Stat 521A
Lecture 22

Outline

- Algorithms for finding MAP structure (18.4)
- MCMC over DAG structure (18.5)
- Dynamic programming + MH
- Stochastic search

Computationally intractable

- There are $O(d!2^{\binom{d}{2}})$DAGs on d nodes

d	2	3	4	5	6	7	8	9
#G(d)	3	25	543	29,281	3,781,503	1.1e9	7.8e11	1.2e15

Trees

- Can learn optimal tree using MST algo in $O(n^2 \log n + n^2 M)$ time, n=#num nodes, M=#cases

Definition 18.4.1: *A network structure $\mathcal{G}$ is called* tree-structured *if each variable X has at most one parent in $\mathcal{G}$, i.e., $|\mathrm{Pa}_X^{\mathcal{G}}| \leq 1$.* ■

$$\Delta(\mathcal{G}) = \mathrm{score}(\mathcal{G} : \mathcal{D}) - \mathrm{score}(\mathcal{G}_\emptyset : \mathcal{D})$$

$$\Delta(\mathcal{G}) = \sum_{i, \mathrm{Pa}_i^{\mathcal{G}} \neq \emptyset} \left(\mathrm{FamScore}(X_i \mid \mathrm{Pa}_i^{\mathcal{G}} : \mathcal{D}) - \mathrm{FamScore}(X_i : \mathcal{D})\right)$$

$$w_{X_j \to X_i} = \mathrm{FamScore}(X_i \mid X_j : \mathcal{D}) - \mathrm{FamScore}(X_i : \mathcal{D})$$

$$\Delta(\mathcal{G}) = \sum_{X_j \to X_i \in \mathcal{G}} w_{X_j \to X_i}$$

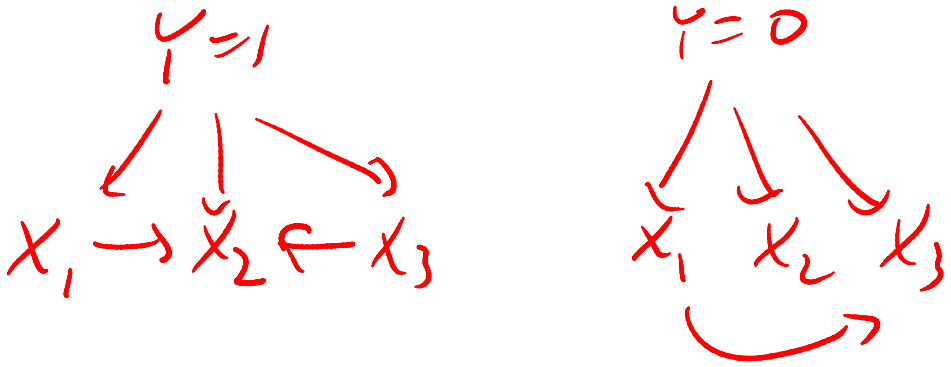
Undirected max weight spanning tree

Score equivalence => $w_{X_i \to X_j} = w_{X_j \to X_i}$.

$$\mathrm{score}_L(\mathcal{G}_1 : \mathcal{D}) - \mathrm{score}_L(\mathcal{G}_0 : \mathcal{D}) = M \sum_{x,y} \hat{P}(x,y) \log \frac{\hat{P}(y \mid x)}{\hat{P}(y)} = M \cdot I_{\hat{P}}(X;Y)$$

TAN classifiers

- Tree-augmented naïve Bayes
- Can learn tree structure for each class conditional density



Mixtures of trees

- We can fit mixtures of trees using EM: just run MST algorithm in M step
- Analogous to mixture of diagonal Gaussians

DAG with known order

- Can find optimal set of parents for each node independently

Proposition 18.4.2: *Let $\prec$ be an ordering over $\mathcal{X}$, and let* $\text{score}(\mathcal{G} : \mathcal{D})$ *be a decomposable score. If we choose $\mathcal{G}$ to be the network where*

$$\text{Pa}_i^{\mathcal{G}} = \arg \max_{\boldsymbol{U} \subseteq \{X_j : X_j \prec X_i\}} \text{FamScore}(X_i \mid \boldsymbol{U}_i : \mathcal{D})$$

for each i, then $\mathcal{G}$ maximizes the score among the structures consistent with $\prec$.

- If at most d parents, last node Xn must select from

$$X_n \text{ is } 1 + \binom{n-1}{2} + \ldots + \binom{n-1}{d} = O(d\binom{n-1}{d})$$

- If CPDs are GLMs, can use lasso to find parents
- If order unknown, can search over orders.

Dependency networks

- A depnet is a set of full conditionals p(Xi|X(-i)) learned independently.
- There may not be any joint which is consistent with these conditionals.
- However, one can define a (non-unique) joint by using an ordered Gibbs sampler.
- If the conditionals are learned from (lots of) data, they are likely to be consistent.
- By performing variable selection at each node independently, we get a sparse graph.
- Provides a fast way to visualize dependencies.

Collaborative filtering

- One succesful application of depnets is CF.
- Xi=1 if item i has been bought, Xi=0 otherwise
- Assume S=set of bought items, Sbar = not bought items, i = target item. Compute p(Xi|S=1,Sbar=0).
- In a depnet, this is a simple lookup – all other nodes are observed.
- In a DGM, this is also fairly simple – product of CPDs in the Markov blanket.
- Both techniques have similar predictive accuracy, but depnet is much faster to learn.
- Ships in Microsoft's ecommerce package.

DAG with unknown order

- Thm 18.4.3 It is NP-hard to find the optimal DAG with d >= 2 parents.
- Standard approach: heuristic local search (eg hill climbing), using add/ delete/ reverse edge (n^2 neighbors to each DAG).
- Diameter of space is $O(n^2)$: to get from G1 to G2, delete all edges of G1 then add all edges of G2.
- If too many neighbors, use first best instead of evaluating all of them.
- Often there will be large plateaus of I-equivalent DAGs. Can use tabu search to escape these.
- Multiple restarts or data perturbation.

Data perturbation

- Can be used to escape local minima for many ML algorithms, where score = sum_i score(D_i)
- Idea: use weights w_i, and perturb them at random (or more cleverly – rather like boosting)

```
1    G ← Search(G_∅, D, score, O)
2    G_best ← G
3    t ← t_0
4    for i = 1, ... until convergence
5      D' ← Perturb(D, t)
6      G ← Search(G, D', score, O)
7      if score(G : D) > score(G_best : D) then
8        G_best ← G
9      t ← γ · t
10
11   return G_best
```

Efficient scoring of proposed new graph

$$\delta(\mathcal{G} : o) = \text{score}(o(\mathcal{G}) : \mathcal{D}) - \text{score}(\mathcal{G} : \mathcal{D})$$

to be the change of score associated with applying o on $\mathcal{G}$. Using score decomposition, we can compute this quantity relatively efficiently.

Proposition 18.4.4: *Let $\mathcal{G}$ be a network structure, and* score *be a decomposable score.*

- *If o is "Add $X \rightarrow Y$", and $X \rightarrow Y \notin \mathcal{G}$, then*

$$\delta(\mathcal{G} : o) = \text{FamScore}(Y, \text{Pa}_Y^{\mathcal{G}} \cup \{X\} : \mathcal{D}) - \text{FamScore}(Y, \text{Pa}_Y^{\mathcal{G}} : \mathcal{D})$$

- *If o is "Delete $X \rightarrow Y$" and $X \rightarrow Y \in \mathcal{G}$, then*

$$\delta(\mathcal{G} : o) = \text{FamScore}(Y, \text{Pa}_Y^{\mathcal{G}} - \{X\} : \mathcal{D}) - \text{FamScore}(Y, \text{Pa}_Y^{\mathcal{G}} : \mathcal{D})$$

- *If o is "Reverse $X \rightarrow Y$" and $X \rightarrow Y \in \mathcal{G}$, then*

$$\begin{aligned}\delta(\mathcal{G} : o) = {} & \text{FamScore}(X, \text{Pa}_X^{\mathcal{G}} \cup \{Y\} : \mathcal{D}) + \text{FamScore}(Y, \text{Pa}_Y^{\mathcal{G}} - \{X\} : \mathcal{D}) \\ & - \text{FamScore}(X, \text{Pa}_X^{\mathcal{G}} : \mathcal{D}) - \text{FamScore}(Y, \text{Pa}_Y^{\mathcal{G}} : \mathcal{D})\end{aligned}$$

Efficient update of cached scores

- After accepting change, only have to update scores of affected families - O(n) operators

Proposition 18.4.5: *Let $\mathcal{G}$ and $\mathcal{G}'$ be two network structures, and* score *be a decomposable score.*

- *If o is either "Add $X \rightarrow Y$" or "Delete $X \rightarrow Y$" and $\mathrm{Pa}_Y^{\mathcal{G}} = \mathrm{Pa}_Y^{\mathcal{G}'}$, then $\delta(\mathcal{G} : o) = \delta(\mathcal{G}' : o)$.*

- *If o is "Reverse $X \rightarrow Y$", $\mathrm{Pa}_Y^{\mathcal{G}} = \mathrm{Pa}_Y^{\mathcal{G}'}$, and $\mathrm{Pa}_X^{\mathcal{G}} = \mathrm{Pa}_X^{\mathcal{G}'}$, then $\delta(\mathcal{G} : o) = \delta(\mathcal{G}' : o)$.*

