Parameter Estimation: Cracking Incomplete Data

Khaled S. Refaat

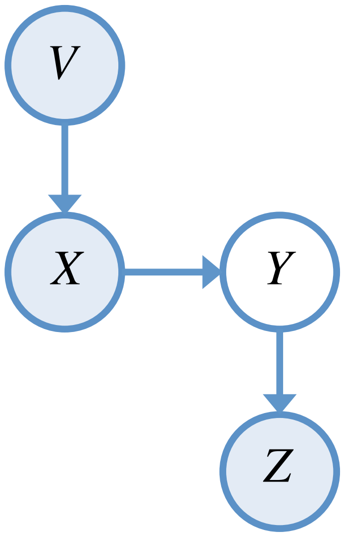
Collaborators: Arthur Choi and Adnan Darwiche

Agenda

- Learning Graphical Models
- Complete vs. Incomplete Data
- Exploiting Data for Decomposition
- EDML vs. EM

Learning Graphical Models

Learning Graphical Model Parameters



V	X	Y	Z
False	True	?	False
True	False	?	True
True	True	?	False

Our goal is to find parameter estimates that maximize the likelihood:

$$L(\theta|\mathcal{D}) = \prod_{i=1}^{N} Pr_{\theta}(\mathbf{d}_i)$$

$$\theta^{\star} = \operatorname*{argmax}_{\theta} L(\theta|\mathcal{D})$$

Complete vs. Incomplete Data

Complete Data

V	X	Y	Z
False	True	True	False
True	False	False	True
True	True	True	False

Incomplete Data

V	X	Y	Z
False	True	?	False
?	False	?	True
True	True	?	False

Complete Data

V	X	Y	Z
False	True	True	False
True	False	False	True
True	True	True	False

Incomplete Data

V	X	Y	Z
False	True	?	False
?	False	?	True
True	True	?	False

closed-form or a convex optimization problem

Complete Data

V	X	Y	Z
False	True	True	False
True	False	False	True
True	True	True	False

closed-form or a convex optimization problem

Incomplete Data

V	X	Y	Z
False	True	?	False
?	False	?	True
True	True	?	False

hard non-convex optimization problem

Complete Data

V	X	Y	Z
False	True	True	False
True	False	False	True
True	True	True	False

closed-form or a convex optimization problem

Incomplete Data

V	X	Y	Z
False	True	?	False
?	False	?	True
True	True	?	False

hard non-convex optimization problem

Incomplete Data

V	X	Y	Z
False	True	?	False
?	False	?	True
True	True	?	False

Incomplete Data

Fully-observed variables

V	X	Y	Z
False	True	?	False
?	False	?	True
True	True	?	False

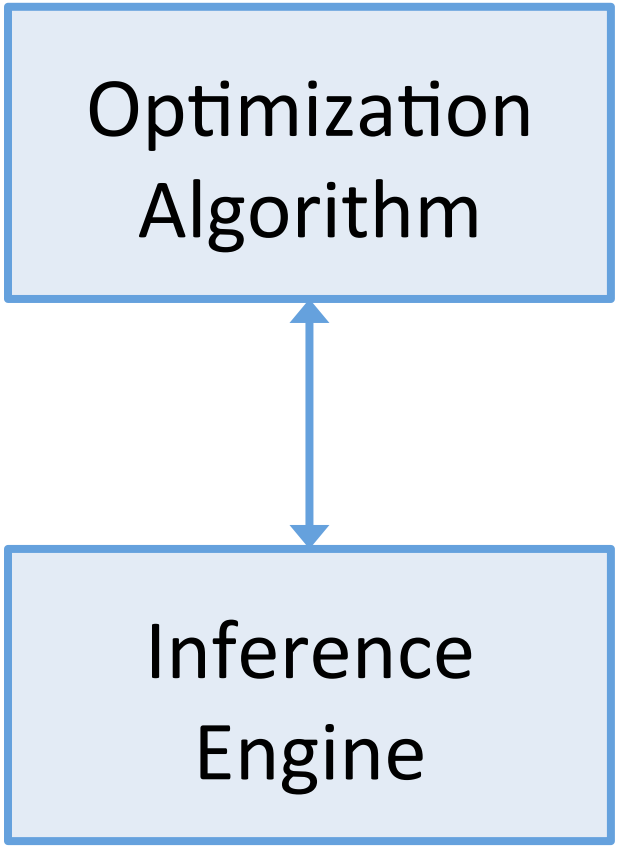

Incomplete Data

Hidden variables

V	X	Y	Z
False	True	?	False
?	False	?	True
True	True	?	False

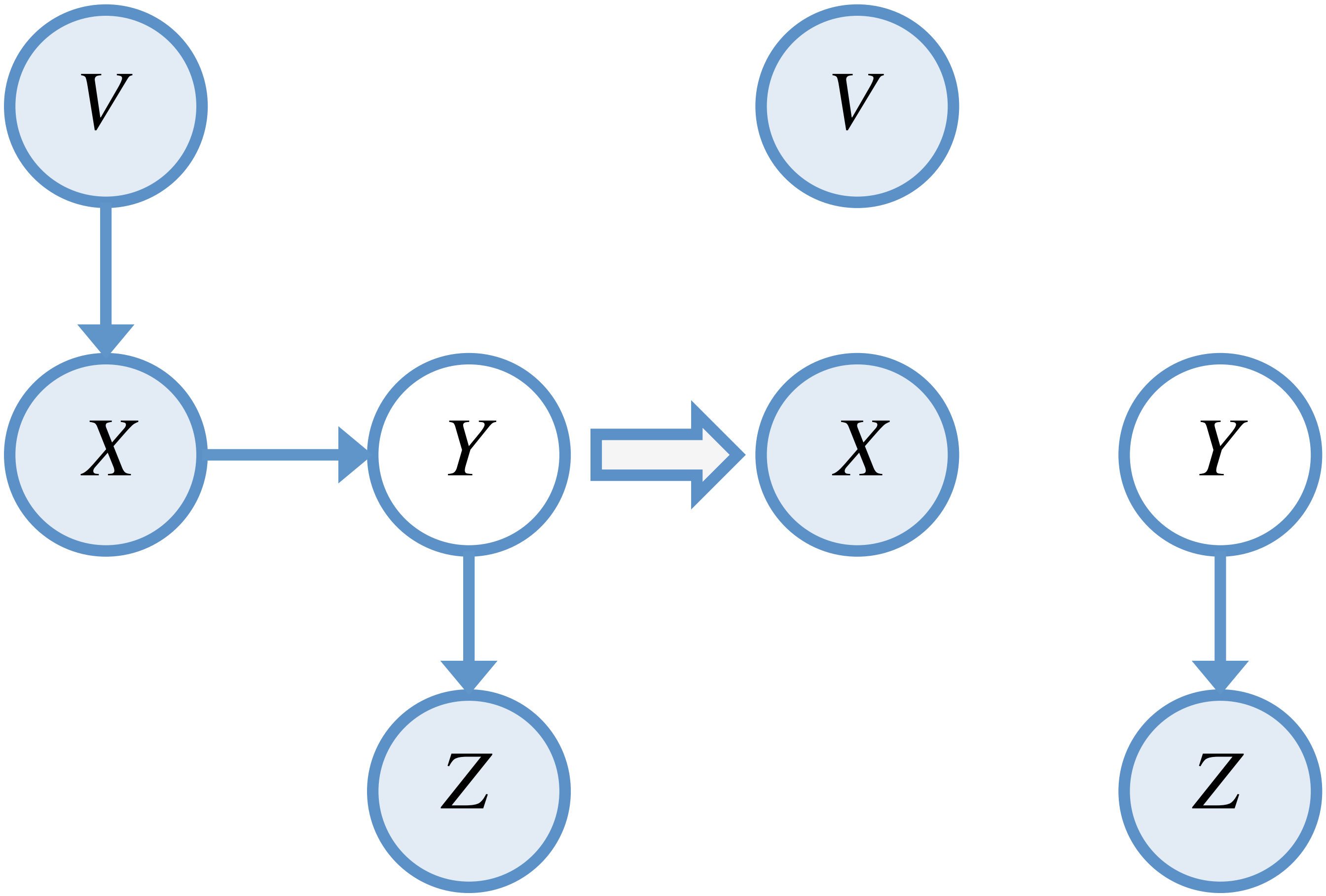
Exploiting Data for Decomposition

Optimization



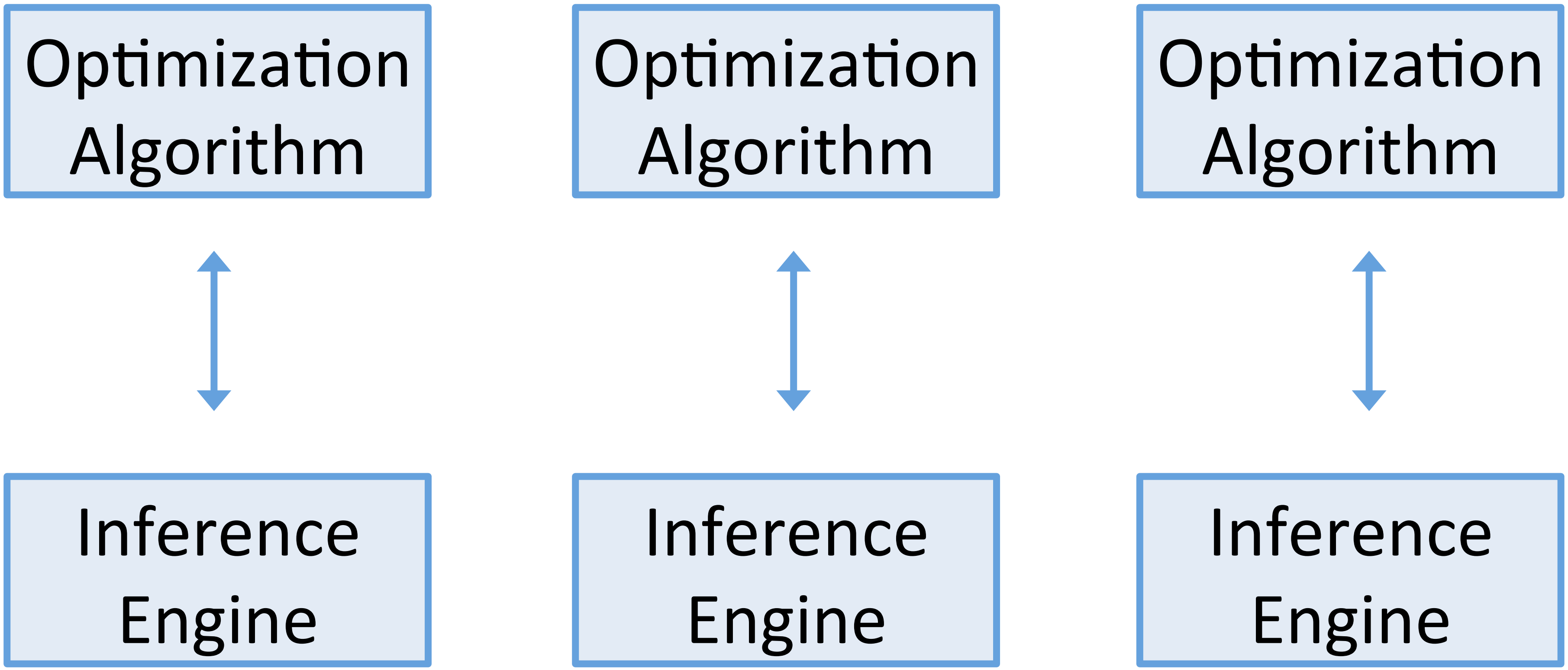
The optimization algorithm (e.g. EM, EDML, Gradient Method) calls the inference engine with every unique data example at each iteration.

Inference Decomposition Techniques



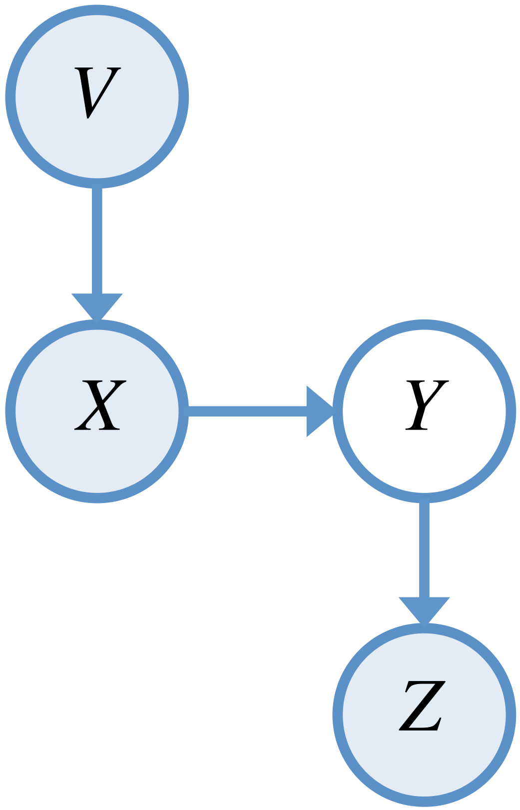
Prune edges outgoing from observed nodes before computing probabilities.

