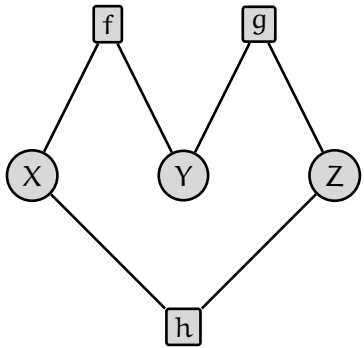
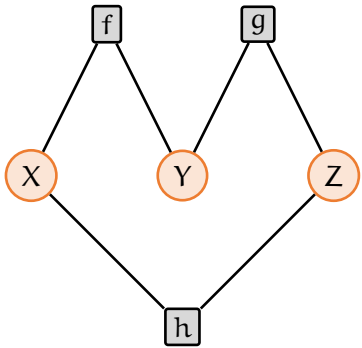
Variable Elimination and Treewidth

CS 221: Finn & Anari

Factor Graph Review



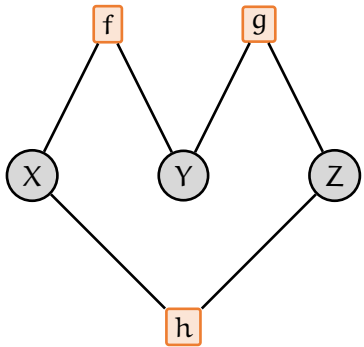
Factor Graph Review



Variables

- The X, Y, Z nodes.
- Each variable takes values from a specified set called its domain.

Factor Graph Review



Variables

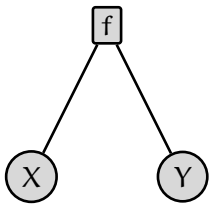
- The X, Y, Z nodes.
- Each variable takes values from a specified set called its domain.

Factors

- The f, g, h nodes.
- Each factor is a local function whose input consists of values for connected variables.

Weight Database

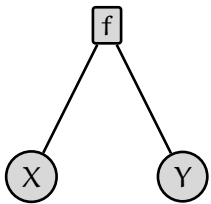
- Each factor is a small table indexed by connected variables.



X	Y	f
0	0	1.5
0	1	2.5
1	0	0.3
1	1	1.9

Weight Database

- Each factor is a small table indexed by connected variables.



X	Y	f
0	0	1.5
0	1	2.5
1	0	0.3
1	1	1.9

- Together they form the global weight database.

X	Y	Z	weight $= f \cdot g \cdot h$
0	0	0	$f(0,0) \cdot g(0,0) \cdot h(0,0)$
0	0	1	$f(0,0) \cdot g(0,1) \cdot h(0,1)$
0	1	0	$f(0,1) \cdot g(1,0) \cdot h(0,0)$
0	1	1	$f(0,1) \cdot g(1,1) \cdot h(0,1)$
1	0	0	$f(1,0) \cdot g(0,0) \cdot h(1,0)$
1	0	1	$f(1,0) \cdot g(0,1) \cdot h(1,1)$
1	1	0	$f(1,1) \cdot g(1,0) \cdot h(1,0)$
1	1	1	$f(1,1) \cdot g(1,1) \cdot h(1,1)$

- Weight database is exponentially large. 

$$n \text{ variables each with domain } \{0, 1\} \implies 2^n \text{ rows}$$

- Weight database is exponentially large. 

$$n \text{ variables each with domain } \{0, 1\} \implies 2^n \text{ rows}$$

- Factors are more manageable as long as their arity is small.

$$\begin{aligned} \text{unary factor} &\implies 2 \text{ rows} \\ \text{binary factor} &\implies 4 \text{ rows} \\ &\vdots \end{aligned}$$

Inference Problem

- **Input:** Description of factors.
- **Output:** Weight database?

Inference Problem

- **Input:** Description of factors. 
- **Output:** Weight database? 

Inference Problem

- **Input:** Description of factors. 
- **Output:** Reduction of weight database. 

Reduction

Key Idea

Produce table on just a small subset of variables.

Reduction

Key Idea

Produce table on just a small subset of variables.

How do we reduce to a subset of variables?

Reduction

Key Idea

Produce table on just a small subset of variables.

How do we reduce to a subset of variables?
Using a reduction/aggregation operator.

Sum ($\sum$)

Max ($\bigvee$)

Reduction

Key Idea

Produce table on just a small subset of variables.

How do we reduce to a subset of variables?
Using a reduction/aggregation operator.

Sum ($\sum$) Max ($\bigvee$)

- Partial reduction (subset $\neq \emptyset$):

$$\text{output}(x) := \sum_{y} \sum_{z} \text{weight}(x, y, z)$$

Reduction

Key Idea

Produce table on just a small subset of variables.

How do we reduce to a subset of variables?
Using a reduction/aggregation operator.

