the lattice of Figure 2, apply the single-table Bayes net learner to the extended join for the relationship set.

5 Evaluation

All experiments were done on a QUAD CPU Q6700 with a 2.66GHz CPU and 8GB of RAM. The LAJ code and datasets are available on the world-wide web

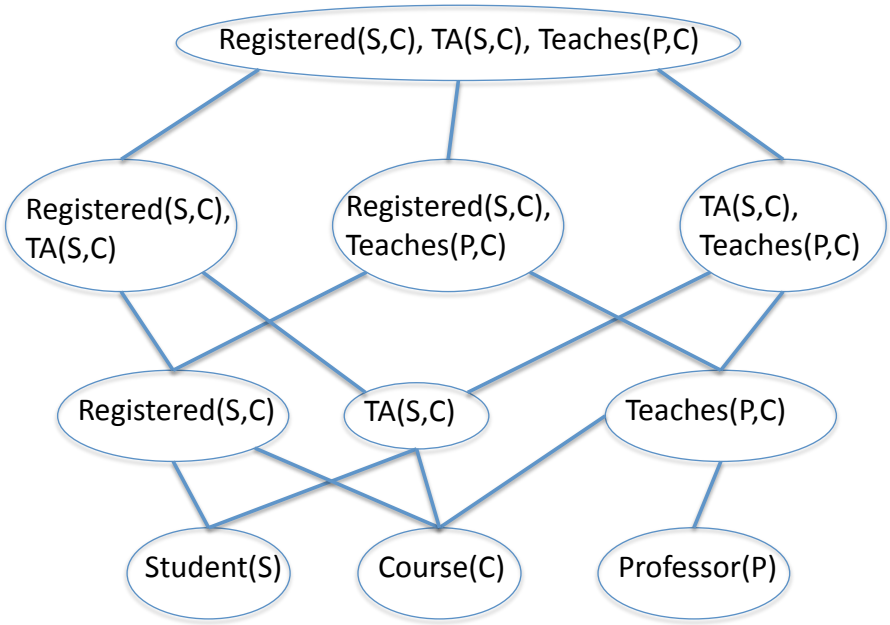

Fig. 2. A lattice of relationship sets for the university schema of Table 1. Links from entity tables to relationship tables correspond to foreign key pointers.

[6]. We made use of the following single-table Bayes Net search implementation: GES search [1] with the BDeu score as implemented in version 4.3.9-0 of CMU's Tetrad package (structure prior uniform, ESS=10; [20]).

Methods Compared We compared the following methods.

LAJ The previous LAJ method without link correlations (Algorithm 1).
LAJ+ The new LAJ method that has the potential to find link correlations (Algorithm 1 with the extended join tables instead of natural join tables).
Flat Applies the single-table Bayes net learner to the full join table.

To implement Flat Search and the LAJ+ algorithm efficiently, we apply the Fast Möbius Transform to compute tables of sufficient statistics that involve negated relationships. We discuss the details further in Section 6.

Performance Metrics We report learning time, log-likelihood, Bayes Information Criterion (BIC), and the Akaike Information Criterion (AIC). BIC and AIC are standard scores for Bayes nets [1], defined as follows. We write

$$L(\hat{G}, \mathbf{d})$$

Algorithm 1 Pseudocode for previous Learn-and-Join Structure Learning for Lattice Search.

Input: Database $\mathcal{D}$ with $E_1, ..E_e$ entity tables, $R_1, ...R_r$ Relationship tables,
Output: Bayes Net for $\mathcal{D}$
Calls: PBN: Any propositional Bayes net learner that accepts edge constraints and a single table of cases as input.
Notation: PBN$(T, \text{Econstraints})$ denotes the output DAG of PBN. Get-Constraints(G) specifies a new set of edge constraints, namely that all edges in G are required, and edges missing between variables in G are forbidden.

```
Add descriptive attributes of all entity and relationship tables as variables to G.
  Add a boolean indicator for each relationship table to G.
Econstraints = ∅ [Required and Forbidden edges]
for m=1 to e do
    Econstraints += Get-Constraints(PBN(E_m , ∅))
end for
for m=1 to r do
    N_m := natural join of R_m and entity tables linked to R_m
    Econstraints += Get-Constraints(PBN(N_m, Econstraints))
end for
for all N_i and N_j with a foreign key in common do
    K_ij := join of N_i and N_j
    Econstraints += Get-Constraints(PBN(K_ij, Econstraints))
end for
return Bayes Net defined by Econstraints.
```

