Understanding Approximations for Finding Maximum Likelihood Bounded Tree-width Markov Networks

Percy Liang

LIDS Student Conference

January 29, 2004

Overview

- Problem: approximating multivariate distributions using maximum likelihood bounded tree-width Markov networks

Overview

- Problem: approximating multivariate distributions using maximum likelihood bounded tree-width Markov networks

- Algorithm: finding the maximum weight hypertree using windmill farms

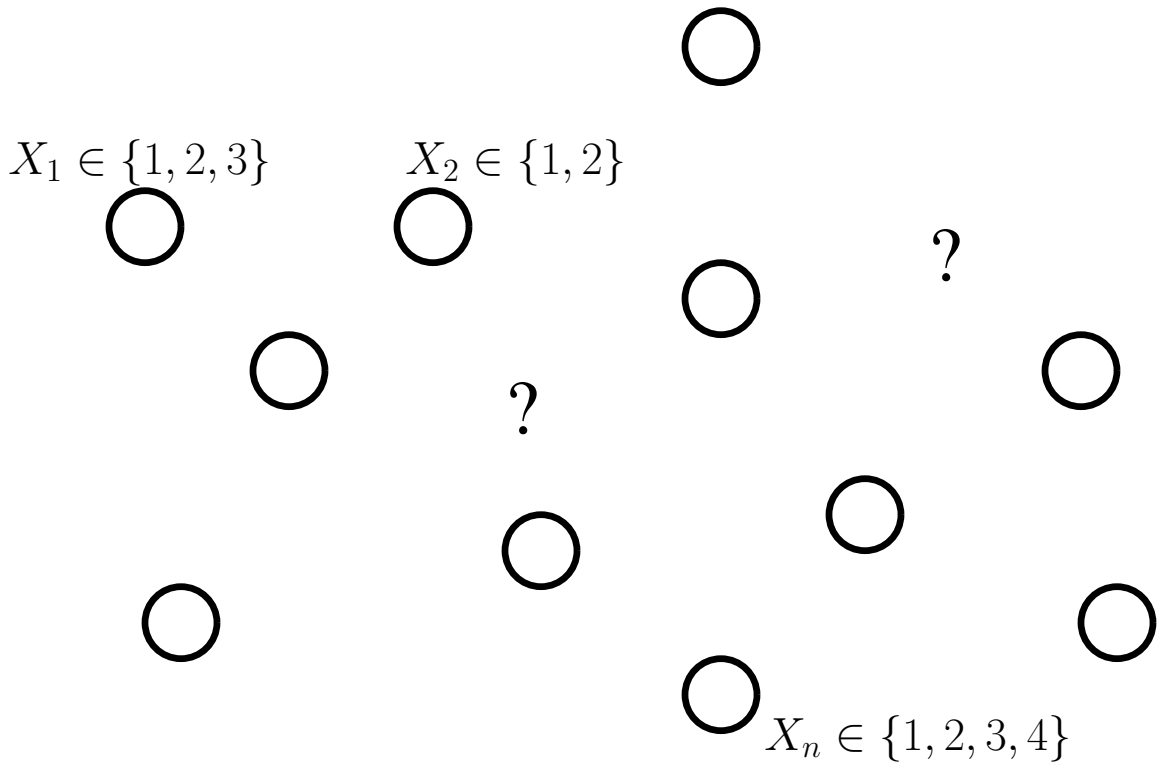
Overview

- Problem: approximating multivariate distributions using maximum likelihood bounded tree-width Markov networks

- Algorithm: finding the maximum weight hypertree using windmill farms

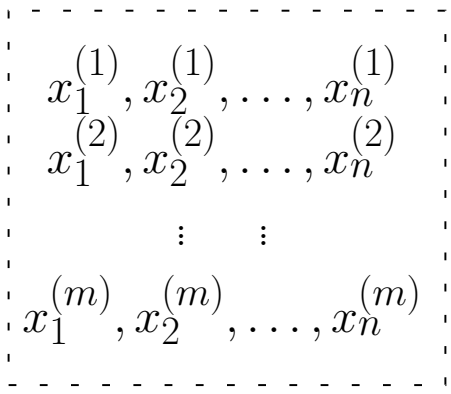
- Analysis: windmill farm coverage of hypertrees

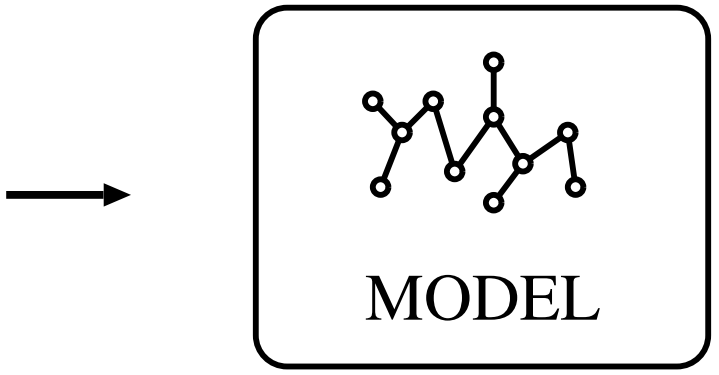
Situation: n interdependent variables



Maximum likelihood multivariate density estimation

Problem:





$\hat{P}(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$

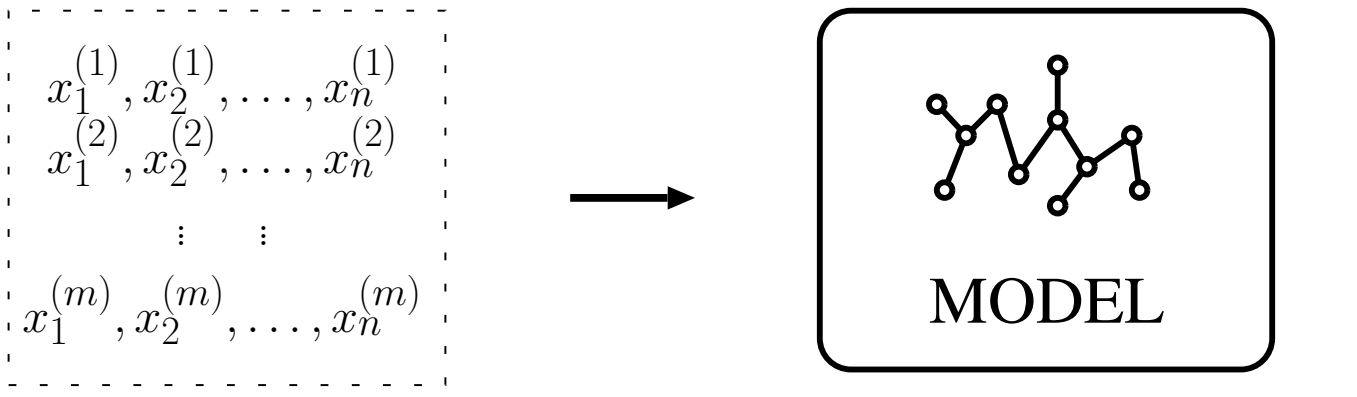
$P_\Theta(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$

D

Θ

Maximum likelihood multivariate density estimation

Problem:



