Inference in Graphical Models

王立威

北京大学
信息科学技术学院

Outline

- What is inference in GM
- The hardness of inference in GM
- Exact inference algorithms
- Approximate inference algorithms
- Future research directions

What is inference in GM

- Input: a graph and the local conditional distributions.

- Goal: two types of inference
  - Conditional probability

$$\Pr(X \mid E = e)$$

  - MAP inference

$$\max_x \Pr(X = x \mid E = e)$$

The hardness of inference in GM

- Exact inference in GM is hard
  - Decision version of exact inference is **NP**C.
  - Exact inference is **#P** complete.

- Approximate inference in GM is hard
  - ε-approximate inference is **NP**-hard for every ε

- Thm.1: Decide $\Pr(X = x) > 0$ is **NP** complete.

  Proof: Reduction to 3SAT.

- Thm.2: Compute $\Pr(X = x)$ is **#P** complete.

  Proof: Use above reduction to #3SAT. It is a Levin reduction, certificates are one-to-one.

- Thm.3: For every $\varepsilon > 0$, compute an ε-approximate of $\Pr(X = x)$ is **NP**-hard.

Proof: $\hat{p}$ is an ε-approximation of $\Pr(X = x)$ means that

$$\frac{\Pr(X = x)}{1+\varepsilon} \leq \hat{p} \leq \Pr(X = x)(1+\varepsilon)$$

Clearly, if one has an ε-approximation of $\Pr(X = x)$, one can solve the **NP**C problem $\Pr(X = x) > 0$.

- Thm.4: Exact and approximate MAP inference are hard.

- **Conclusion**: The worst-case complexity of the inferences, both exact and approximate are **NP-**hard.

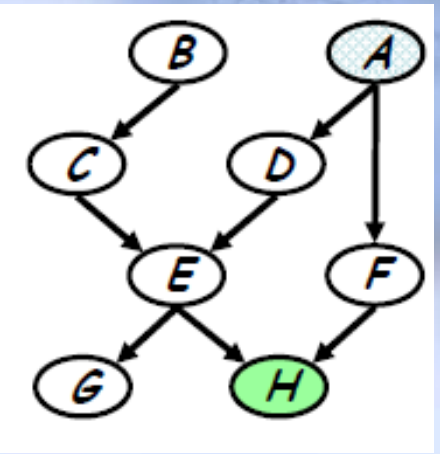
Exact Inference Algorithms

- Exact inference algorithms:
  - The variable elimination algorithm.
  - Message passing---the sum-product algorithm.
  - Belief propagation algorithm.

- Why study exact inference algorithms?
  - Gain intuition and insight for developing approximate algorithms.

- The variable elimination algorithm

query: $\Pr(A = a \mid H = h)$



distribution:

$$P(A)P(B)P(C \mid B)P(D \mid A)P(E \mid C, D)P(F \mid A)P(H \mid E, F)P(G \mid E)$$

solve:

$$P(a, h)$$

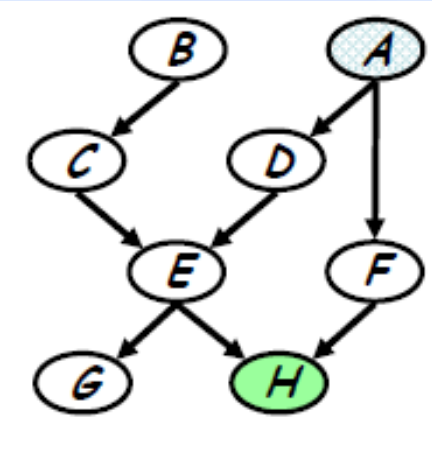
$$= \sum_{b,c,d,e,f,g} P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c, d)P(f \mid a)P(h \mid e, f)P(g \mid e)$$