for the log-likelihood score, where $\hat{G}$ is the BN G with its parameters instantiated to be the maximum likelihood estimates given the dataset $\mathbf{d}$, and the quantity $L(\hat{G}, \mathbf{d})$ is the log-likelihood of $\hat{G}$ on $\mathbf{d}$.

The BIC score is defined as follows [1, 14]

$$BIC(G, \mathbf{d}) = L(\hat{G}, \mathbf{d}) - par(G)/2 \times ln(m)$$

where the data table size is denoted by m, and $par(G)$ is the number of free parameters in the structure G. The AIC score is given by

$$AIC(G, \mathbf{d}) = L(\hat{G}, \mathbf{d}) - par(G).$$

Selection by AIC is asympotically equivalent to selection by cross-validation, so we may view it as a closed-form approximation to cross-validation, which is computationally demanding for relational datasets.

Datasets We used one synthetic and three benchmark real-world databases, with the modifications described by Schulte and Khosravi [16]. See that article for more details.

University Database. We manually created a small dataset, based on the schema given in Table 1. The dataset is small and is used as a testbed for the correctness of our algorithms.

Dataset	#tuples
University	662
Movielens	1585385
Mutagenesis	1815488
Hepatitis	2965919
Small-Hepatitis	19827

Table 2. Size of datasets in total number of table tuples.

MovieLens Database. A dataset from the UC Irvine machine learning repository. The data are organized in 3 tables (2 entity tables, 1 relationship table, and 7 descriptive attributes).
Mutagenesis Database. A dataset widely used in ILP research. It contains two entity tables and two relationships.
Hepatitis Database. A modified version of the PKDD'02 Discovery Challenge database. The data are organized in 7 tables (4 entity tables, 3 relationship tables and 16 descriptive attributes). In order to make the learning feasible, we undersampled Hepatitis database to keep the ratio of positive and negative link indicator equal to one.

Dataset	Data Processing Time
University	1.205
Movielens	1.539
Mutagenesis	0.723
Small-Hepatitis	57.794

Table 3. Data Preprocessing: Table Join Time in seconds.

Dataset	Flat	LAJ+	LAJ
University	1.916	1.183	0.291
Movielens	38.767	18.204	1.769
Mutagenesis	3.231	3.448	0.982
Small-Hepatitis	9429.884	7.949	10.617

Table 4. Model Structure Learning Time in seconds.

5.1 Results

Learning Times Table 3 shows the data preprocessing time that the different methods require for table joins. This is the same for all methods, namely the

cost of computing the full join table using the fast Möbius transform described in Section 6. Table 4 provides the model search time for each of the link analysis methods. On the smaller and simpler datasets, all search strategies are fast, but on the medium-size and more complex datasets (Hepatitis, MovieLens), hierarchical search is much faster due to its use of constraints.

University	BIC	AIC	log-likelihood	# Parameter
Flat	-17638.27	-12496.72	-10702.72	1767
LAJ+	-13495.34	-11540.75	-10858.75	655
LAJ	-13043.17	-11469.75	-10920.75	522

MovieLens	BIC	AIC	log-likelihood	# Parameter
Flat	-4912286.87	-4911176.01	-4910995.01	169
LAJ+	-4911339.74	-4910320.94	-4910154.94	154
LAJ	-4911339.74	-4910320.94	-4910154.94	154

Mutagenesis	BIC	AIC	log-likelihood	# Parameter
Flat	-21844.67	-17481.03	-16155.03	1289
LAJ+	-47185.43	-28480.33	-22796.33	5647
LAJ	-30534.26	-25890.89	-24479.89	1374

Hepatitis	BIC	AIC	log-likelihood	# Parameter
Flat	-7334391.72	-1667015.81	-301600.81	1365357
LAJ+	-457594.18	-447740.51	-445366.51	2316
LAJ	-461802.76	-452306.05	-450018.05	2230

Table 5. Statistical Performance of different Searching Algorithms by dataset.

