MCMC over Model Structures defined by Stochastic Logic Programs.

James Cussens and Nicos Angelopoulos

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

Department of Computer Science,

University of York.

Learning distributions

Given

D	data
$\mathcal{M}$	set of possible models explaining the data

Fit

$\pi(M)$	probability distribution over $\mathcal{M}$

Bayesian learning

Given

D	data
$\mathcal{M}$	set of possible models explaining the data
$p(M)$	prior probability of a model
$L(D \mid M)$	likelihood of a model

Fit

$\pi(M)$	posterior probability over $\mathcal{M}$

$$\pi(M) = p(M|D) = \frac{L(D|M)p(M)}{\sum_M L(D|M)p(M)}$$

MCMC for Bayesian learning

Use MCMC to approximate normalisation factor in Bayes' theorem

$$\sum_M L(D|M)p(M)$$

Markov chain

Construct a chain of visited models, $M_1, M_2, \ldots, M_n$, by proposing M_* from M_i , with probability $q(M_*, M_i)$. Stochastically accept proposed model.

Monte Carlo

Approximate the posterior $\pi(M)$ by $\frac{\#M_i}{\sum_k \#M_k}$

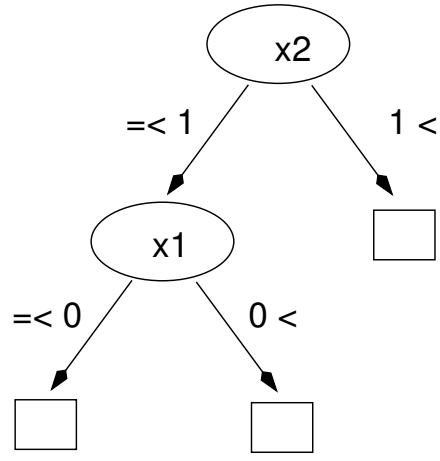
SLP defined Cart Priors

```
1/3 : cart( leaf ).
2/3 : cart( node(F,V,L,R) ) :-
        feature( F ),
        value_of( F, V ),
        cart( L ),
        cart( R ).

1/2 : feature( a ).
1/2 : feature( b ).

1/2 : value_of( _, 0 ).
1/2 : value_of( _, 1 ).
```

SLP defined Cart Priors

```
1/3 : cart( leaf ).
2/3 : cart( node(F,V,L,R) ) :-
        feature( F ),
        value_of( F, V ),
        cart( L ),
        cart( R ).

1/2 : feature( x1 ).
1/2 : feature( x2 ).

1/2 : value_of( _, 0 ).
1/2 : value_of( _, 1 ).
```

?- cart(M).

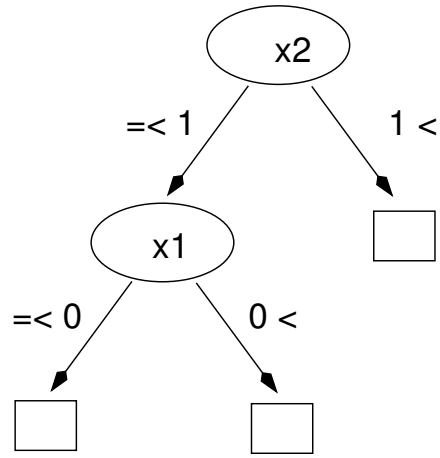


M = node(b, 1, node(a,0,leaf,leaf), leaf)