$\hat{P}(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$

D

$P_\Theta(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$

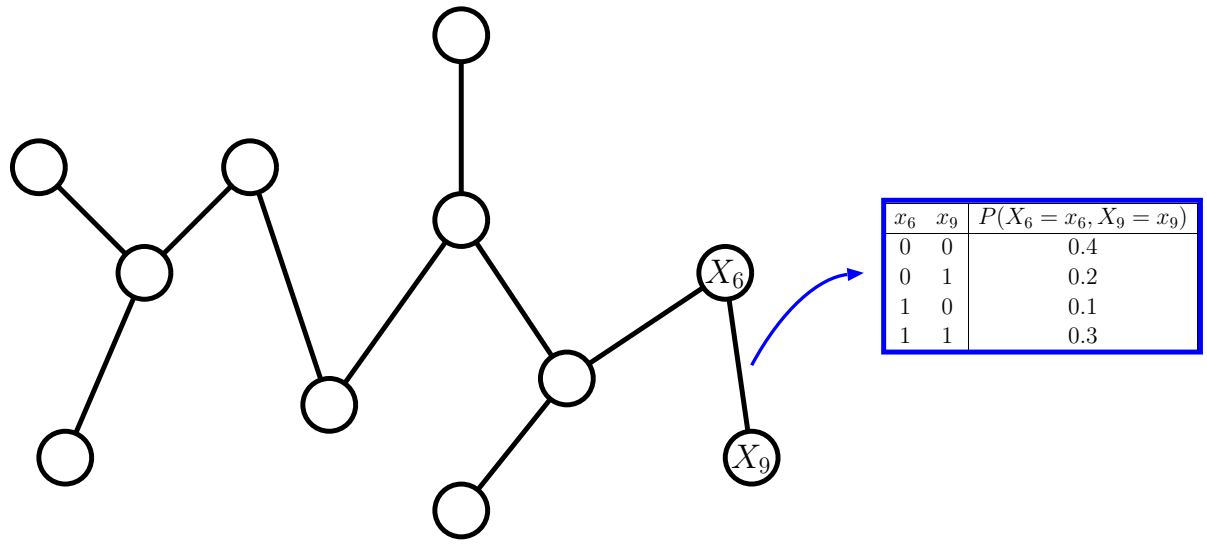
Θ

Maximize total likelihood: $P_\Theta(D)$
Maximize expected log-likelihood: $E_{x \sim \hat{P}}[\log P_\Theta(x)]$

Tree-width 1 Markov networks (trees)

Model pairwise dependencies.

$k = 1$



x_6	x_9	$P(X_6 = x_6, X_9 = x_9)$
0	0	0.4
0	1	0.2
1	0	0.1
1	1	0.3

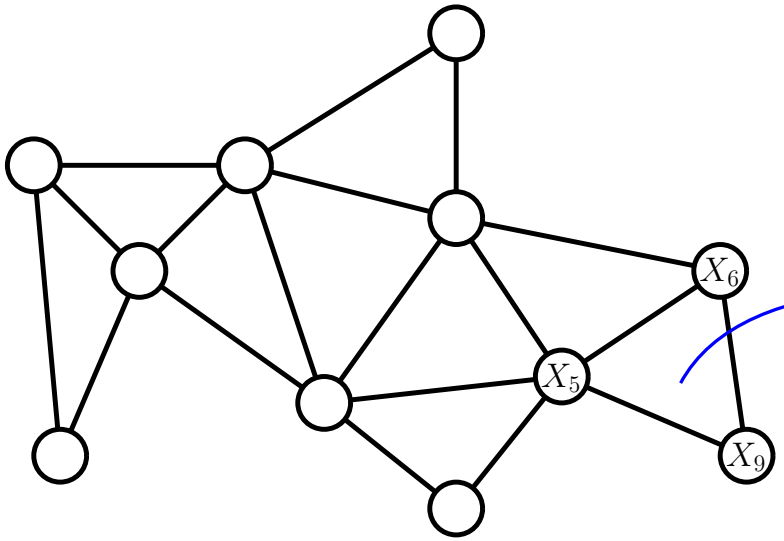
Θ: tree structure, marginal probabilities for each edge

Triangulated tree-width k Markov networks (hypertrees)

Model $(k+1)$-way dependencies.

Tree-width k: maximum size of clique (hyperedge) $= k+1$

$k = 2$



x_5	x_6	x_9	$P(X_5 = x_5, X_6 = x_6, X_9 = x_9)$
0	0	0	0.1
0	0	1	0.1
0	1	0	0.6
0	1	1	0.05
1	0	0	0.01
1	0	1	0
1	1	0	0.14
1	1	1	0

$\ominus$: hypertree structure, marginal probabilities for each clique

Maximizing the likelihood (given hypertree structure)

- Find: $\text{argmax}_\Theta P_\Theta(D) = \text{argmax}_\Theta E_{\hat{P}}[\log P_\Theta(x)]$.

Maximizing the likelihood (given hypertree structure)

- Find: $\text{argmax}_\Theta P_\Theta(D) = \text{argmax}_\Theta E_{\hat{P}}[\log P_\Theta(x)]$.

- $E_{\hat{P}}[P_\Theta(x)] = \sum_c w_c$

Maximizing the likelihood (given hypertree structure)

- Find: $\text{argmax}_\Theta P_\Theta(D) = \text{argmax}_\Theta E_{\hat{P}}[\log P_\Theta(x)]$.
- Probability factors into clique potentials: $P_\Theta(x) = \prod_{c \in C} \phi_c(x_c)$
- $E_{\hat{P}}[P_\Theta(x)] = \sum_c w_c$

Maximizing the likelihood (given hypertree structure)

- Find: $\text{argmax}_\Theta P_\Theta(D) = \text{argmax}_\Theta E_{\hat{P}}[\log P_\Theta(x)]$.
- Probability factors into clique potentials: $P_\Theta(x) = \prod_{c \in C} \phi_c(x_c)$
- To compute a potential: $\phi_c(x_c) = \frac{\hat{P}_c(x_c)}{\prod_{c' \subsetneq c} \phi_{c'}(x_{c'})}$
- $E_{\hat{P}}[P_\Theta(x)] = \sum_c w_c$

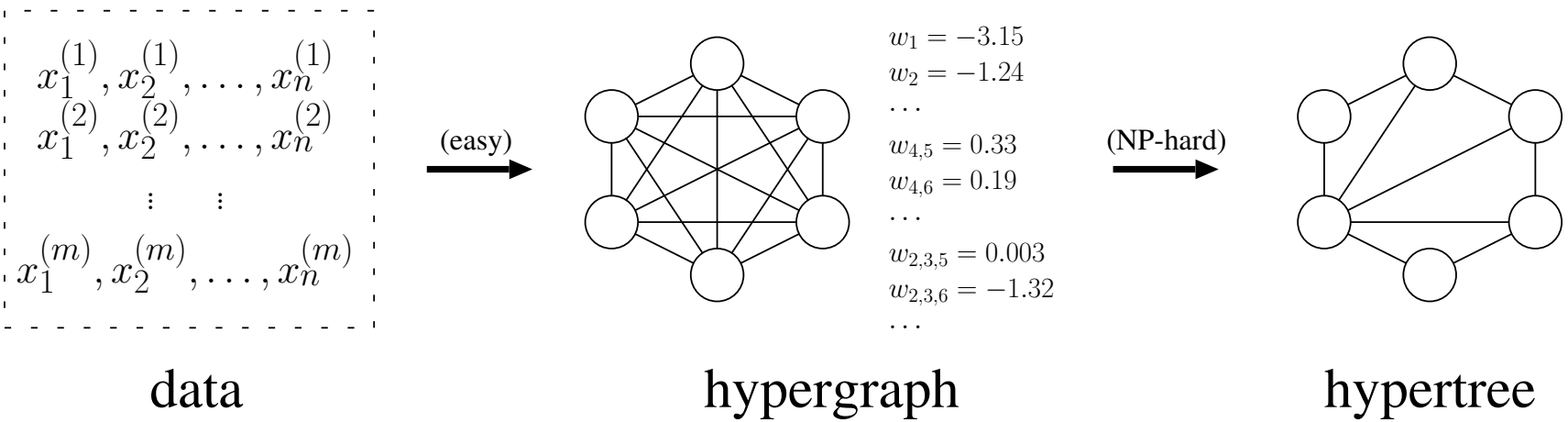
Maximizing the likelihood (given hypertree structure)