Statistical Scores As expected, adding edges between link nodes improves the statistical data fit: the link analysis methods LAJ+ and Flat perform better than the learn-and-join baseline in terms of log-likelihood on all datasets shown in table 5, except for MovieLens where the Flat search has a lower likelihood. On the small synthetic dataset University, flat search appears to overfit whereas the hierarchical search methods are very close. On the medium-sized dataset MovieLens, which has a simple structure, all three methods score similarly. Hierarchical search finds no new edges involving the single link indicator node (i.e., LAJ and LAJ+ return the same model).

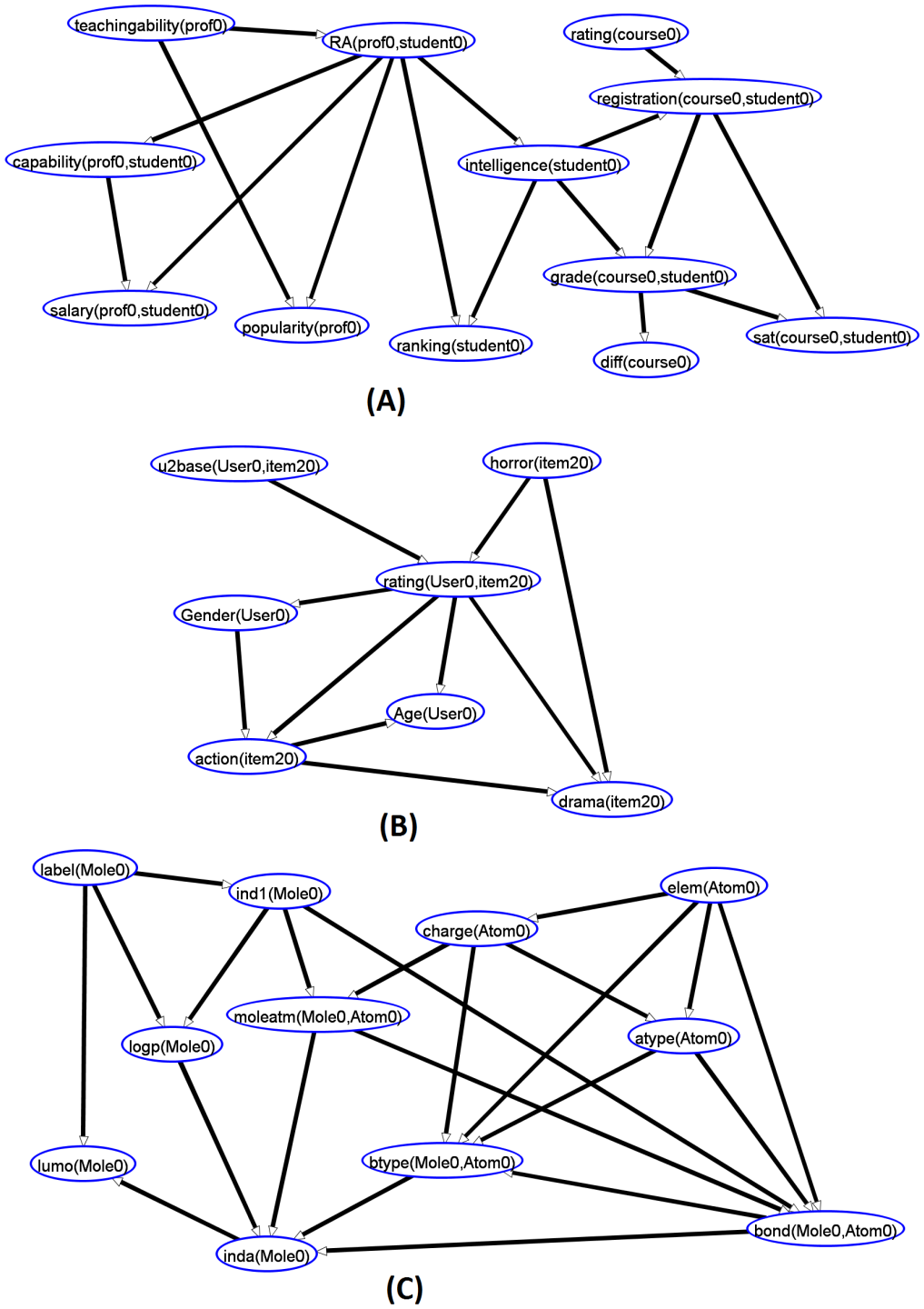

Fig. 3. Learned Parametrized Bayes Net for 3 complete datasets: (A) University Database, (B) MovieLens Database, (C) Mutagenesis Database.

The most complex dataset, Hepatitis, is a challenge for flat search, which seems to overfit severely with a huge number of parameters that result in a model selection score that is an order of magnitude worse than for hierarchical search. Because of the complex structure of the Hepatitis schema, the hierarchy constraints appear to be effective in combating overfitting.

The situation is reversed on the Mutagenesis dataset where flat search does well: compared to previous LAJ algorithm, it manages to fit the data better with a less complex model. Hiearchical search performs very poorly compared to flat search (lower likelihood yet many more parameters in the model). Investigation of the models shows that the reason for this phenomenon is a special property of the Mutagenesis dataset: The two relationships in the dataset, Bond and MoleAtm, involve the same descriptive attributes. The hierarchical search learns two separate Bayes nets for each relationship, then propagates both sets to the final result. However, the union of the two graphs may not be a compact model of the associations that hold in the entire database. A solution to this problem would be to add a final pruning phase where redundant edges can be eliminated. We expect that with this change, hierarchical search would be competitive with flat search on the Mutagenesis dataset as well.

