— . . .

MCMC for SLPs

James Cussens and Nicos Angelopoulos

{jc,nicos}@cs.york.ac.uk.

Department of Computer Science,
University of York, England.

stochastic logic programs

A stochastic logic program (SLP), is a logic program in which some predicates might be probabilistic. Each clause, of a probabilistic predicate, has attached to it a *parameter* (or *label*).

Example program $\mathcal{S}_0$

```
0.4 :: s(X) :- p(X), p(X).
0.6 :: s(X) :- q(X).

0.3 :: p(a).        0.2 :: q(a).
0.7 :: p(b).        0.8 :: q(b).
```

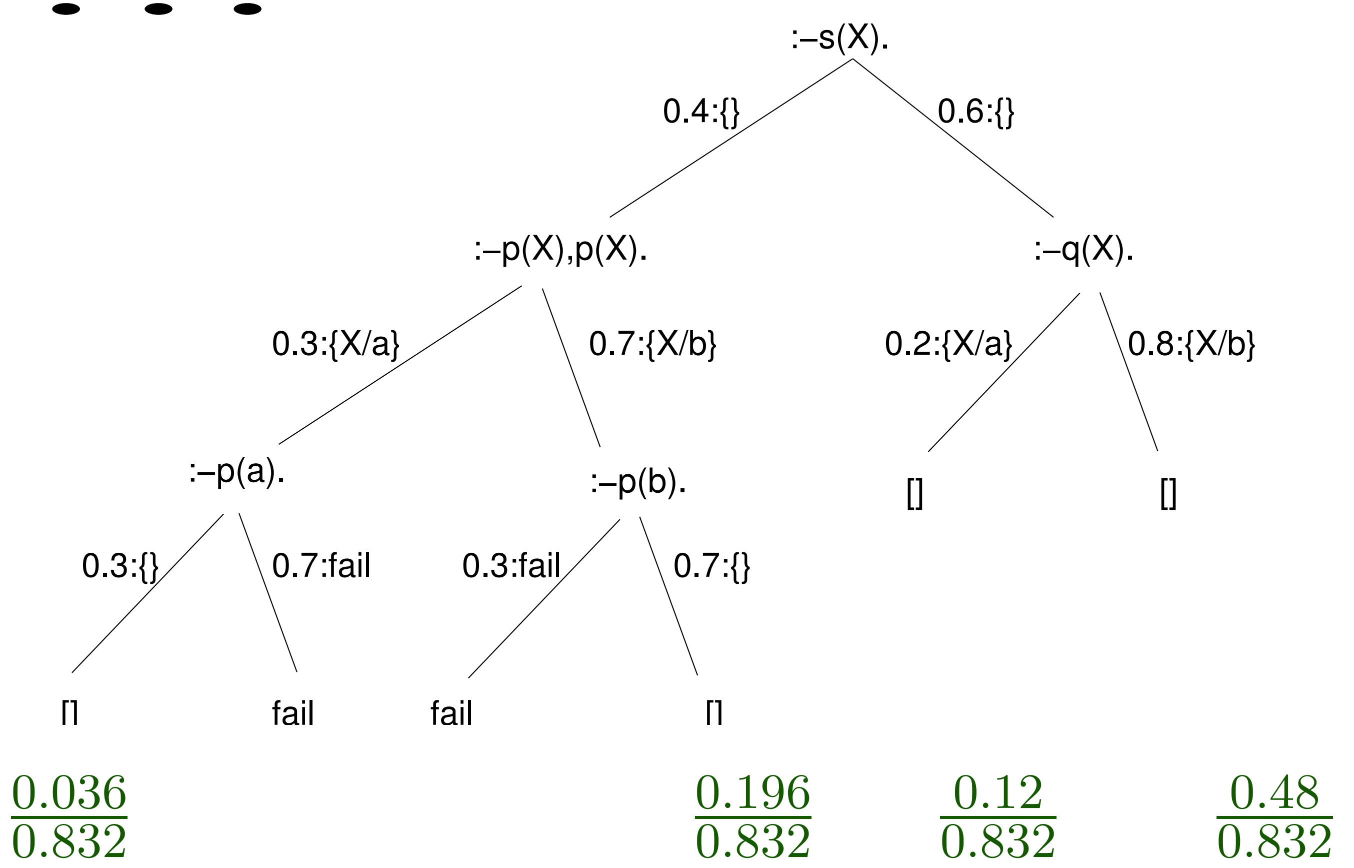
An SLP might be *pure* or/and *normalised*.