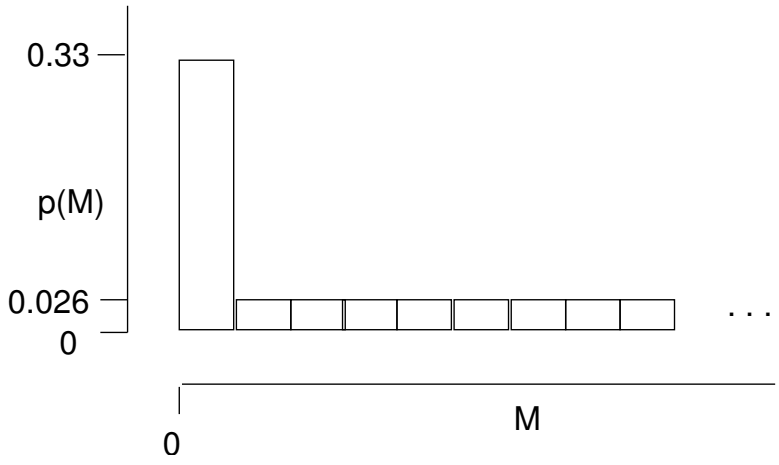
SLP defined Cart Priors

```
1/3 : cart( leaf ).
2/3 : cart( node(F,V,L,R) ) :-
        feature( F ),
        value_of( F, V ),
        cart( L ),
        cart( R ).

1/2 : feature( x1 ).
1/2 : feature( x2 ).

1/2 : value_of( _, 0 ).
1/2 : value_of( _, 1 ).
```

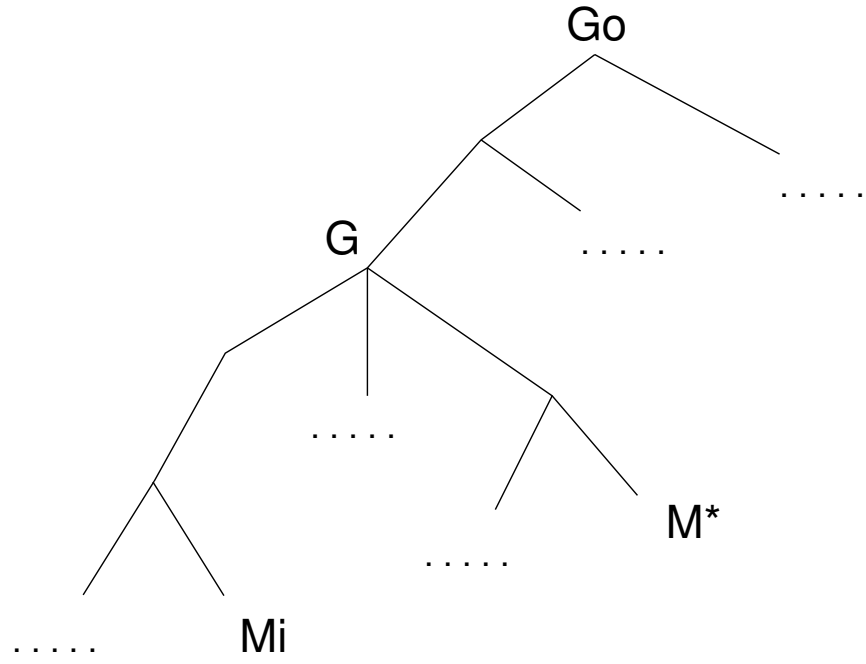
?- cart(M).



M = node(b, 1, node(a,0,leaf,leaf), leaf)



proposal distro q(M_i ,M_*)



Metropolis-Hastings

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

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

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

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

Metropolis-Hastings

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

$$\alpha(M_i, M_*) = min\left\{\kappa(M_i, M_*)\frac{L(D|M_*)}{L(D|M_i)}, 1\right\}$$

