CMPUT463/563
Probabilistic Graphical Models

Exact Inference: Variable Elimination

Lili Mou

Dept. Computing Science, University of Alberta

lmou@ualberta.ca

Outline

- Types of probabilistic queries
- Naïve calculation is expensive
- Dynamic programming on a chain
- Variable elimination in general
  - Evidence potentials falsify incompatible assignments
  - Eliminate a variable by fully connecting its neighbors
  - Sum-product and max-product are semirings
  - Directed and undirected graphs work in a similar way
- Efficiency of VE depends on the induced-width
  - Finding the best order is NP-hard
  - Intuition helps

Inference Questions

We group the variables of a sample into three sets $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$

- $\boldsymbol{X}$: observed; $\boldsymbol{Y}$: variables in question
- $\boldsymbol{Z}$: to be marginalized out

Query types

- Probabilistic queries: $P(\boldsymbol{Y} \mid \boldsymbol{X})$
  - Special case $\boldsymbol{Y} = \emptyset$: probability of evidence $P(\boldsymbol{X})$
- Max a posteriori (MAP) inference: $\mathrm{argmax}\, P(\boldsymbol{Y} \mid \boldsymbol{X})$
    - Most likely values for **some** variables
  - Special case
    - Most probable explanation (MPE): when $\boldsymbol{Z} = \emptyset$
    - Most likely values for **all other** variables

Inference Questions - Applications

Query types

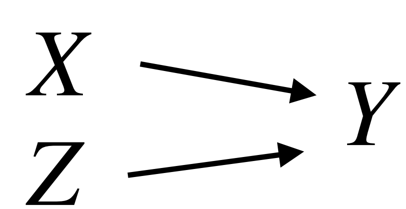
- Probabilistic queries: $P(\boldsymbol{Y} \mid \boldsymbol{X})$
  - Special case $\boldsymbol{Y} = \emptyset$: probability of evidence $P(\boldsymbol{X})$

Application: Outlier detection; Used in parameter learning

- Max a posteriori (MAP) inference: $\operatorname{argmax} P(\boldsymbol{Y} \mid \boldsymbol{X})$
  - Most likely values for **some** variables
  - Special case
    - Most probable explanation (MPE): $\operatorname{argmax}_{Y,Z} P(\boldsymbol{Y}, Z \mid \boldsymbol{X})$
    - Most likely values for **all other** variables

MAP and MPE may be different for Y

Applications: Image segmentation, POS tagging
Text generation from continuous latent space



Joint probability tells everything

- Probabilistic queries: $P(\boldsymbol{Y} \mid \boldsymbol{X})$

$$P(\boldsymbol{Y} \mid \boldsymbol{X}) \propto_Y \sum_{z} P(\boldsymbol{Y}, \boldsymbol{z} \mid \boldsymbol{X})$$

- Max a posteriori (MAP) inference: $\operatorname{argmax} P(\boldsymbol{Y} \mid \boldsymbol{X})$

$$\underset{Y}{\operatorname{argmax}} \sum_{z} P(\boldsymbol{Y}, \boldsymbol{z} \mid \boldsymbol{X})$$

  - Most probable explanation (MPE): when $\boldsymbol{Z} = \emptyset$

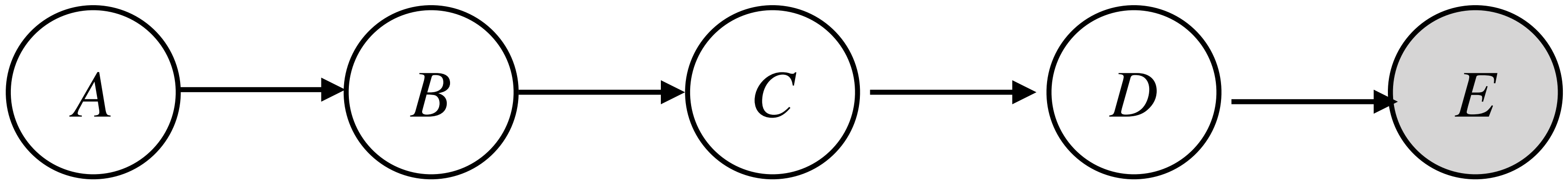
$$\underset{Y}{\operatorname{argmax}} P(\boldsymbol{Y} \mid \boldsymbol{X})$$

Complexity of naïve computation?