Main Idea (NIPS'14)



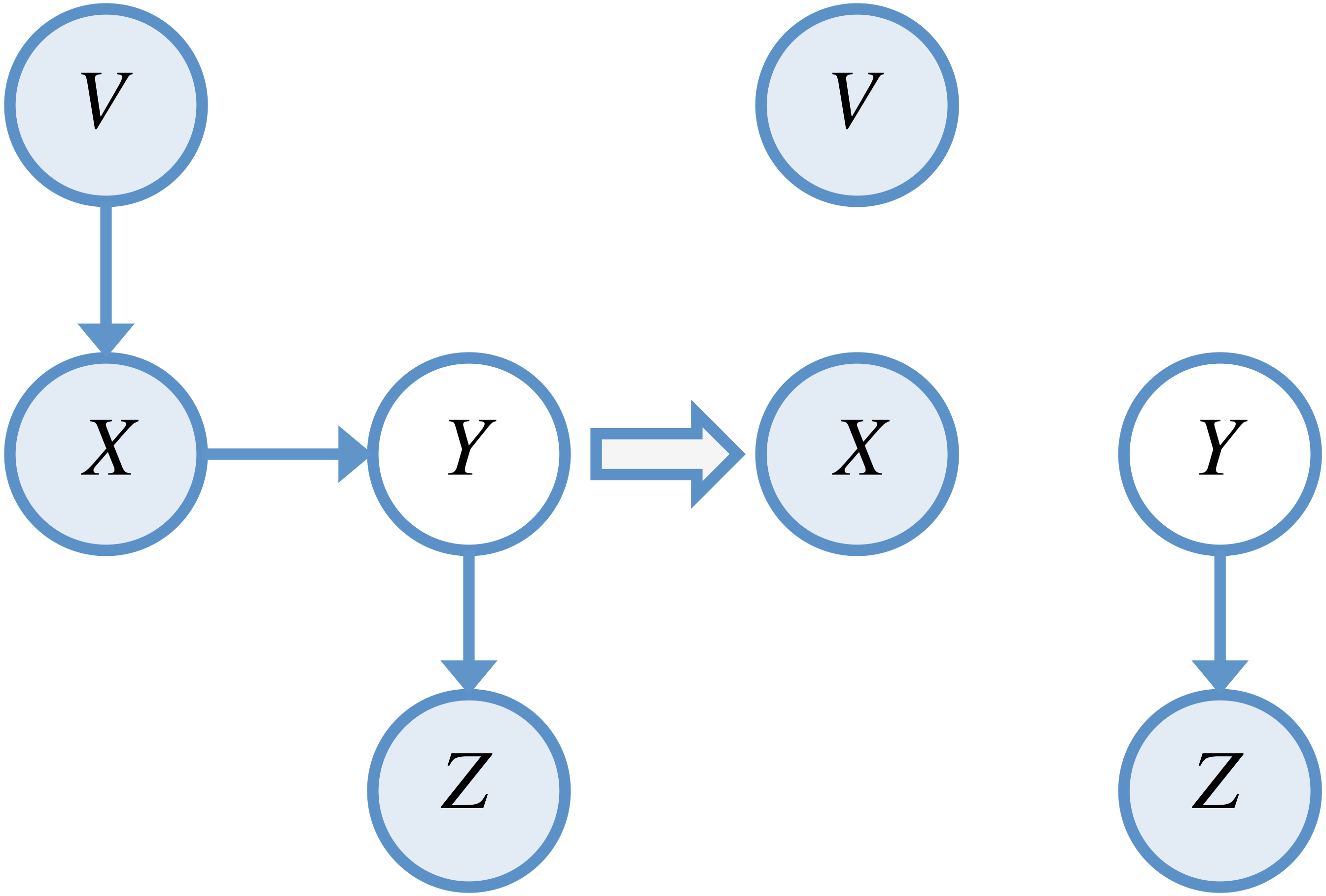
We decompose the optimization problem itself to get decomposed convergence and data compression.

Learning from Incomplete Data



V	X	Y	Z
False	True	?	False
True	False	?	True
True	True	?	False

Decomposing the Optimization Problem

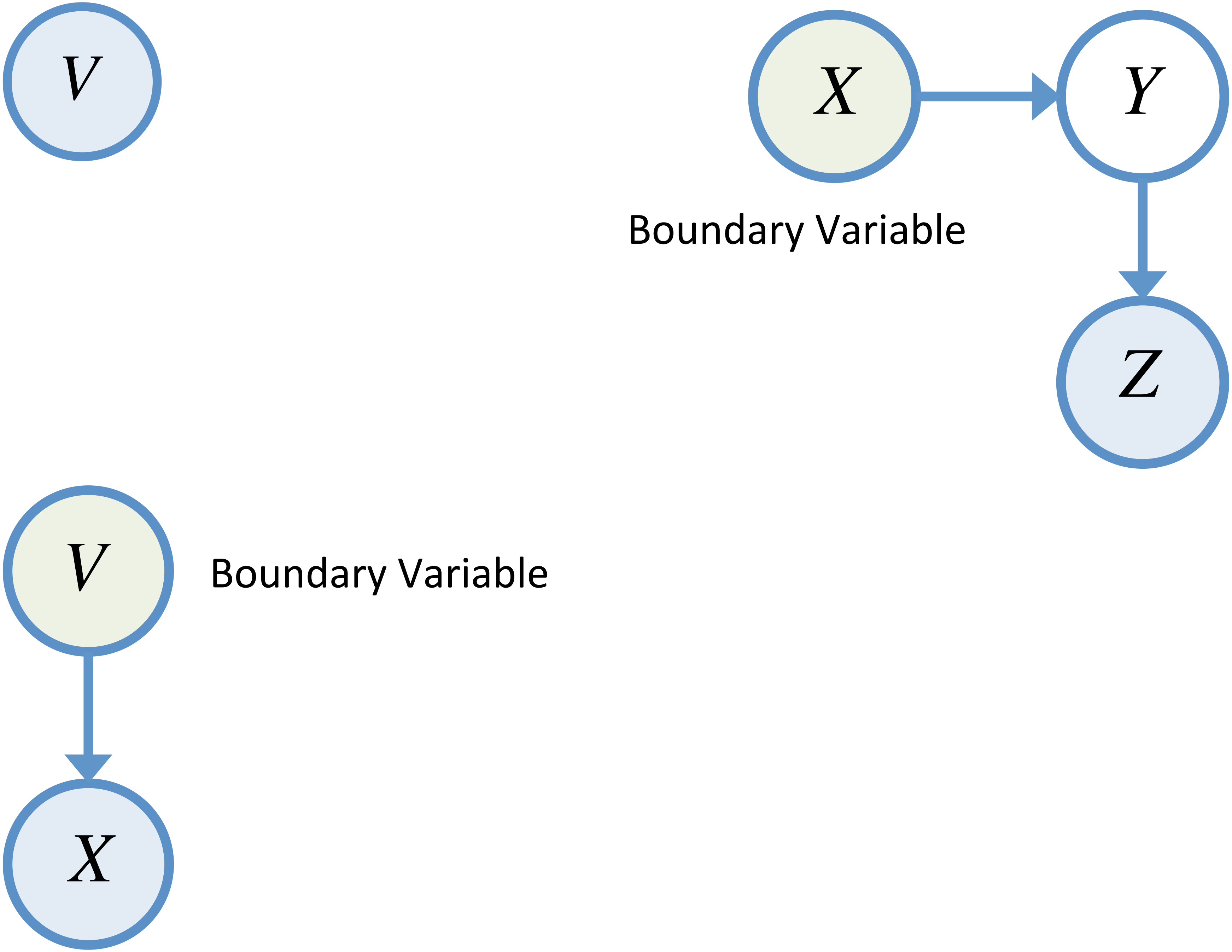


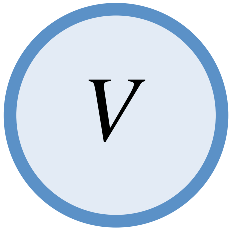
Get three components:

$$\mathbf{S}_1 = \{V\}$$
$$\mathbf{S}_2 = \{X\}$$
$$\mathbf{S}_3 = \{Y, Z\}$$

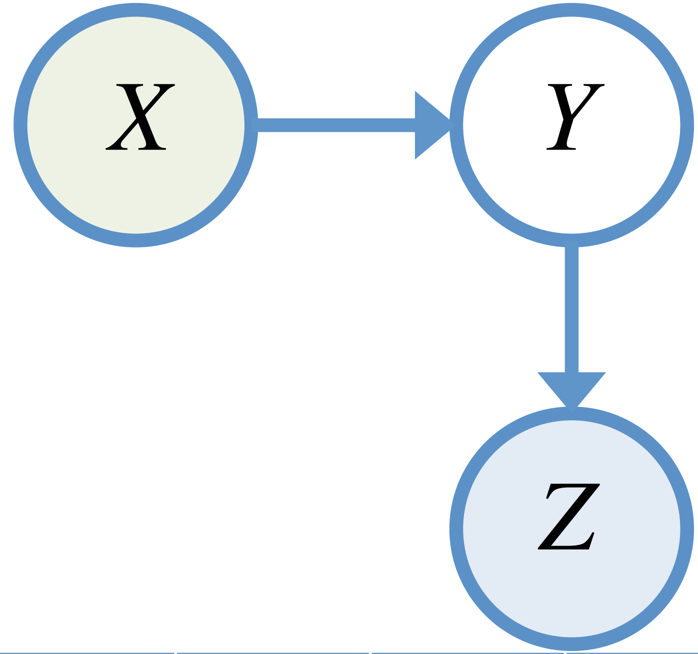
The components of a network partition its parameters into groups:

$$\begin{aligned}
\mathbf{S}_1 &: \quad \{\theta_v, \theta_{\overline{v}}\} \\
\mathbf{S}_2 &: \quad \{\theta_{x|v}, \theta_{\overline{x}|v}, \theta_{x|\overline{v}}, \theta_{\overline{x}|\overline{v}}\} \\
\mathbf{S}_3 &: \quad \{\theta_{y|x}, \theta_{\overline{y}|x}, \theta_{y|\overline{x}}, \theta_{\overline{y}|\overline{x}}, \theta_{z|y}, \theta_{\overline{z}|y}, \theta_{z|\overline{y}}, \theta_{\overline{z}|\overline{y}}\}.
\end{aligned}$$

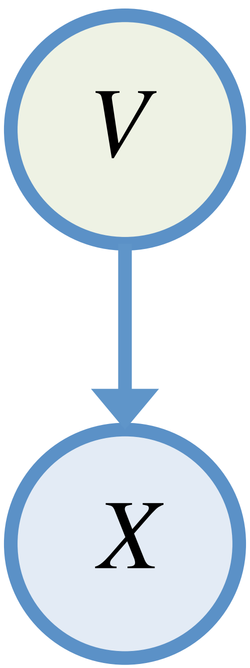

V
X
Y
Z
Boundary Variable
V
Boundary Variable
X



V	Count
False	1
True	2

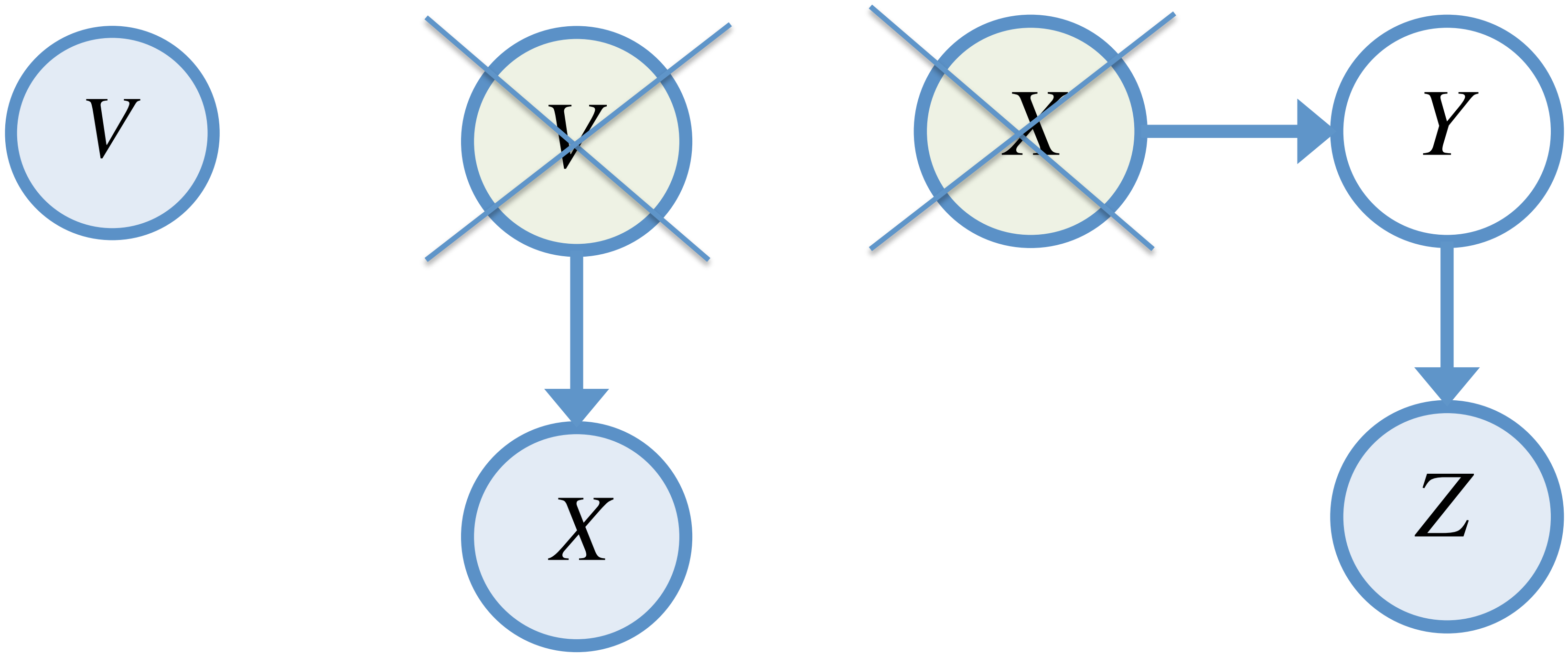


X	Y	Z	Count
True	?	False	2
False	?	True	1



V	X	Count
False	True	1
True	False	1
True	True	1

Learned Parameters



Theorem (NIPS'14)

- Any stationary points for the sub-problems combine to create a stationary point for the original problem.