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

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

Cart priors - for real

Chipman et.al. : $P_{\text{split}}(\eta) = \alpha(1 + d_\eta)^{-\beta}$

α, β: user parameters controlling tree size

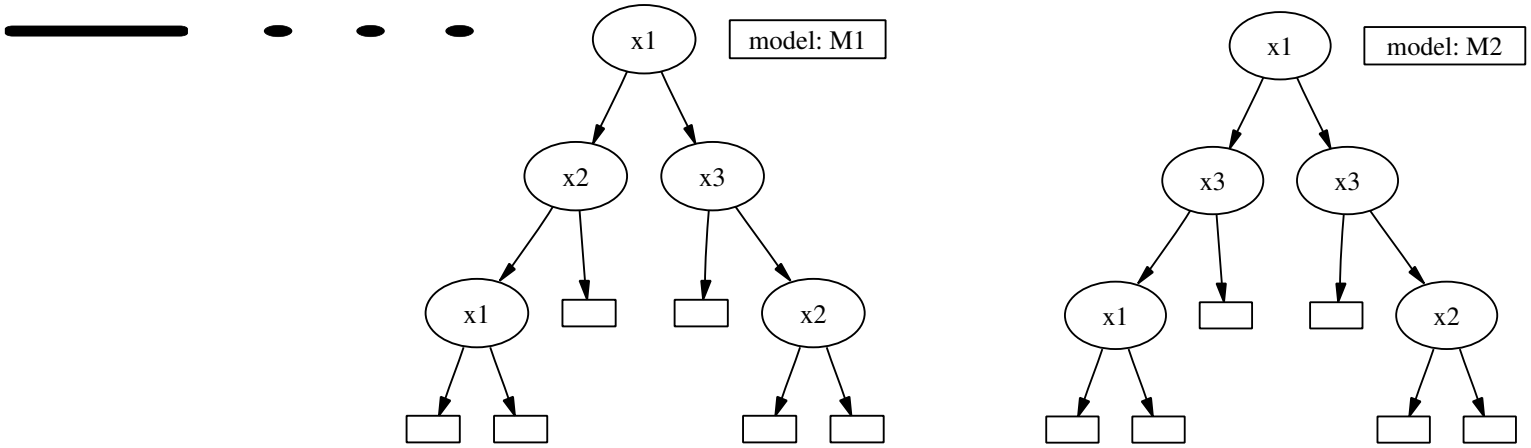
Cart priors - for real

Chipman et.al. : $P_{\text{split}}(\eta) = \alpha(1 + d_\eta)^{-\beta}$

α, β: user parameters controlling tree size

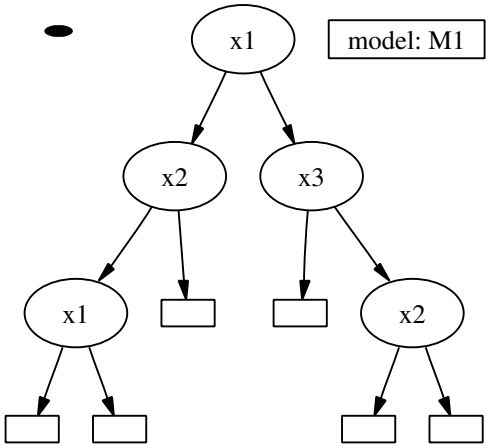
```
1 - Sp: [Sp]: cart( Data, D, A/B, leaf(Data) ).
Sp    : [Sp]: cart( Data, D, A/B, node(F,V,L,R) ) :
         branch( Data, F, V, LData, RData ),
         D1 is D + 1,
         NxtSp is A * ((1 + D1) ^ -B),
         [NxtSp] : cart( LData, D1, A/B, L ),
         [NxtSp] : cart( RData, D1, A/B, R ).
```