Variable elimination---dynamic programming

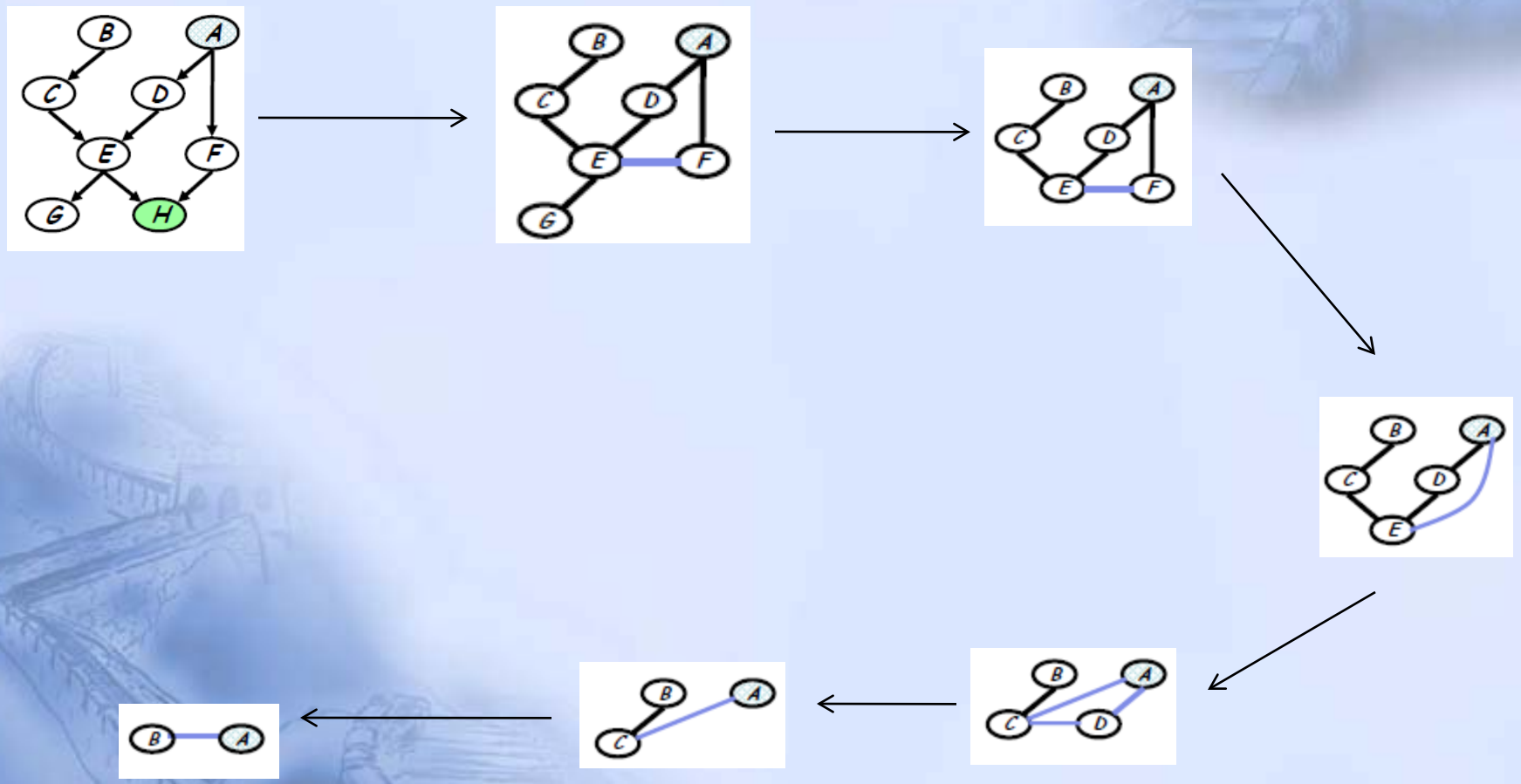
$$\sum_{b,c,d,e,f,g} P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(h|e,f)P(g|e)$$

$$= P(a)\sum_{b} P(b)\sum_{c} P(c|b)\sum_{d} P(d|a)\sum_{e} P(e|c,d)\sum_{f} P(f|a)P(h|e,f)\sum_{g} P(g|e)$$