- Find: $\text{argmax}_\Theta P_\Theta(D) = \text{argmax}_\Theta E_{\hat{P}}[\log P_\Theta(x)]$.
- Probability factors into clique potentials: $P_\Theta(x) = \prod_{c \in C} \phi_c(x_c)$
- To compute a potential: $\phi_c(x_c) = \frac{\hat{P}_c(x_c)}{\prod_{c' \subsetneq c} \phi_{c'}(x_{c'})}$
- Clique (hyperedge) weights: $w_c = E_{\hat{P}}[\log \phi_c]$
- $E_{\hat{P}}[P_\Theta(x)] = \sum_c w_c$

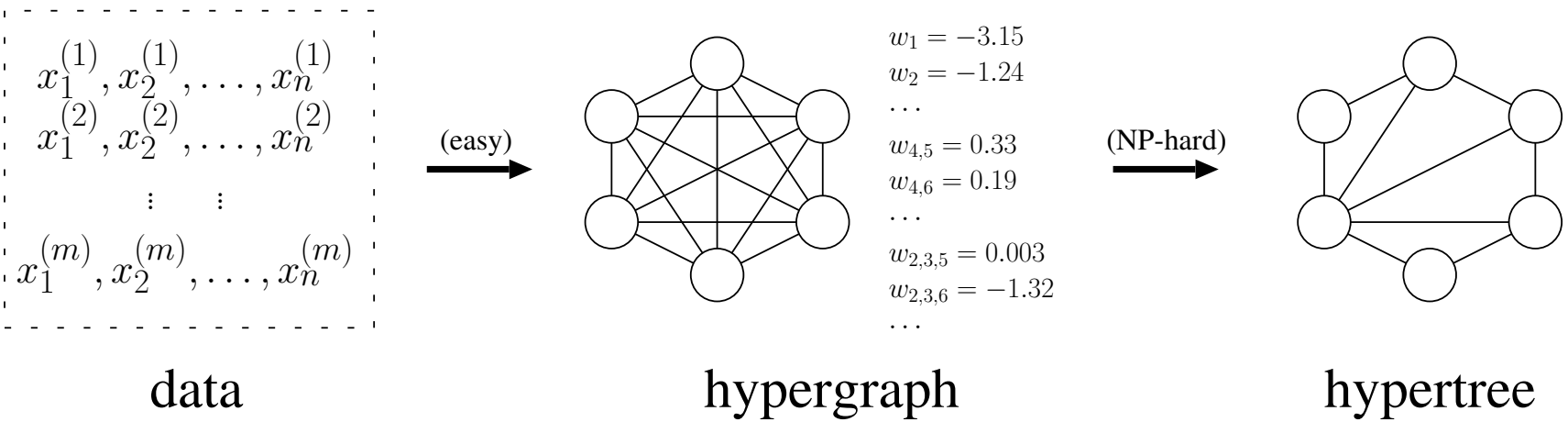
Maximizing the likelihood (given hypertree structure)

- Find: $\text{argmax}_\Theta P_\Theta(D) = \text{argmax}_\Theta E_{\hat{P}}[\log P_\Theta(x)]$.
- Probability factors into clique potentials: $P_\Theta(x) = \prod_{c \in C} \phi_c(x_c)$
- To compute a potential: $\phi_c(x_c) = \frac{\hat{P}_c(x_c)}{\prod_{c' \subsetneq c} \phi_{c'}(x_{c'})}$
- Clique (hyperedge) weights: $w_c = E_{\hat{P}}[\log \phi_c]$
- $E_{\hat{P}}[P_\Theta(x)] = \sum_c w_c$ (key: w_c independent of structure)

Maximizing the likelihood (overview)



Maximizing the likelihood (overview)

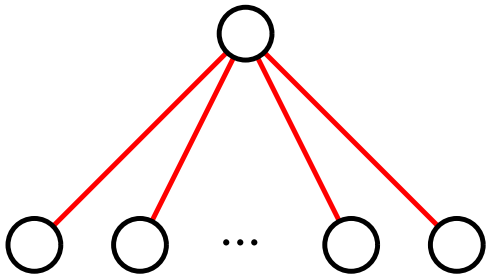


The maximum hypertree problem:

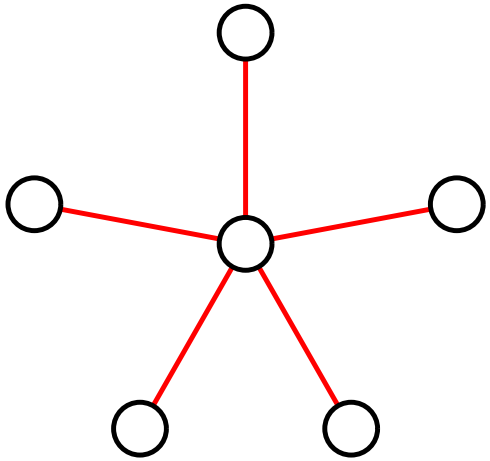
Input: k-hypergraph (H, w)