Categorization of Inference Algorithms

- Exact inference
  - Variable elimination, message passing, junction tree
  - With dynamic programming, efficiency is better than enumeration, but still NP-hard (NP-complete or harder) for a general graph
  - Tree structures: linear
- Approximate inference
  - Asymptotically correct: Monte Carlo approaches
  - Deterministically wrong: variational inference

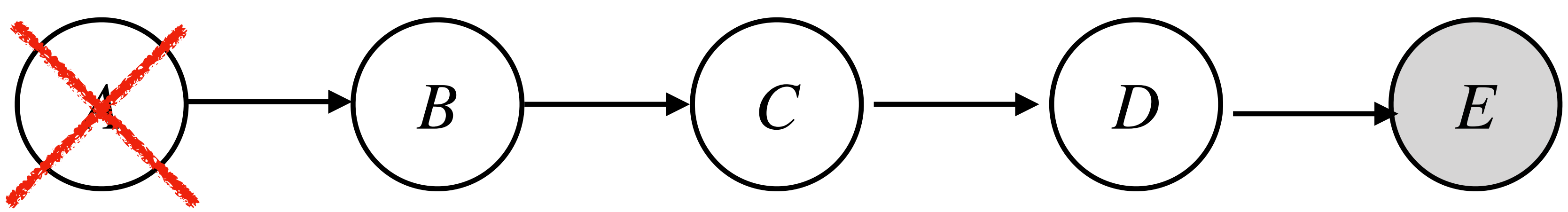
Why dynamic programming is possible?



- Consider the query $P(e)$

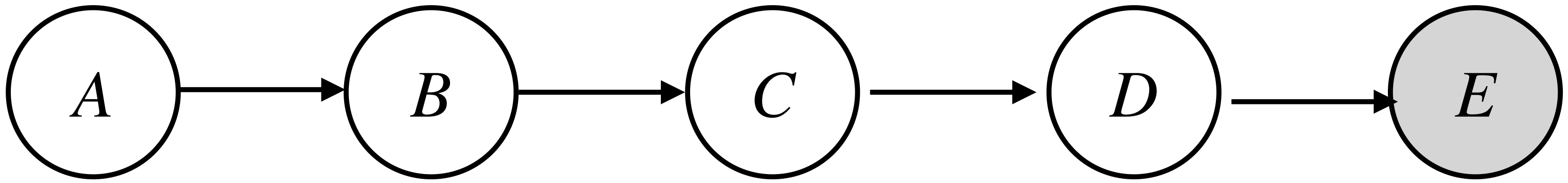
$$P(e) = \sum_d \sum_c \sum_b \underline{\sum_a P(a)P(b\,|\,a)}P(c\,|\,b)P(d\,|\,c)P(e\,|\,d)$$

$$= \sum_d \sum_c \sum_b \left[\underbrace{\left(\sum_a P(a)P(b\,|\,a) \right)}_{P(b)} P(c\,|\,b)P(d\,|\,c)P(e\,|\,d) \right]$$



Variable A is eliminated

Why dynamic programming is possible?

