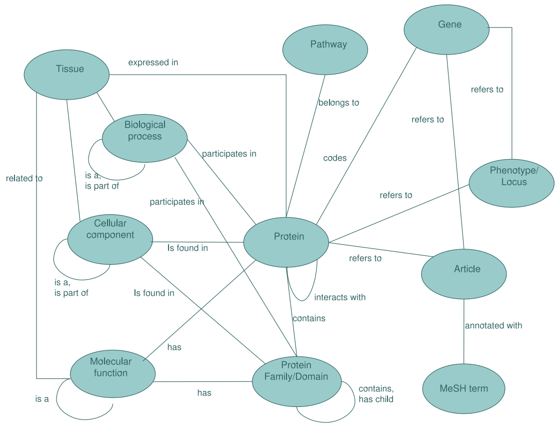
ProbLog

A Probabilistic Prolog and its Application in Link Discovery

Luc De Raedt, Katholieke Universiteit Leuven
Angelika Kimmig, Albert-Ludwigs-Universität Freiburg
Hannu Toivonen, University of Helsinki

Dutch-Belgian Database Day 2006

Motivating Example: Biological Networks



Scheme of biological database

- Huge amounts of molecular biological data available
- Probabilities of direct links obtained by various prediction techniques
- Probability of some items being in a specific kind of possibly complex relationship?

Simple ProbLog Encoding of Biological Network

```
0.3779:edge('EntrezProtein_4885045','HGNC_620').
0.4928:edge('HGNC_620','PubMed_12653567').
0.6054:edge('EntrezProtein_4885045','HGNC_12850').
0.9022:edge('PubMed_2322535','HGNC_983').
0.8750:edge('HomoloGene_20065','HGNC_983').
...

1.0:path(X,Y):-edge(X,Y).
1.0:path(X,Y):-edge(X,Z),path(Z,Y).
```

What is the probability of `path('HGNC_620','HGNC_983')`?

Of course, more complex encodings and queries possible...

ProbLog: Idea

ProbLog = Prolog + Probability Labels

- Each clause has a label (value between 0 and 1) indicating the probability that the clause is in the logic program.
- Each clause independent of all other clauses
- No other assumptions or restrictions

Inference in ProbLog

- Given a ProbLog program T and a query q, what is the probability that there is some proof of q in T?

Semantics

- ProbLog program $T = \{p_1 : c_1, \ldots, p_n : c_n\}$, logical part $L_T = \{c_1, \ldots, c_n\}$
- Sampling logic programs $L \subseteq L_T$:

$$P(L|T) = \prod_{c_i \in L} p_i \prod_{c_i \in L_T \setminus L} (1 - p_i)$$

- Given T and a query q:

$$P(q|T) = \sum_{L \subseteq L_T} P(q|L) \cdot P(L|T)$$

where $P(q|L) = 1$ if there is a proof of q in L, else $P(q|L) = 0$

Inference: Calculating $P(q|T)$

Use monotone DNF formula d describing all proofs of q in L_T

- Boolean variable b_i for each $p_i : c_i \in T$
- Concrete proof: conjunction of clause variables
- Some proof: disjunction of formulae for all proofs

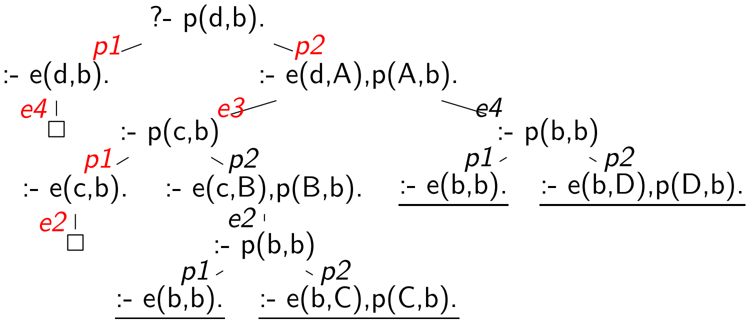
$$d = \bigvee_{b \in proofs(q)} \bigwedge_{b_i \in clauses(b)} b_i$$

- Formula constructed using standard SLD-tree
- Calculating probability of monotone DNF formula is NP-hard.

Construction of DNF Formula

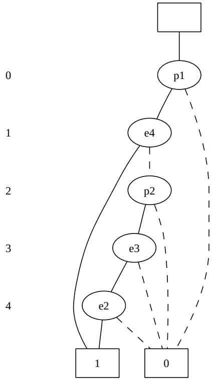
Example

```
1.0 : path(X,Y) :- edge(X,Y).
1.0 : path(X,Y) :- edge(X,Z), path(Z,Y).
0.9 : edge(a,c).   0.7 : edge(c,b).
0.6 : edge(d,c).   0.9 : edge(d,b).
```



$d = (p_1 \wedge e_4) \vee (p_2 \wedge e_3 \wedge p_1 \wedge e_2)$

DNF Formula as Binary Decision Diagram (BDD)