Sufficient statistics

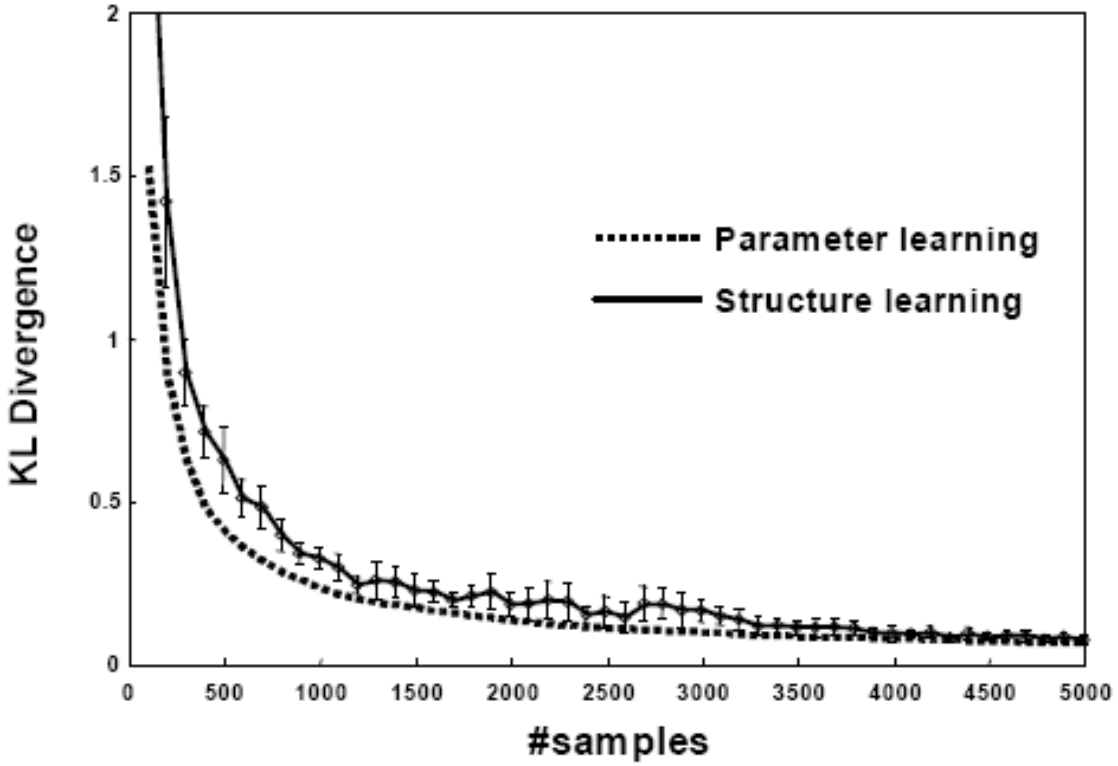
- Need to walk over M rows for all the columns in a given family
- If we need to update n operators, this is O(nM) time
- Can use AD-trees for discrete data, or KD-trees for cts data, to do this more efficiently (possibly subject to approximation error)

Heaps

- Need to search over O(n^2) operators to find best at each step
- Can use a heap to find the best in O(1) time if we do O(n log n) time to update it when scores change

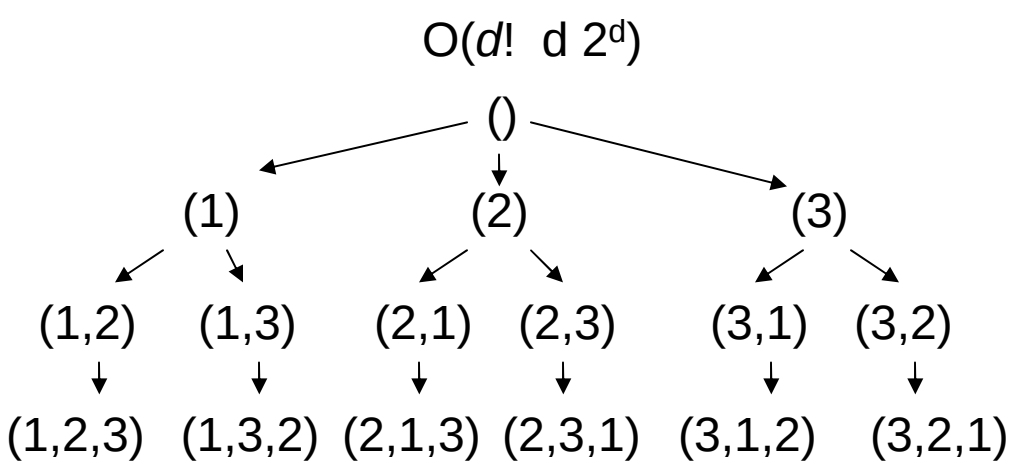
Learning params vs structure

- ICU Alarm network

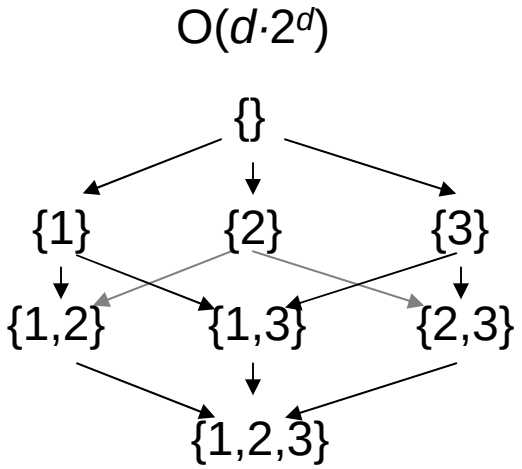


Dynamic programming (DP)

- Can analytically marginalize over *d*! orderings and all possible subsets in O(d 2^d) time/ space using DP and fast Mobius transform
- Since order of parents does not matter, eg $p(X_1|X_2,X_3) == p(X_1|X_3,X_2)$, we can share work
- Can find exact global MAP DAG



Ordered permutation tree



Unordered permutation lattice

Equivalence classes

- Can search through PDAG space - smaller than DAG space, and fewer (if any) plateau
- To evaluate score of a PDAG, convert to a DAG then use score for DAG
- To find neighbors: convert PDAG to DAG, add or delete edge; this changes skeleton hence moves to a new PDAG
- Greedy Equivalence Search: start with empty PDAG, add best edge until local max, then delete best edge till local max. If M->infty, this will provably find optimal PDAG given any consistent scoring fn.
- Performing local updates to score of a PDAG is harder.

Bayes model averaging

- When the sample size is small, the posterior p(G|D) gives support to multiple (non equivalent) models
- We should perform BMA when performing prediction

$$P(\xi[M+1] \mid \mathcal{D}) = \sum_{\mathcal{G}} P(\xi[M+1] \mid \mathcal{D}, \mathcal{G}) P(\mathcal{G} \mid \mathcal{D})$$

- And when computing E[f(G)|D], where f(G) is some feature, eg f(G)=there is an edge X->Y in G, average path length in G

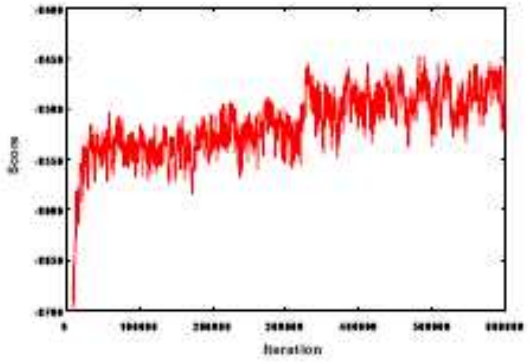
$$\mathbb{E}_{P(\mathcal{G}|\mathcal{D})}[f(\mathcal{G})] = \sum_{\mathcal{G}} f(\mathcal{G}) P(\mathcal{G} \mid \mathcal{D}).$$

MC3

- Markov Chain Monte Carlo Model Composition
- Use MH in space of DAGs, with proposal = uniform over neighbors (add/delete/reverse edge)
- Does not mix well in more than ~10 dimensions. Also, posterior gets more peaky as sample size increases (can use parallel tempering).