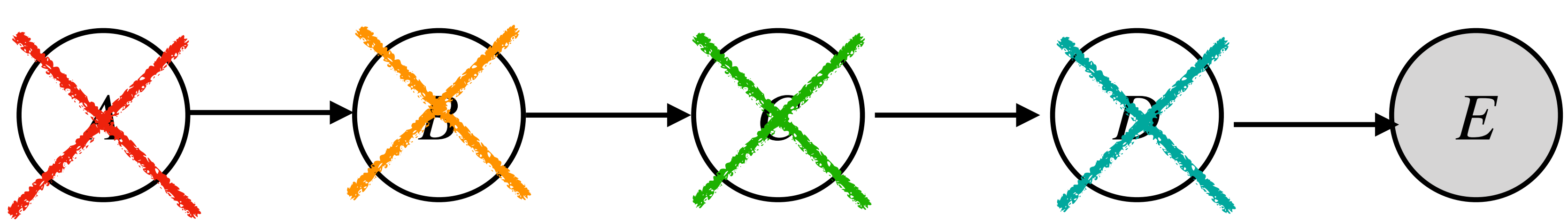


- Consider the query $P(e)$

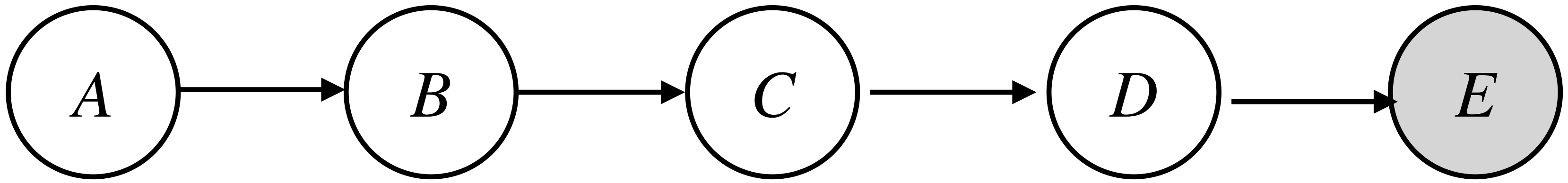
$$P(e) = \sum_d \sum_c \underbrace{\sum_b P(b)P(c\,|\,b)}_{P(c)} P(d\,|\,c)P(e\,|\,d)$$

$$= \sum_d \underbrace{\sum_c P(c)P(d\,|\,c)}_{P(d)} P(e\,|\,d)$$

$$= \underline{\sum_d P(d)P(e\,|\,d)}$$



Such DP also works for argmax



- Consider the query $\max_{a,b,c,d} P(e)$

Subscripts of $m(\cdot)$ indicate they're different functions

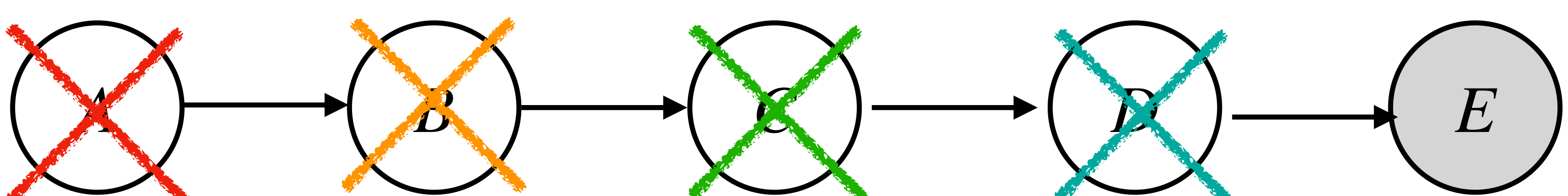
$$\max_{a,b,c,d} P(e) = \max_d \max_c \max_b \underbrace{\max_a P(a)P(b\,|\,a)}_{m_B(b)} P(b)P(c\,|\,b)P(d\,|\,c)P(e\,|\,d)$$

$$= \max_d \max_c \underbrace{\max_b m(b)P(c\,|\,b)}_{m_C(c)} P(d\,|\,c)P(e\,|\,d)$$

$$= \max_d \underbrace{\max_c m(c)P(d\,|\,c)}_{m_D(d)} P(e\,|\,d)$$

$$= \underbrace{\max_d m(d)P(e\,|\,d)}$$

This is dynamic programming (DP), not a greedy algorithm. For example, when computing $\max_c P(c)P(d\,|\,c)$, you get a function of d without a specific c. The choice of c depends on d, so back-pointers are needed.



Semiring

Compared with a ring, semiring does not require inverse of addition

- Algebraic structure: ($\oplus$, $\odot$) on a set R is a semiring if
  - $\oplus$ is associative and commutative with identity 0
    - $0 \oplus a = a \oplus 0 = a$
    - $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
    - $a \oplus b = b \oplus a$
  - $\odot$ is associative with zero 0 and identity 1
    - $0 \odot a = a \odot 0 = 0, 1 \odot a = a \odot 1 = a$
    - $(a \odot b) \odot c = a \odot (b \odot c)$
  - $\odot$ distributive wrt $\oplus$