Theorem (NIPS'14)

- Any stationary points for the sub-problems combine to create a stationary point for the original problem.
- Every stationary point for the original problem induces stationary points for the sub-problems.

Experimental Setting

Experimental Setting

- EM: uses an inference engine that decomposes inference.

Experimental Setting

- EM: uses an inference engine that decomposes inference.
- D-EM: decomposes the optimization problem itself, solves each sub-problem using EM, and combines the solutions.

The Computational Benefit of Decomposition

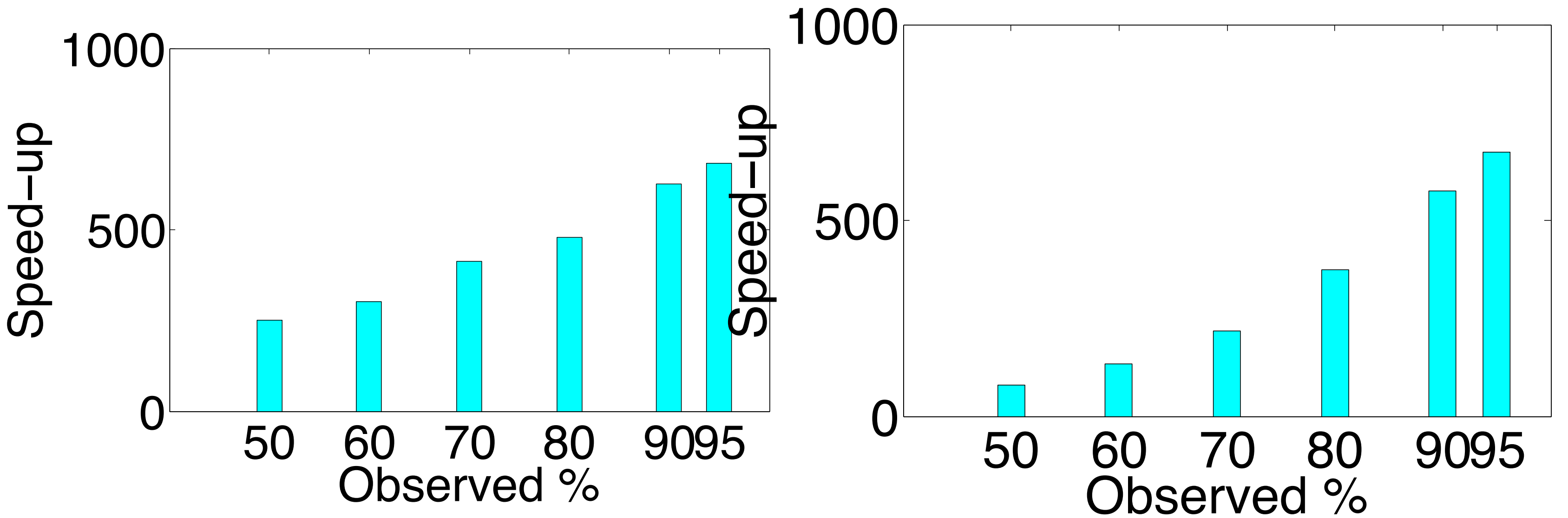


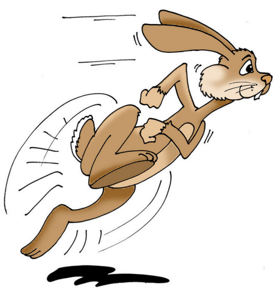
Figure: Speed-up of D-EM over EM on chain networks: three chains (180, 380, and 500 variables) (left), and tree networks (63, 127, 255, and 511 variables) (right).

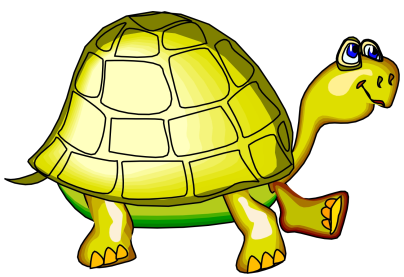
Observed %	Network	Speed-up D-EM	Network	Speed-up D-EM	Network	Speed-up D-EM
95.0%	alarm	267.67x	diagnose	43.03x	andes	155.54x
90.0%	alarm	173.47x	diagnose	17.16x	andes	52.63x
80.0%	alarm	115.4x	diagnose	11.86x	andes	14.27x
70.0%	alarm	87.67x	diagnose	3.25x	andes	2.96x
60.0%	alarm	92.65x	diagnose	3.48x	andes	0.77x
50.0%	alarm	12.09x	diagnose	3.73x	andes	1.01x
95.0%	win95pts	591.38x	water	811.48x	pigs	235.63x
90.0%	win95pts	112.57x	water	110.27x	pigs	37.61x
80.0%	win95pts	22.41x	water	7.23x	pigs	34.19x
70.0%	win95pts	17.92x	water	1.5x	pigs	16.23x
60.0%	win95pts	4.8x	water	2.03x	pigs	4.1x
50.0%	win95pts	7.99x	water	4.4x	pigs	3.16x

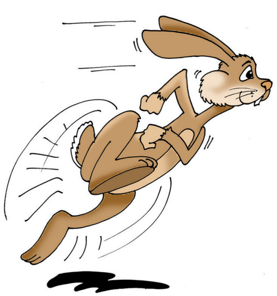
Table: Speed-up of D-EM over EM on UAI networks.

Reasons for Speed-up

Decomposed Convergence







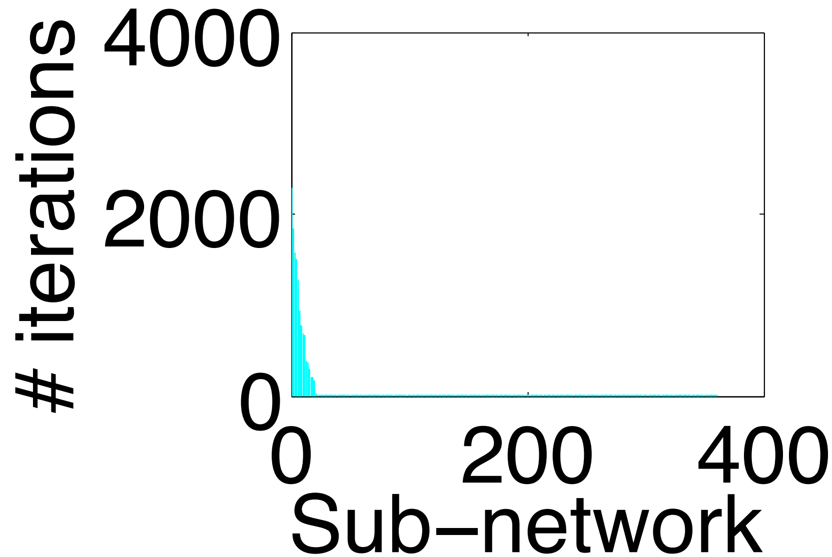
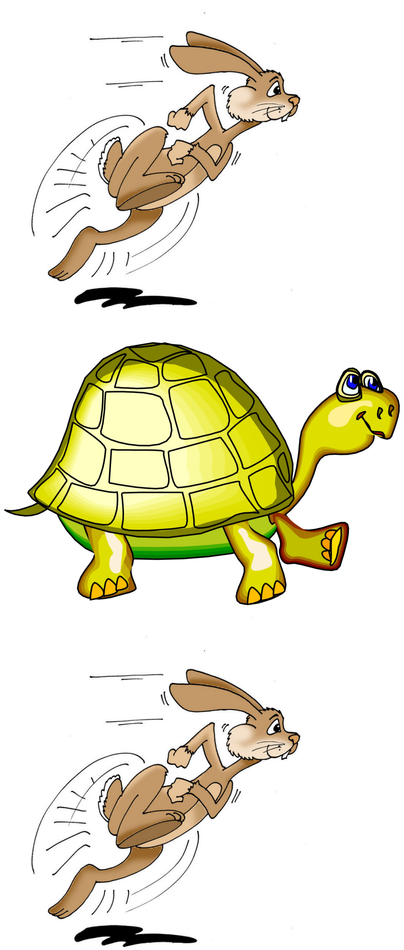


Figure: Graph showing the number of iterations required by each sub-network sorted descendingly.

Decomposed Convergence



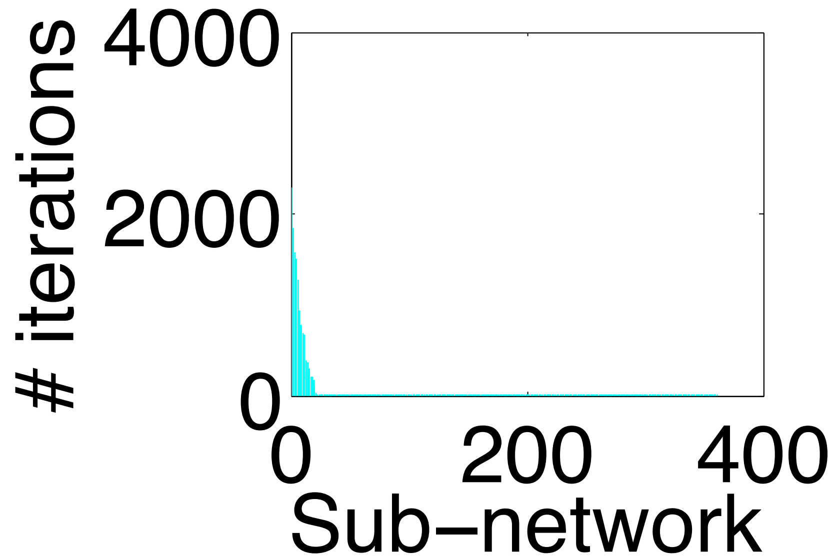
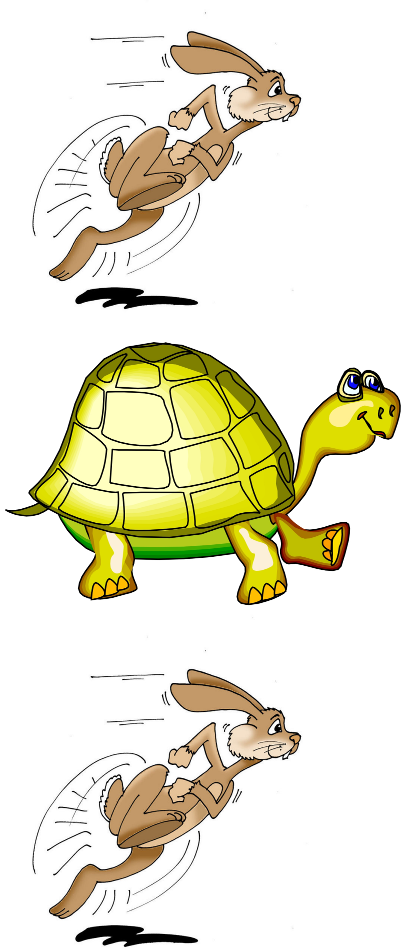


Figure: Graph showing the number of iterations required by each sub-network sorted descendingly.

Decomposed Convergence



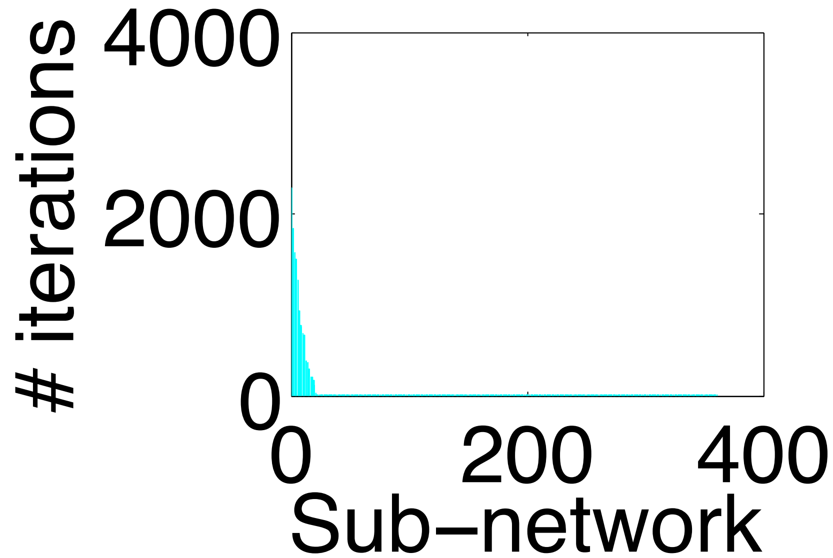
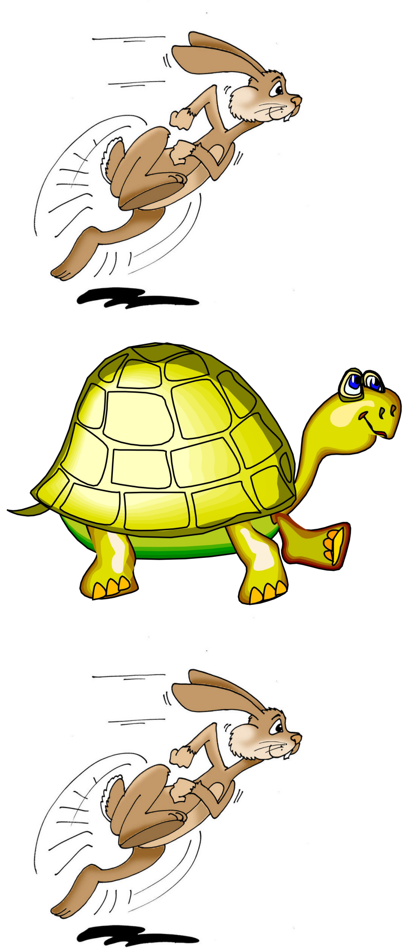


Figure: Graph showing the number of iterations required by each sub-network sorted descendingly.