Sum ($\sum$)

- Partial reduction (subset $\neq \emptyset$):

$$\text{output}(x) := \sum_{y} \sum_{z} \text{weight}(x, y, z)$$

Max ($\bigvee$)

- Full reduction (subset $= \emptyset$):

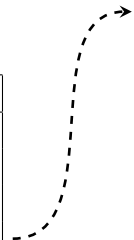
$$\text{output} = \bigvee_{x} \bigvee_{y} \bigvee_{z} \text{weight}(x, y, z)$$

global weight database

X	Y	Z	weight $= f \cdot g \cdot h$
0	0	0	$f(0,0) \cdot g(0,0) \cdot h(0,0)$
0	0	1	$f(0,0) \cdot g(0,1) \cdot h(0,1)$
0	1	0	$f(0,1) \cdot g(1,0) \cdot h(0,0)$
0	1	1	$f(0,1) \cdot g(1,1) \cdot h(0,1)$
1	0	0	$f(1,0) \cdot g(0,0) \cdot h(1,0)$
1	0	1	$f(1,0) \cdot g(0,1) \cdot h(1,1)$
1	1	0	$f(1,1) \cdot g(1,0) \cdot h(1,0)$
1	1	1	$f(1,1) \cdot g(1,1) \cdot h(1,1)$

global weight database

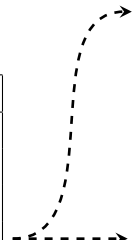
X	Y	Z	weight $= f \cdot g \cdot h$
0	0	0	$f(0,0) \cdot g(0,0) \cdot h(0,0)$
0	0	1	$f(0,0) \cdot g(0,1) \cdot h(0,1)$
0	1	0	$f(0,1) \cdot g(1,0) \cdot h(0,0)$
0	1	1	$f(0,1) \cdot g(1,1) \cdot h(0,1)$
1	0	0	$f(1,0) \cdot g(0,0) \cdot h(1,0)$
1	0	1	$f(1,0) \cdot g(0,1) \cdot h(1,1)$
1	1	0	$f(1,1) \cdot g(1,0) \cdot h(1,0)$
1	1	1	$f(1,1) \cdot g(1,1) \cdot h(1,1)$



X	Y	output
0	0	$\sum_z f(0,0) \cdot g(0,z) \cdot h(0,z)$
0	1	$\sum_z f(0,1) \cdot g(1,z) \cdot h(0,z)$
1	0	$\sum_z f(1,0) \cdot g(0,z) \cdot h(1,z)$
1	1	$\sum_z f(1,1) \cdot g(1,z) \cdot h(1,z)$

global weight database

X	Y	Z	weight $= f \cdot g \cdot h$
0	0	0	$f(0,0) \cdot g(0,0) \cdot h(0,0)$
0	0	1	$f(0,0) \cdot g(0,1) \cdot h(0,1)$
0	1	0	$f(0,1) \cdot g(1,0) \cdot h(0,0)$
0	1	1	$f(0,1) \cdot g(1,1) \cdot h(0,1)$
1	0	0	$f(1,0) \cdot g(0,0) \cdot h(1,0)$
1	0	1	$f(1,0) \cdot g(0,1) \cdot h(1,1)$
1	1	0	$f(1,1) \cdot g(1,0) \cdot h(1,0)$
1	1	1	$f(1,1) \cdot g(1,1) \cdot h(1,1)$

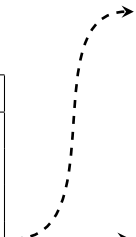


X	Y	output
0	0	$\sum_z f(0,0) \cdot g(0,z) \cdot h(0,z)$
0	1	$\sum_z f(0,1) \cdot g(1,z) \cdot h(0,z)$
1	0	$\sum_z f(1,0) \cdot g(0,z) \cdot h(1,z)$
1	1	$\sum_z f(1,1) \cdot g(1,z) \cdot h(1,z)$

X	output
0	$\sum_y \sum_z f(0,y) \cdot g(y,z) \cdot h(0,z)$
1	$\sum_y \sum_z f(1,y) \cdot g(y,z) \cdot h(1,z)$

global weight database

X	Y	Z	weight $= f \cdot g \cdot h$
0	0	0	$f(0,0) \cdot g(0,0) \cdot h(0,0)$
0	0	1	$f(0,0) \cdot g(0,1) \cdot h(0,1)$
0	1	0	$f(0,1) \cdot g(1,0) \cdot h(0,0)$
0	1	1	$f(0,1) \cdot g(1,1) \cdot h(0,1)$
1	0	0	$f(1,0) \cdot g(0,0) \cdot h(1,0)$
1	0	1	$f(1,0) \cdot g(0,1) \cdot h(1,1)$
1	1	0	$f(1,1) \cdot g(1,0) \cdot h(1,0)$
1	1	1	$f(1,1) \cdot g(1,1) \cdot h(1,1)$



X	Y	output
0	0	$\sum_z f(0,0) \cdot g(0,z) \cdot h(0,z)$
0	1	$\sum_z f(0,1) \cdot g(1,z) \cdot h(0,z)$
1	0	$\sum_z f(1,0) \cdot g(0,z) \cdot h(1,z)$
1	1	$\sum_z f(1,1) \cdot g(1,z) \cdot h(1,z)$

X	output
0	$\sum_y \sum_z f(0,y) \cdot g(y,z) \cdot h(0,z)$
1	$\sum_y \sum_z f(1,y) \cdot g(y,z) \cdot h(1,z)$

output
$\sum_x \sum_y \sum_z f(x,y) \cdot g(y,z) \cdot h(x,z)$

Reducing n variables with $\{0, 1\}$ domains naively takes $\geqslant 2^n$ time. 

$$\underbrace{\sum_x \sum_y \sum_z \cdots}_{2^n \text{ terms in these sums}}$$

Reducing n variables with $\{0, 1\}$ domains naively takes $\geqslant 2^n$ time. 

$$\underbrace{\sum_x \sum_y \sum_z \cdots}_{2^n \text{ terms in these sums}}$$

Can we do better?