$$a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c)$$

$$(b \oplus c) \odot a = (b \odot a) \oplus (c \odot a)$$

Semiring

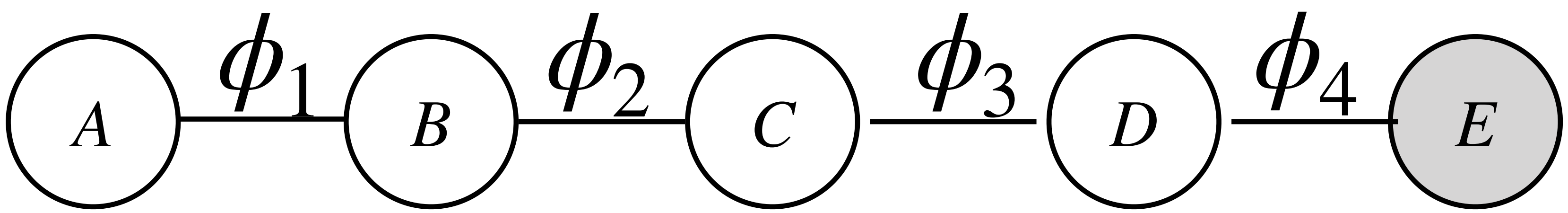
- Both $(+, \cdot)$ and $(\max, \cdot)$ are semirings
- The above DP algorithms only involve the operations defined in a semiring
- Thus, the two algorithms are almost identical.

$$P(e) = \sum_d \sum_c \sum_b \sum_a P(a)P(b \mid a)P(c \mid b)P(d \mid c)P(e \mid d)$$

$$\max_{a,b,c,d} P(e) = \max_d \max_c \max_b \max_a P(a)P(b \mid a)P(b)P(c \mid b)P(d \mid c)P(e \mid d)$$

To obtain argmax, we need back-pointers

Such DP also works for undirected graphs



$$P(e) = \sum_d \sum_c \sum_b \sum_a \frac{1}{Z} \phi_1(a,b)\phi_2(b,c)\phi_3(c,d)\phi_4(d,e)$$

$$= \frac{1}{Z} \underbrace{\sum_d \underbrace{\sum_c \underbrace{\sum_b \underbrace{\sum_a \phi_1(a,b)}_{m_B(b)} \phi_2(b,c)}_{m_C(c)} \phi_3(c,d)}_{m_D(d)} \phi_4(d,e)}_{m_D(d)}$$

How about Z?

- Sum-product: marginalize out E
- Max-product: can be ignored because Z is a constant

Beyond a chain structure

From now on, we consider undirected graphs

For BNs, a conditional probability $P(X \mid \mathrm{Par}(X))$ can immediately be interpreted as $\phi(X, \mathrm{Par}(X))$

Suppose our query is $P(Y \mid x)$ and we need to eliminate $\boldsymbol{Z}$

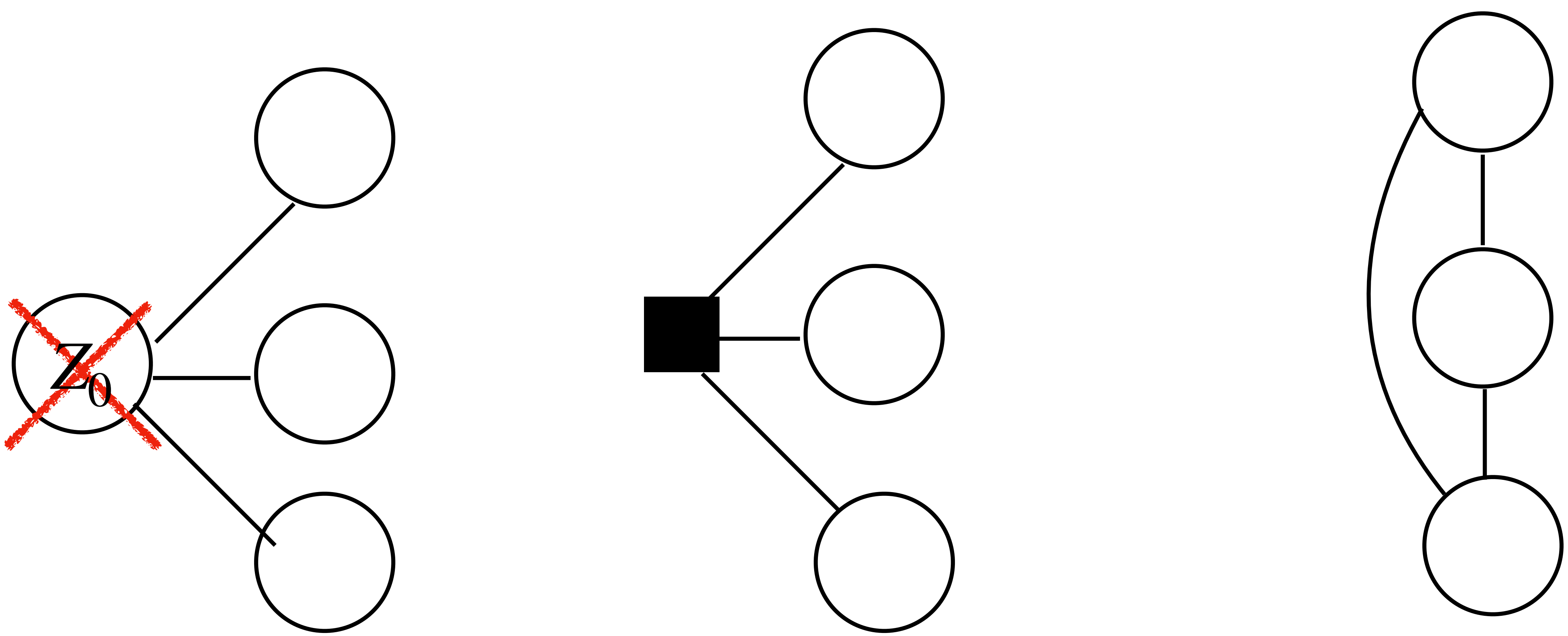
$$P(Y|x) \propto P(Y, x) = \sum_{\boldsymbol{z}} P(Y, \boldsymbol{z}, x)$$