Decomposed Convergence



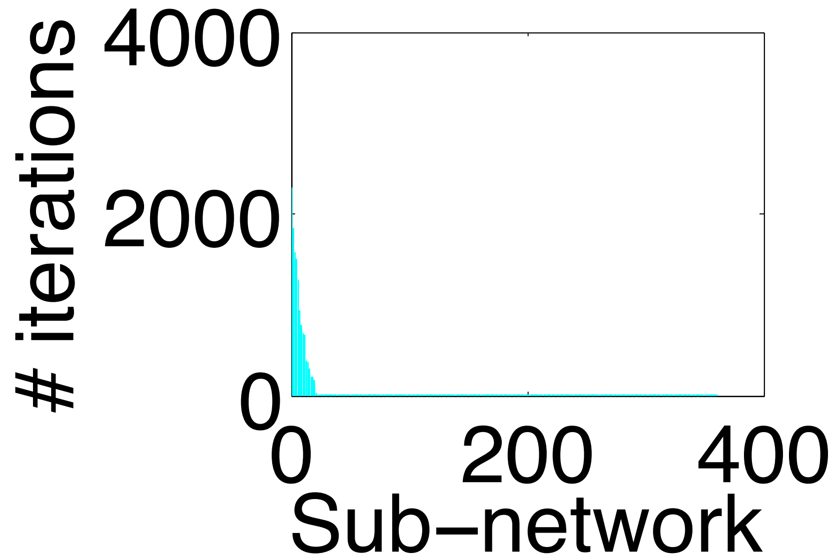
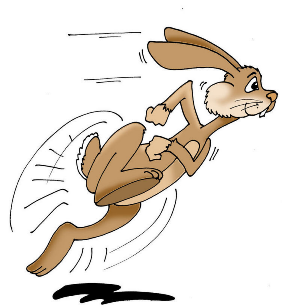
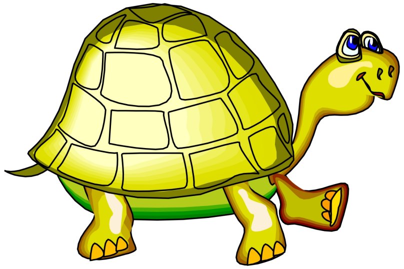
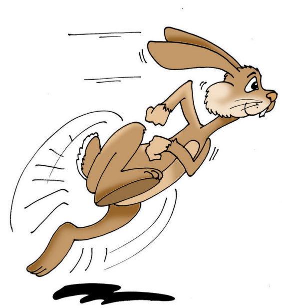


Figure: Graph showing the number of iterations required by each sub-network sorted descendingly.

Decomposed Convergence







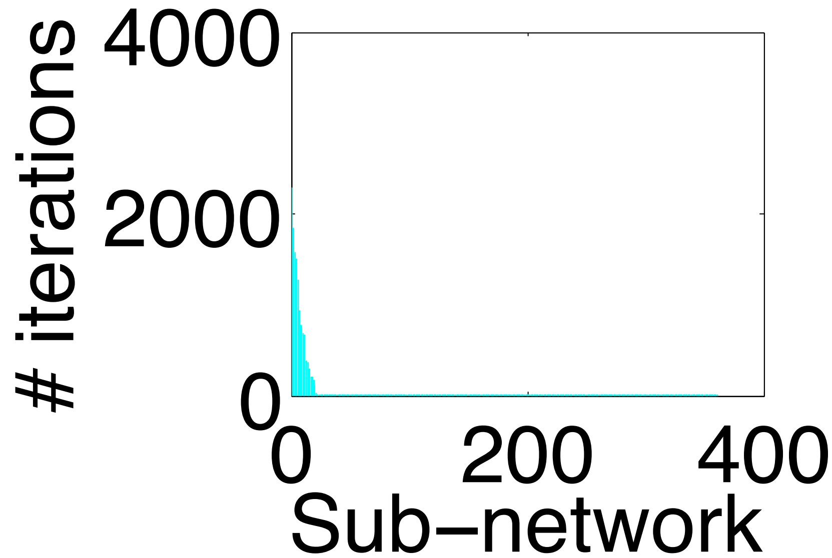
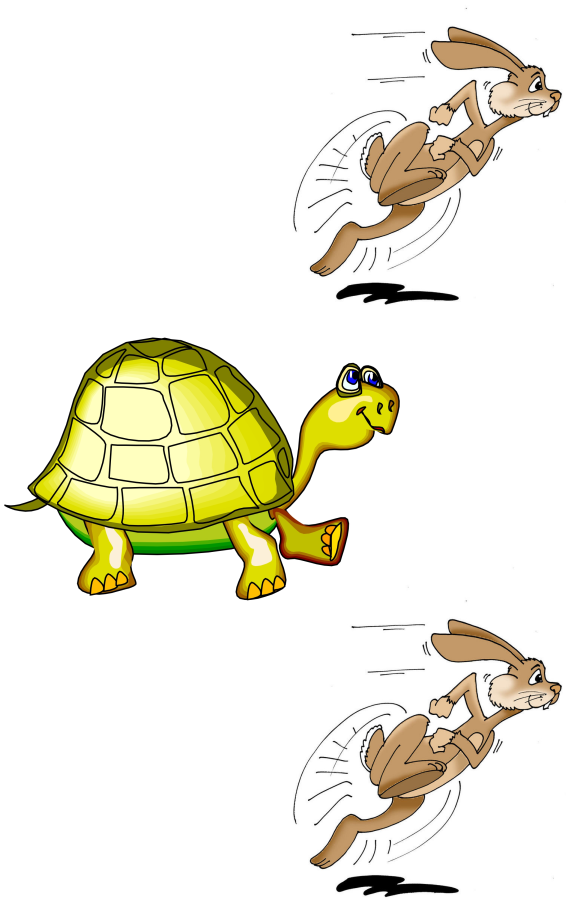


Figure: Graph showing the number of iterations required by each sub-network sorted descendingly.

Decomposed Convergence



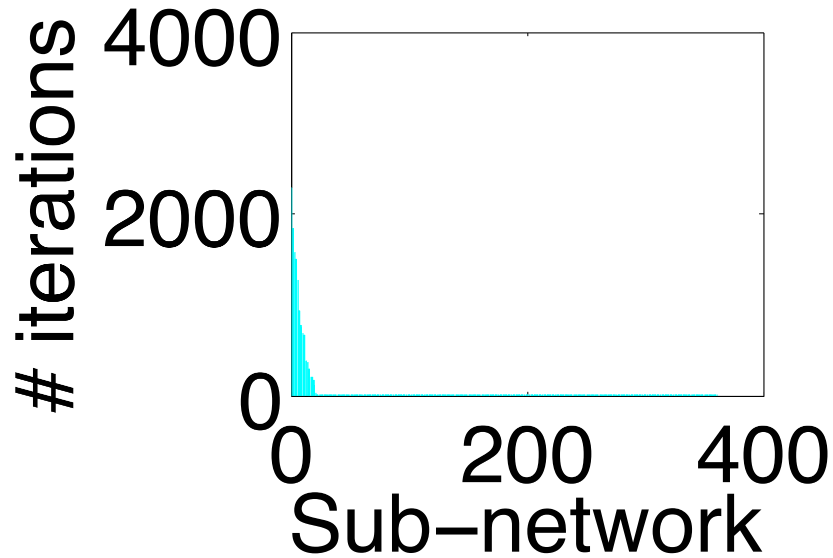
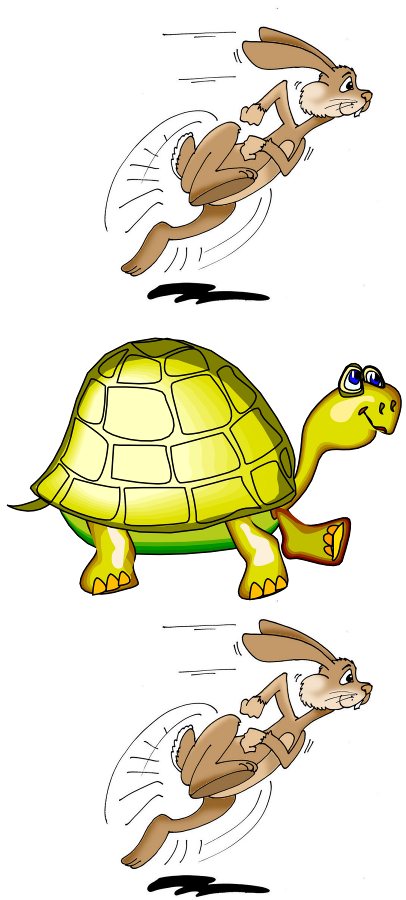


Figure: Graph showing the number of iterations required by each sub-network sorted descendingly.

Decomposed Convergence



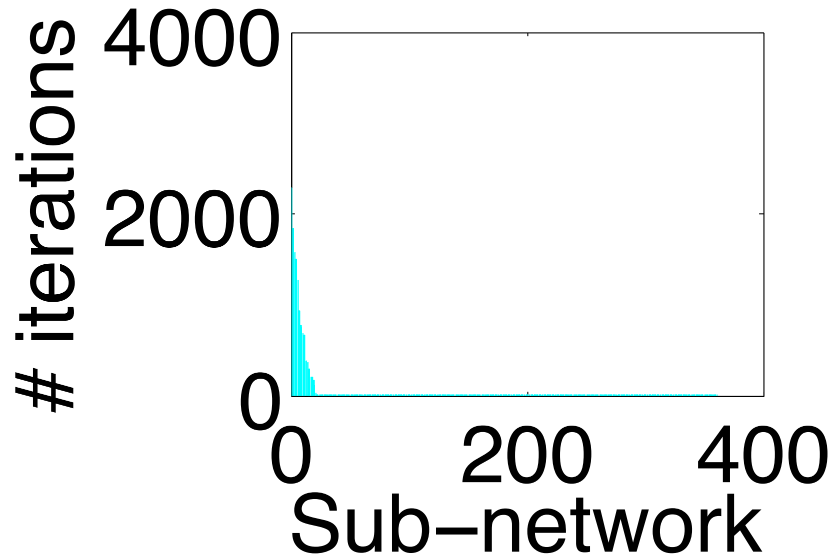
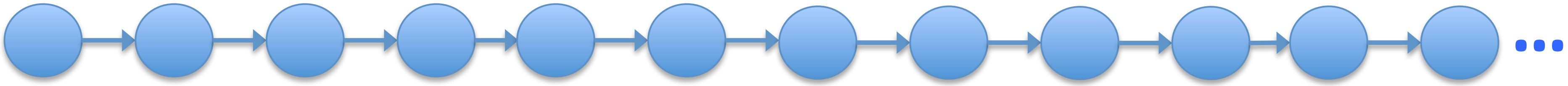
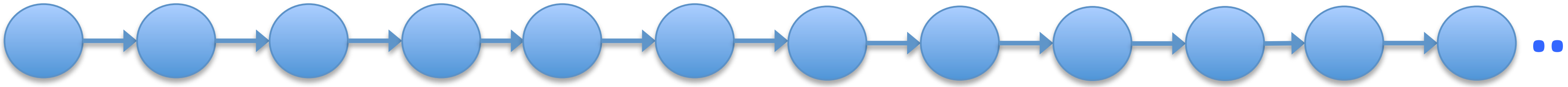


Figure: Graph showing the number of iterations required by each sub-network sorted descendingly.

Data Compression

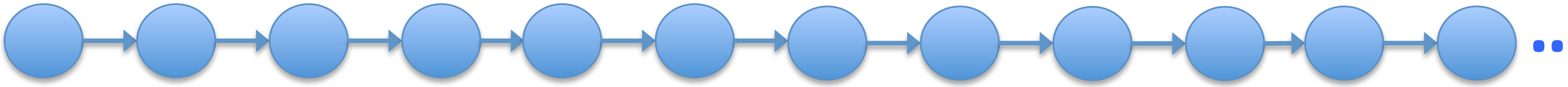


Data Compression

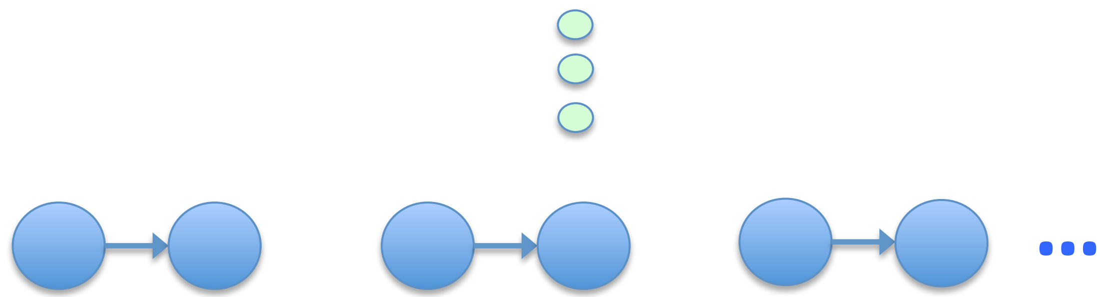


A	B	C	D	E	F	G	H	I	J	K	L	…

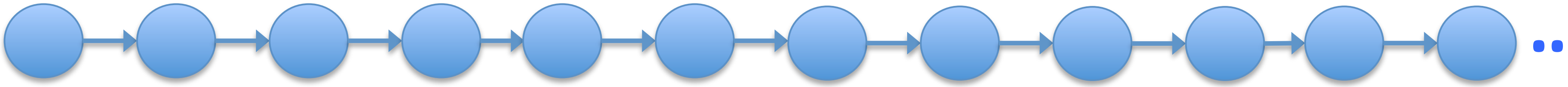
Data Compression



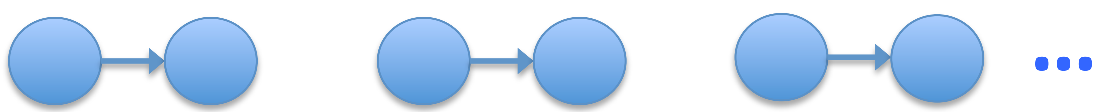
A	B	C	D	E	F	G	H	I	J	K	L	...