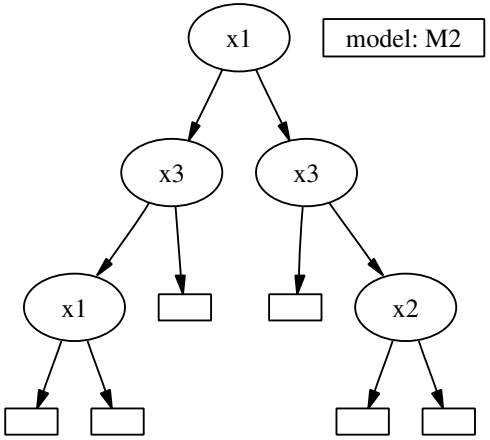
Operations on branches

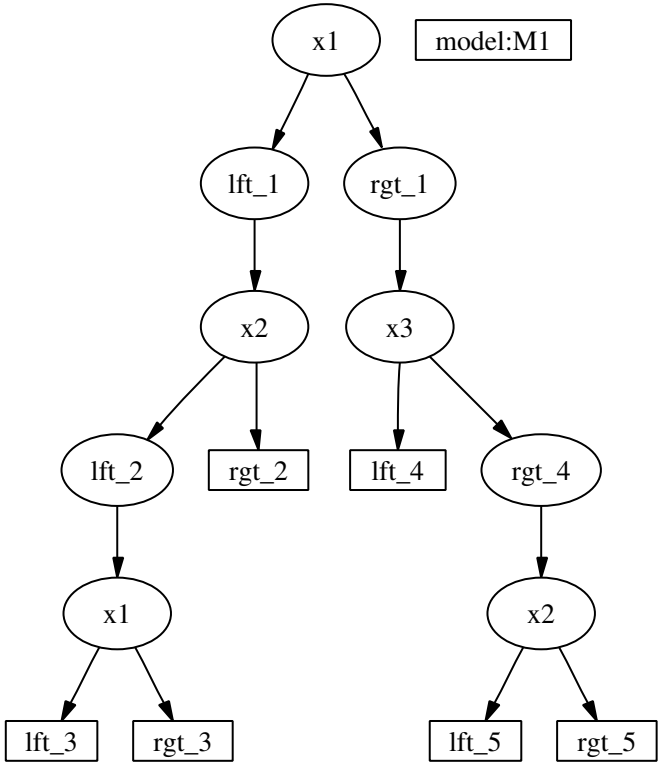


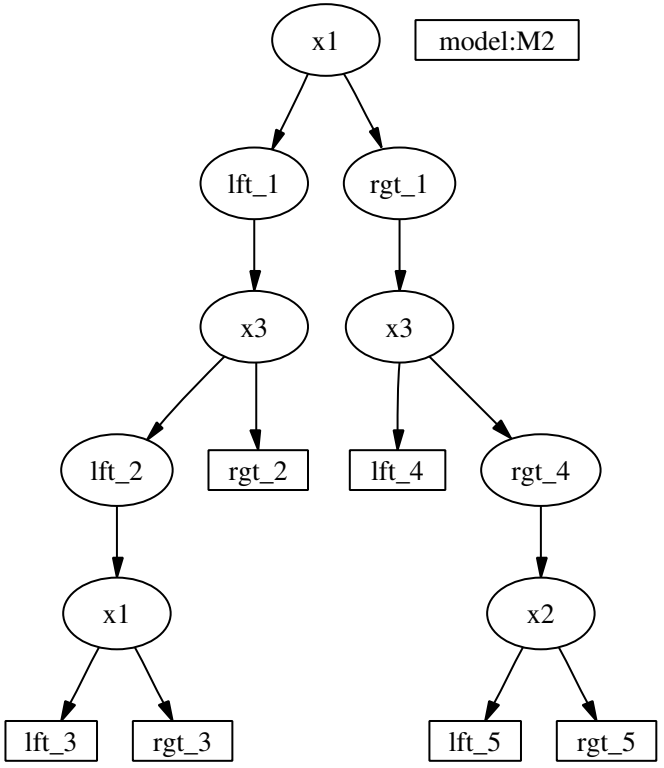

```
1 - Sp: [Sp]: cart( Data, D, A/B, leaf(Data) ).
Sp    : [Sp]: cart( Data, D, A/B, node(F,V,L,R) ) :-
         branch( Data, F, V, LData, RData ),
         D1 is D + 1,
         NxtSp is A * ((1 + D1) ^ -B),
         [NxtSp] : cart( LData, D1, A/B, L ),
         [NxtSp] : cart( RData, D1, A/B, R ).
```