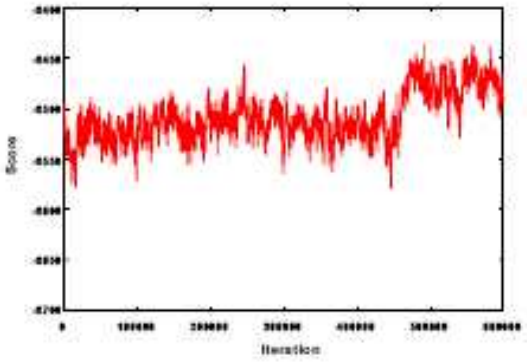
MH on Alarm

Init empty

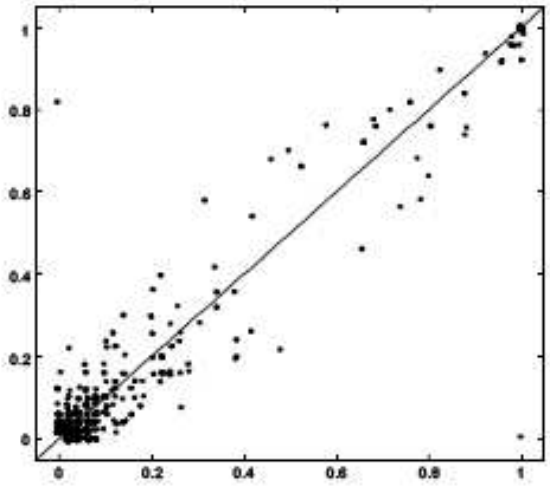


(a)

Init local search

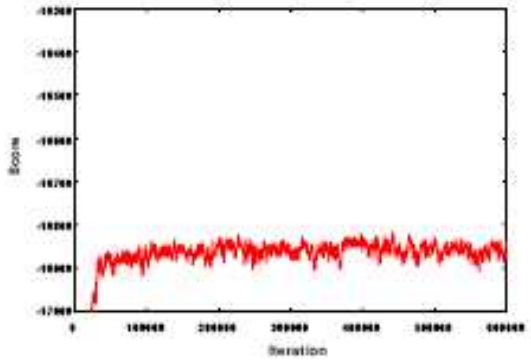


(b)

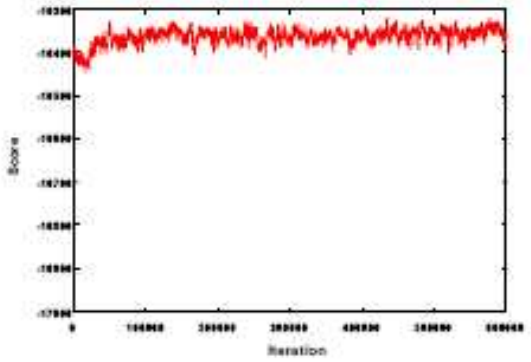


(c)

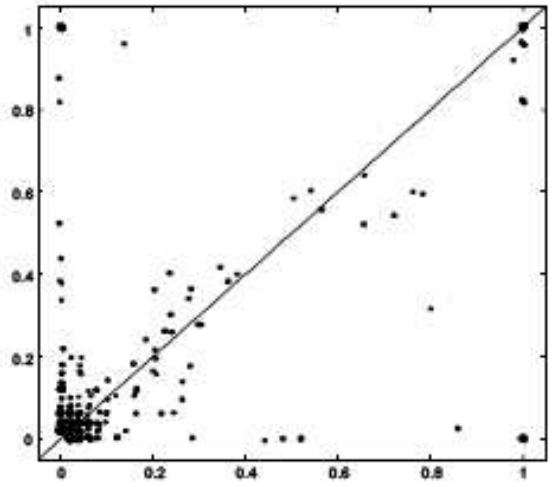
N=500



(a)



(b)



Edge marginals

N=1000

MH in order space

- Given a known order, we can integrate over all possible graphs by summing over all parents sets
- Hence use MH to sample over orders, using traveling-salesman like moves

$$(X_{i_1} \dots X_{i_j} \dots X_{i_d} \dots X_{i_n}) \mapsto (X_{i_1} \dots X_{i_d} \dots X_{i_j} \dots X_{i_n}).$$

$$\min\left[1, \frac{P(\prec', \mathcal{D}) T^{\mathcal{Q}}(\prec' \to \prec)}{P(\prec, \mathcal{D}) T^{\mathcal{Q}}(\mathcal{G} \to \prec')}\right]$$

Target distribution

$$\begin{aligned} P(D \mid \prec) &= \sum_{\mathcal{G} \in \mathcal{G}_{d,\prec}} \prod_i \exp\{\text{FamScore}_B(X_i \mid \text{Pa}_{X_i}^{\mathcal{G}} : \mathcal{D})\} \\ &= \prod_i \sum_{\boldsymbol{U}_i \in \mathcal{U}_{i,\prec}} \exp\{\text{FamScore}_B(X_i \mid \boldsymbol{U}_i : \mathcal{D})\} \end{aligned}$$

$$\mathcal{U}_{i,\prec} = \{\boldsymbol{U} : \boldsymbol{U} \prec X_i, |\boldsymbol{U}| \leq k\}.$$

Posterior features

- Given samples from p(<|D) we compute

$$P(f \mid \mathcal{D}) \approx \frac{1}{T} \sum_{t=1}^{T} P(f \mid \mathcal{D}, \prec_t).$$

- Parent features

Proposition 18.5.1:

$$P(\mathrm{Pa}_{X_i}^{\mathcal{G}} = \boldsymbol{U} \mid D, \prec) = \frac{\exp\{\mathrm{FamScore}_B(X_i \mid \boldsymbol{U} : \mathcal{D})\}}{\sum_{\boldsymbol{U}' \in \mathcal{U}_{i,\prec}} \exp\{\mathrm{FamScore}_B(X_i \mid \boldsymbol{U}' : \mathcal{D})\}}.$$

- Edge features

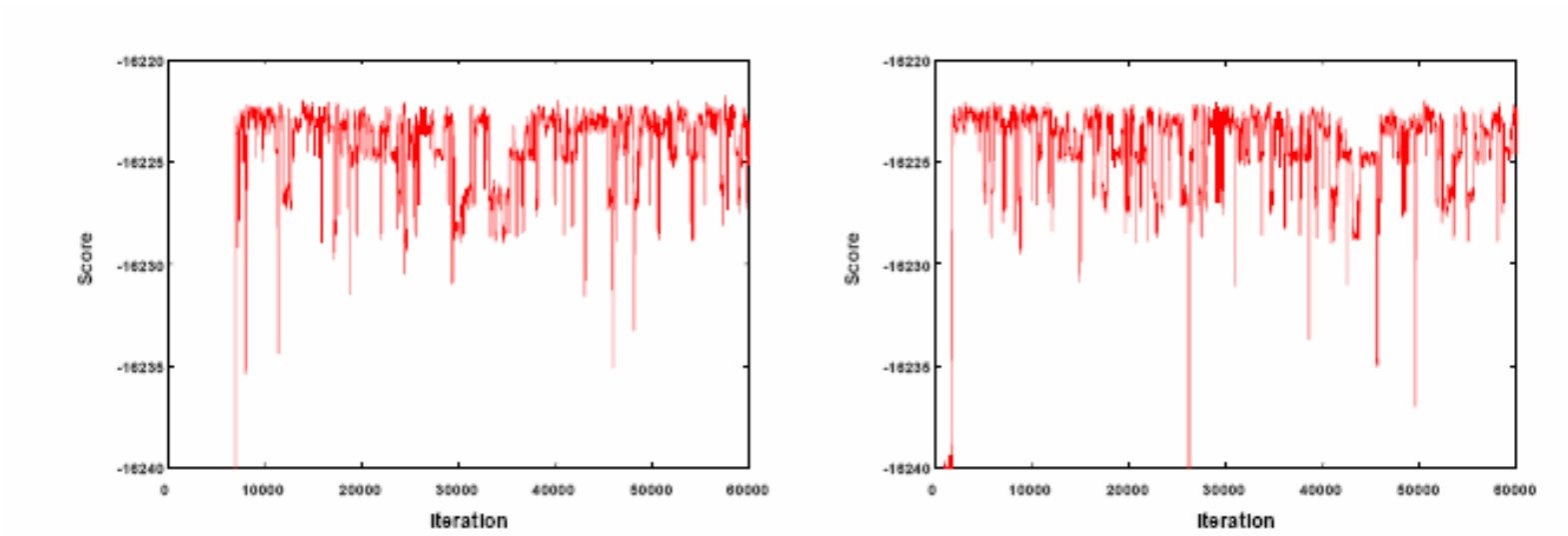
Proposition 18.5.2:

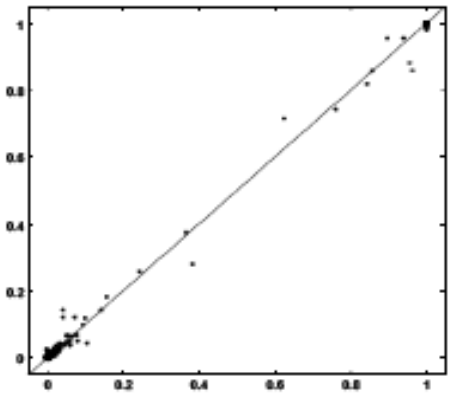
$$P(X_j \in \mathrm{Pa}_{X_i}^{\mathcal{G}} \mid \prec, \mathcal{D}) = \frac{\sum_{\{\boldsymbol{U} \in \mathcal{U}_{i,\prec} \,:\, X_j \in \boldsymbol{U}\}} \exp\{\mathrm{FamScore}_B(X_i \mid \boldsymbol{U} : \mathcal{D})\}}{\sum_{\boldsymbol{U} \in \mathcal{U}_{i,\prec}} \exp\{\mathrm{FamScore}_B(X_i \mid \boldsymbol{U} : \mathcal{D})\}}$$

- General features: sample G given <, then use

$$P(f \mid \prec, \mathcal{D}) = \frac{P(f, \mathcal{D} \mid \prec)}{P(\mathcal{D} \mid \prec)}. \qquad P(f, \mathcal{D} \mid \prec) = \sum_{\mathcal{G} \in \boldsymbol{\mathcal{G}}_{d,\prec}} f(\mathcal{G}) P(\mathcal{G} \mid \prec) P(\mathcal{D} \mid \mathcal{G})$$