Data Compression



A	B	C	D	E	F	G	H	I	J	K	L	…



A	B	Count
True	True	1000
True	False	5000
False	True	3000
False	False	8000

Data Compression

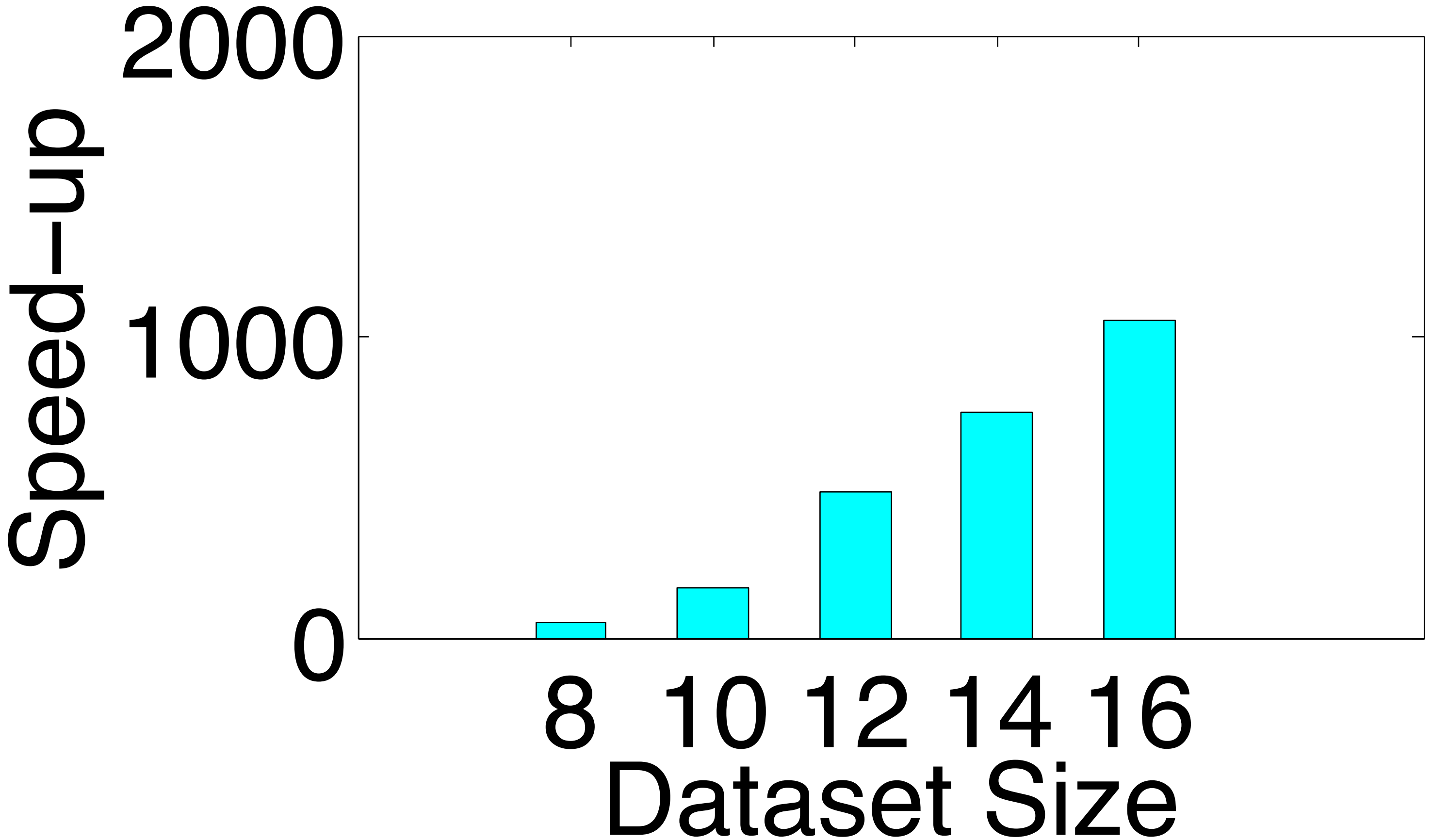
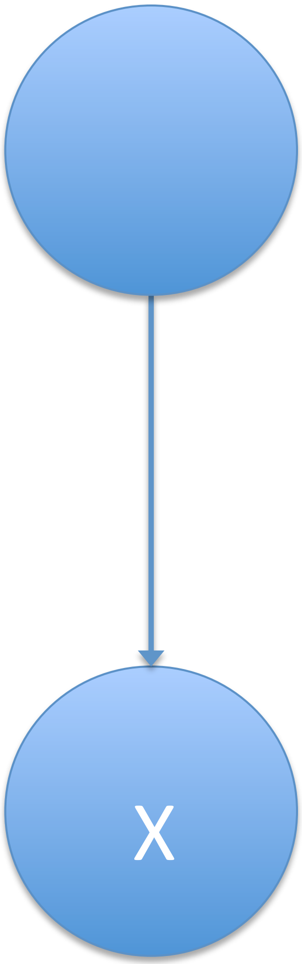


Figure: Speed-up of D-EM over EM as a function of dataset size **(log-scale)**.

EDML vs. EM

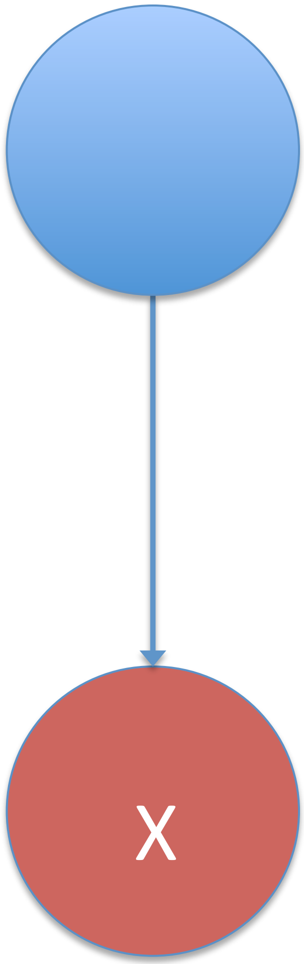
Soft Evidence

Hard Evidence



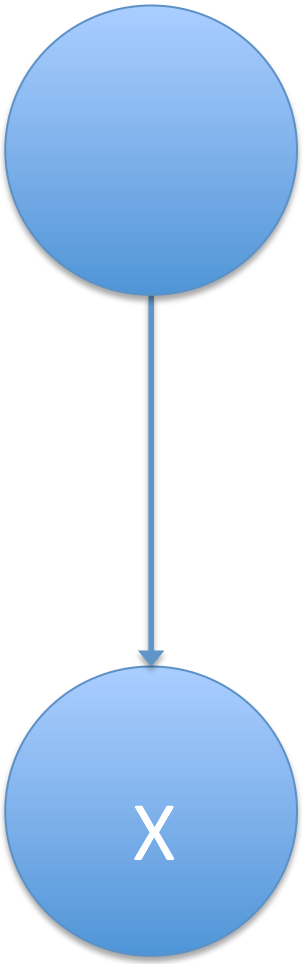
$$X \in \{S_1, S_2, S_3\}$$

Hard Evidence



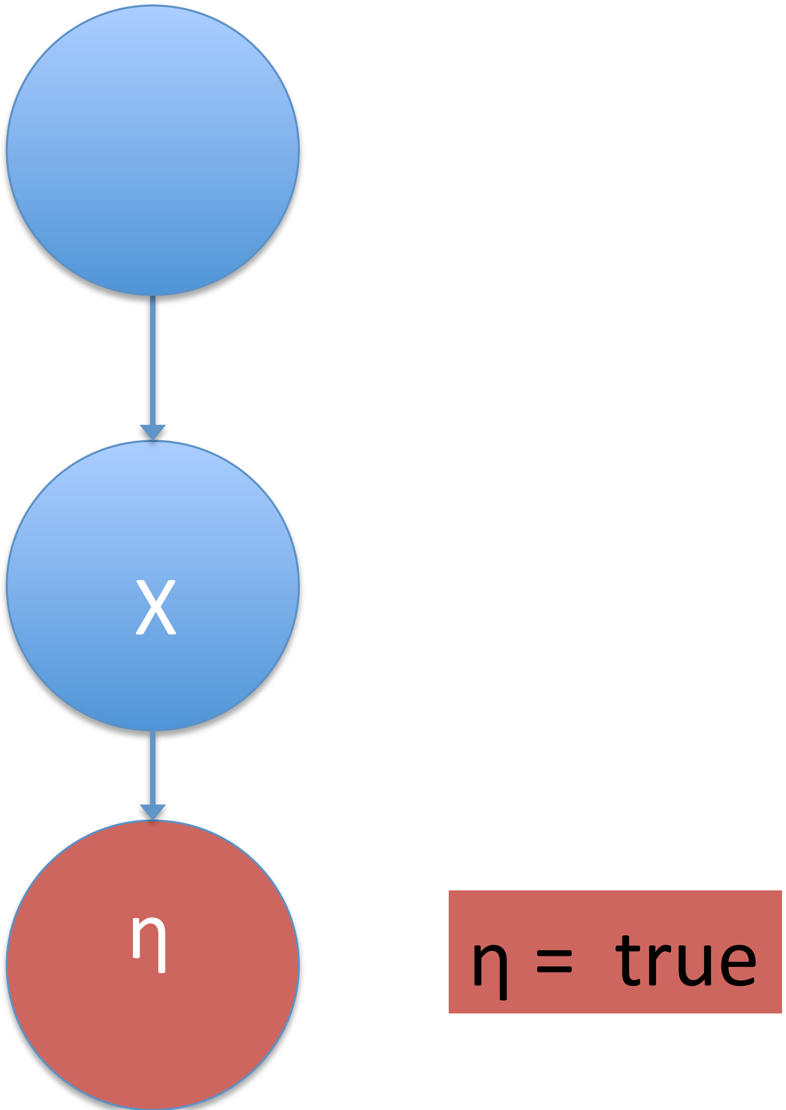
$$X = S_1$$

Soft Evidence



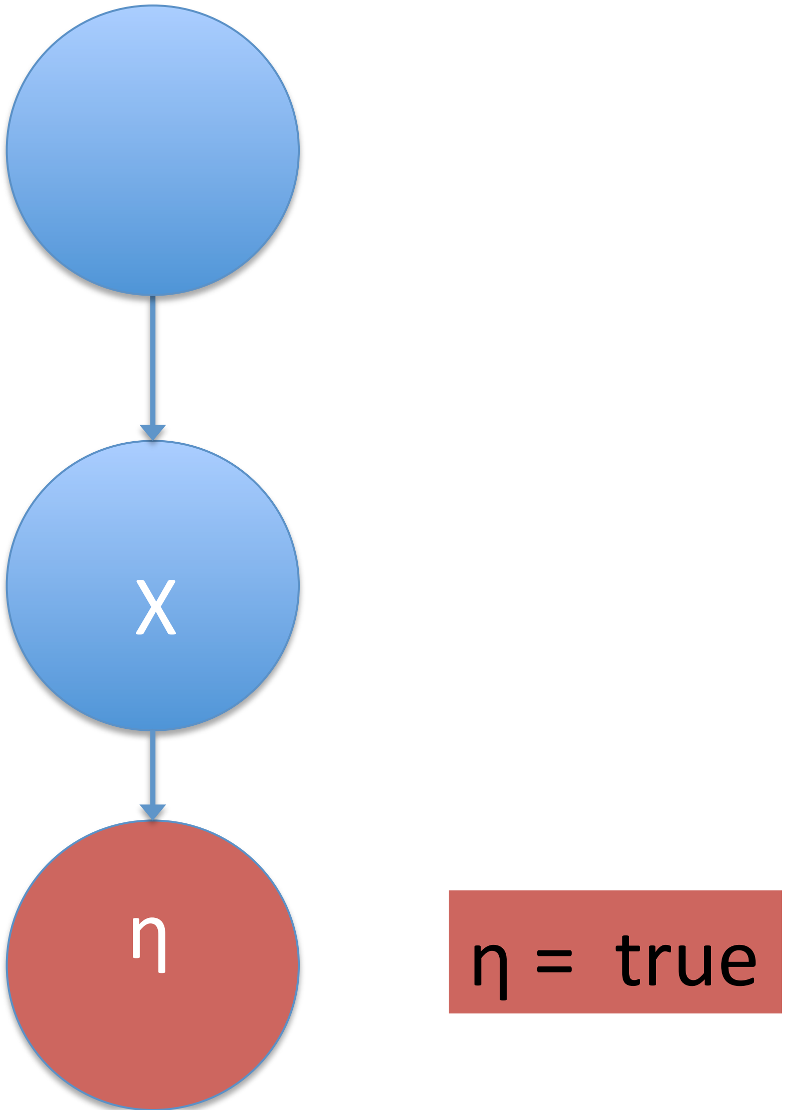
$X = S_1$ with some probability

Soft Evidence



Soft Evidence

η	X	p(η | X)
true	S_1	λ_1
true	S_2	λ_2
true	S_3	λ_3



Edge Deletion

Edge Deletion (cont.)