Output: maximum weight k-hypertree $T \subset H$

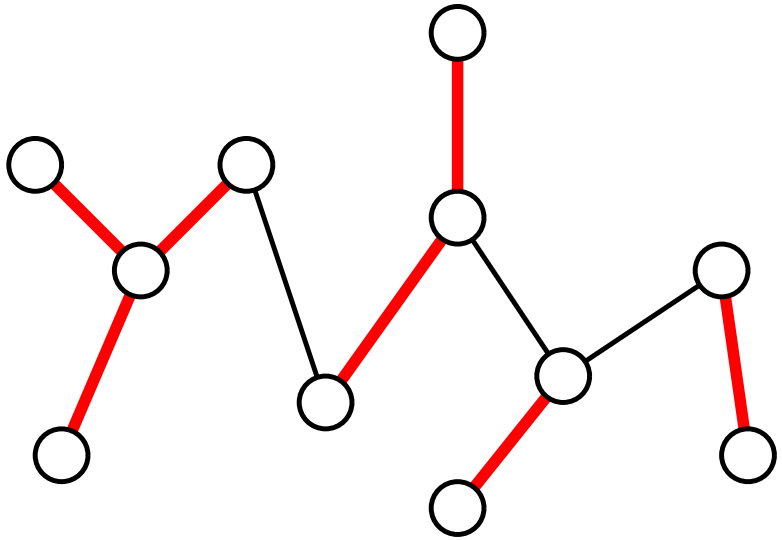
1-*windmill farms (constellations)*



star

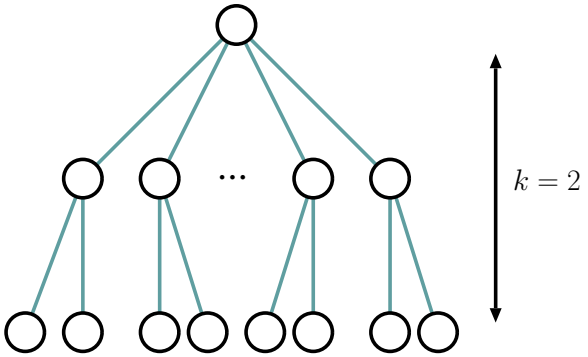


star

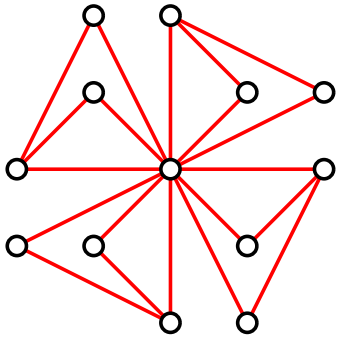


constellation in a tree

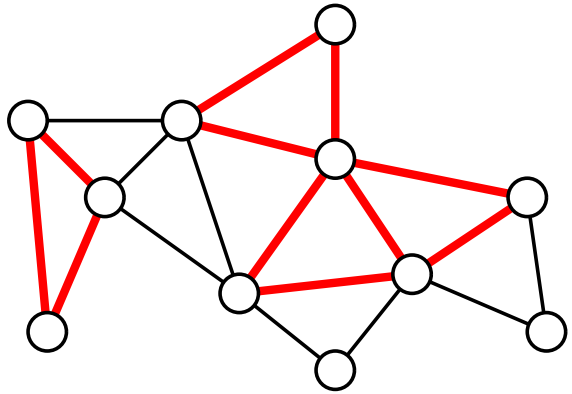
k-windmill farms



representing (rooted) forest for a 2-windmill

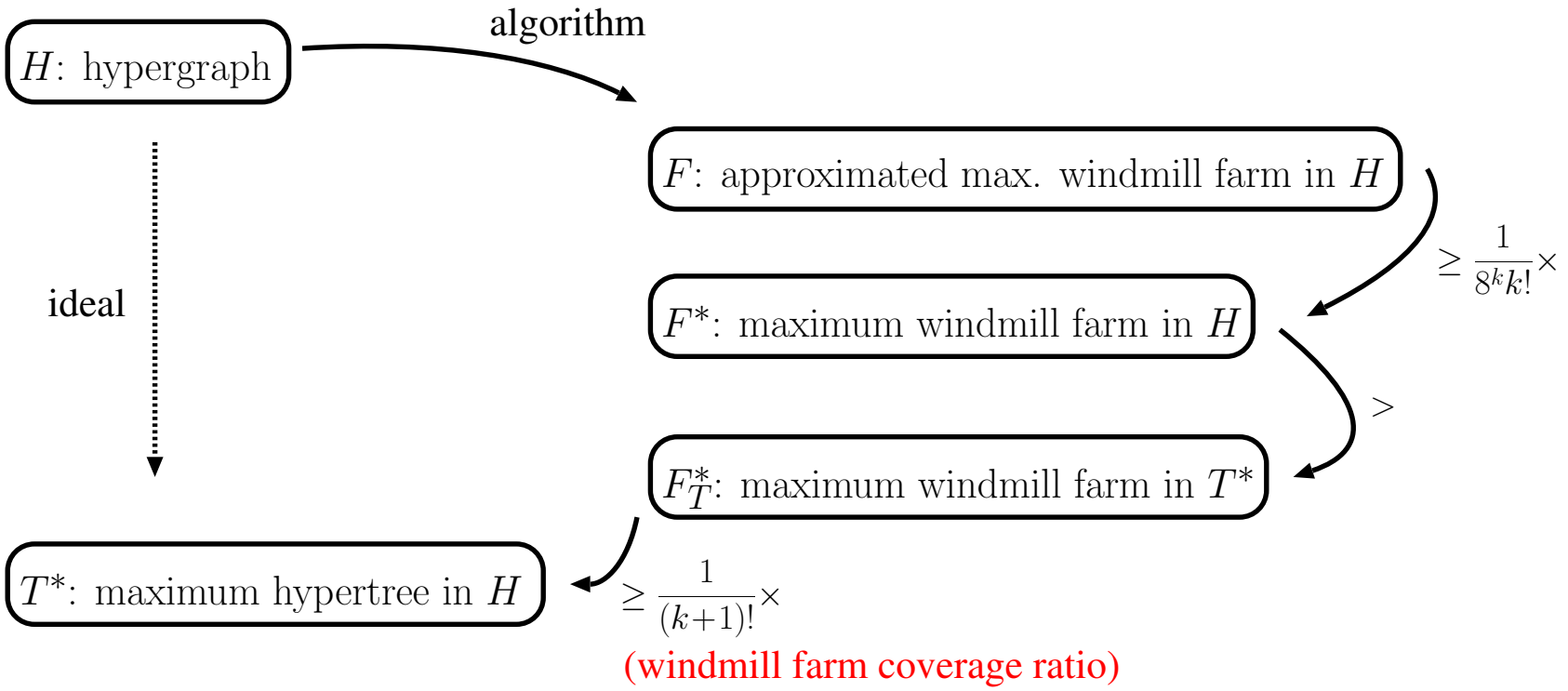


2-windmill



2-windmill farm in a hypertree

The approximation algorithm



Conclusion: $w(F) \geq \frac{1}{8^k k!(k+1)!} w(T^*)$

Analyzing the windmill coverage ratio: 3 problems

Assume weight of hypertree $w(T) = 1$.

1. Given a weighted hypertree [tree] (T, w), find the maximum weight windmill farm [constellation] F.

$$C_k(T, w) = \max_{F \subset T} w(F)$$

Analyzing the windmill coverage ratio: 3 problems

Assume weight of hypertree $w(T) = 1$.

1. Given a weighted hypertree [tree] (T, w), find the maximum weight windmill farm [constellation] F.

$$C_k(T, w) = \max_{F \subset T} w(F)$$

2. Given an unweighted hypertree [tree] structure T, find the "worst" assignment of weights w.

$$C_k(T) = \min_{w} \max_{F \subset T} w(F)$$

Analyzing the windmill coverage ratio: 3 problems

Assume weight of hypertree $w(T) = 1$.