RB MH on Alarm





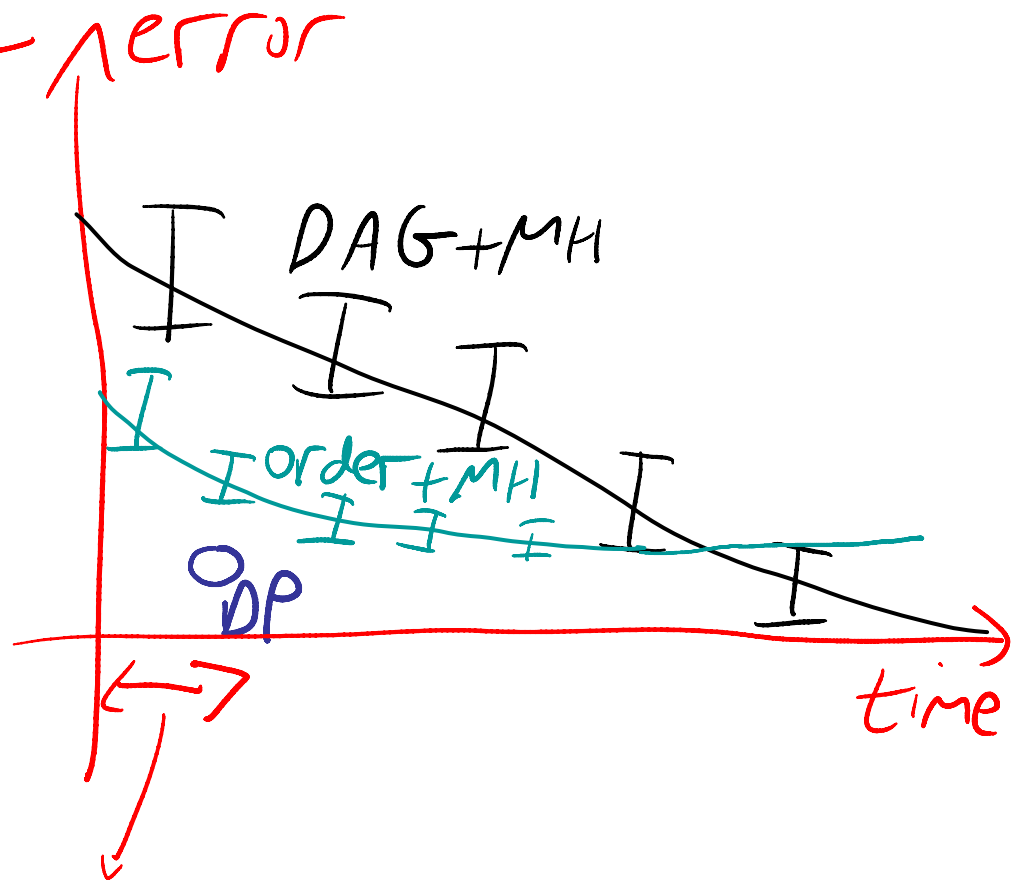
(c)

Dynamic programming

- Koivisto & Sood showed how to compute all edge marginals p(Gij=1|D) exactly in O(n 2^n) time
- Requires special (“modular”) prior p(G) which can be unnatural (see later)

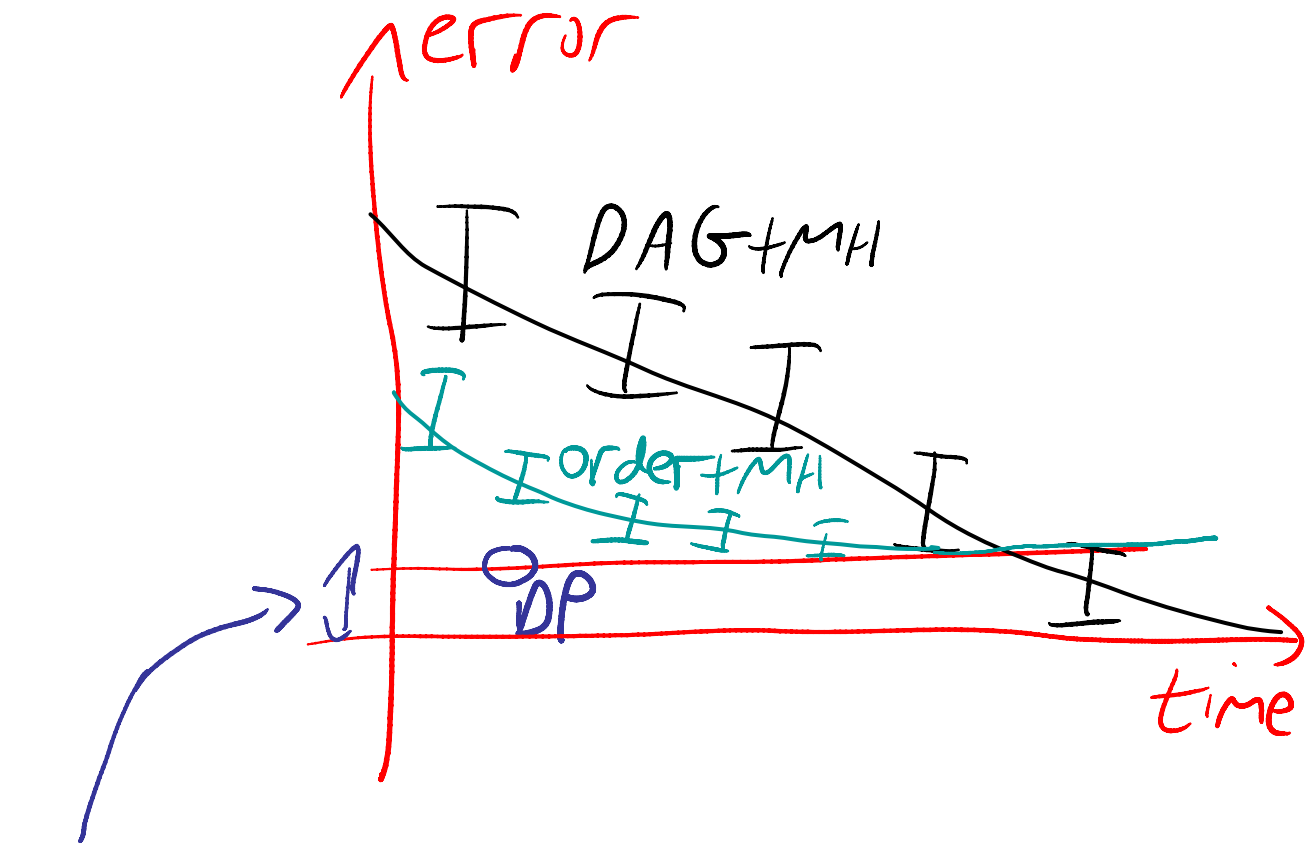
Comparison of approaches

$$\sum_{ij} |\hat{P}(G_{ij} = 1|D) - P(G_{ij} = 1|D)|$$



5 sec (d=10) to 5 mins (d=20)

Error floor due to wrong p(G)



Error due to wrong p(G)

p(G) needed by DP and MH+order

- Joint ("modular") prior on G and $\prec$

$$p(G, \prec) = \frac{1}{Z} \prod_{i=1}^{d} \rho_i(G_i) q_i(\prec_i) I(G, \prec \text{ is valid})$$

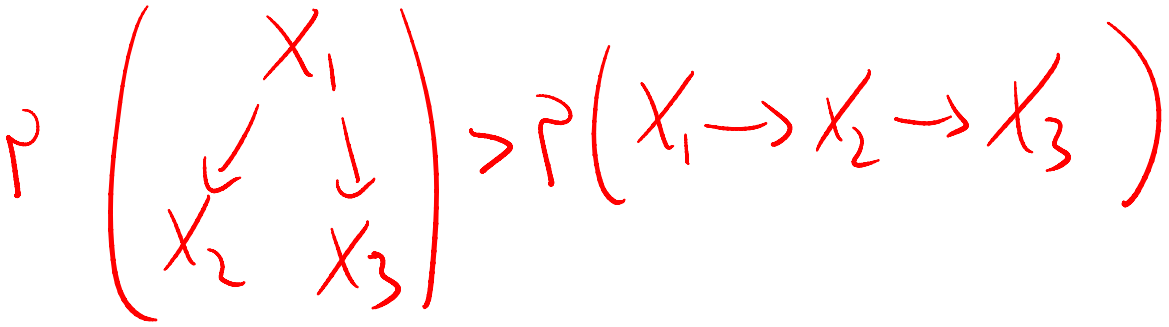
unordered set of parents

ordering of predecessors

- Induced prior on p(G)

$$p(G) = \sum_{\prec} p(G, \prec)$$

Graphs consistent with more orderings are more probable



- Effect will not get erased even with infinite data, since both models are likelihood equivalent

Problems with induced p(G)

- Prior is highly non-uniform
- Effect will not get erased even with infinite data
- Cannot encode arbitrary prior knowledge in p(G)