Variable Elimination

$$\sum_{x} \sum_{y} \sum_{z} f(x, y) \cdot g(y, z) \cdot h(x, z)$$

Variable Elimination

$$\sum_{x} \sum_{y} \sum_{z} f(x, y) \cdot g(y, z) \cdot h(x, z)$$

Variable Elimination

$$\sum_{y} \sum_{z} \sum_{x} f(x, y) \cdot g(y, z) \cdot h(x, z)$$

Variable Elimination

$$\sum_{y} \sum_{z} \sum_{x} f(x, y) \cdot g(y, z) \cdot h(x, z)$$

Variable Elimination

$$\sum_{y} \sum_{z} g(y,z) \cdot \left(\sum_{x} f(x,y) h(x,z) \right)$$

Variable Elimination

$$\sum_{y} \sum_{z} g(y,z) \cdot \underbrace{\left(\sum_{x} f(x,y)h(x,z) \right)}_{\text{this is a function of } y,z}$$

Variable Elimination

$$\sum_{y} \sum_{z} g(y,z) \cdot \underbrace{\left(\sum_{x} f(x,y)h(x,z) \right)}_{\text{this is a function of } y,z}$$

- We can treat the highlighted term as a factor on y, z.

Variable Elimination

$$\sum_y \sum_z g(y,z) \cdot \underbrace{\left(\sum_x f(x,y)h(x,z) \right)}_{\text{this is a function of } y,z}$$

- We can treat the highlighted term as a factor on y, z.
- Precompute its table by enumerating y, z, and each time summing over x.

Y	Z	output
0	0	$\sum_x f(x,0) \cdot \cdot h(x,0)$
0	1	$\sum_x f(x,0) \cdot h(x,1)$
1	0	$\sum_x f(x,1) \cdot h(x,0)$
1	1	$\sum_x f(x,1) \cdot h(x,1)$

Variable Elimination

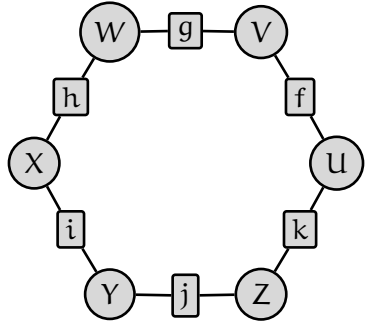
$$\sum_y \sum_z g(y,z) \cdot \underbrace{\left(\sum_x f(x,y)h(x,z) \right)}_{\text{this is a function of } y,z}$$

- We can treat the highlighted term as a factor on y, z.
- Precompute its table by enumerating y, z, and each time summing over x.

Y	Z	output
0	0	$\sum_x f(x,0) \cdot \cdot h(x,0)$
0	1	$\sum_x f(x,0) \cdot h(x,1)$
1	0	$\sum_x f(x,1) \cdot h(x,0)$
1	1	$\sum_x f(x,1) \cdot h(x,1)$

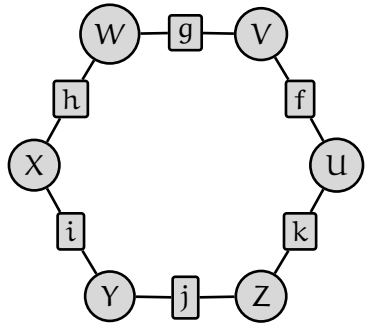
- For this particular example, this does not seem like much help. 

Example: Cycle Factor Graph



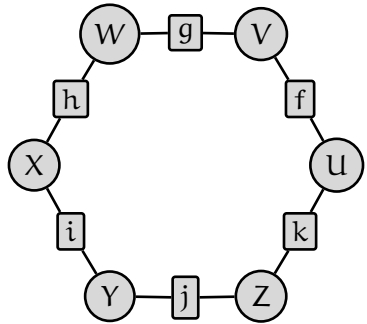
$$\sum_{u}\sum_{v}\sum_{w}\sum_{x}\sum_{y}\sum_{z} f(u,v)\cdot g(v,w)\cdot h(w,x)\cdot i(x,y)\cdot j(y,z)\cdot k(z,u)$$

Example: Cycle Factor Graph



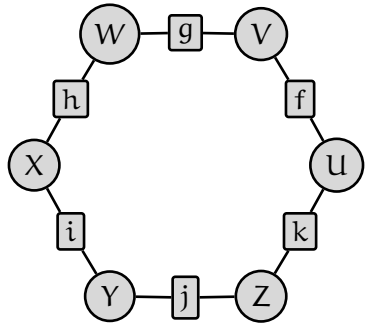
$$\sum_{u}\sum_{v}\sum_{w}\sum_{x}\sum_{y}\sum_{z} f(u,v)\cdot g(v,w)\cdot h(w,x)\cdot i(x,y)\cdot j(y,z)\cdot k(z,u)$$