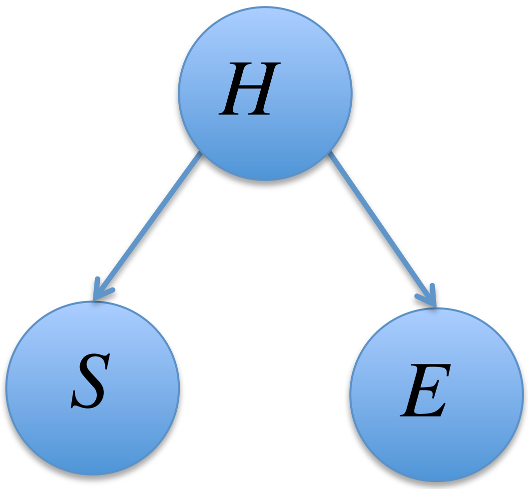
Choi *et al* 2006

Assert Soft Evidence

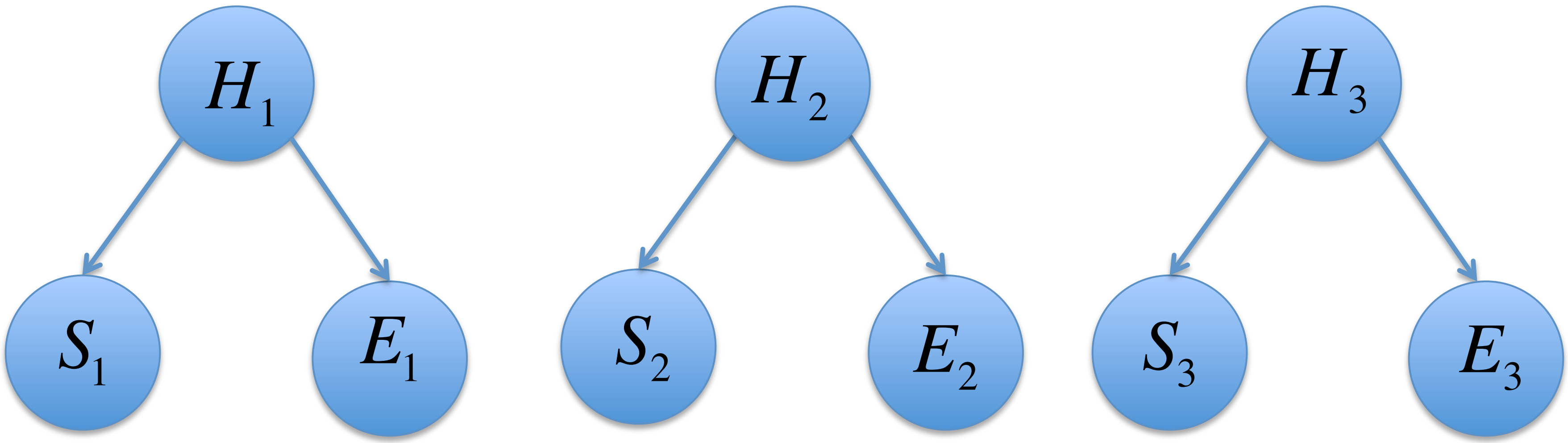
Problem Definition

- Original Bayesian Network:

H	S	E
?	true	?
true	?	?
?	?	true

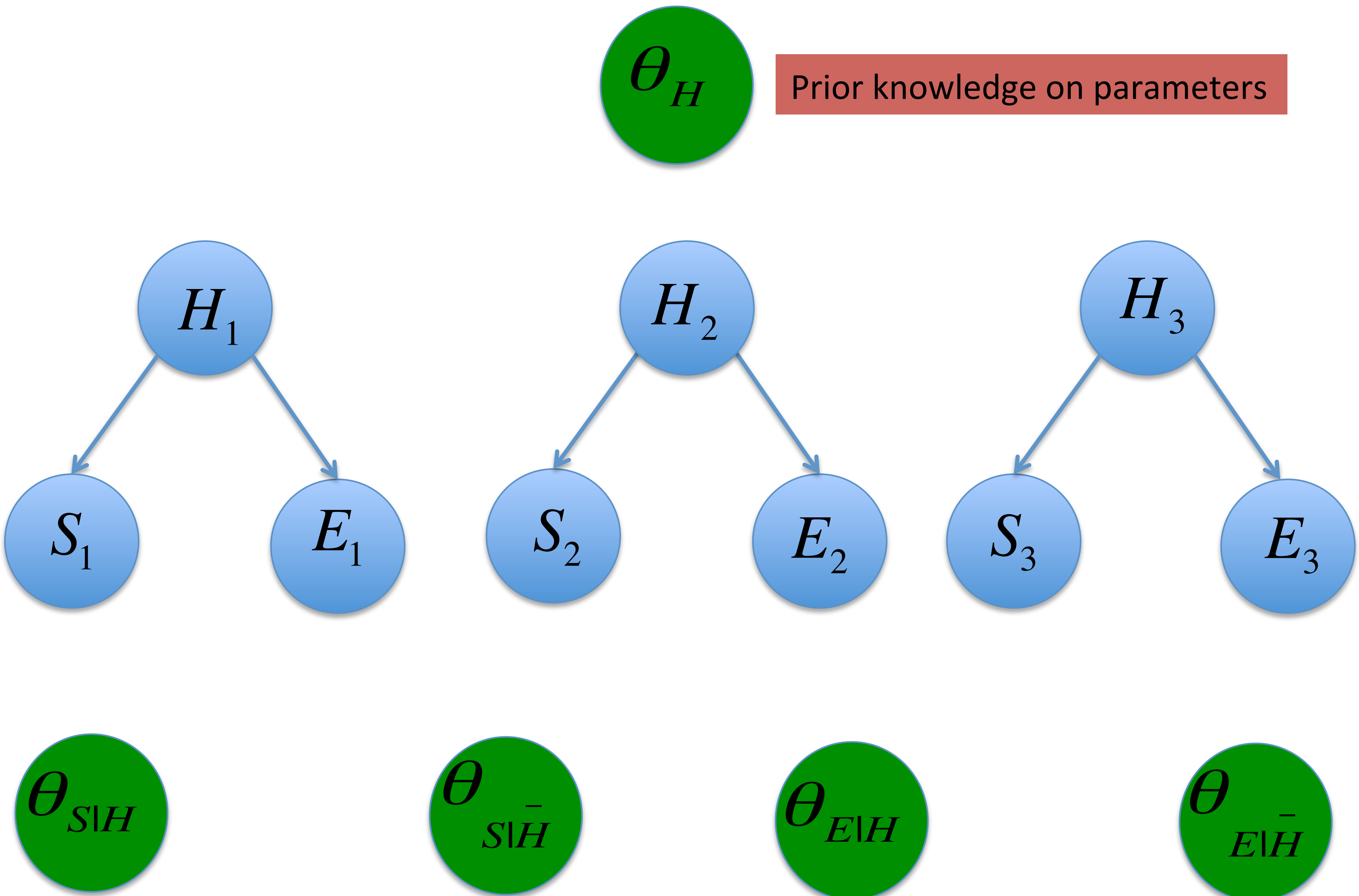


Meta Network Creation

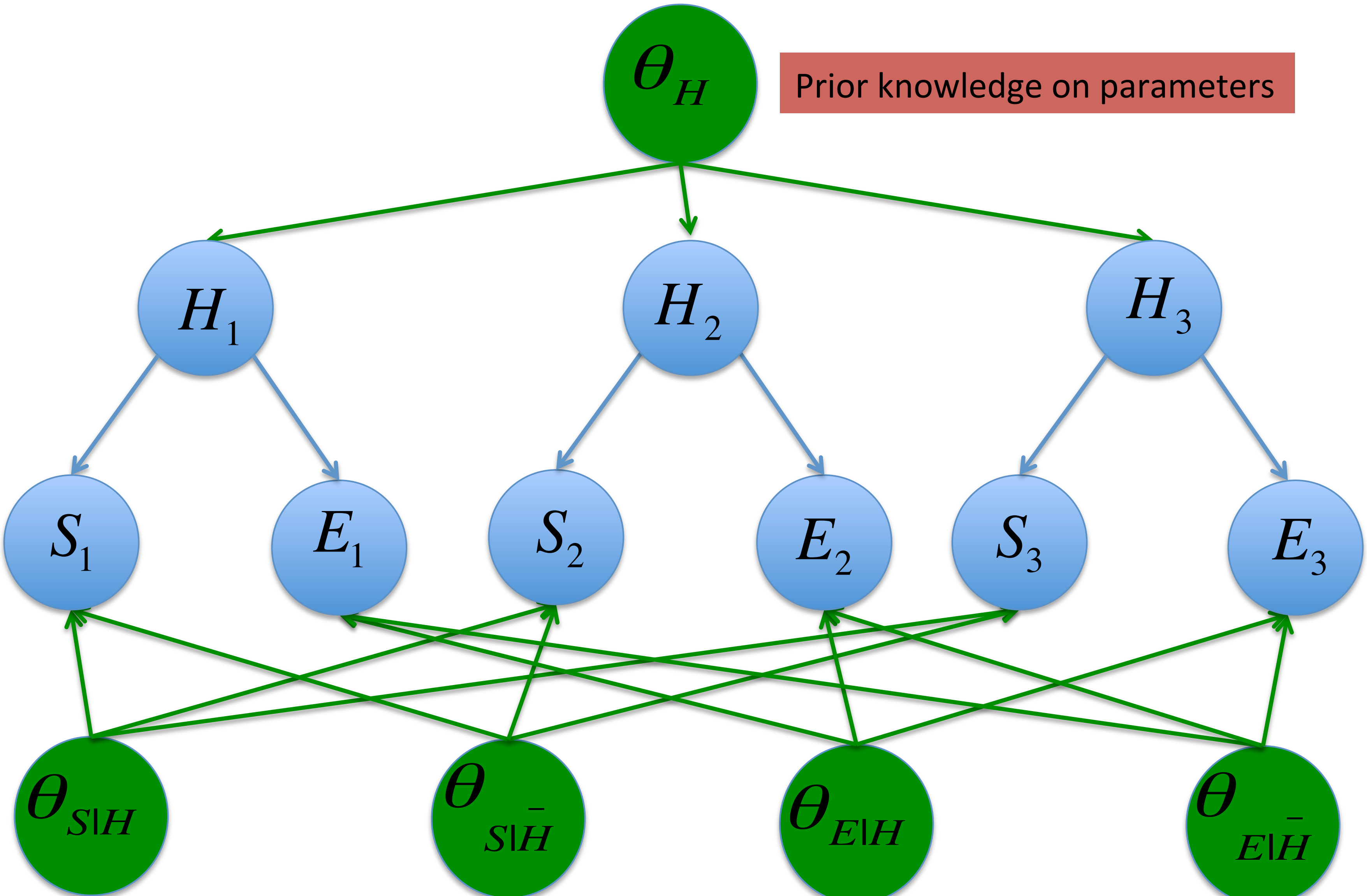


Example 1 Example 2 Example 3

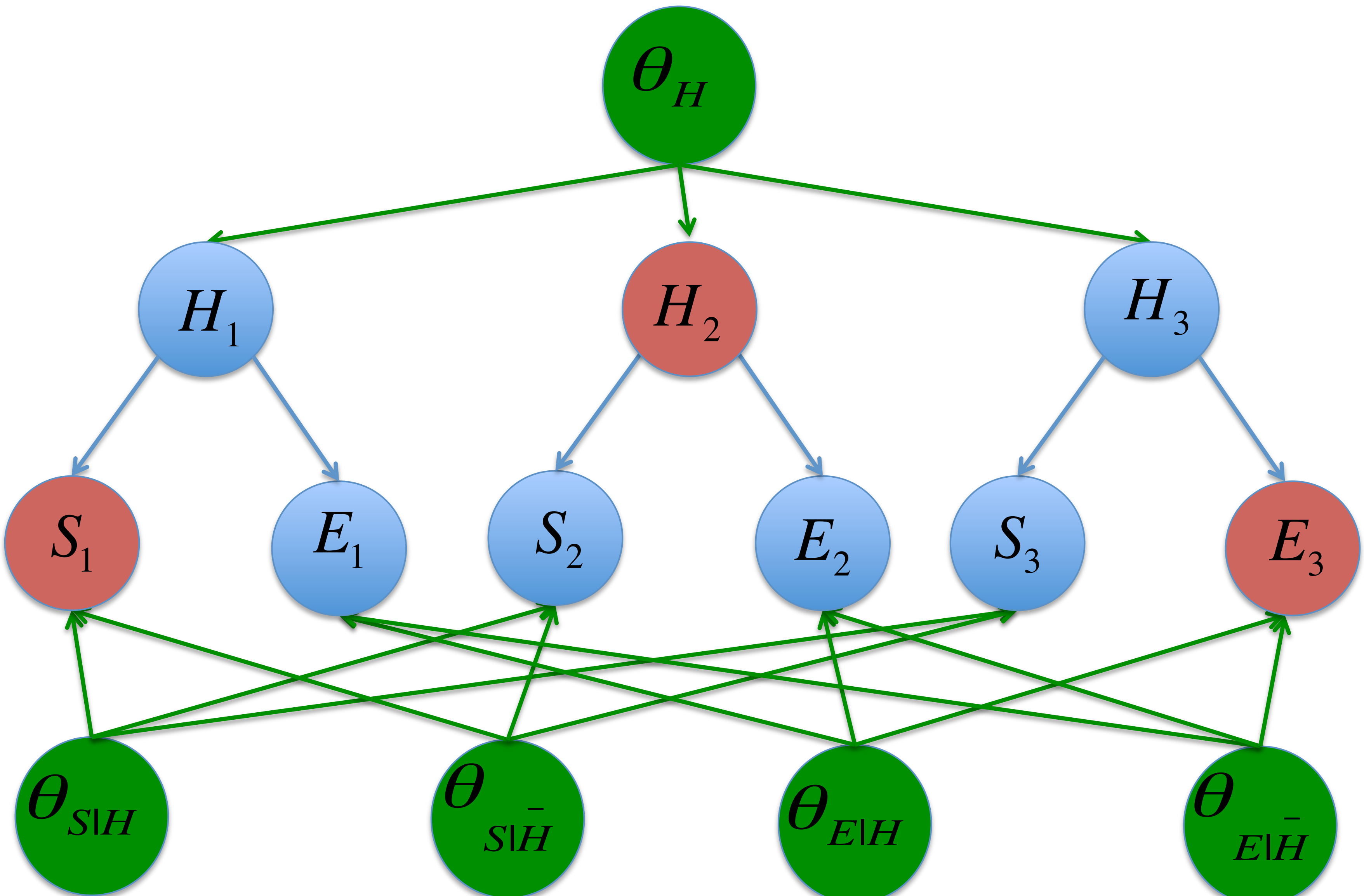
Meta Network Creation (cont.)