$$\propto \sum_{\boldsymbol{z} \setminus \{z_0\}} \underbrace{\sum_{z_0} \prod_{\phi : z_0 \in \mathrm{Scope}(\phi)} \phi}_{\varphi \text{ with scope } \bigcup_{\phi : z_0 \in \mathrm{Scope}(\phi)} \mathrm{scope}(\phi) \setminus \{z_0\}} \prod_{\psi : z_0 \notin \mathrm{Scope}(\psi)} \psi$$

Eliminate a Variable

To eliminate a variable Z_0:

- Consider all factors involving Z_0
- Eliminate Z_0 by summing all the values of Z_0
- Obtain a factor φ with a scope of Z_0' neighbors (fully connects the neighbor)
- Put φ back to the graphical model



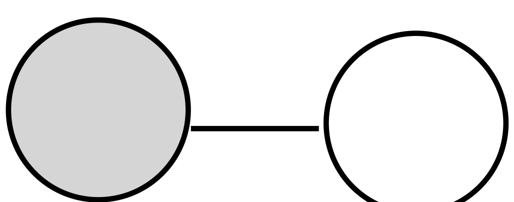
Handling Evidence

$P(Y|x) \propto P(Y,x)$, i.e., we only considers assignments compatible to evidence.

This can be handled by introducing an evidence potential

$$\delta_i = \begin{cases} 1, \text{ if } X_i = x_i \\ 0, \text{ if } X_i = x_i \end{cases}$$

Multiply every potential containing X_i with δ_i



$X_i = 1$

X_i	$\cdots$	ϕ
0	0	2
0	1	0.1
1	0	0.2
1	1	10

X_i	$\cdots$	$\phi\delta_i$
0	0	~~2~~ 0
0	1	~~0.1~~ 0
1	0	0.2
1	1	10

Variable Elimination

Input: Variables $\boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{X} = \boldsymbol{x}$, set of factors Φ

Elimination order (assuming $Z_1, Z_2, \cdots, Z_k$ wlog)