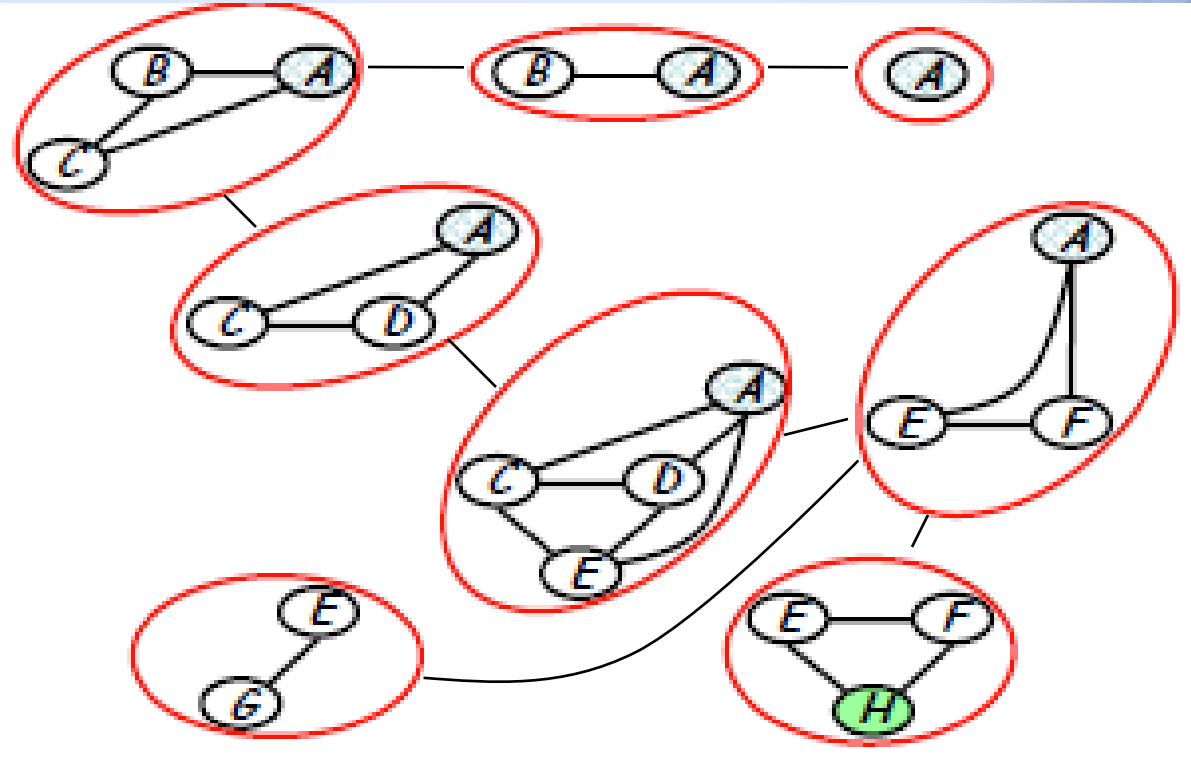
$$P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)\phi_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)\phi_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)\phi_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)\phi_e(a,c,d)$$
$$\Rightarrow P(a)P(b)P(c|d)\phi_d(a,c)$$
$$\Rightarrow P(a)P(b)\phi_c(a,b)$$
$$\Rightarrow P(a)\phi_b(a)$$



Variable elimination on the graph

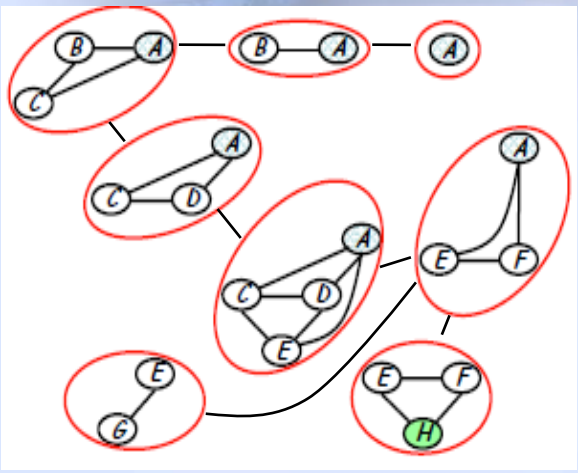


- Message Passing: the SP algorithm
  - Variable elimination induced clique tree



- Variable elimination as message passing on clique tree:

$$
\begin{aligned}
& P(a)P(b)P(c\mid d)P(d\mid a)P(e\mid c,d)P(f\mid a)P(g\mid e)P(h\mid e,f) \\
\Rightarrow\ & P(a)P(b)P(c\mid d)P(d\mid a)P(e\mid c,d)P(f\mid a)P(g\mid e)\phi_h(e,f) \\
\Rightarrow\ & P(a)P(b)P(c\mid d)P(d\mid a)P(e\mid c,d)P(f\mid a)\phi_h(e,f) \\
\Rightarrow\ & P(a)P(b)P(c\mid d)P(d\mid a)P(e\mid c,d)\phi_f(a,e) \\
\Rightarrow\ & P(a)P(b)P(c\mid d)P(d\mid a)\phi_e(a,c,d) \\
\Rightarrow\ & P(a)P(b)P(c\mid d)\phi_d(a,c) \\
\Rightarrow\ & P(a)P(b)\phi_c(a,b) \\
\Rightarrow\ & P(a)\phi_b(a)
\end{aligned}
$$



$$\phi_e(a,c,d)=\sum_e P(e\mid c,d)\phi_f(a,e)=\sum_{\{a,c,d,e\}\setminus\{a,c,d\}} P(e\mid c,d)\phi_f(a,e)$$

- General message passing from clique C_i to C_j

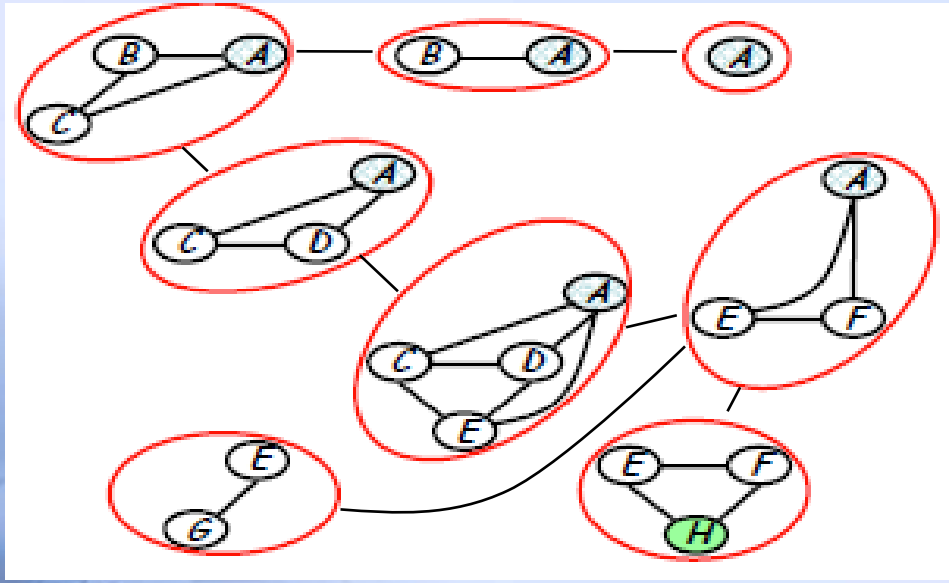
$$\delta_{i\to j} = \sum_{C_i \setminus S_{ij}} \psi_i \prod_{k \in Nb(i)\setminus\{j\}} \delta_{k\to i}$$

$$S_{ij} = C_i \cap C_j$$

ψ_i : original terms (potential) of C_i

- The Sum-Product (SP) algorithm:
  - Construct a clique tree that satisfies the running intersection property.
  - Choose clique where the query variable lies in as the root.
  - Message passing from the leaves.
  - After all messages arrive at the root, sum over all other variables in the root clique other than the query variable.

- What if we want to compute k queries?
  - Message passing k times?
  - No! Message can be reused.
  - For each edge in the clique tree, twice is enough, one for each direction.