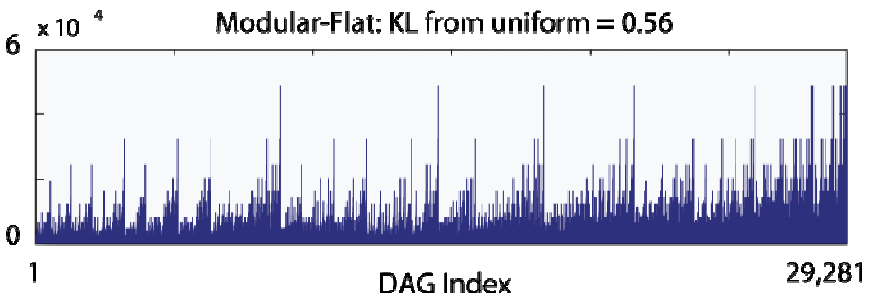
$$p(G) = \sum_{\prec} \frac{1}{Z} \prod_{i=1}^{d} \rho_i(G_i) q_i(\prec_i) I(G, \prec \text{ is valid})$$

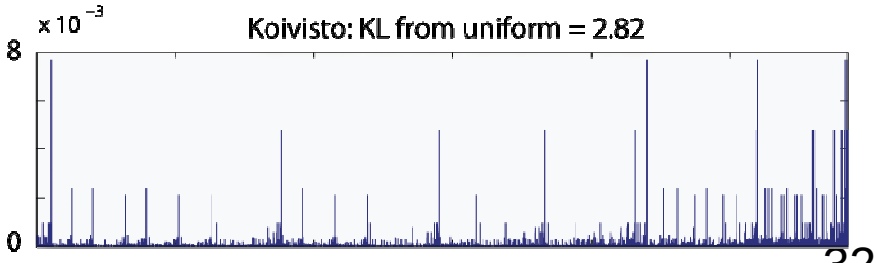
unordered set of parents

ordering of predecessors, $q_i \propto 1$

$$\rho_i(G_i) = 1$$

$$\rho_i(G_i) = \binom{d-1}{|G_i|}^{-1}$$





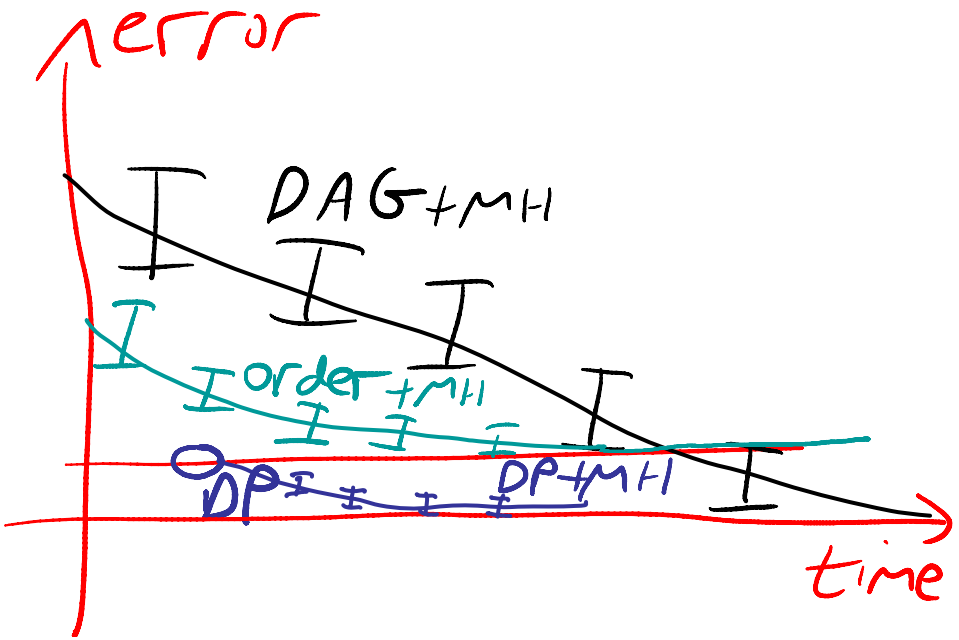
Solutions to p(G) problem

- Importance sampling -- Ellis & Wong ’06
  - Use MH+order as proposal
  - #P-hard to compute exact IS weights

$$
\begin{aligned}
w(G) &= \frac{p^*(G)}{p(G)} = \frac{\frac{1}{Z}\prod_i \rho_i^*(G_i)}{\sum_{\prec} \frac{1}{Z}\prod_{i=1}^{d} \rho_i(G_i) q_i(\prec_i) I(G, \prec \text{ is valid})} \\
&= \frac{1}{\sum_{\prec} I(\text{consistent}(\prec, G))}
\end{aligned}
$$

Solutions to p(G) problem

- Importance sampling -- Ellis & Wong ’06
- Metropolis Hastings -- this paper
  - Use DP marginals as proposal for MH



MH with DP+local proposal

- Compute $p_{ij}=p(G_{ij}=1|D)$ offline using DP
- wp β, we use a standard local move
- wp 1-β, sample a new graph ~ p_{ij}
- If β=0 (global) independence sampler
- If β=1 (local) standard proposal

Why MH?

- DP alone has 3 problems
  1. Modular prior p(G)
  2. Cannot compute prob. of “long range” features (e.g., path from i to j), only edge features.
  3. Very slow to compute predictive density
     $p(x|D) = \sum_G p(x|G)\, p(G|D)$

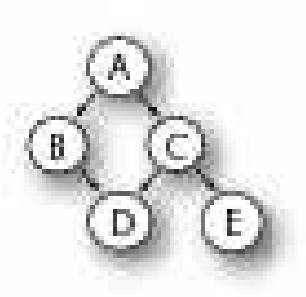
MH allows any $p(G)$

- Propose using $q(G'|G)$
- Accept wp α

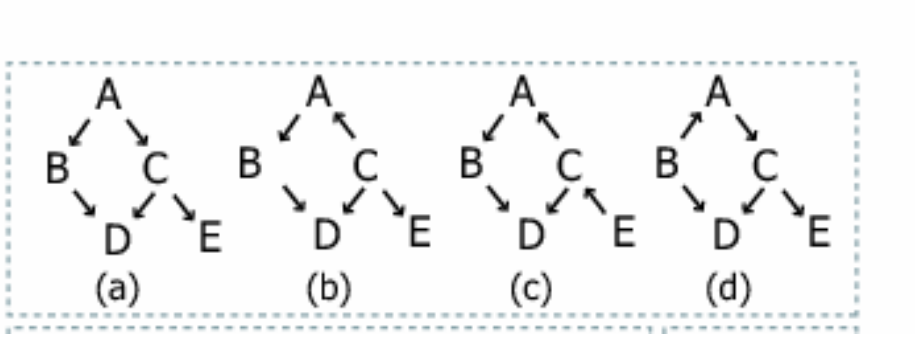
$$\alpha = \min\left(1, \frac{p(D|G')p(G')}{p(D|G)p(G)} \frac{q(G|G')}{q(G'|G)}\right)$$

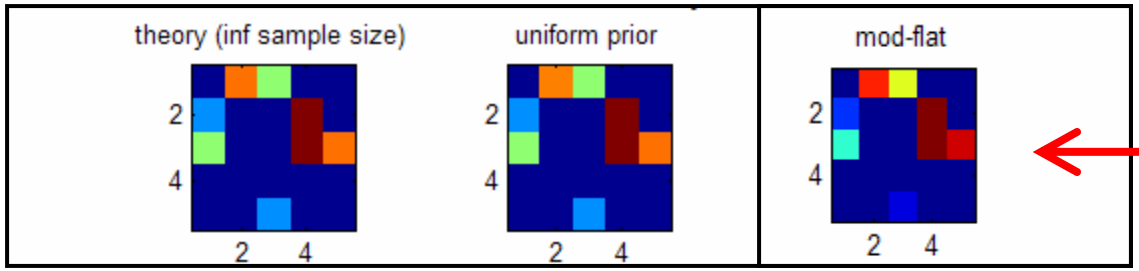
Modular vs uniform p(G)

5 node "cancer" network

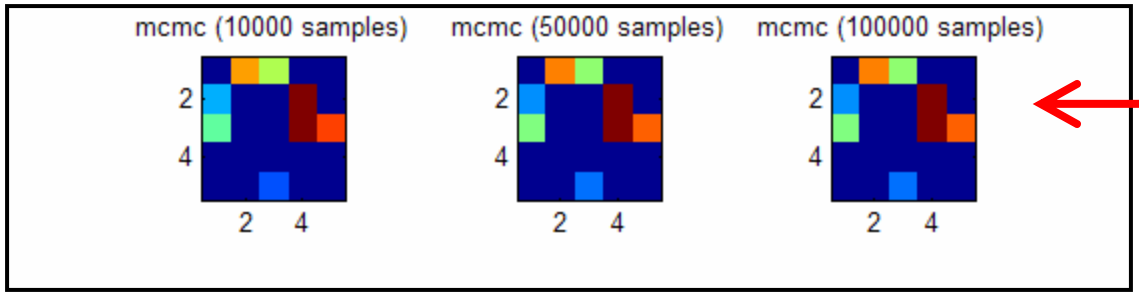


Markov equivalence class





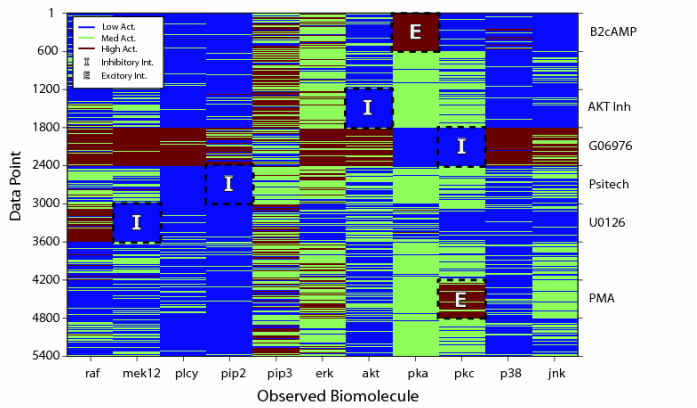
Modular prior biases posterior even as $|D| \rightarrow \infty$