distribution

For program $\mathcal{S}$, query $\mathcal{Q}$, we define p over computed answers (yields) yk

$$p_\lambda(y_k) = Z_\lambda^{-1} \sum_{r \in X(y_k)} \prod_{i=1}^{n} \lambda_i^{\nu_i(r)}$$

$p_\lambda(s(a))$



$$p_\lambda(s(a)) = \frac{0.036 + 0.12}{0.832} = 0.1875$$

Bayesian learning

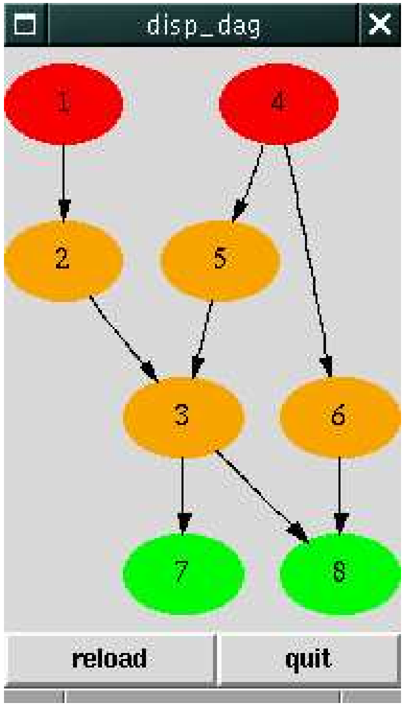
Bayesian learning approaches built statistical models from data. Unlike searching for best model which explains the data, many models may contribute features to learned model.

We used SLPs to construct Bayesian network (BN) models. The SLP program generates all possible BNs and each particular BN can be the computed answer. Its prior probability is defined from the labels in the SLP program.

We used MCMC algorithm to do Bayesian learning over the generator SLP.

example BN (Asia)

For example ?- bn([1,2,3,4,5,6,7,8], M).
M = [1-[],2-[1],3-[2,5],4-[],5-[4],6-[4],7-[3],8-[3,6]].



simple BN Generator

```
bn([],[]).
bn([RV|RVs],BN) :-
    bn(RVs,TailBN),
    connect(RV,RVs,TailBN,BN),
    no_cycle_introduced(RV,BN).

connect(RV,RVs,TailBN,BN) :-
    choose_edges(RVs,RV,Edges),
    add_vertices(TailBN,[RV],BN1),
    add_edges(BN1,Edges,BN).

choose_edges([],RV,[]).
choose_edges([H|T],RV,Edges) :-
    choose_edges1([H|T],RV,Edges).
```

simple BN Generator (Cont.)

```
%RV parent of H
0.3 :: choose_edges1([H|T],RV,[H-RV|Rest]) :-
    choose_edges(T,RV,Rest).
%RV child of H
0.4 :: choose_edges1([H|T],RV,[RV-H|Rest]) :-
    choose_edges(T,RV,Rest).
%no direct connection
0.3 :: choose_edges1([_H|T],RV,Rest) :-
    choose_edges(T,RV,Rest).
```

MCMC

Markov Chain - construct a chain by visiting possible models. Given that the process of next visited model satisfies some weak conditions, then the chain converges to the true posterior.

Monte Carlo - integrate over desirable features by dividing the number of models having the feature, by the number of visited models.

algorithm

MCMC is used to estimate $P(M \mid D)$ from a prior distribution $p(M)$.