Example: Cycle Factor Graph



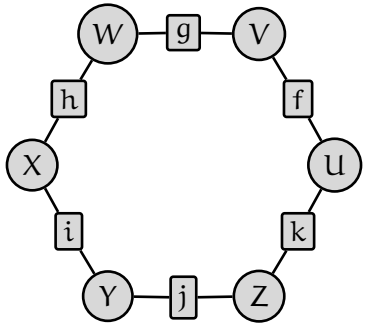
$$\sum_{v}\sum_{w}\sum_{x}\sum_{y}\sum_{z}\sum_{u} f(u,v)\cdot g(v,w)\cdot h(w,x)\cdot i(x,y)\cdot j(y,z)\cdot k(z,u)$$

Example: Cycle Factor Graph



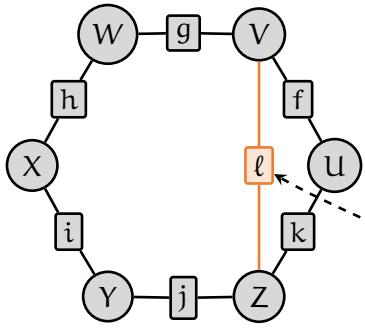
$$\sum_{v}\sum_{w}\sum_{x}\sum_{y}\sum_{z}\sum_{u} f(u,v)\cdot g(v,w)\cdot h(w,x)\cdot i(x,y)\cdot j(y,z)\cdot k(z,u)$$

Example: Cycle Factor Graph



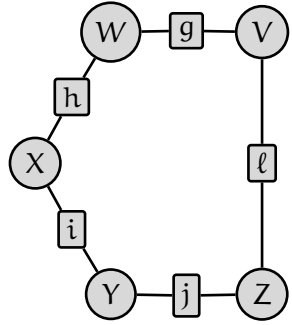
$$\sum_{v}\sum_{w}\sum_{x}\sum_{y}\sum_{z} g(v,w)\cdot h(w,x)\cdot i(x,y)\cdot j(y,z)\cdot\left(\sum_{u} f(u,v)\cdot k(z,u)\right)$$

Example: Cycle Factor Graph



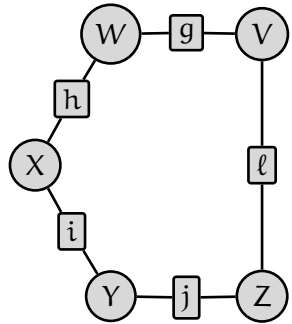
$$\sum_{v}\sum_{w}\sum_{x}\sum_{y}\sum_{z} g(v,w)\cdot h(w,x)\cdot i(x,y)\cdot j(y,z)\cdot \left(\sum_{u} f(u,v)\cdot k(z,u)\right)$$

Example: Cycle Factor Graph



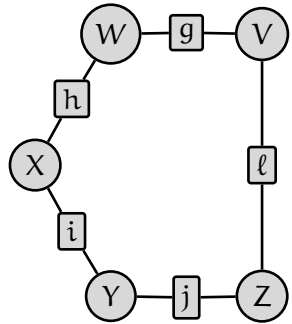
$$\sum_{v}\sum_{w}\sum_{x}\sum_{y}\sum_{z} g(v,w)\cdot h(w,x)\cdot i(x,y)\cdot j(y,z)\cdot \ell(v,z)$$

Example: Cycle Factor Graph



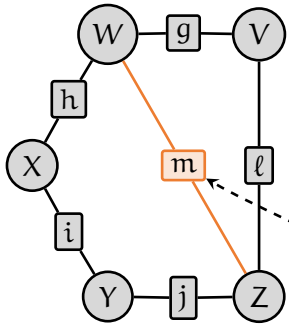
$$\sum_{v} \sum_{w} \sum_{x} \sum_{y} \sum_{z} g(v,w) \cdot h(w,x) \cdot i(x,y) \cdot j(y,z) \cdot \ell(v,z)$$

Example: Cycle Factor Graph



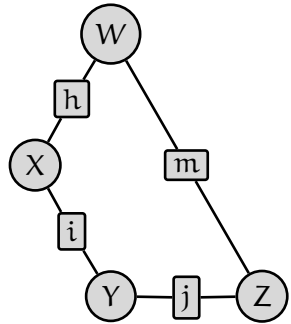
$$\sum_{w}\sum_{x}\sum_{y}\sum_{z} h(w,x)\cdot i(x,y)\cdot j(y,z)\cdot\left(\sum_{v} g(v,w)\cdot \ell(v,z)\right)$$

Example: Cycle Factor Graph



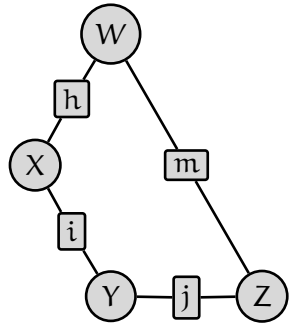
$$\sum_{w}\sum_{x}\sum_{y}\sum_{z} h(w,x)\cdot i(x,y)\cdot j(y,z)\cdot \overbrace{\left(\sum_{v} g(v,w)\cdot \ell(v,z)\right)}$$