1. Given a weighted hypertree [tree] (T, w), find the maximum weight windmill farm [constellation] F.

$$C_k(T, w) = \max_{F \subset T} w(F)$$

2. Given an unweighted hypertree [tree] structure T, find the "worst" assignment of weights w.

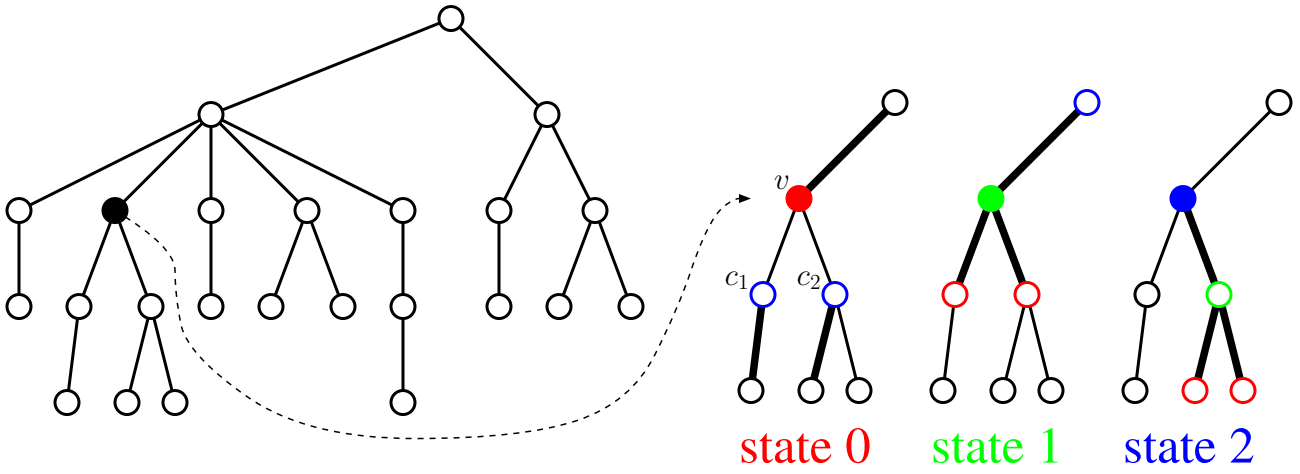
$$C_k(T) = \min_{w} \max_{F \subset T} w(F)$$

3. Find the "worst" weighted hypertree (T, w).

$$C_k = \min_{T} \min_{w} \max_{F \subset T} w(F)$$

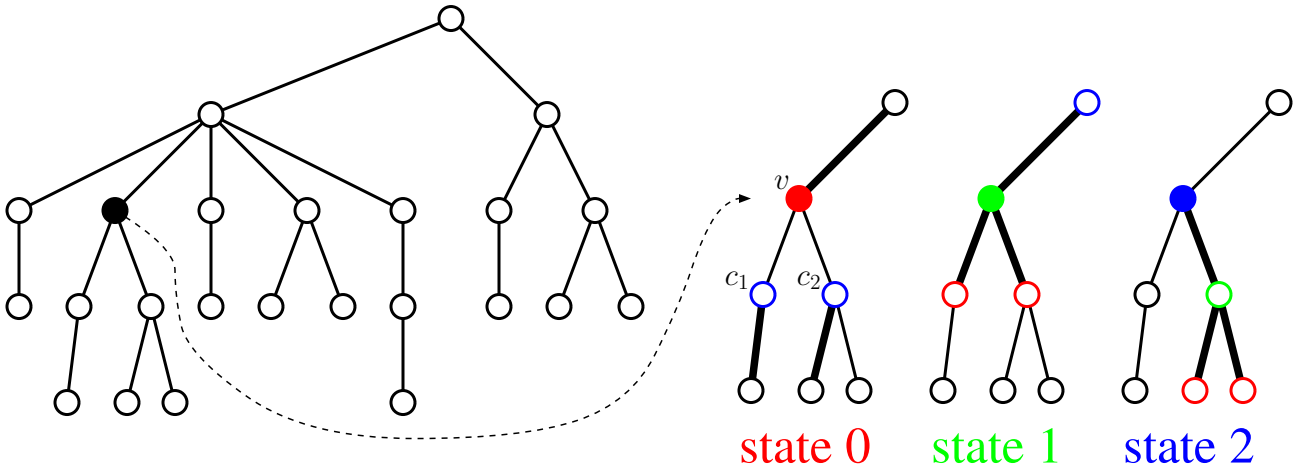
Problem 1: compute $C_1(T, w)$ (trees)

- Given a weighted tree (T, w), find the maximum weight constellation $F \subset T$ via dynamic programming.



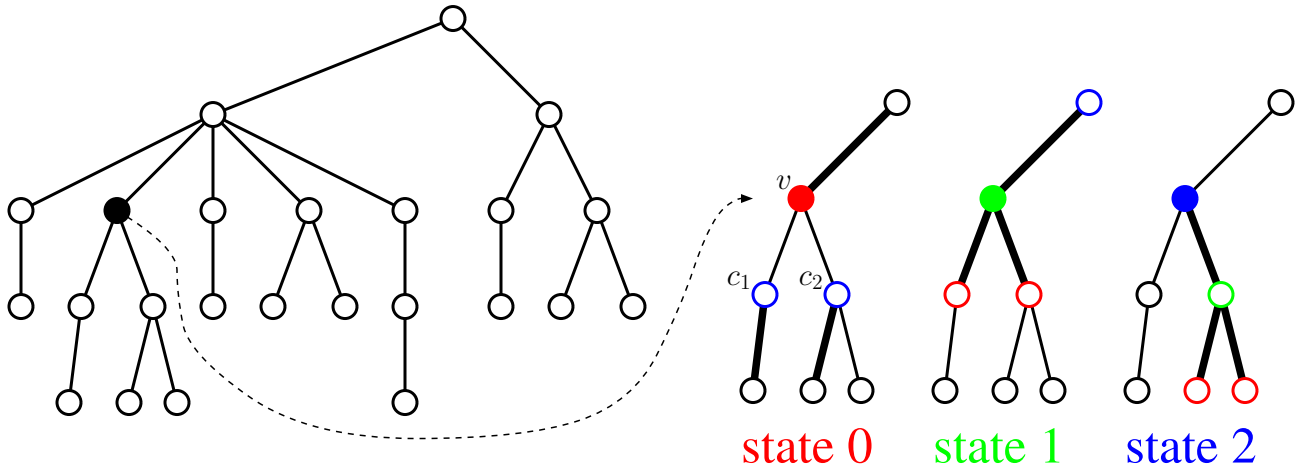
Problem 1: compute $C_1(T, w)$ (trees)

- Given a weighted tree (T, w), find the maximum weight constellation $F \subset T$ via dynamic programming.
- $f(v, q)$ = maximum weight of the extension of a constellation into the subtree rooted at vertex v given it is in state q



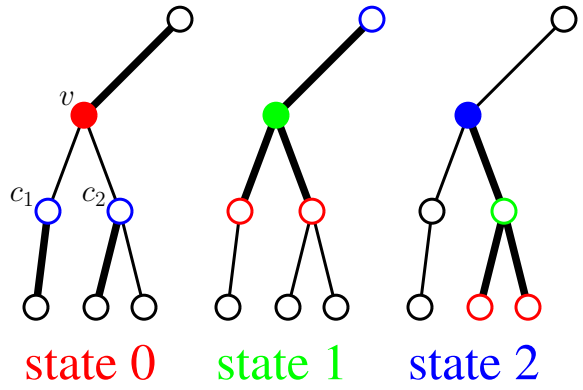
Problem 1: compute $C_1(T, w)$ (trees)

- Given a weighted tree (T, w), find the maximum weight constellation $F \subset T$ via dynamic programming.
- $f(v, q)$ = maximum weight of the extension of a constellation into the subtree rooted at vertex v given it is in state q
- $C_1(T, w) = f(\text{root}(T), 2)$



Problem 1: compute $C_1(T, w)$ (trees)

$$
\begin{aligned}
f(v,0) &= \sum_{i=1}^{n} f(c_i, 2) \\
f(v,1) &= \sum_{i=1}^{n} \max(f(c_i,2), w(v,c_i) + f(c_i,0)) \\
f(v,2) &= \max\left\{ f(v,1), f(v,0) + \max_{1 \le i \le n} w(v,c_i) + f(c_i,1) - f(c_i,2) \right\}
\end{aligned}
$$