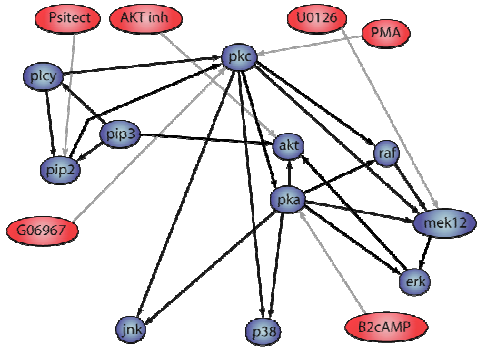
MH fixes bias

T-cell signaling network

Protein phosphorylation (d=11, N=5400)

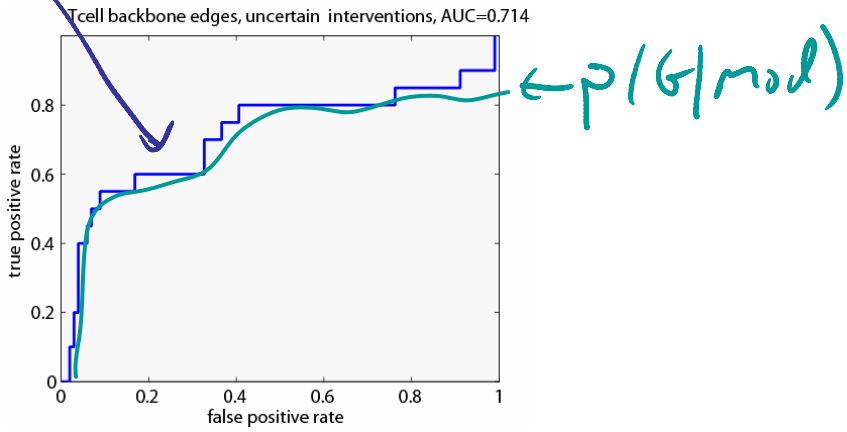


Ground truth DAG



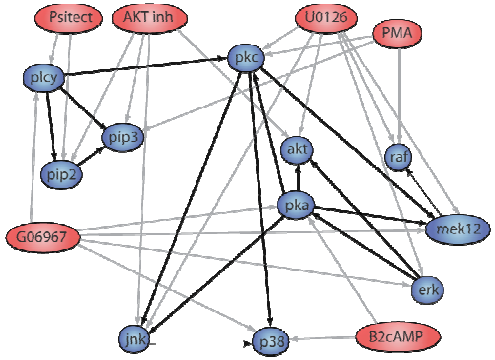
p(G|mij)

ROC



←p(G|mod)

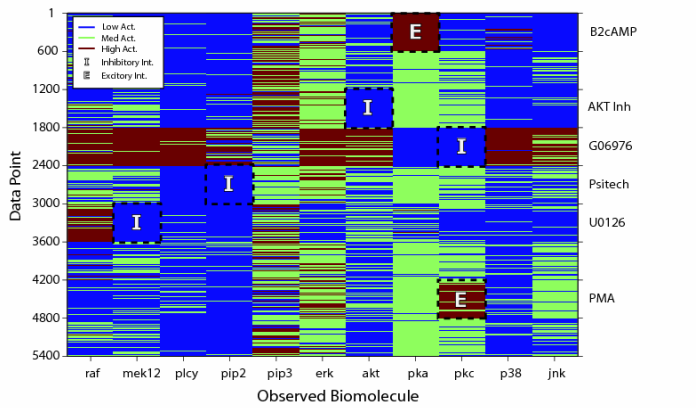
Exact $P(G_{ij}=1|D)$



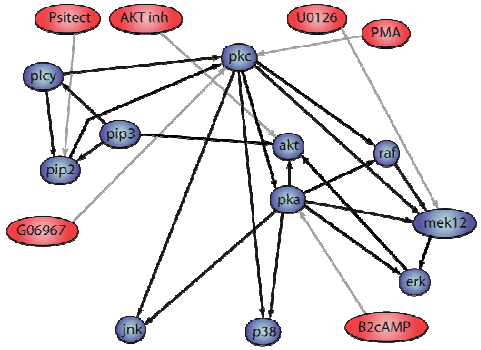
"Causal Protein-Signaling Networks derived from Multiparameter Single-Cell Data", Sachs, Perez, Pe'er, Lauffenberger, Nolan, Science 2005

Informative p(G)

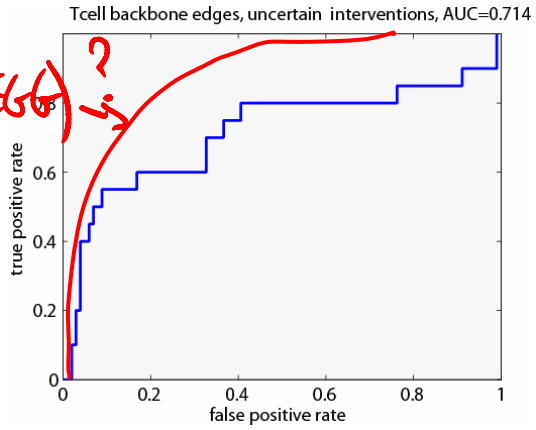
Protein phosphorylation (d=11, N=5400)



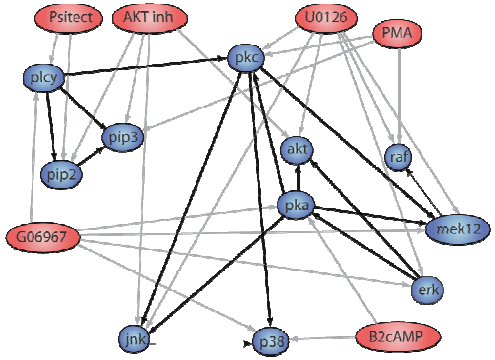
Ground truth DAG



ROC



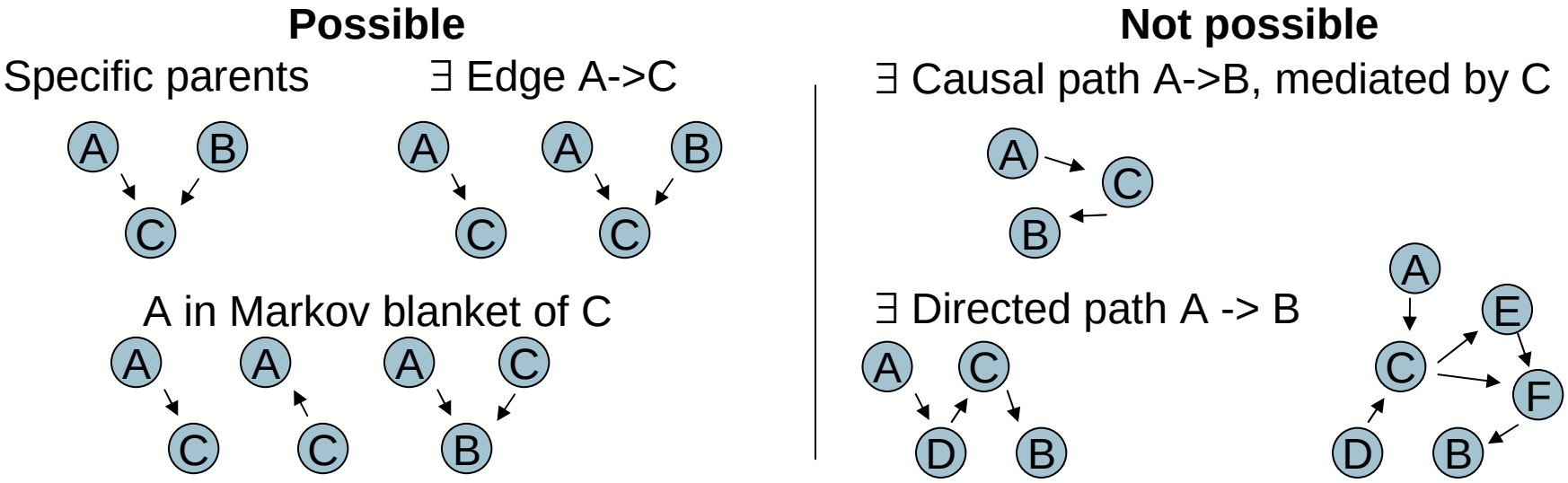
$P(G_{ij}=1|D)$



"Reconstructing Gene Regulatory Networks with Bayesian Networks by Combining Expression Data with Multiple Sources of Prior Knowledge", Werhli & Husmeier, 2007

Sampling G allows any features

- DP can only compute posterior of features that are functions of a local family topology