1. $\Phi \leftarrow \Phi$ with evidence potentials

2. For $i = 1, \cdots, k$

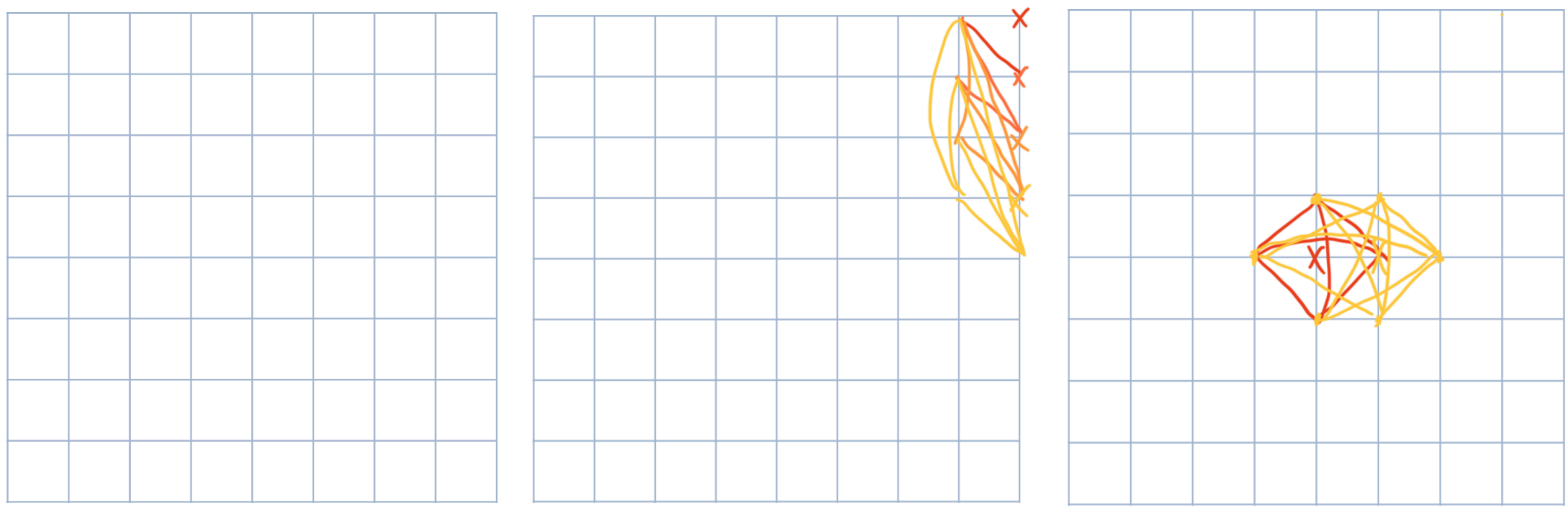
2.1 $\Phi' \leftarrow \{\phi \in \Phi' : Z_i \in \text{scope}(\phi)\}$

2.2. $\varphi \leftarrow \sum_{z_i} \prod_{\phi \in \Phi'} \phi$

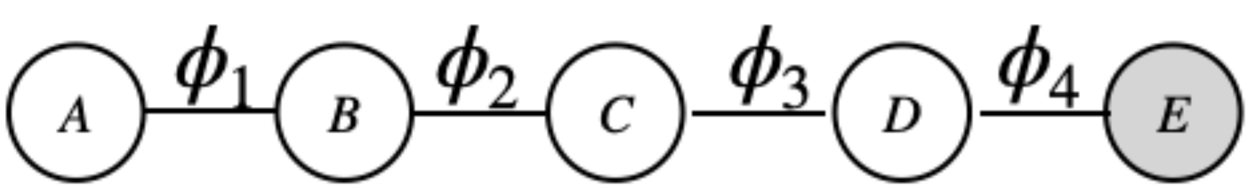
2.3 $\Phi = \Phi \setminus \Phi' \cup \{\varphi\}$

Return $\prod_{\phi \in \Phi} \phi$ as a factor on $\boldsymbol{Y}$ given $\boldsymbol{X}$

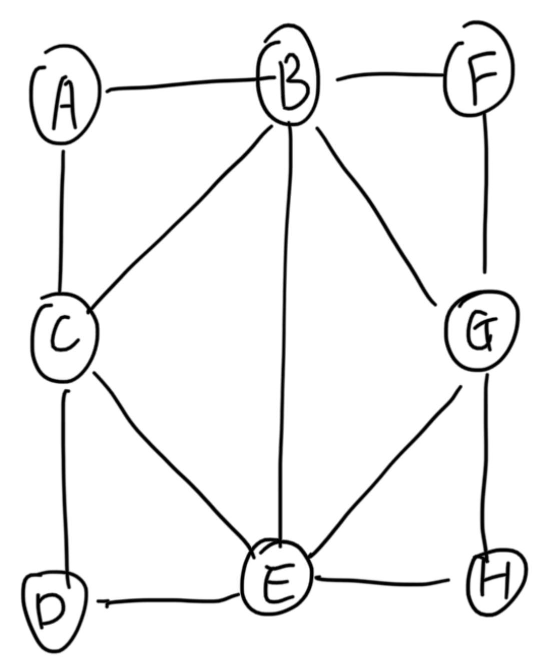
Some graphs don't have efficient elimination anyway