We also provide learned Bayes net for three databases in terms of the best statistical performance in Figure 3.

6 Computing Data Join Tables

The learning algorithms described in this paper rely on the availability of the extended relational tables R^+ (see Figure 1). A naive implementation constructs this tables using standard joins. However, the cross-products grow exponentially with the number of relations joined, and therefore do not scale to large datasets. In this section we describe a "virtual join" algorithm that computes the required data tables without the quadratic cost of materializing a cross-product.

Our first observation is that Bayes net learners do not require the entire extended table, but only the *sufficient statistics*, namely the counts of how many times each combination of values occurs in the data [11]. For instance, the attribute-relationship table of Figure 1, each combination is observed exactly once. Our second observation is that to compute the sufficient statistics, it suffices to compute the *frequencies* of value combinations rather than the counts, because counts can be obtained from frequencies by multiplying with the appropriate domain sizes. An efficient virtual join algorithm for computing frequencies in relational data was recently published by Schulte *et al.* [18]. Their algorithm is based on the fast Möbius transform (FMT). We outline it briefly.

Consider a set of relationship indicator nodes $R_1 = \cdot, R_2 = \cdot, \ldots, R_m$ and attribute nodes $f_1, \ldots, f_j$. The sufficient statistics correspond to a joint distribution over these random variables.

m Boolean relationship random variables:

$$P(R_1 = \cdot, R_2 = \cdot, \ldots, R_m = \cdot; f_1 = \cdot, \ldots, f_j = \cdot).$$

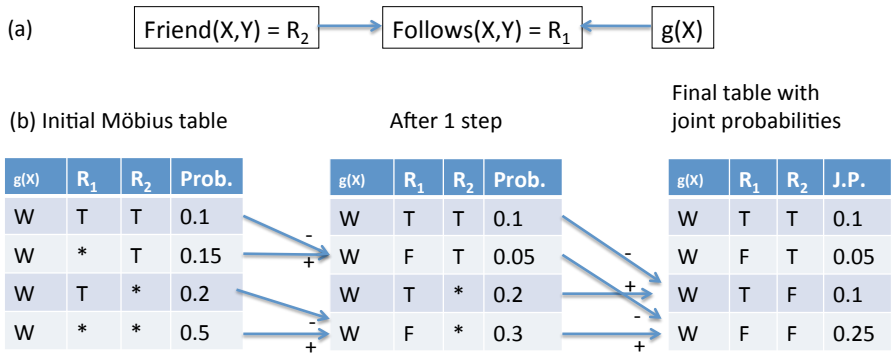

Fig. 4. (a) A Bayes net with two relationship nodes. (b) An illustrative trace of the fast Möbius transform.