Example: Cycle Factor Graph



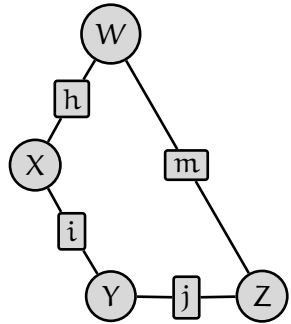
$$\sum_{w}\sum_{x}\sum_{y}\sum_{z} \mathfrak{h}(w,x)\cdot \mathfrak{i}(x,y)\cdot \mathfrak{j}(y,z)\cdot \mathfrak{m}(w,z)$$

Example: Cycle Factor Graph



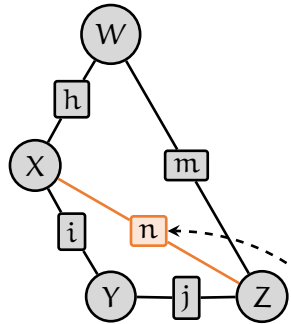
$$\sum_{w} \sum_{x} \sum_{y} \sum_{z} \mathfrak{h}(w,x) \cdot \mathfrak{i}(x,y) \cdot \mathfrak{j}(y,z) \cdot \mathfrak{m}(w,z)$$

Example: Cycle Factor Graph



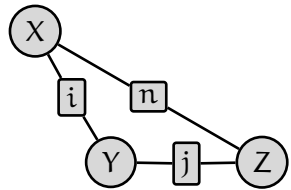
$$\sum_{x}\sum_{y}\sum_{z} i(x,y)\cdot j(y,z)\cdot\left(\sum_{w} h(w,x)\cdot m(w,z)\right)$$

Example: Cycle Factor Graph



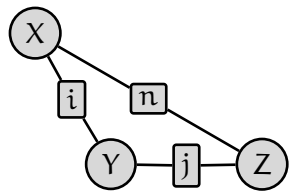
$$\sum_{x}\sum_{y}\sum_{z} i(x,y)\cdot j(y,z)\cdot \overbrace{\left(\sum_{w} h(w,x)\cdot m(w,z)\right)}$$

Example: Cycle Factor Graph



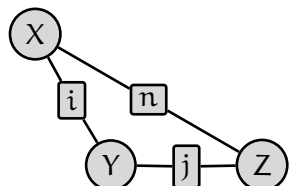
$$\sum_{x} \sum_{y} \sum_{z} i(x,y) \cdot j(y,z) \cdot n(x,z)$$

Example: Cycle Factor Graph



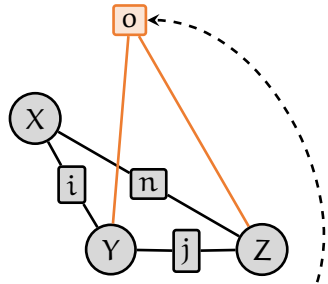
$$\sum_{x} \sum_{y} \sum_{z} i(x,y) \cdot j(y,z) \cdot n(x,z)$$

Example: Cycle Factor Graph



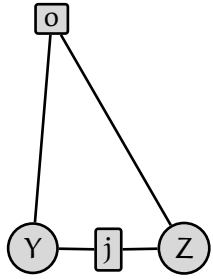
$$\sum_{y} \sum_{z} j(y,z) \cdot \left(\sum_{x} i(x,y) \cdot n(x,z) \right)$$

Example: Cycle Factor Graph



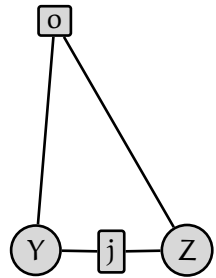
$$\sum_{y} \sum_{z} j(y,z) \cdot \overbrace{\left(\sum_{x} i(x,y) \cdot n(x,z) \right)}$$

Example: Cycle Factor Graph



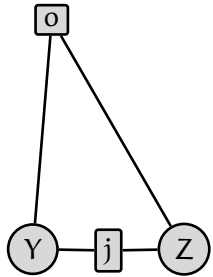
$$\sum_{y} \sum_{z} j(y,z) \cdot o(y,z)$$

Example: Cycle Factor Graph



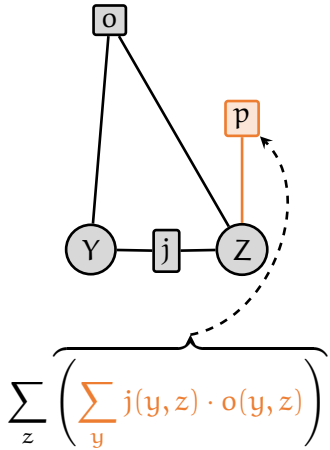
$$\sum_{y} \sum_{z} j(y,z) \cdot o(y,z)$$

Example: Cycle Factor Graph

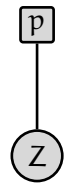


$$\sum_{z} \left(\sum_{y} j(y,z) \cdot o(y,z) \right)$$

Example: Cycle Factor Graph

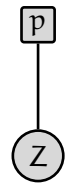


Example: Cycle Factor Graph



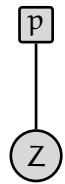
$$\sum_{z} \mathrm{p}(z)$$

Example: Cycle Factor Graph



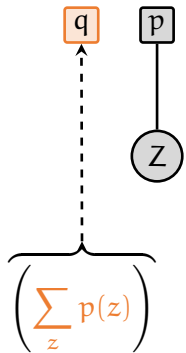
$$\sum_{z} p(z)$$

Example: Cycle Factor Graph



$$\left(\sum_{z} p(z) \right)$$

Example: Cycle Factor Graph



Example: Cycle Factor Graph

q

- In the process we only created binary, unary, and 0-ary factors.