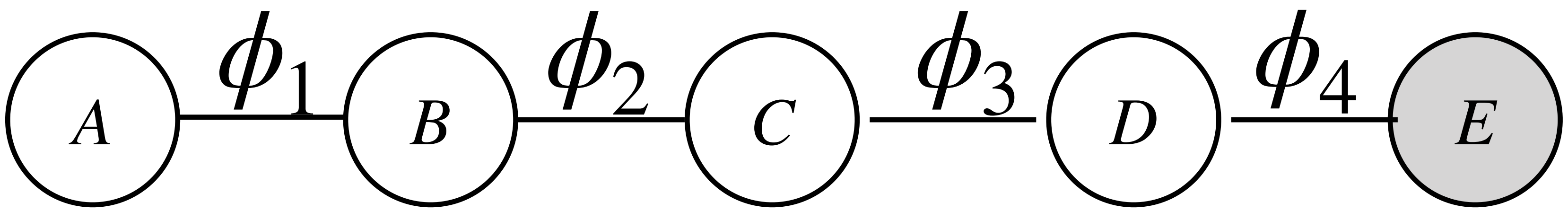
Certain graphs can be eliminated efficiently



Chordal



Example



$$P(e) = \sum_d \sum_c \sum_b \sum_a \frac{1}{Z} \phi_1(a,b)\phi_2(b,c)\phi_3(c,d)\phi_4(d,e)$$

$$= \frac{1}{Z} \sum_d \sum_c \sum_b \sum_a \phi_1(a,b)\phi_2(b,c)\phi_3(c,d)\phi_4(d,e)$$

$m_B(b)$

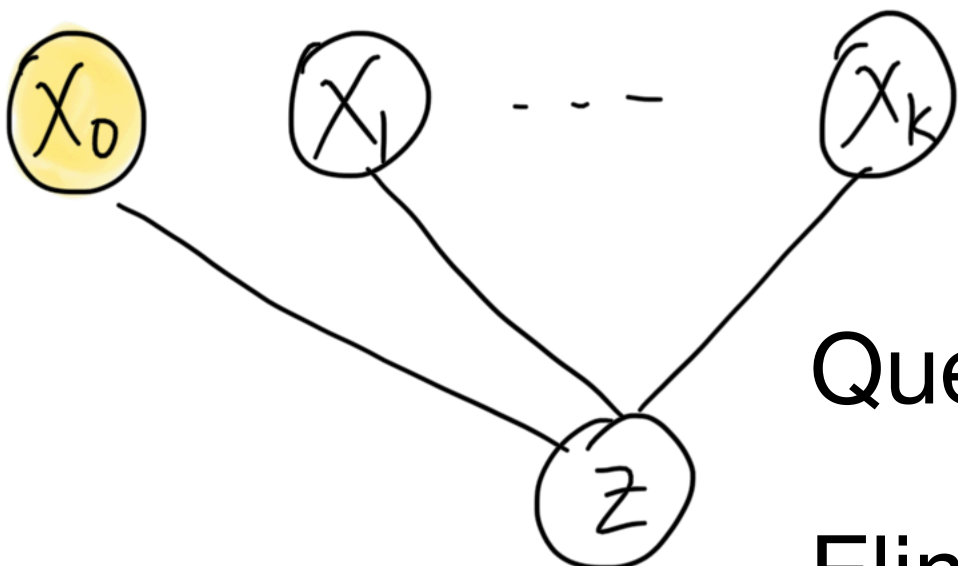
$m_C(c)$

$m_D(d)$

$m_D(d)$

In fact, you may eliminate either direction in this example.
However, the eliminator order matters a lot

Example



Query: $P(X_k \mid X_0)$

Eliminate $X_0, \cdots, X_{k-1}, Z$: linear complexity

Eliminate Z first: exponential wrt k

The *induced-width* is a size of the largest scope during variable elimination given an induction order. The *tree-width* is the minimum induced width.

Unfortunately, TreeWidth$\leq$ M is NP-complete. Finding the smallest tree-width is NP-hard. Intuition helps.

Summary

- Types of probabilistic queries
- Naïve calculation is expensive
- Dynamic programming on a chain
- Variable elimination in general
  - Evidence potentials falsify incompatible assignments
  - Eliminate a variable by fully connecting its neighbors
  - Sum-product and max-product are semirings
  - Directed and undirected graphs work in a similar way
- Efficiency of VE depends on the induced-width
  - Finding the best order is NP-hard
  - Intuition helps