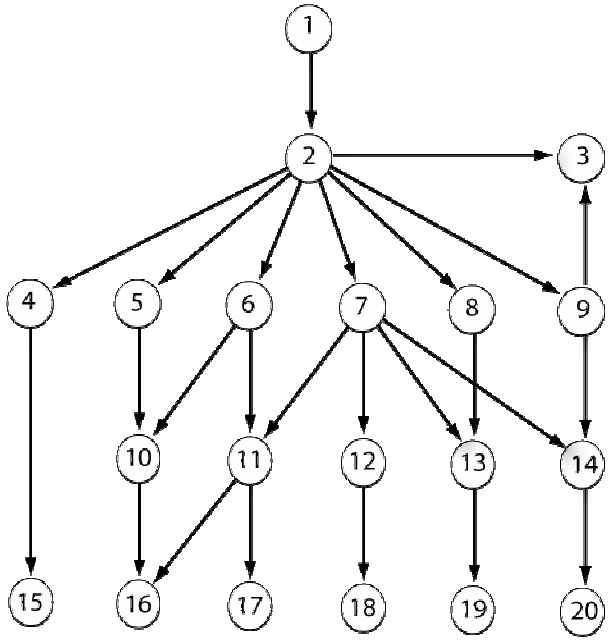


- By sampling DAGs, we can compute E[f(G)] for arbitrary features f

Posterior features

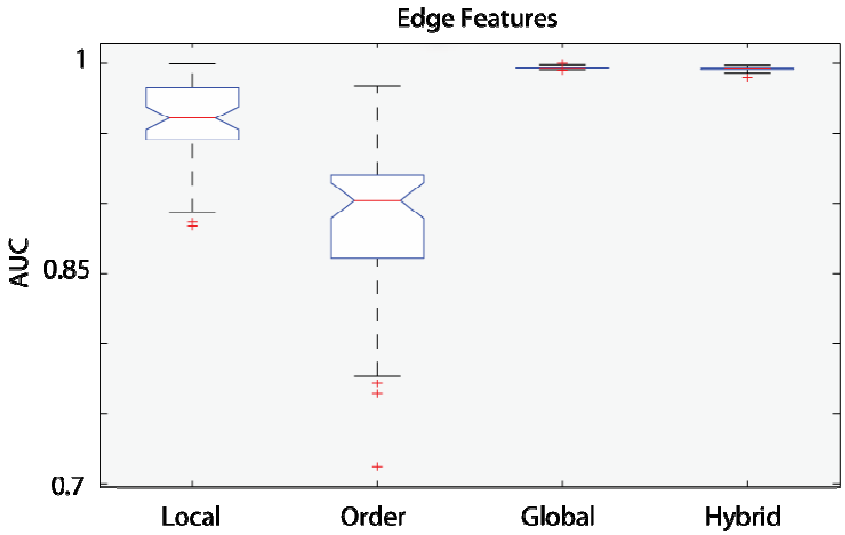
- We sampled N=10k data from d=20 node graph with random CPTs
- Compute p(edge i->j|D) and p(path i->j|D)

"child" network

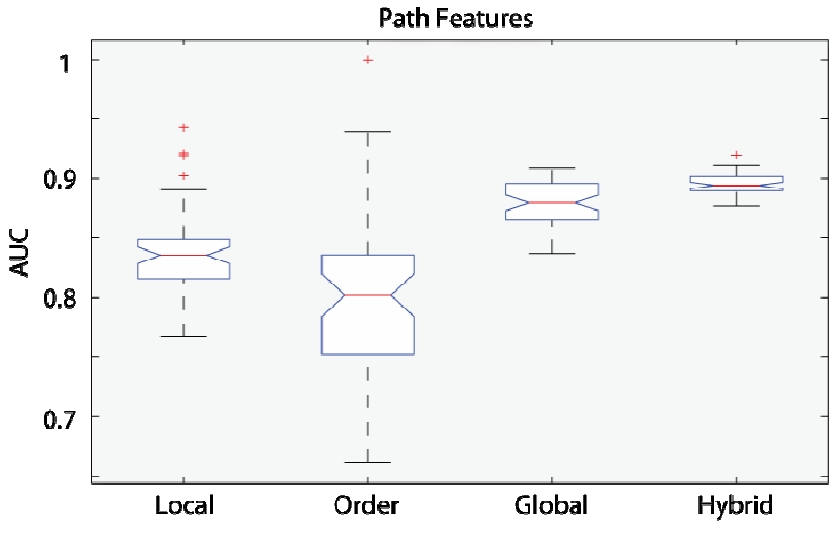


AUC for p(feature=1|D)

Area under the ROC curve after 200 seconds of wall clock time*



β=1 β=0 β=0.1



β=1 β=0 β=0.1

All algorithms were implemented in Matlab/C and run on a standard desktop

Sampling G allows fast prediction

- DP can compute the marginal likelihood of data p(D)
- Hence can compute the predictive likelihood of a test point x:

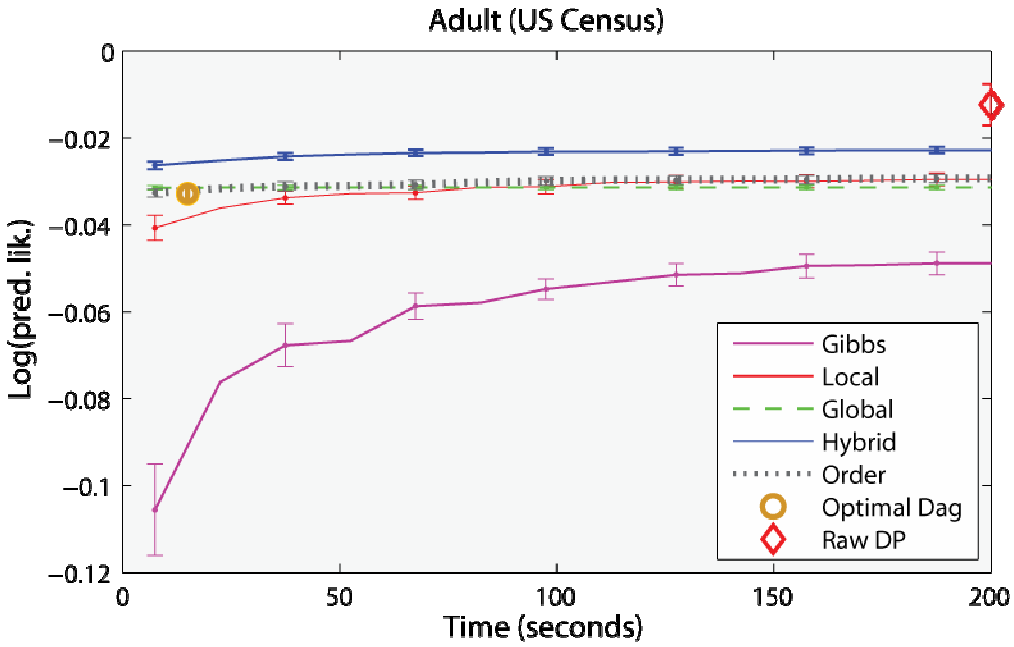
$$p(x|D) = \frac{p(x,D)}{p(D)}$$

- Since DP integrates out G, we have to keep D, and re-run algorithm for each x, which is very slow

- Our approach: keep a sample of $G^s \sim p(G|D)$ and compute posterior mean parameters $\overline{\theta}^s$ for each G^s

$$p(x|D) = \sum_G \int_\theta p(x|G,\theta)p(\theta|G,D)p(G|D) \approx \frac{1}{M}\sum_{s=1}^{M} p(x|G^s,\overline{\theta}^s)$$

US census data (d=15,N=49k)



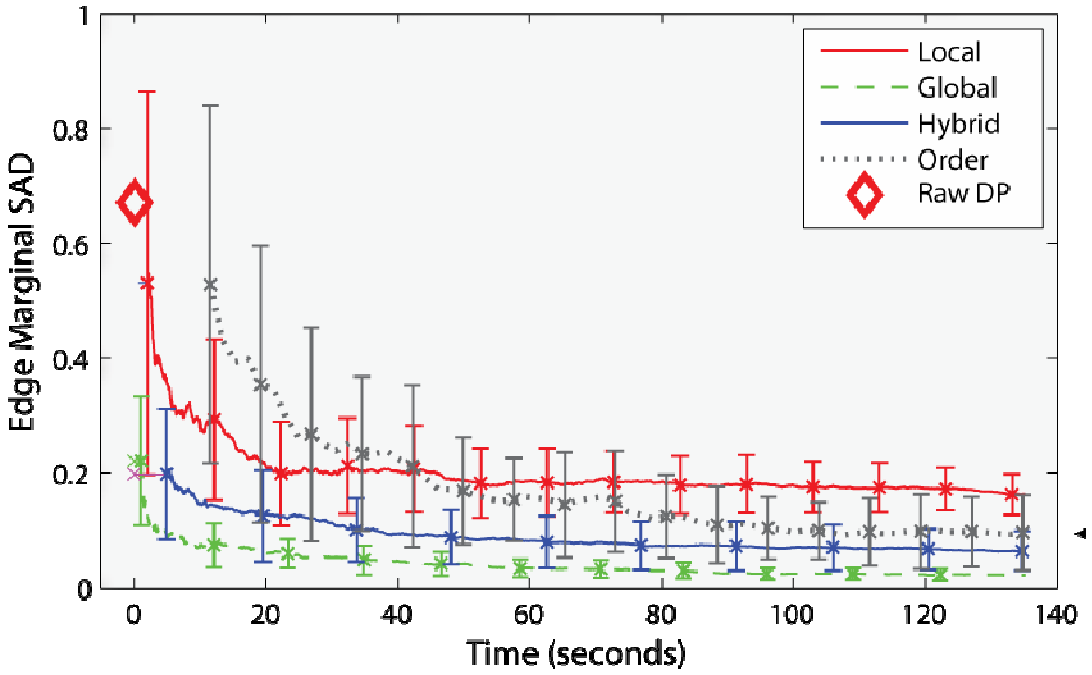
1. Exact BMA (but takes 350h!)
2. MH-DP hybrid β=0.1
3. Plug-in MAP-optimal DAG, MH-DP global β=0, MH-order
4. MH-local β=1
5. Gibbs

Why MH+DP?