- In the process we only created binary, unary, and 0-ary factors.
- The computation was bounded in each step, regardless of cycle length. 

- In the process we only created binary, unary, and 0-ary factors.
- The computation was bounded in each step, regardless of cycle length. 

Measure of Performance

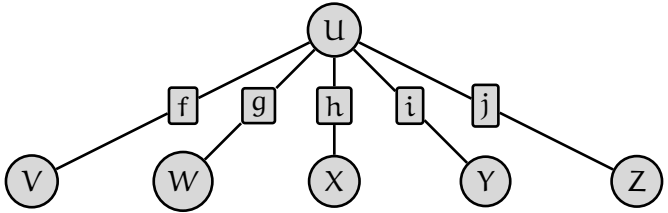
Maximum arity of new factors we create.

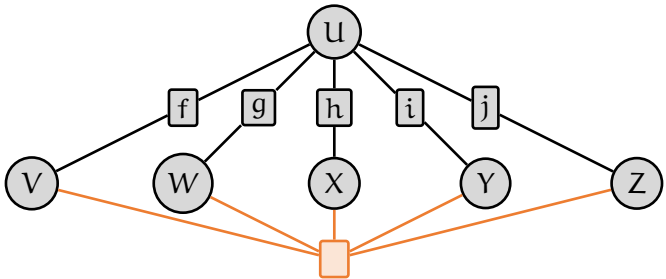
- In the process we only created binary, unary, and 0-ary factors.
- The computation was bounded in each step, regardless of cycle length. 

Measure of Performance

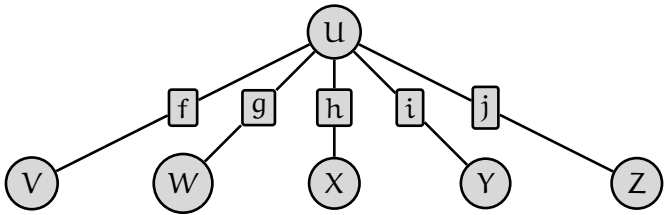
Maximum arity of new factors we create.

- In general, performance depends on the order of eliminating variables (not for the cycle graph though).

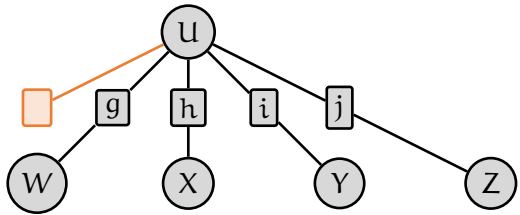

U
f
g
h
i
j
V
W
X
Y
Z



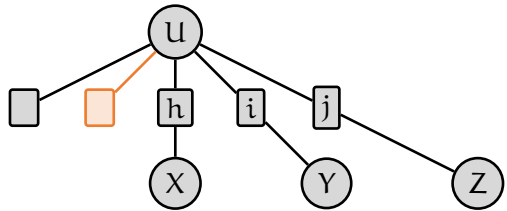
- If we eliminate u first, we create a 5-ary factor. ☹



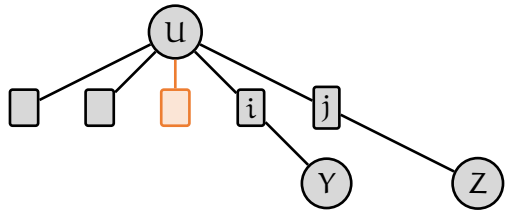
- If we eliminate U first, we create a 5-ary factor. ☹
- If we eliminate V, W, X, Y, Z and then U, we never create more than unary factors. ☺



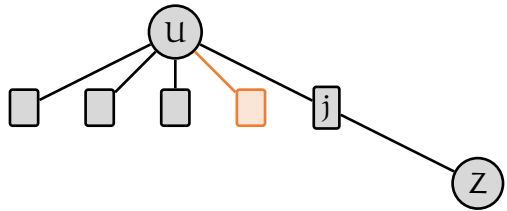
- If we eliminate U first, we create a 5-ary factor. ☹
- If we eliminate V, W, X, Y, Z and then U, we never create more than unary factors. ☺



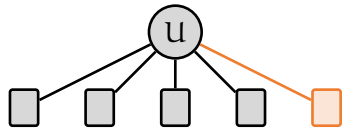
- If we eliminate U first, we create a 5-ary factor. ☹
- If we eliminate V, W, X, Y, Z and then U, we never create more than unary factors. ☺



- If we eliminate U first, we create a 5-ary factor. ☹
- If we eliminate V, W, X, Y, Z and then U, we never create more than unary factors. ☺



- If we eliminate U first, we create a 5-ary factor. 😕
- If we eliminate V, W, X, Y, Z and then U, we never create more than unary factors. 🙂



- If we eliminate u first, we create a 5-ary factor. 
- If we eliminate v, w, x, y, z and then u, we never create more than unary factors. 