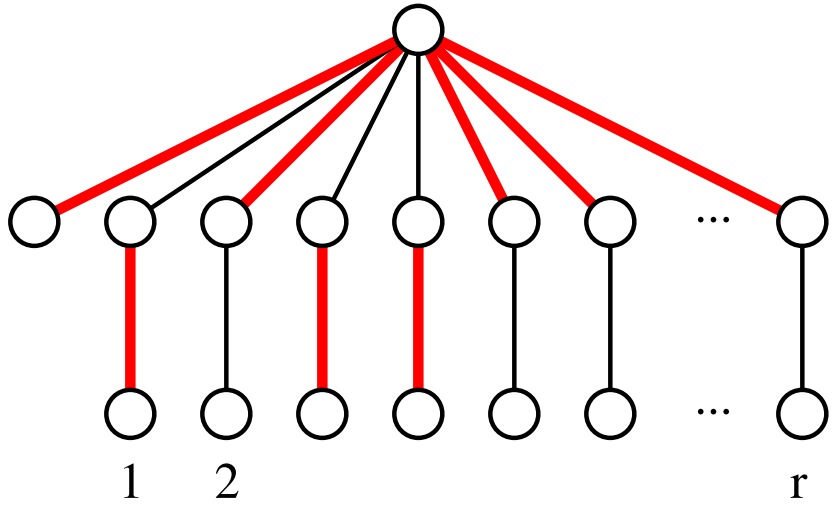


Problem 2: compute $C_1(T)$ (trees)

Convert the dynamic program into a linear program.

- Variables:
  - dynamic programming states $f(\cdot, \cdot)$
  - edge weights $w(\cdot)$
  - results of intermediate $\max\{\cdot\}$ expressions
- Equations:
  Directly use linear equations.
  $a = \max\{b, c\}$ becomes two equations: $a \geq b$ and $a \geq c$.

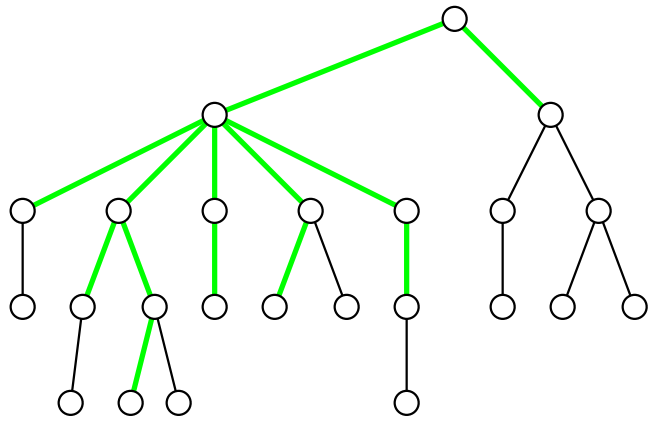
Problem 3: compute C_1 (trees)



C_1 approaches 1/2 from above as $r \to \infty$.
$C_1 \geq \frac{1}{(1+1)!} = 1/2$, so the bound is tight.

Vine conjecture

A vine is a tree in which every non-leaf vertex is adjacent to a leaf vertex.

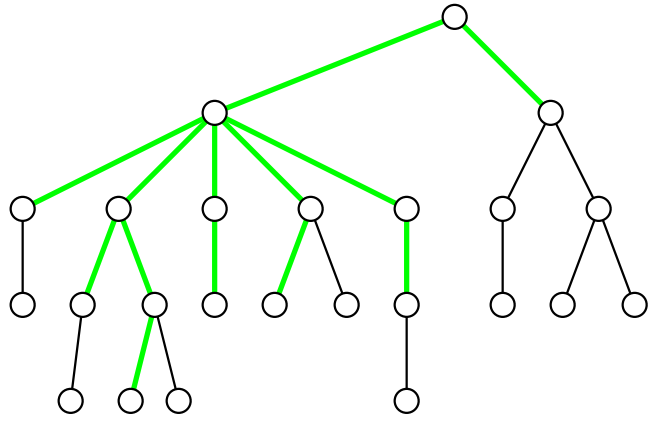


Vine conjecture

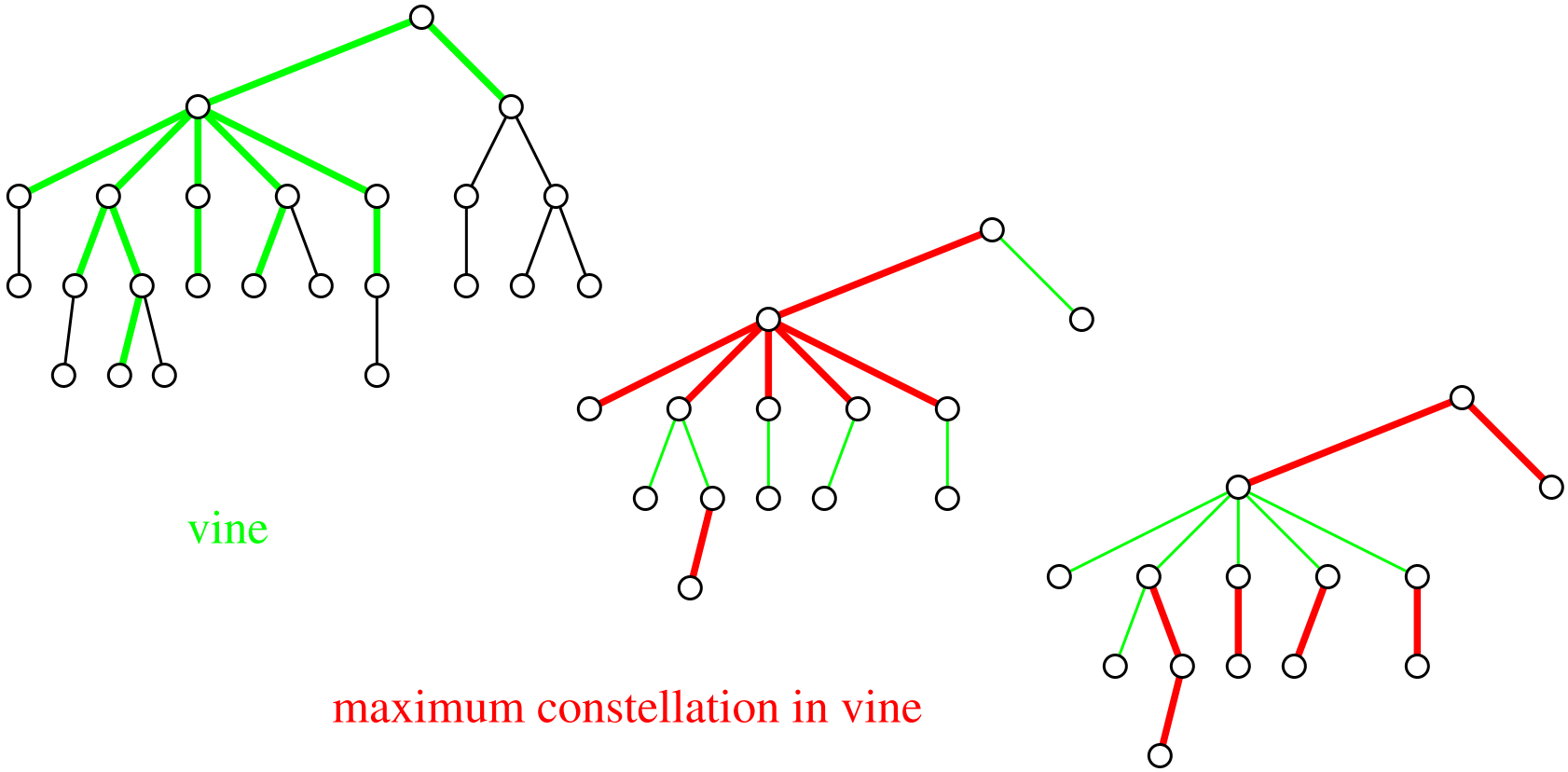
A vine is a tree in which every non-leaf vertex is adjacent to a leaf vertex. Conjecture:

$$C_k(T) = \frac{n}{2(n-1)} \text{ is achieved by: } w(e) = \begin{cases} \frac{1}{n-1} & \text{if } e \in V \\ 0 & \text{otherwise,} \end{cases}$$

where V is the maximum vine $V \subset T$ and n is the number of vertices in V.



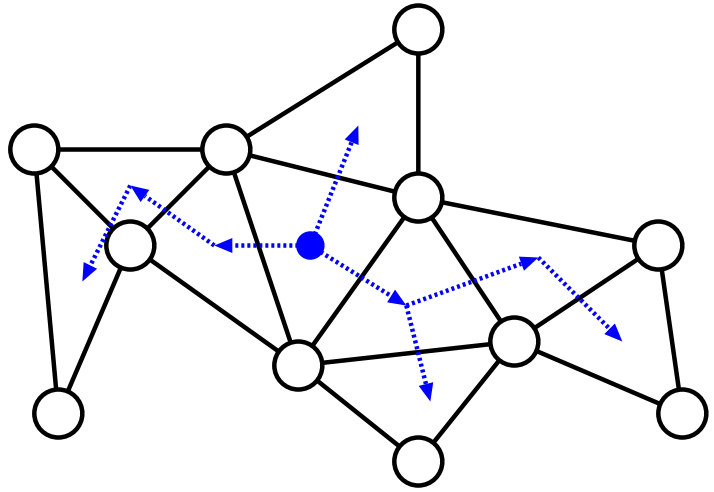
Vine conjecture (examples)



In a vine, every maximal constellation has $n/2$ edges.

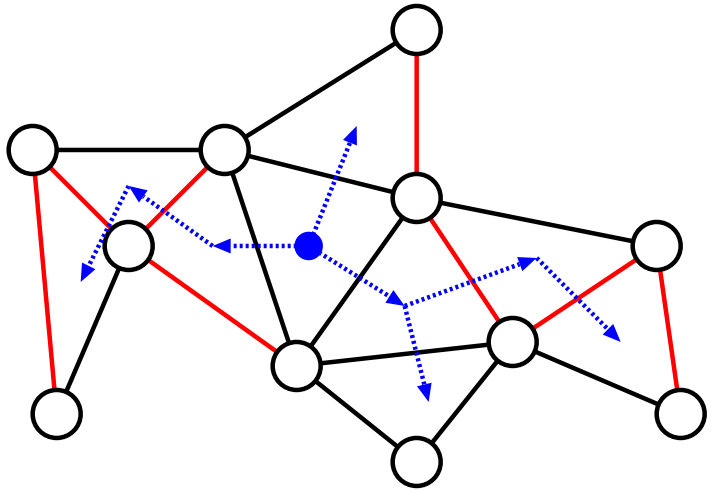
Problem 1: compute $C_k(T, w)$ (hypertrees)

Tree decomposition: tree over hyperedges. Each hyperedge separates the hypertree.



Dynamic programming states: something like $f(h, \text{state}(h))$

Problem 1: compute $C_k(T, w)$ (hypertrees)



representing forest of a windmill farm

tree decomposition

Dynamic programming states: $f(h, i, R, F)$, where

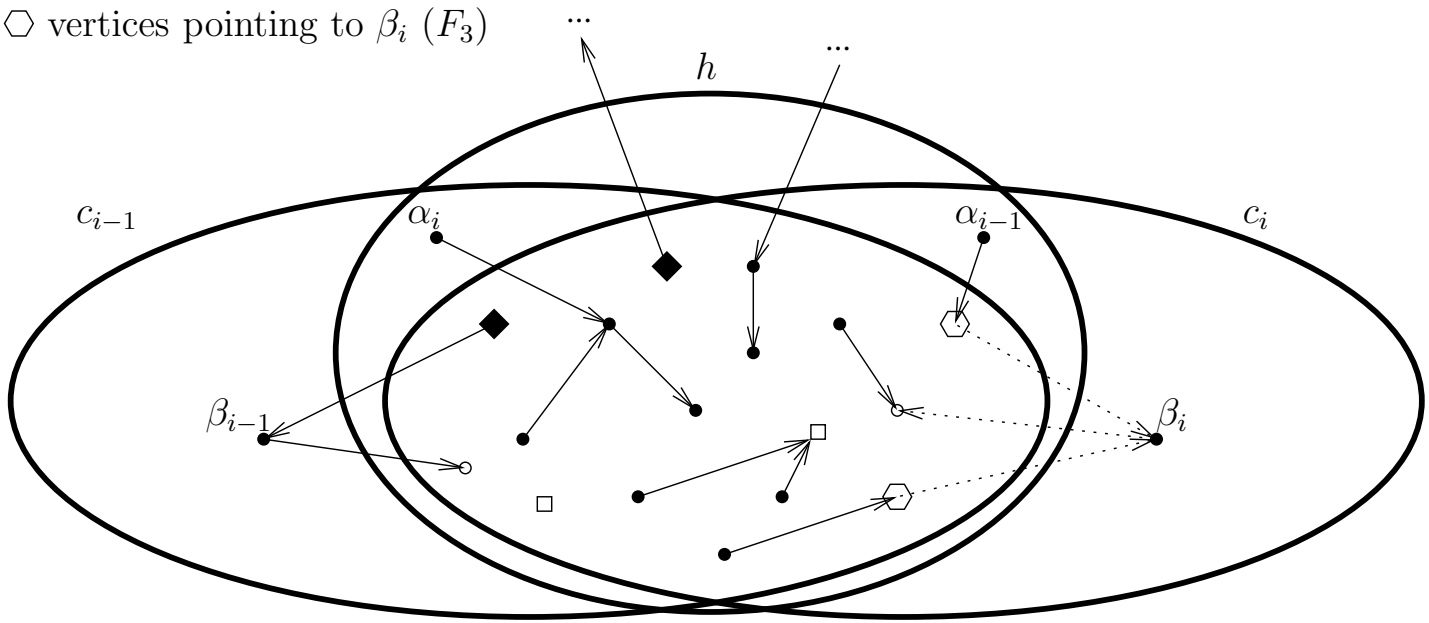
- h: current hyperedge
- i: which child of h to extend R into
- R: representing forest of the windmill farm restricted to h
- F: free vertices

Problem 1: compute $C_k(T, w)$ (hypertrees)

- • vertices that point to some vertex in the representing forest (R)
- ◆ blocked vertices
- ○ vertices free for $c_{i+1}, \ldots, c_n$ (F_1)
- □ vertices free in c_i (F_2)
- ⬡ vertices pointing to β_i (F_3)