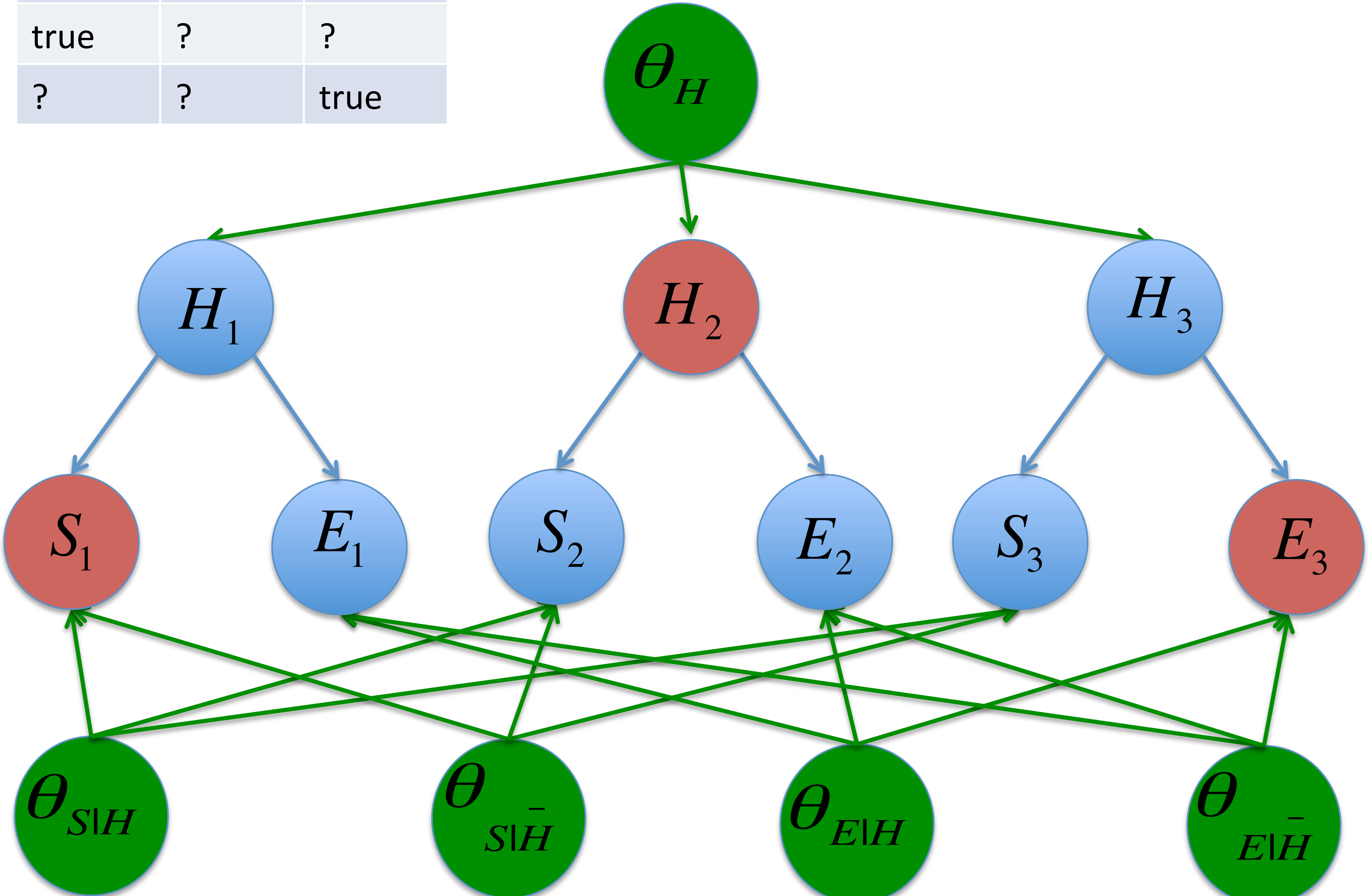
Meta Network Creation (cont.)



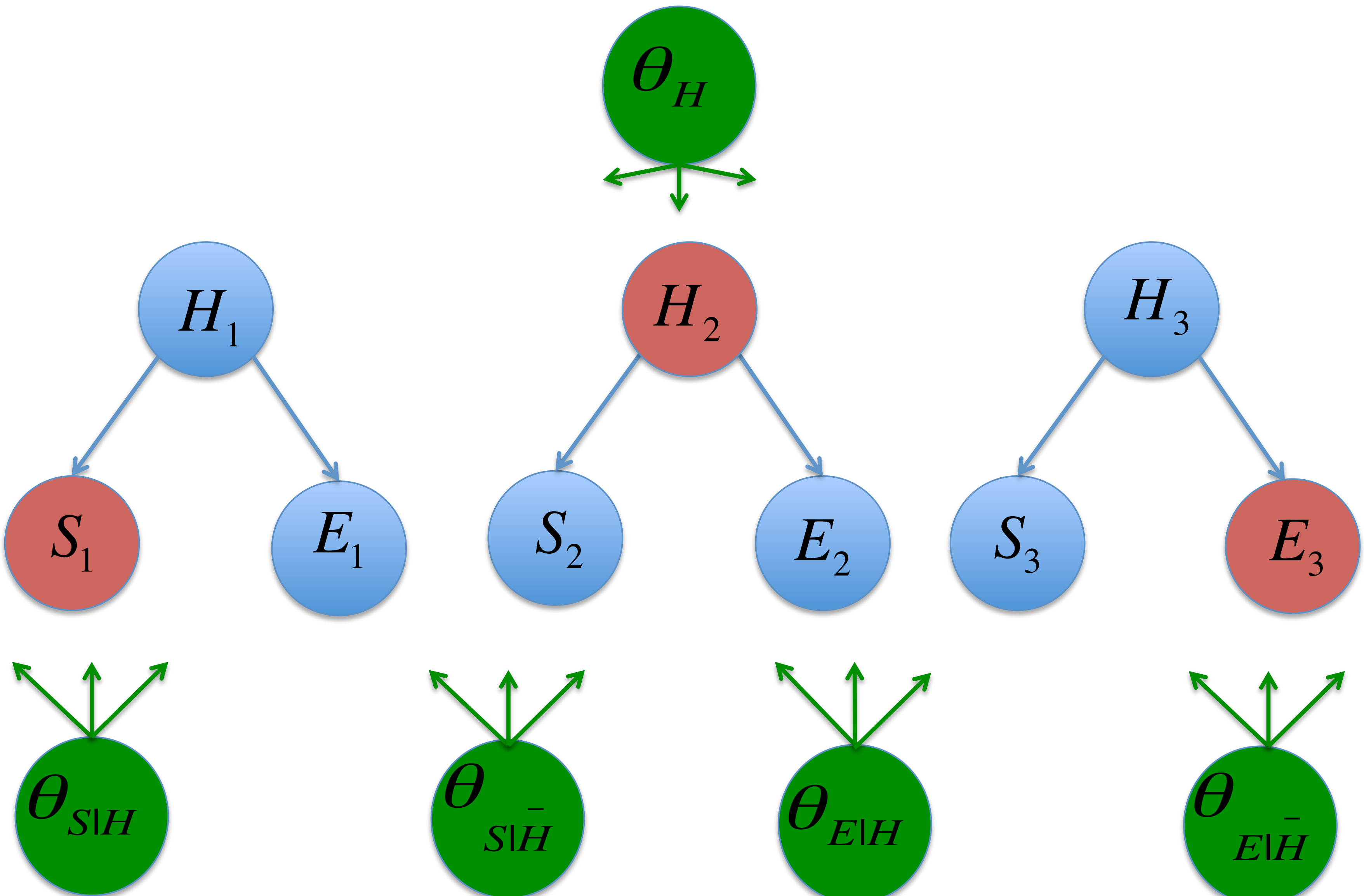
Assert Data as Evidence



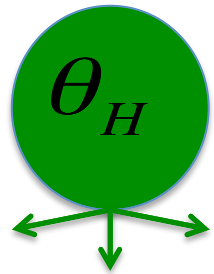
H	S	E
?	true	?
true	?	?
?	?	true



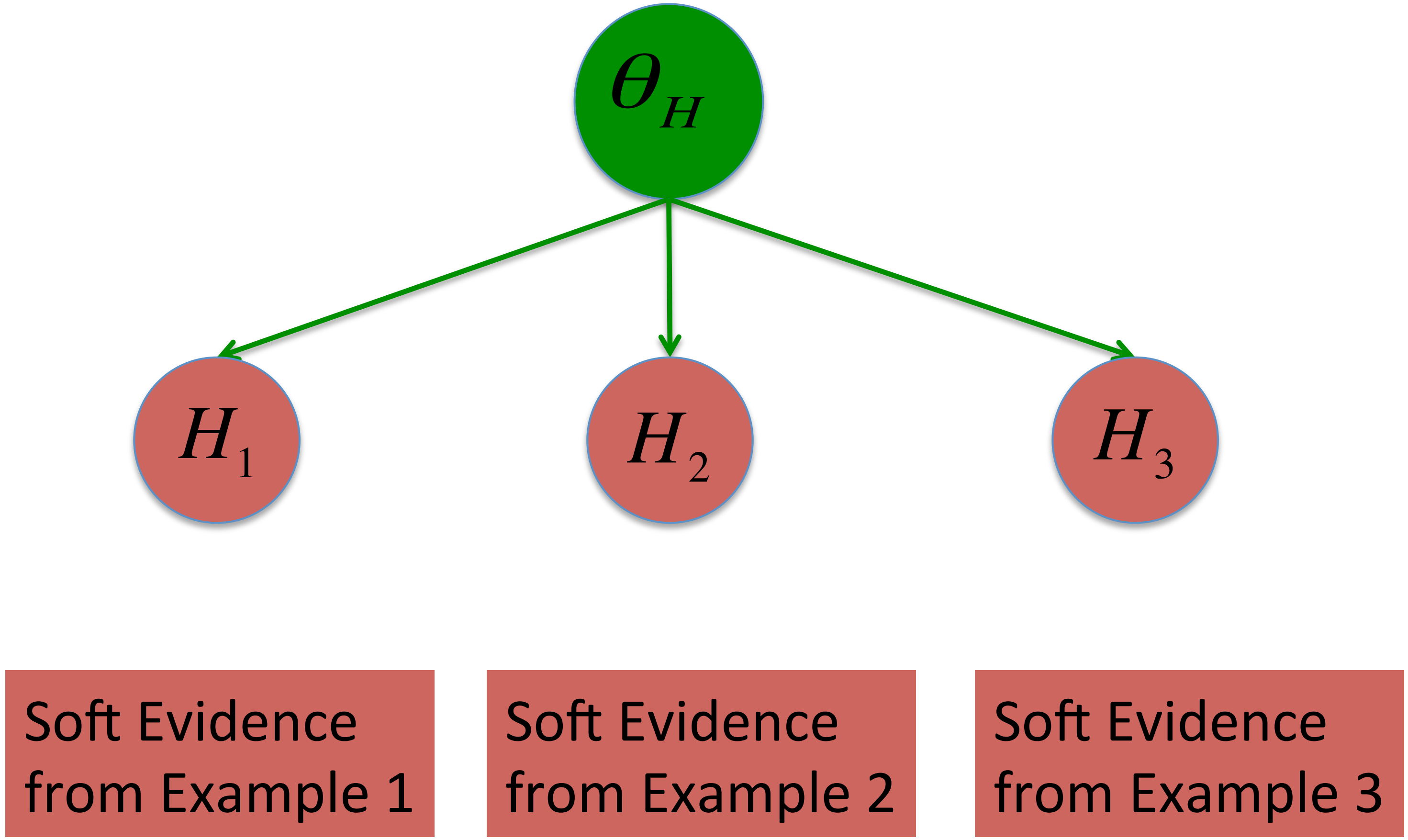
EDML (Delete Edges)



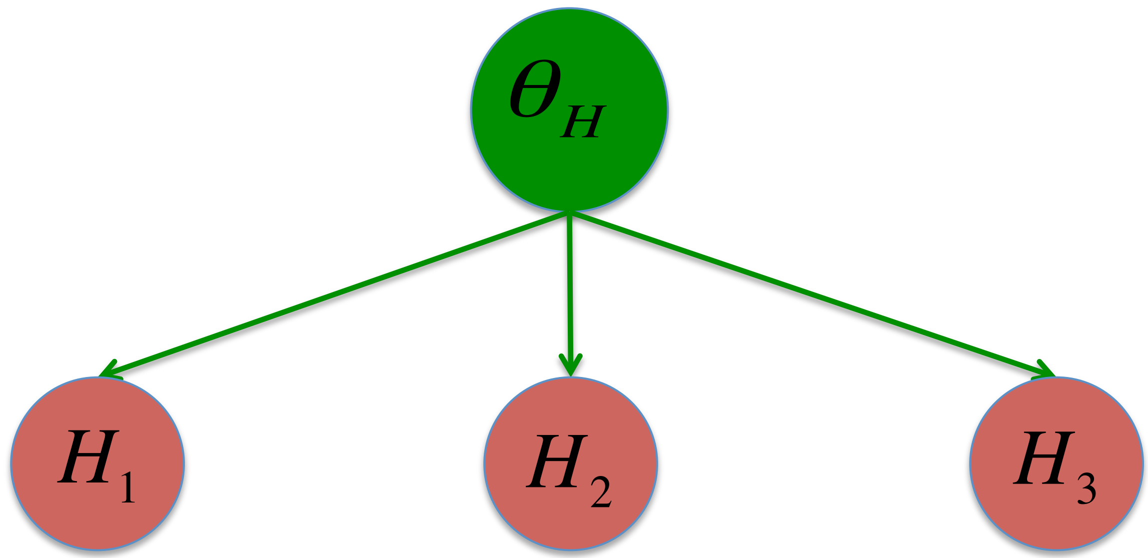
EDML (Learning from Soft Evidence)



EDML (Learning from Soft Evidence)



EDML (Learning from Soft Evidence)



Maximizing the posterior probability is a convex optimization problem (UAI'11, UAI'12).