Operations on branches





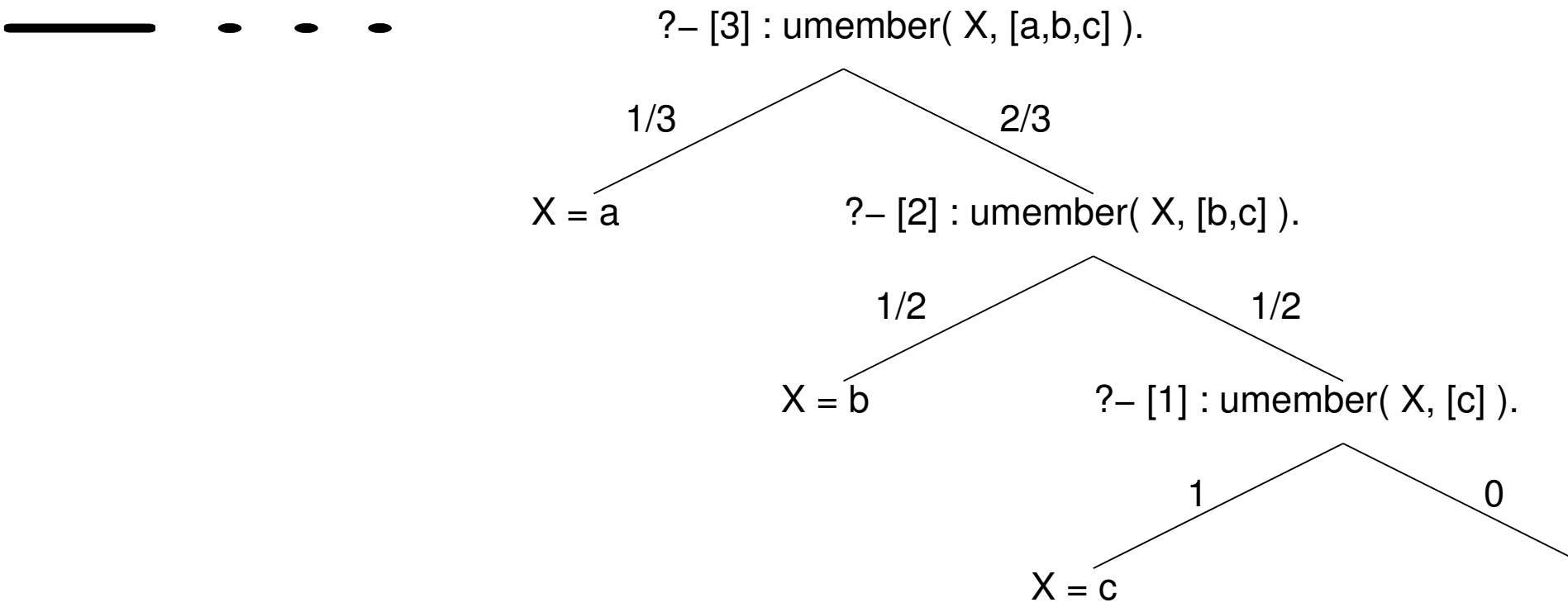




tail recursion optimisation

```
1/L         : [L] : umember( El, [El|T] ).
1 - (1/L) : [L] : umember( El, [_|T] ) :-
                  [L-1] : umember( El, T ).
```

tail recursion optimisation




```
1/L         : [L] : umember( El, [El|T] ).
1 - (1/L)   : [L] : umember( El, [_|T] ) :-
                    [L-1] : umember( El, T ).
```