$(p_1 \wedge e_4) \quad \vee$
$(p_2 \wedge e_3 \wedge p_1 \wedge e_2)$

- Efficient graphical representation for Boolean functions [Bryant,1986]
- Nodes labeled with Boolean variables
- Solid edge: variable assigned 1
- Dashed edge: variable assigned 0
- Probability of formula starting at node *n*:

$$\begin{aligned} prob(n) = {} & p_n \cdot prob(n=1) \\ & + (1-p_n) \cdot prob(n=0) \end{aligned}$$

- Experiments show: up to 100.000 conjuncts feasible, depending on formula

Approximate Inference

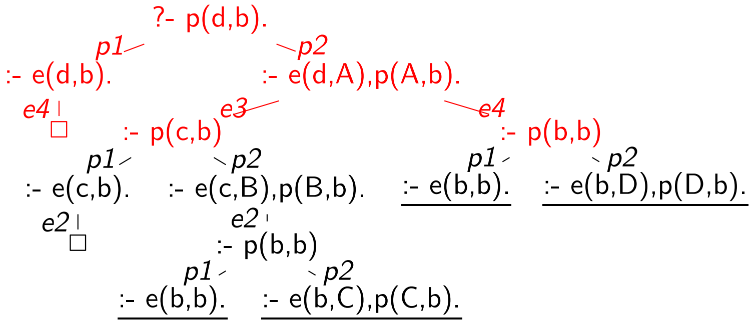
- Incremental, levelwise search for proofs
- After each level, construct two DNF formulae:
  - *low*: all proofs found up to current level
  - *up*: all derivations up to current level that have not yet failed
- $low \models d \models up$
- Bounds on probability obtained using BDDs for *low* and *up*:

$$P(low) \leq P(q|T) \leq P(up)$$

- Stop if $|P(up) - P(low)| \leq \delta$ for some small δ
- Similar to [Poole,1992]

Approximate Inference

Example revisited



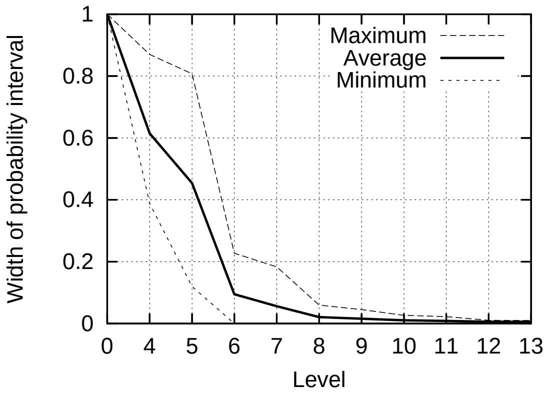
$$
\begin{aligned}
low &= (p_1 \wedge e_4) \\
up &= (p_1 \wedge e_4) \vee (p_2 \wedge e_3) \vee (p_2 \wedge e_4)
\end{aligned}
$$

$$d = (p_1 \wedge e_4) \vee (p_2 \wedge e_3 \wedge p_1 \wedge e_2)$$

Experiments

- Real biological graph G around four random Alzheimer genes (5220 nodes, 11530 edges)
- Example query: connection between two of the genes
- 10 sequences of subgraphs $G_1 \subset G_2 \subset \ldots$ of sizes $200, 400, \ldots$ edges obtained by randomly subsampling edges from G
- Each G_i consists of exactly one connected component and contains both genes used in the query.
- Approximate inference using interval width $\delta = 0.01$
- First level of search contains proofs with up to 4 clauses, this bound is incremented by one clause on each level.

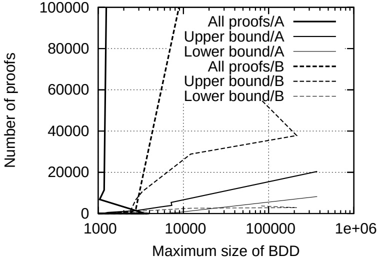
Convergence of Probability Interval Width



- Results for 10 graphs with 1400 edges each
- 6 to 13 levels taking 15 seconds to 4 minutes

Results of Scalability Experiments

- Good scalability as often only small fraction of proofs needed for approximation
- Example query solved for graphs with up to 1400 to 4600 edges, depending on the random sample
- Runtimes from some seconds up to four hours for larger graphs
- Runtimes influenced by many factors, difficult to predict based on size of graph



Conclusions

- ProbLog: a simple Probabilistic Prolog assuming independence between clauses
- Probability calculation using
  - Monotone DNF formulae
  - Binary decision diagrams
- Effective and efficient approximation algorithm
- Experimental evaluation on real-world problem of mining large biological networks

References

R.E. Bryant.
Graph-based algorithms for boolean function manipulation.
IEEE Trans. Computers, 35(8):677–691, 1986.

L. De Raedt, A. Kimmig, H. Toivonen.
ProbLog: A Probabilistic Prolog and its Application in Link Discovery.
In *Proc. 20th IJCAI*, 2007, in Press.

D. Poole.
Logic programming, abduction and probability.
In *Fifth Generation Computing Systems*, pages 530–538, 1992.

P. Sevon, L. Eronen, P. Hintsanen, K. Kulovesi, H. Toivonen.
Link discovery in graphs derived from biological databases.
In *Data Integration in the Life Sciences 2006*, volume 4075 of *LNBI*, pages 35 – 49, 2006. Springer.