Treewidth

The best case measure of performance, among all orderings.

Treewidth

The best case measure of performance, among all orderings.

- It is NP-hard to compute treewidth. 

Treewidth

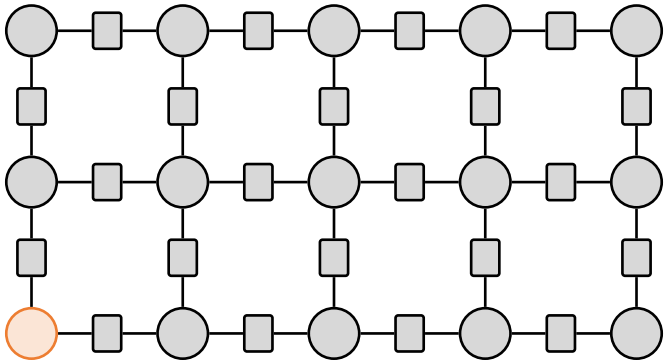
The best case measure of performance, among all orderings.

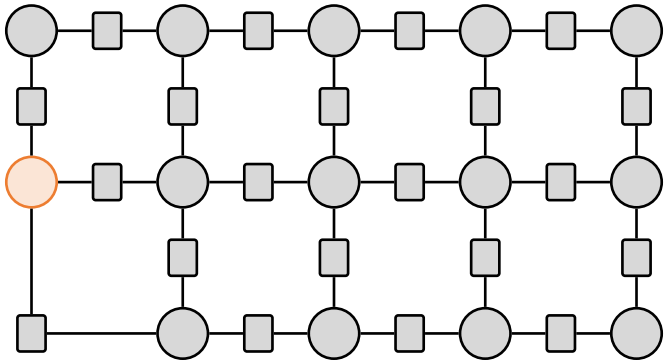
- It is NP-hard to compute treewidth. 
- Greedy heuristic: Eliminate the variable with the least number of neighboring variables each time.

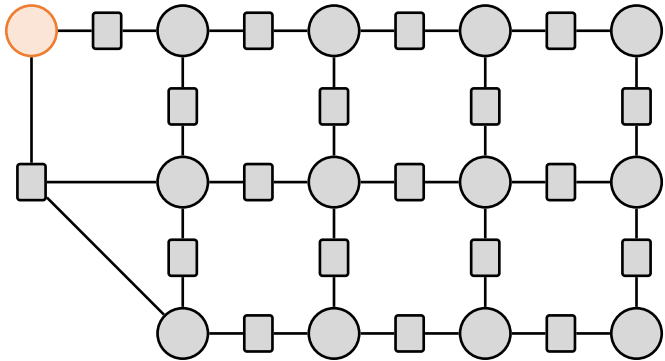
Treewidth

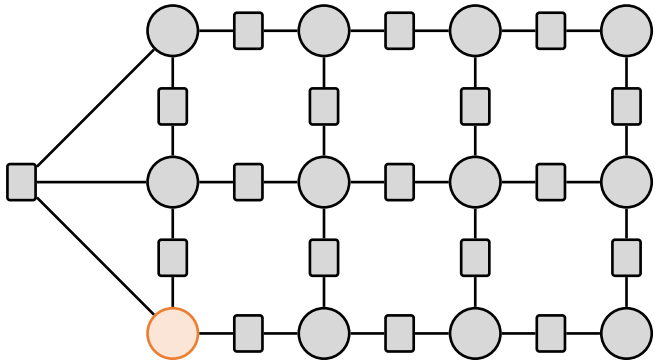
The best case measure of performance, among all orderings.

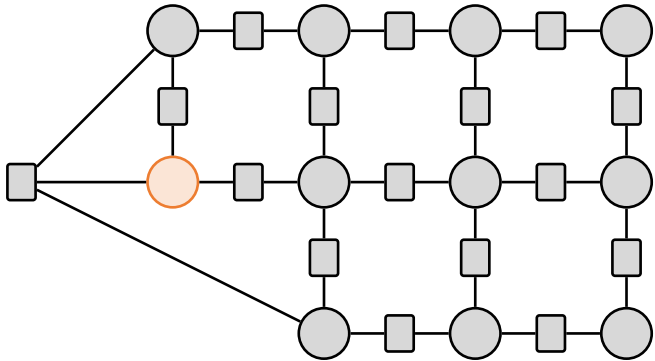
- It is NP-hard to compute treewidth. 
- Greedy heuristic: Eliminate the variable with the least number of neighboring variables each time.
- If the factor graph has no loop (i.e., a tree), the greedy algorithm is provably optimal. It only creates unary factors. 