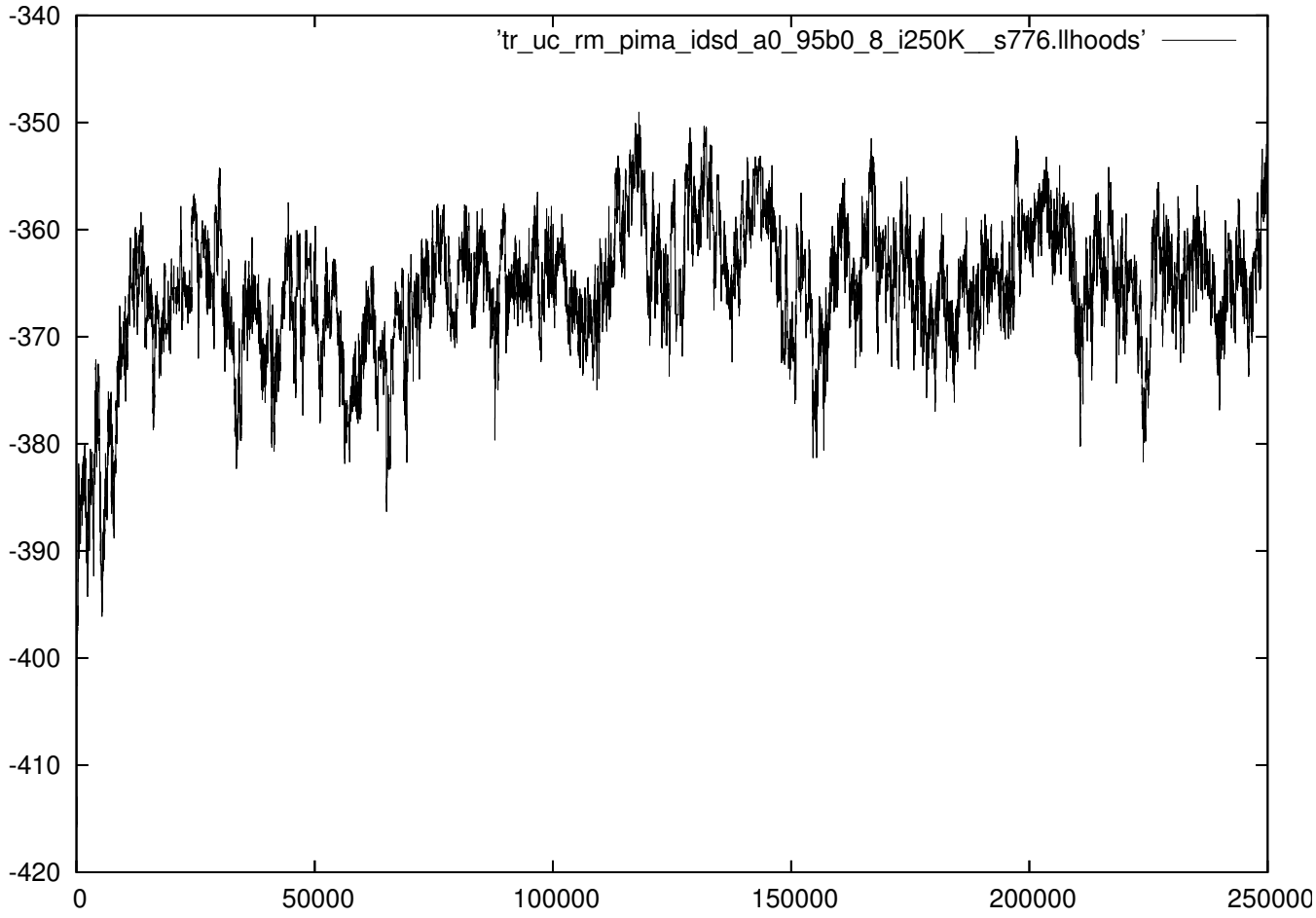
Experiment

Pima Indians Diabetes Database
768 complete entries of 8 variables.