Our goal is to compute sufficient statistics of this form for relational data. The FMT allows us to efficiently find sufficient statistics for binary random variables. We apply it with a fixed set of values for the attribute nodes, which corresponds to a joint distribution over the m Boolean relationship random variables:

$$P(R_1 = \cdot, R_2 = \cdot, \ldots, R_m = \cdot; f_1 = v_1, \ldots, f_j = v_j).$$

The FMT uses the local update operation

$$P(R = F, \mathbf{R}) = P(\mathbf{R}) - P(R = T, \mathbf{R}) \tag{1}$$

where $\mathbf{R}$ is a conjunction of relationship specifications, possibly with both positive and negative relationships. The equation continues to hold if we extend relationship specifications with any fixed set of value assignments $f_1 = v_1, \ldots, f_j = v_j$ to attribute functor nodes $f_1, \ldots, f_j$. Using the convention that $R = *$ means that the value of relationship R is unspecified, the equation (1) can be rewritten as

$$P(R = F, \mathbf{R}) = P(R = *, \mathbf{R}) - P(R = T, \mathbf{R}). \tag{2}$$

The FMT begins with an initial table of sufficient statistics where all relationship nodes have the value T or $*$ but not F. Since these sufficient statistics do not involve false relationships, they can be computed efficiently from a relational database using table joins. The procedure then goes through the relationship nodes $R_1, \ldots, R_m$ in order, at stage i replacing all occurrences of $R_i = *$ with $R_i = F$, and applying the local update equation to obtain the probability value for the modified row. At termination, all $*$ values have been replaced by F and the table specifies all joint frequencies as required. Algorithm 2 gives pseudo code and Figure 4 presents an example of the transform step. For example, the

Algorithm 2 The inverse Möbius transform for parameter estimation in a Parametrized Bayes Net.

Input: database $\mathcal{D}$; a set of nodes divided into attribute nodes $f_1, \ldots, f_j$ and relationship nodes $R_1, \ldots, R_m$.
Output: joint probability table specifying the data frequencies for each joint assignment to the input nodes.

1: **for all** attribute value assignments $f_1 := v_1, \ldots, f_j := v_j$ **do**
2: initialize the table: set all relationship nodes to either T or $*$; find joint frequencies with data queries.
3: **for** $i = 1$ to m **do**
4: Change all occurrences of $R_i = *$ to $R_i = F$.
5: Update the joint frequencies using (1).
6: **end for**
7: **end for**

probability entry for the second row of the middle table is computed by applying the equation

$$P(R_1 = F, R_2 = T; g(X) = W) =$$
$$P(R_1 = *, R_2 = T; g(X) = W) - P(R_1 = T, R_2 = T; g(X) = W) =$$
$$0.15 - 0.1 = 0.05$$

which is an instance of Equation (2).

7 Conclusion

We described different methods for extending relational Bayes net learning to correlations involving links. Statistical measures indicate that Bayes net methods succeed in finding relevant correlations. There is a trade-off between statistical power and computational feasibility (full table search vs constrained search). Hierarchical search often does well on both dimensions, but needs to be extended with a pruning step to eliminate redundant edges.

A key issue for scalability is that most of the learning time is taken up by forming table joins, whose size is the cross product of entity tables. These table joins provide the sufficient statistics required in model selection. To improve scalability, computing sufficient statistics needs to be feasible for cross product sizes in the millions or more. A solution is applying virtual join methods that compute sufficient statistics without materializing table joins, such as the Fast Möbius Transform [17, 22].