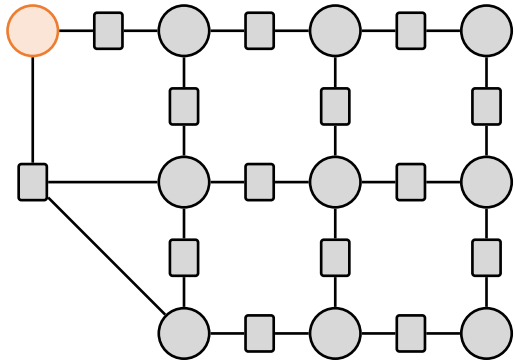


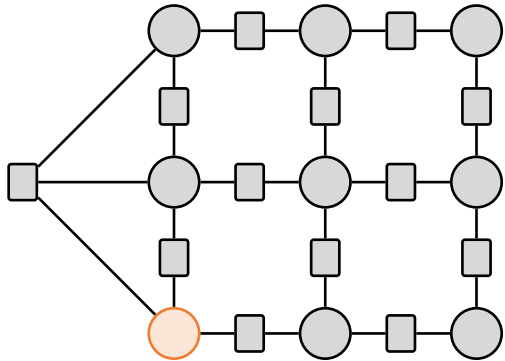


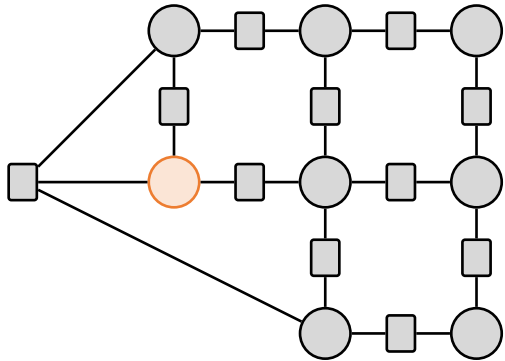


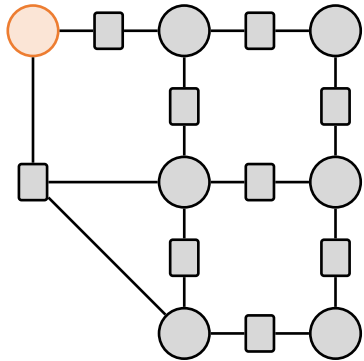


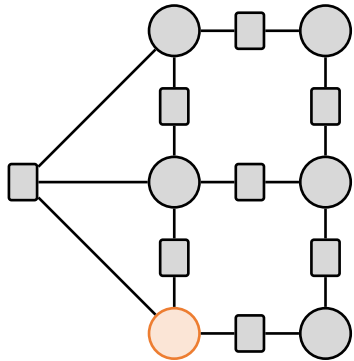


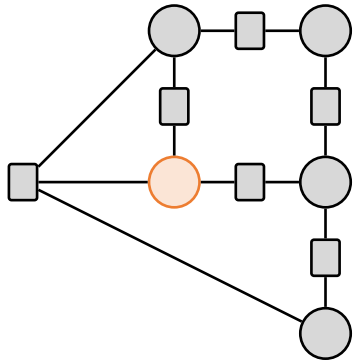


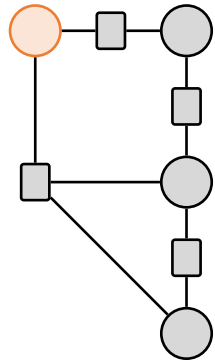


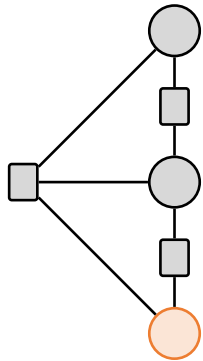


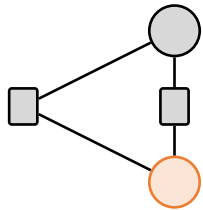


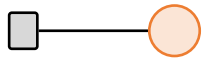












- For an $m \times n$ grid, the treewidth is $\min\{m, n\}$.

- For an $\mathbf{m} \times \mathbf{n}$ grid, the treewidth is $\mathbf{min}\{\mathbf{m}, \mathbf{n}\}$.
- From row-by-row elimination and column-by-column elimination, only one of them achieves $\mathbf{min}\{\mathbf{m}, \mathbf{n}\}$.

- For an $m \times n$ grid, the treewidth is $\min\{m, n\}$.
- From row-by-row elimination and column-by-column elimination, only one of them achieves $\min\{m, n\}$.
- Even for grids this is much better than naive aggregation:

$$2^{\min\{m,n\}} \ll 2^{m \times n}.$$

- For an $m \times n$ grid, the treewidth is $\min\{m, n\}$.
- From row-by-row elimination and column-by-column elimination, only one of them achieves $\min\{m, n\}$.
- Even for grids this is much better than naive aggregation:

$$2^{\min\{m,n\}} \ll 2^{m \times n}.$$

- Advanced: In some sense, grids are the only obstructions to having small treewidth. Having no grid minor of size $k \times k$ implies treewidth $\leqslant f(k)$.

- For an $m \times n$ grid, the treewidth is $\min\{m, n\}$.
- From row-by-row elimination and column-by-column elimination, only one of them achieves $\min\{m, n\}$.
- Even for grids this is much better than naive aggregation:

$$2^{\min\{m,n\}} \ll 2^{m \times n}.$$

- Advanced: In some sense, grids are the only obstructions to having small treewidth. Having no grid minor of size $k \times k$ implies treewidth $\leqslant f(k)$.
- Advanced: You can eliminate more than one variable at a time.