Denison et.al. run 250,000 iterations of local perturbations.
Their best likelihood model: -343.056

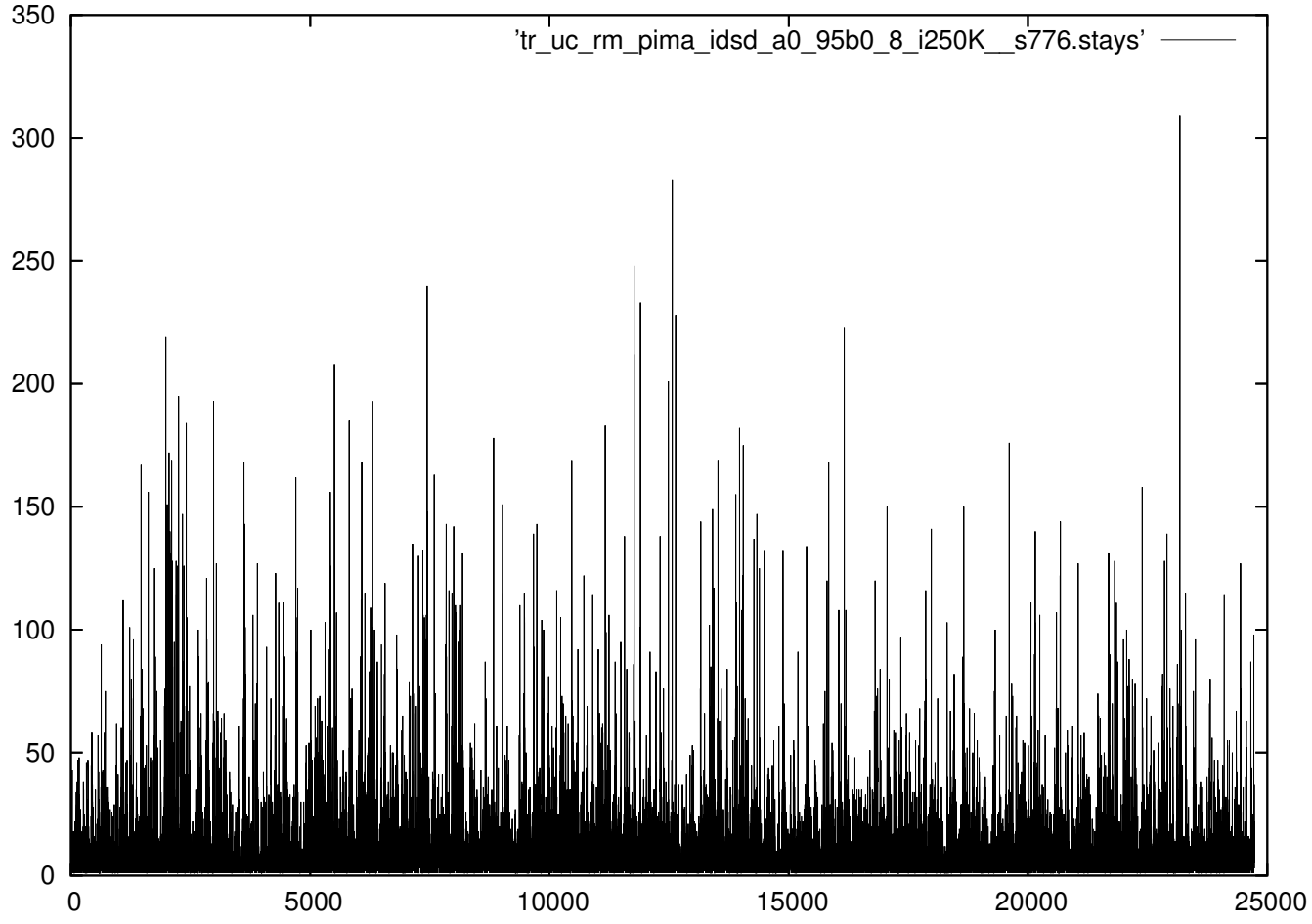
Our experiment run for 250,000 iterations with branch replacing.
Parameters: uniform choice proposal, $\alpha = .95$ $\beta = .8$
Our best likelihood model: -347.651