- MH + DP mixes faster than MH + other

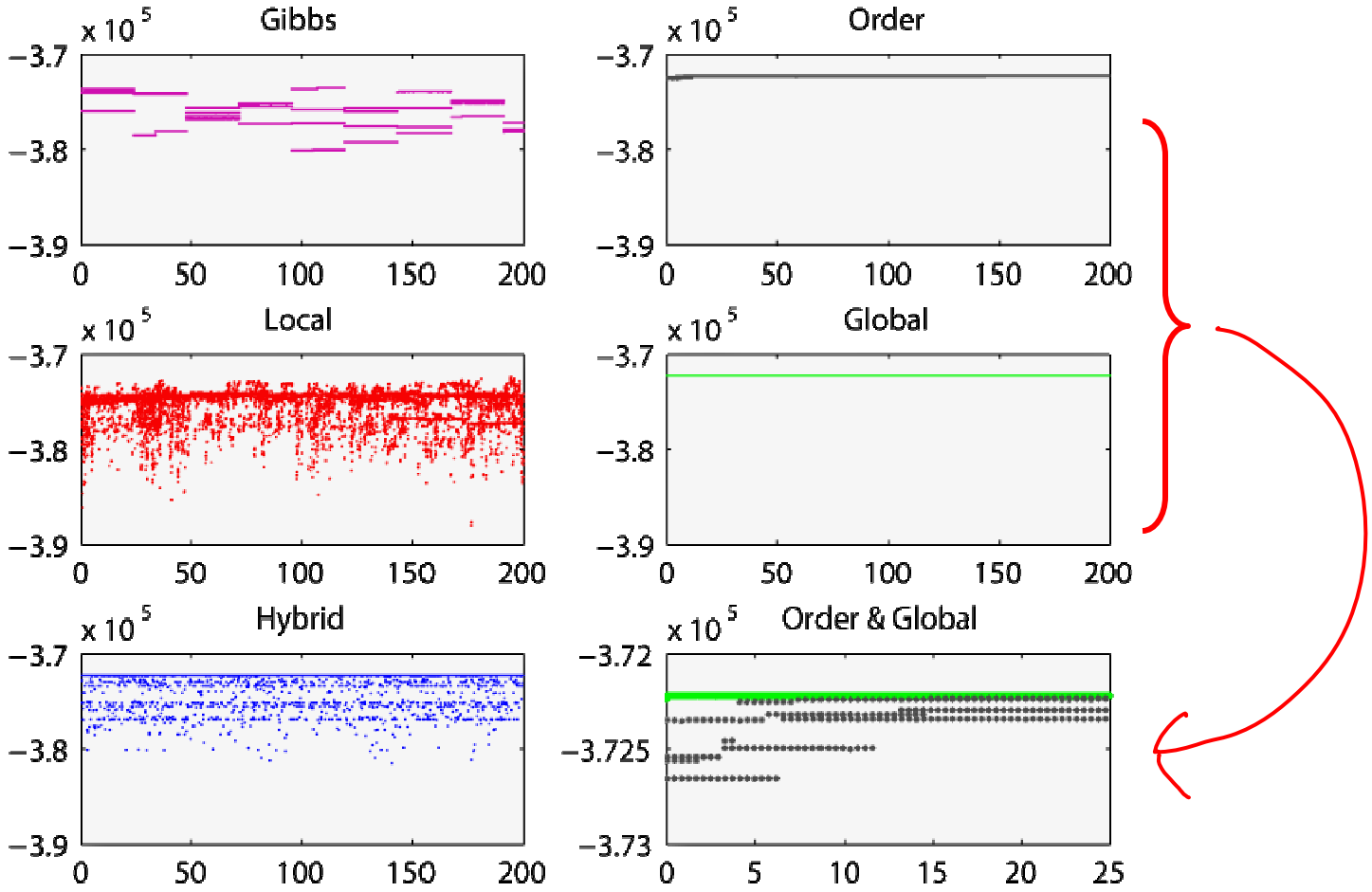
Edge marginal error vs time

$$\sum_{ij} |p(G_{ij} = 1|D) - \hat{p}_t(G_{ij} = 1|D)|$$



d=5 cancer network

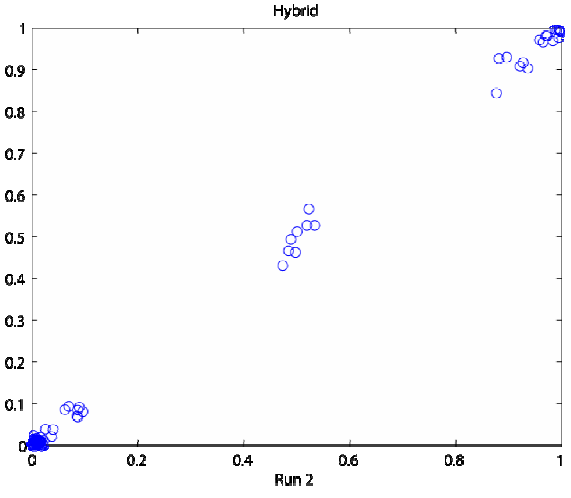
Traceplots of log p(G,D)



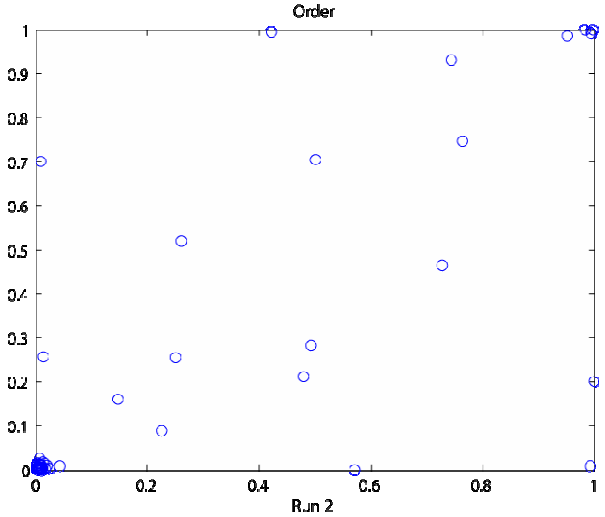
US census (d=14, N=49k)

Repeatability

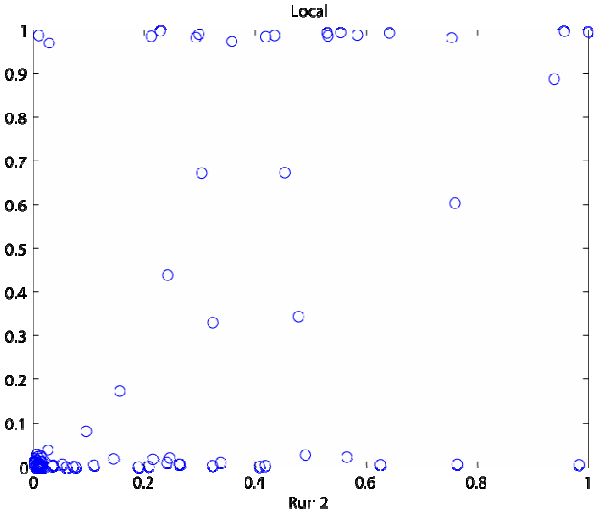
MH+DP (hybrid)



MH-order



MH-local



US census (d=15, N=49k)

We plot edge marginals after two runs from different random starting points

Stochastic search

- MCMC approximates p(M=m|D) by counting how many samples are equal to m.
- Since we can compute p(m,D) exactly, we don't need to visit m more than once. We can approximate

$$p(m|D) \approx \frac{p(m,D)}{\sum_{m' in S} p(m',D)}$$

- It is better to rapidly move through model space, covering as much posterior mass as possible.
- Shotgun stochastic search (SSS), mode oriented stochastic search (MOSS)

Occam's window

- Goal: compute level set of the posterior

$$C(\alpha) = \{m : p(m|D) \geq \alpha p(m^*|D)\}$$

- M^* is unknown, so approximate this by

$$\hat{C}(\alpha) = \{m : p(m|D) \geq \alpha p(\hat{m}^*|D)\}$$

$$\hat{m}^* = \arg\max_{m \in S} p(m|D)$$

- Can find this by beam search, throwing out models that are worse than α time the current best (Raftery, Dobra)