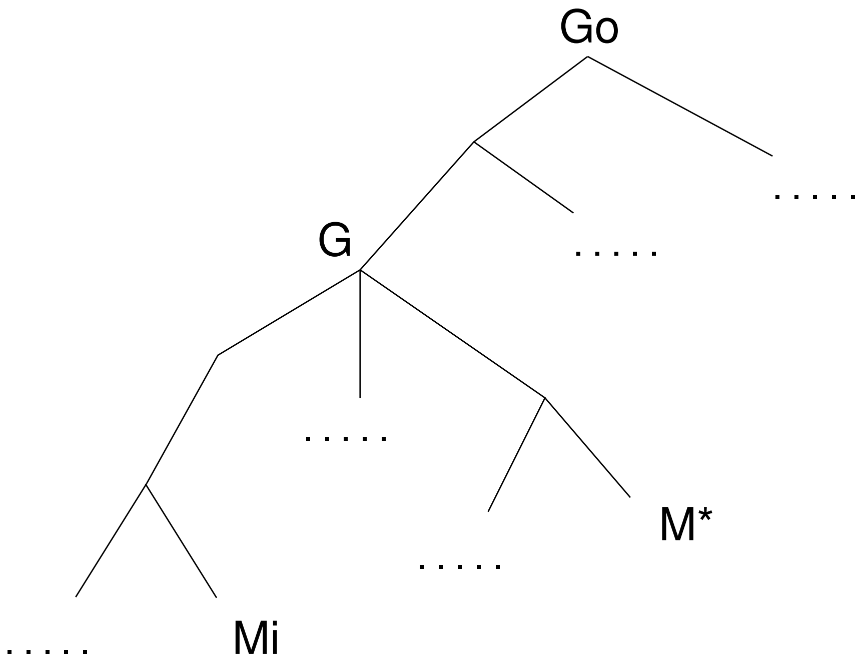
0. Set $i = 0$ and find M_0 using the prior.
1. From M_i produce a candidate model M_*. Let the probability of reaching M_* be $q(M_i, M_*)$.
2. Let

$$\alpha(M_i, M_*) = min \left\{ \frac{q(M_*, M_i)P(D|M_*)P(M*)}{q(M_i, M_*)P(D|M_i)P(M_i)}, 1 \right\}$$

$$M_{i+1} = \begin{cases} M_* & \text{with probability } \alpha(M_i, M_*) \\ M_i & \text{with probability } 1 - \alpha(M_i, M_*) \end{cases}$$