- After message passing on all edges in two directions, each clique has a belief (joint probability).

- Belief: $\beta_i = \psi_i \prod_{k \in Nb(i)} \delta_{k \to i}$

- The system satisfies the calibration property:

$$\sum_{C_i \setminus S_{ij}} \beta_i = \sum_{C_j \setminus S_{ij}} \beta_j$$

Agree on the marginal probability of S_{ij}

- Belief propagation:
  - Message passing:

$$\delta_{i\to j} = \sum_{C_i \setminus S_{ij}} \psi_i \prod_{k \in Nb(i)\setminus\{j\}} \delta_{k\to i}$$

  - Belief: $\beta_i = \psi_i \prod_{k \in Nb(i)} \delta_{k\to i}$
  - So $\delta_{i\to j} = \dfrac{\sum_{C_i \setminus S_{ij}} \beta_i}{\delta_{j\to i}}$

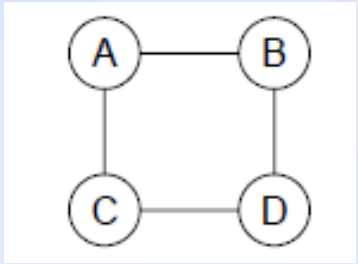
    Propagating beliefs from C_i to C_j

- Belief propagating algorithm:
  - Construct clique tree
  - Initialize: $\beta_i = \psi_i \qquad \mu_{i \to j} = 1$
  - Update: $t_{i \to j} \leftarrow \sum_{C_i \setminus S_{ij}} \beta_i$

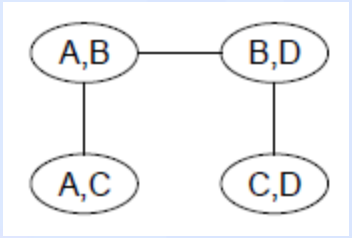
$$\beta_j \leftarrow \frac{\beta_j t_{i \to j}}{\mu_{ij}}$$

$$\mu_{ij} \leftarrow t_{i \to j}$$

- How to construct clique tree?
  - Graph:

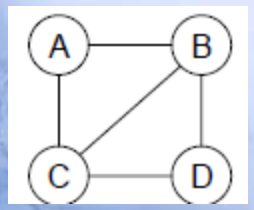


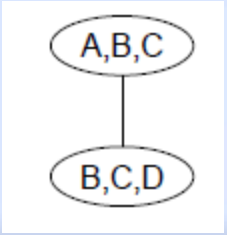
  - Clique tree?



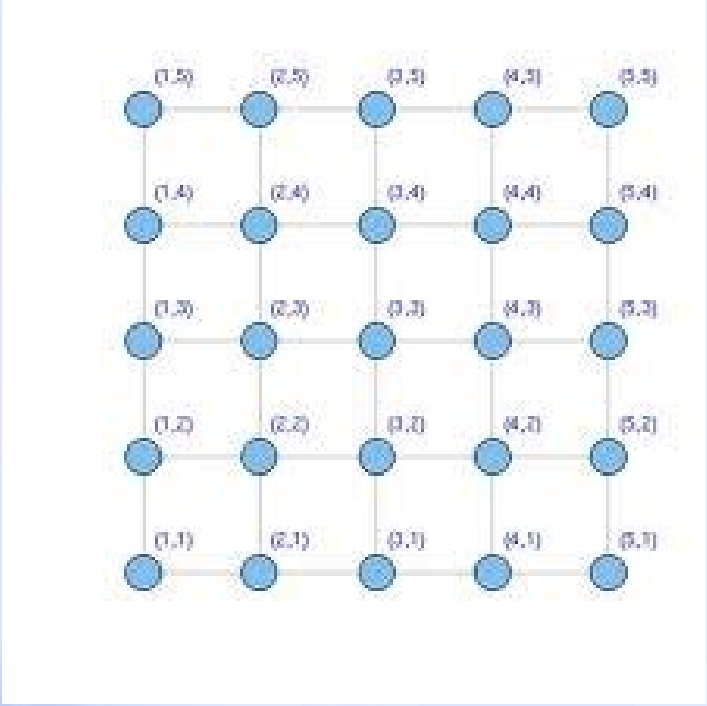
No!