Algorithm 1 Multivalued EDML

input:

G: A Bayesian network structure

$\mathcal{D}$: An incomplete dataset $\mathbf{d}_1, \ldots, \mathbf{d}_N$

θ: An initial parameterization of structure G

ψ: A Dirichlet prior for each parameter set $\theta_{X|\mathbf{u}}$

while not converged **do**

$Pr \leftarrow$ distribution induced by θ and G

Compute soft evidence parameters:

$$\lambda^{i}_{x|\mathbf{u}} \leftarrow Pr(x\mathbf{u}|\mathbf{d}_i)/Pr(x|\mathbf{u}) - Pr(\mathbf{u}|\mathbf{d}_i) + 1 \qquad (1)$$

for each family instantiation $x\mathbf{u}$ and example $\mathbf{d}_i$

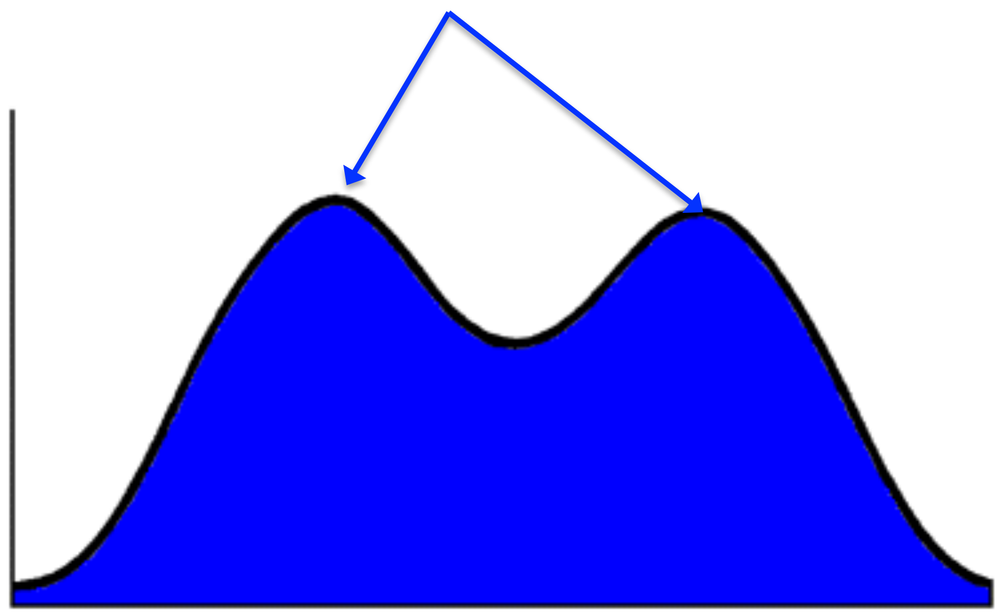
Update parameters:

$$\theta_{X|\mathbf{u}} \leftarrow \underset{\hat{\theta}_{X|\mathbf{u}}}{\operatorname{argmax}} \prod_{x} [\hat{\theta}_{x|\mathbf{u}}]^{\psi_{x|\mathbf{u}}-1} \prod_{i=1}^{N} \sum_{x} \lambda^{i}_{x|\mathbf{u}} \hat{\theta}_{x|\mathbf{u}} \qquad (2)$$

return parameterization θ

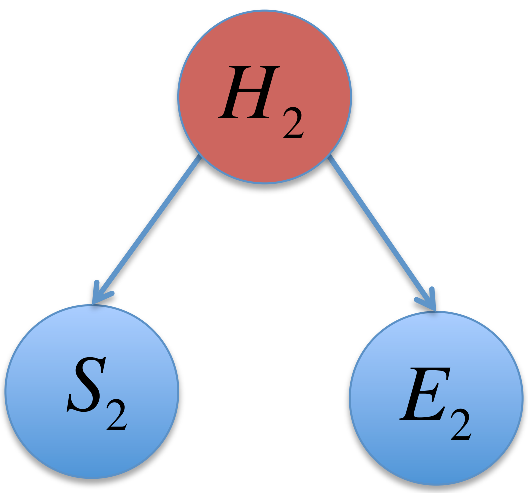
EDML Fixed Points (UAI'12)

Theorem: EDML fixed points are precisely the EM fixed points.



Convergence (UAI'11)

Theorem: When only leaves have missing values, EDML converges in one iteration, whereas EM may not.



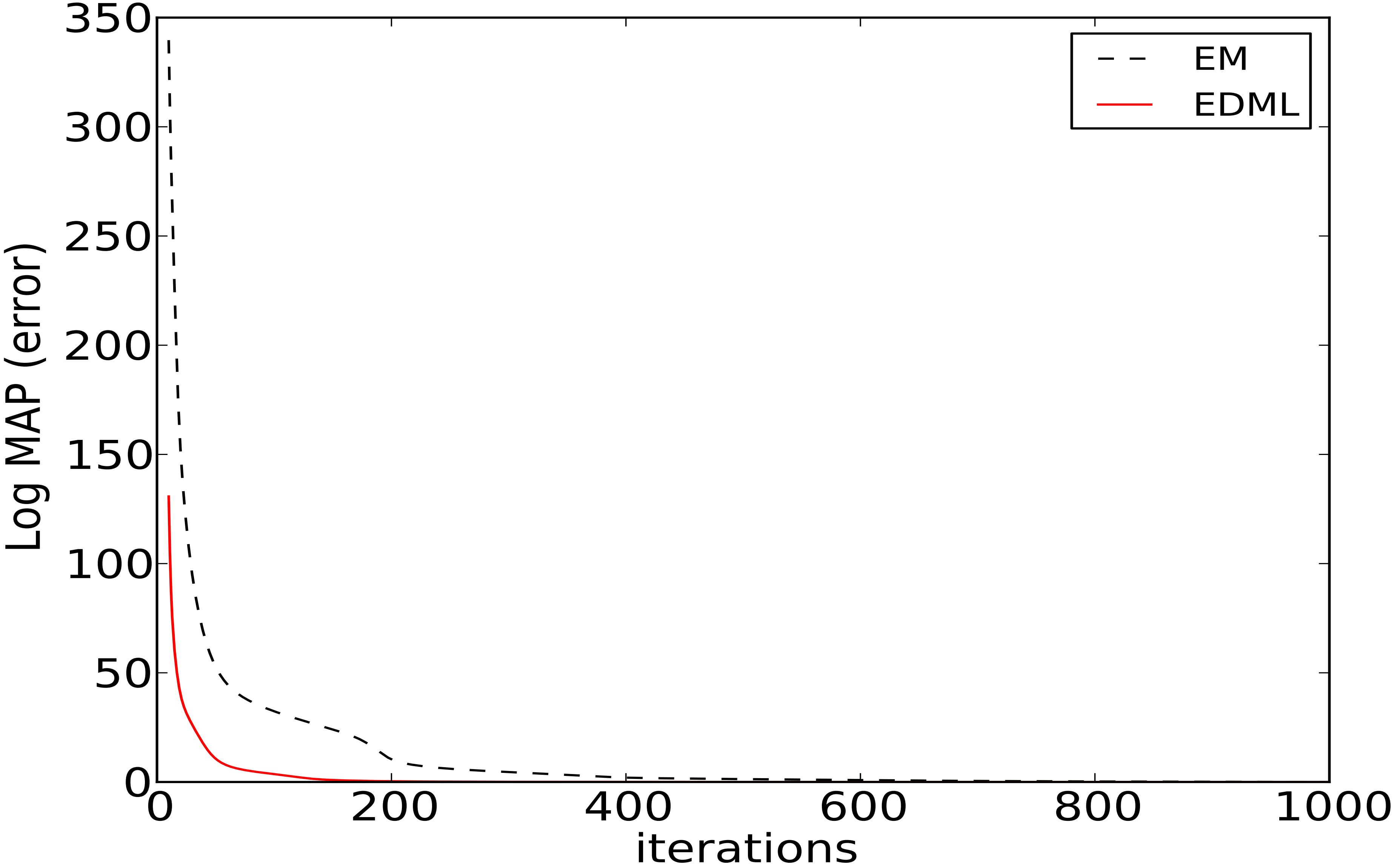
Experiment EM vs. EDML (iterations)

Category	%EDML better	%EM better	EDML Speed-up %	EM Speed-up %
Hiding 10%	93.82%	6.18%	84.59%	87.13%
Hiding 25%	90.95%	9.05%	83.83%	75.70%
Hiding 35%	82.24%	17.76%	86.26%	75.09%
Hiding 50%	77.61%	22.39%	87.80%	80.21%
Hiding 70%	75.65%	24.35%	84.48%	74.21%
Average	**83.05%**	**16.95%**	**85.41%**	**76.96%**

Experiment EM vs. EDML (iterations)

Category	%EDML better	%EM better	EDML Speed-up %	EM Speed-up %
Hiding 10%	93.82%	6.18%	84.59%	87.13%
Hiding 25%	90.95%	9.05%	83.83%	75.70%
Hiding 35%	82.24%	17.76%	86.26%	75.09%
Hiding 50%	77.61%	22.39%	87.80%	80.21%
Hiding 70%	75.65%	24.35%	84.48%	74.21%
Average	**83.05%**	**16.95%**	**85.41%**	**76.96%**

Andes (Hiding 25% of the nodes)



EDML Generalization (NIPS'13)

- We generalized EDML as a parallel coordinate descent algorithm.
- This helps derive new EDML algorithms for other graphical models.

EDML for Learning MRFs from Complete Data (NIPS'13)

Table 1: Speed-up results of EDML over CG and L-BFGS

problem	i_{cg}	i_{edml}	t_{cg}	(S)	$i_{\text{l-bfgs}}$	i'_{edml}	$t_{\text{l-bfgs}}$	(S')
zero	45	105	3.62	3.90×	24	74	1.64	1.98×
one	104	73	8.25	13.26×	58	42	3.87	8.08×
two	46	154	3.73	2.83×	21	87	1.54	1.54×
three	43	169	3.58	2.52×	52	169	3.55	1.93×
four	56	126	4.59	4.31×	61	115	3.90	3.22×
five	43	155	3.48	2.70×	49	155	3.20	1.90×
six	48	150	3.93	3.13×	20	90	1.47	1.40×
seven	57	147	4.64	3.37×	23	89	1.65	1.62×
eight	48	155	3.82	2.84×	57	154	3.83	2.28×
nine	56	168	4.46	3.15×	45	141	2.90	1.94×
54.wcsp	107.33	160.33	6.56	2.78×	68.33	172	1.80	0.72×
orchain42	120.33	27	0.123	31.27×	110	54.33	0.06	6.43×
orchain45	151	33.67	0.14	12.52×	94.33	36.33	0.06	4.85×
orchain147	107.67	18.67	3.27	80.72×	105	58.33	1.63	12.77×
orchain148	122.67	42.33	1	49.04×	80	32	0.28	14.24×
orchain225	181.33	58	0.79	44.14×	137.67	69	0.33	10.76×
rbm20	9	41	30.98	2.38×	30	107.22	30.18	0.99×
Seg2-17	63	83.66	1.77	7.00×	46.67	64.67	0.74	4.14×
Seg7-11	54.3	84	1.86	2.84×	48.66	73.33	1.27	2.32×
Family2Dominant.1.5loci	117.33	88	2.39	5.90×	85.67	78.33	1.04	2.69×
Family2Recessive.15.5loci	111.6	89.7	1.31	3.85×	86.33	81.67	0.74	2.18×
grid10x10.f5.wrap	136.67	239	17.36	6.26×	142	180.33	10.3	4.63×
grid10x10.f10.wrap	101.33	62.33	12.39	20.92×	92.67	59	5.94	9.70×
average	**83.89**	**101.29**	**5.39**	**13.55**×	**66.84**	**94.89**	**3.56**	**4.45**×

Conclusion

- Learning from incomplete data can be difficult.

Conclusion

- Learning from incomplete data can be difficult.
- Good news: patterns of incompleteness may be exploited.

Conclusion

- Learning from incomplete data can be difficult.
- Good news: patterns of incompleteness may be exploited.
- EDML becomes more exact as the data becomes more complete.

Thanks!