3. If i reached limit then terminate, else set $i = i + 1$ and repeat from 1.

model transitions

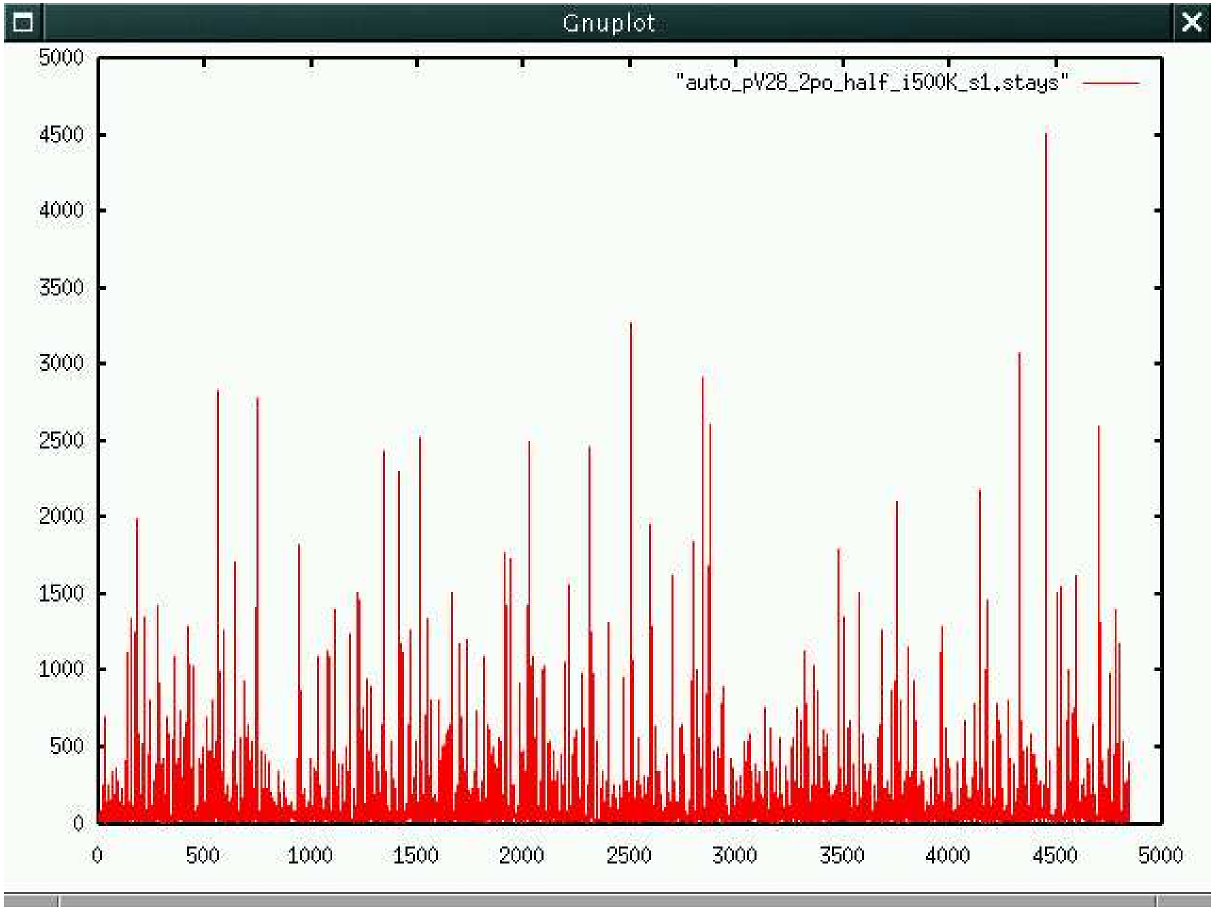


transformations

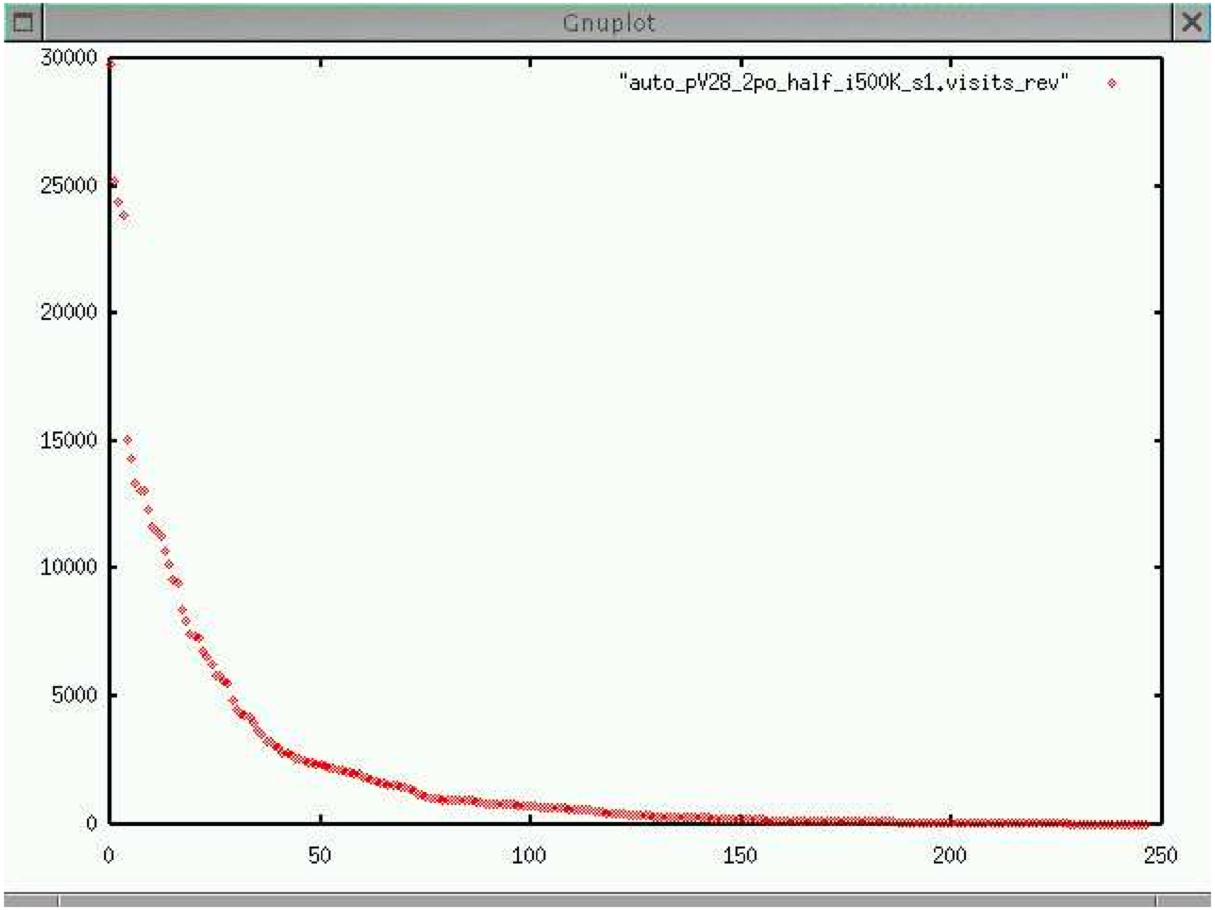
Implementation,

- Add unique Id to each clause.
- Two versions of the followed path (as Ids) are kept as extra arguments.
- All clauses of a probabilistic predicate are collated to a single clause.
- Use run time meta-predicate to perform goal to clause matching.

mixing

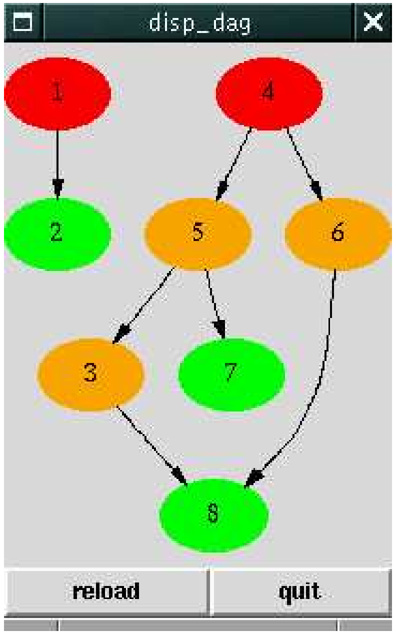

Gnuplot
"auto_pV28_2po_half_i500K_s1.stays"
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

visits


Gnuplot
"auto_pV28_2po_half_i500K_s1.visits_rev"
30000
25000
20000
15000
10000
5000
0
0
50
100
150
200
250

recovery

With topological ordering constraint and a maximum of 2 parents per node, the algorithm recovers most of the BN arcs in 0.5 M iterations. For example for a .99 cut-off we have :



Missing :

- $2 \rightarrow 3$ (.84)
- $3 \rightarrow 7$ (.47)

Superfluous :

- $5 \rightarrow 7$

software

…available from

`http://www.cs.york.ac.uk/~nicos/sware/slps`