  - The graph cannot have a 4-loop (or larger)!
  - Solution: triangulation.





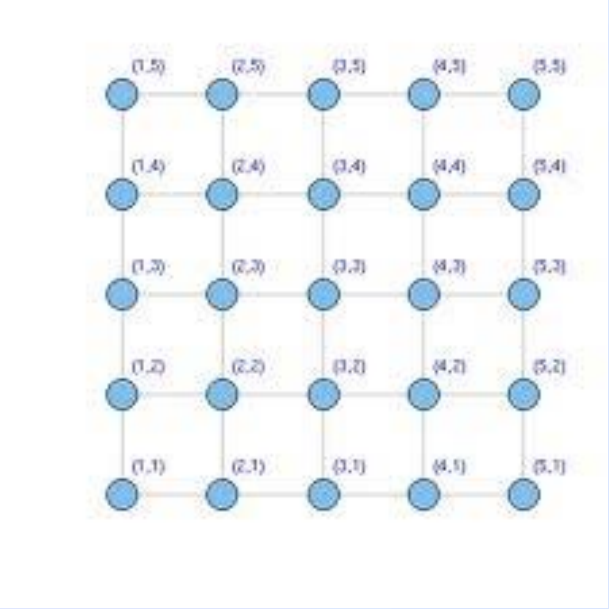
- What if the graph is an Ising model, Triangulation?

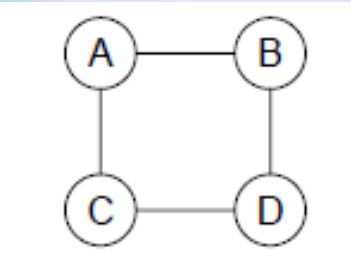


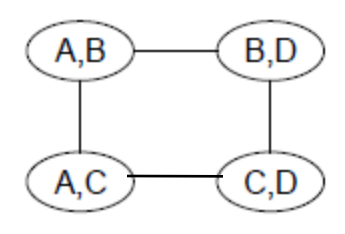
Approximate Inference

- Approximate inference:
  - Variational Methods.
    - (Loopy) belief propagation.
    - Mean field method.
  - Sampling based methods.
    - MCMC
    - …

- (Loopy) belief propagation:







Clique graph

- Loopy belief propagation: just let the belief propagates.
- On clique trees, belief propagation converges after propagating on all edges (two directions).
- For general clique graphs, it is not guaranteed to converge. Even if it converges, it can converge to a wrong answer.

- Variational explanation of the (loopy) belief propagation:
  - Exact inference

  ⇔ Minimizing the relative entropy (free energy) of the target distribution and the CPDs on clique trees under the constraints of calibration (local agreement).

  - Approximate inference

  ⇔ Minimizing the relative entropy of the target distribution and the CPDs on general clique graphs under the calibration constraint.

- One can derive a fixed point equation from the Lagrangian of the constrain optimization problem:

$$\delta_{i\to j} = \sum_{C_i \setminus S_{ij}} \psi_i \prod_{k \in Nb(i)\setminus\{j\}} \delta_{k\to i}$$

- Mean field method:
  - Using first order approximation of the target distribution:

$$P(X_1, \ldots, X_n) = \prod_i P(X_i)$$

- Sampling-based approximation algorithms:
  - Using sample based frequency to approximate the probability.
  - MCMC
    - Metropolis-Hasting
    - Gibbs sampling

Future Research Directions

- Inference is **NP**-hard, what shall we do?
  - Develop practical algorithms. **#P** complete is a worst case result.
  - To solve the inference problem, we are in a situation very similar to solving TSP (**NP**-hard):
    - TSP (decision) is **NP**-complete.
    - Euclidean TSP is NP-hard.
    - Approximate TSP is NP-hard.
    - TSP: Euclidean + approximation → **polynomial** approximation scheme

- Can we find a reasonable class of graphical models such that (approximate) inference has **polynomial** time algorithm?

Thanks