A valuable direction for future work is to compare learning link correlations with directed and undirected models, such as Markov Logic Networks [2]. As we explained in Section 2, current relational learners for undirected models do not scale to most of our datasets. One option is to subsample the datasets so that we can compare the statistical power of directed and undirect learning methods independently of scalability issues. Khosravi *et al.* were able to obtain structure

learning results for Alchemy [7], but did not evaluate the models with respect to link correlations. For the MLN-Boost system, we were able to obtain preliminary results on several benchmark databases (including Mutagenesis and Hepatitis), by selecting the right subset of target predicates. MLN-Boost is the current state-of-the-art learner for Markov Logic Networks [8]. The Bayes net models were competitive with the MLN-Boost models on a standard cross-validation measure of predictive accuracy.

References

1. D. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2003.
2. Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool Publishers, 2009.
3. Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. Probabilistic relational models. In *Introduction to Statistical Relational Learning*, chapter 5, pages 129–173. MIT Press, 2007.
4. Lise Getoor, Benjamin Taskar, and Daphne Koller. Selectivity estimation using probabilistic models. *ACM SIGMOD Record*, 30(2):461–472, 2001.
5. D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.
6. H. Khosravi, T. Man, J. Hu, E. Gao, and O. Schulte. Learn and join algorithm code. URL = `http://www.cs.sfu.ca/~oschulte/jbn/`.
7. Hassan Khosravi, Oliver Schulte, Tong Man, Xiaoyuan Xu, and Bahareh Bina. Structure learning for Markov logic networks with many descriptive attributes. In *AAAI*, pages 487–493, 2010.
8. Tushar Khot, Sriraam Natarajan, Kristian Kersting, and Jude W. Shavlik. Learning Markov logic networks via functional gradient boosting. In *ICDM*, pages 320–329. IEEE Computer Society, 2011.
9. Tushar Khot, Jude Shavlik, and Sriraam Natarajan. Boostr, 2013. URL = `http://pages.cs.wisc.edu/~tushar/Boostr/`.
10. Sriraam Natarajan, Tushar Khot, Kristian Kersting, Bernd Gutmann, and Jude W. Shavlik. Gradient-based boosting for statistical relational learning: The relational dependency network case. *Machine Learning*, 86(1):25–56, 2012.
11. R. E. Neapolitan. *Learning Bayesian Networks*. Pearson Education, 2004.
12. David Poole. First-order probabilistic inference. In *IJCAI*, pages 985–991, 2003.
13. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.
14. Oliver Schulte. A tractable pseudo-likelihood function for Bayes nets applied to relational data. In *SIAM SDM*, pages 462–473, 2011.
15. Oliver Schulte. Challenge paper: Marginal probabilities for instances and classes. ICML-SRL Workshop on Statistical Relational Learning., June 2012.
16. Oliver Schulte and Hassan Khosravi. Learning graphical models for relational data via lattice search. *Machine Learning*, 88(3):331–368, 2012.
17. Oliver Schulte, Hassan Khosravi, Arthur Kirkpatrick, Tianxiang Gao, and Yuke Zhu. Modelling relational statistics with bayes nets. In *Inductive Logic Programming (ILP)*, 2012.

18. Oliver Schulte, Hassan Khosravi, Arthur Kirkpatrick, Tianxiang Gao, and Yuke Zhu. Modelling relational statistics with bayes nets. *Machine Learning*, page forthcoming., 2013.
19. Yizhou Sun and Jiawei Han. *Mining Heterogeneous Information Networks: Principles and Methodologies*, volume 3. Morgan & Claypool Publishers, 2012.
20. The Tetrad Group. The Tetrad project, 2008. http://www.phil.cmu.edu/projects/tetrad/.
21. J. D. Ullman. *Principles of Database S¡ystems*. W. H. Freeman & Co., 2 edition, 1982.
22. Xiaoxin Yin, Jiawei Han, Jiong Yang, and Philip S. Yu. Crossmine: Efficient classification across multiple database relations. In *ICDE*, pages 399–410. IEEE Computer Society, 2004.