——➤ existing pointer

·····➤ new pointer (for which c_i is responsible)

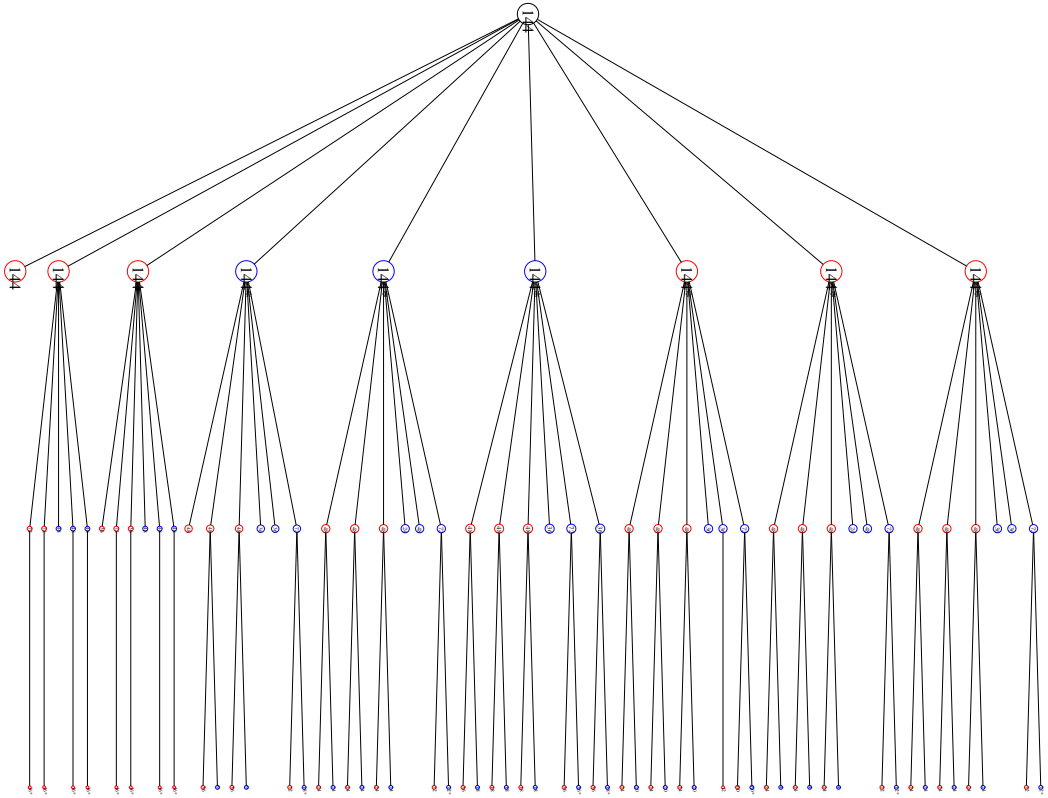


Problem 2: compute $C_k(T)$ (hypertrees)

Use the same tricks as before to convert the dynamic program into a linear program.

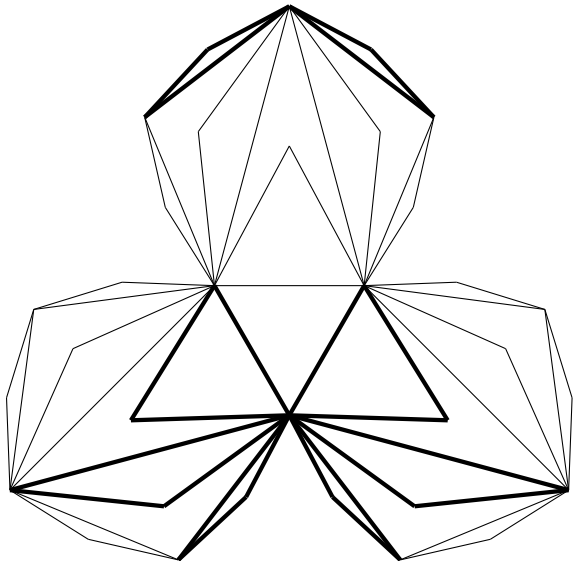
Problem 3: compute C_k (hypertrees)

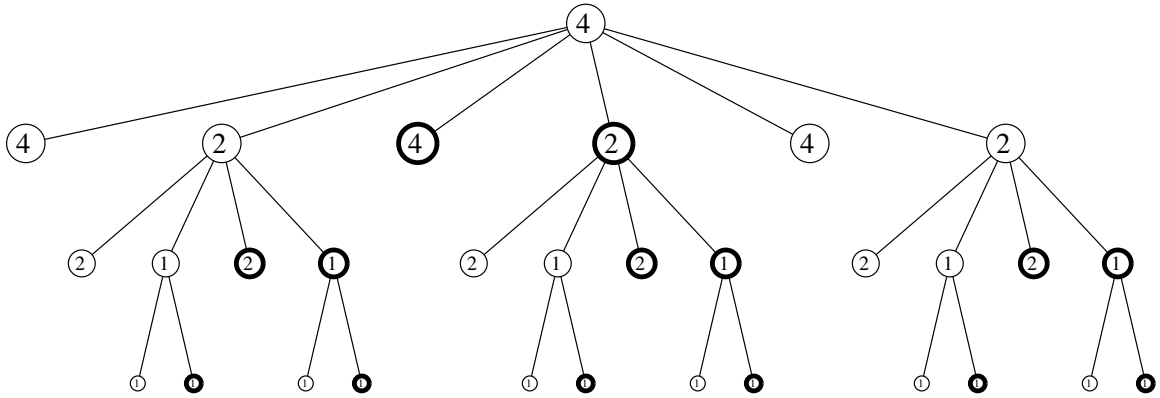
Search over hypertree structures, increasing in size. Resulted in semi-irregular structures and weights. Example of tree decomposition (breadth $3(k+1) = 9$, depth 3):



A particular sequence of weighted hypertrees

Hypertree and tree decomposition of T_d, where d is the depth of the tree decomposition:





Experimental results

k	C_k lower bound	$\lim_{d\to\infty} C_k(T_d)$	C_k upper bound
2	0.166667	$0.2222222 = 2/9$	0.33333
3	0.041667	$0.0953932\ldots$	0.25
4	0.008333	$0.0515625 = 33/640$	0.2
5	0.001389	$0.0258048 = 2016/78125$	0.16667
6	0.000198	$0.0123157\ldots$	0.14286
k	$1/(k+1)!$	$< 1/2^k$??	$1/(k+1)$

Future work

- Windmill coverage analysis:
  - Prove the vine conjecture.
  - Find a analytic closed form bound of $\lim_{d\to\infty} C_k(T_d)$.
  - Better yet, find C_k.

Future work

- Windmill coverage analysis:
  - Prove the vine conjecture.
  - Find a analytic closed form bound of $\lim_{d\to\infty} C_k(T_d)$.
  - Better yet, find C_k.
- Algorithms: find new methods of approximating the maximum hypertree.

Future work

- Windmill coverage analysis:
  - Prove the vine conjecture.
  - Find a analytic closed form bound of $\lim_{d\to\infty} C_k(T_d)$.
  - Better yet, find C_k.
- Algorithms: find new methods of approximating the maximum hypertree.
- Applications: apply bounded tree-width Markov networks to machine learning tasks.