Likelihoods trace

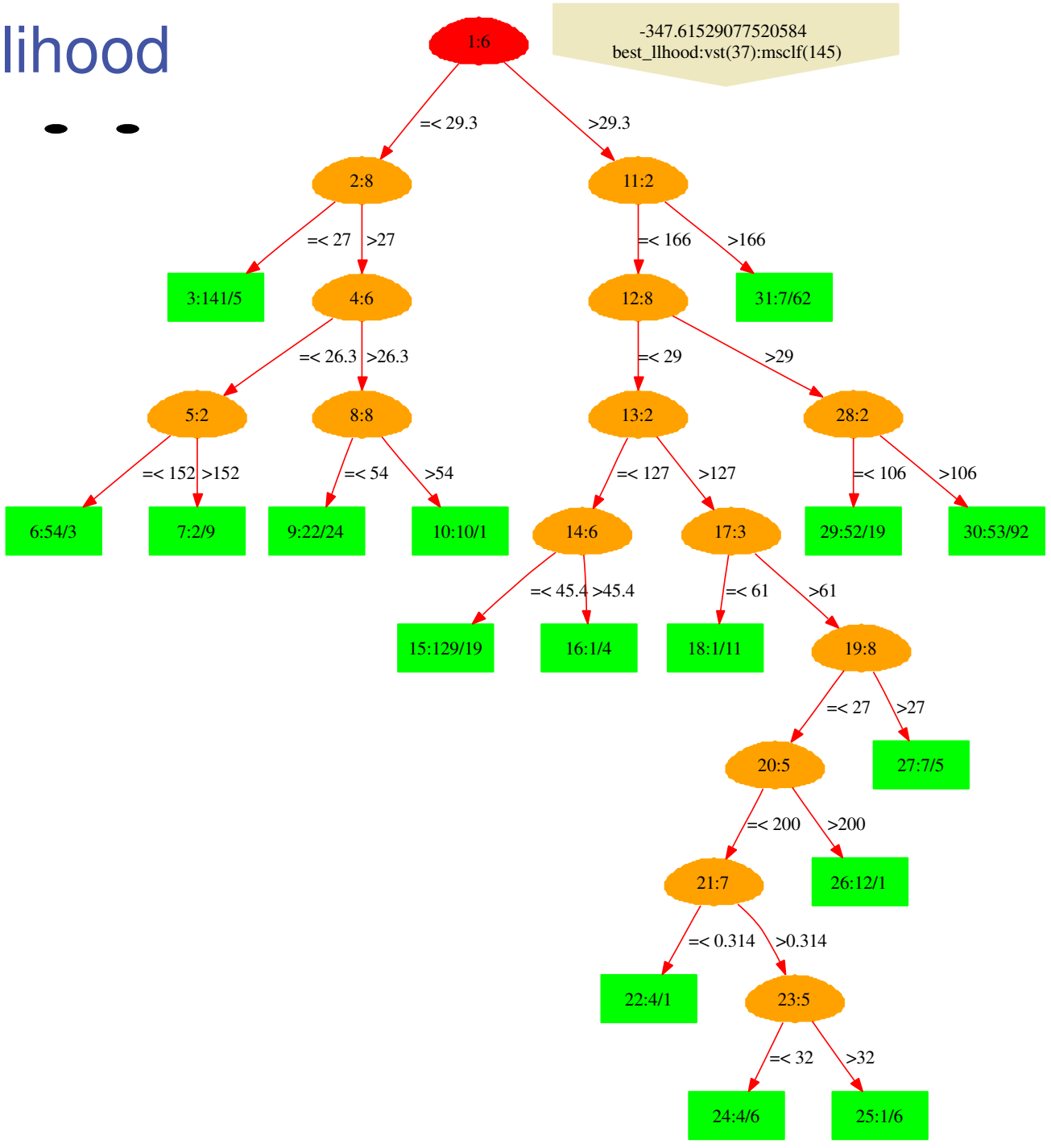


$$\beta = .8, \alpha = .95, proposal = uniform\ choice$$

visits


'tr_uc_rm_pima_idsd_a0_95b0_8_i250K__s776.stays'
350
300
250
200
150
100
50
0
0
5000
10000
15000
20000
25000

Best likelihood



bottom line

Algorithmic reality and representational power need to work in tandem.

Software available from

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