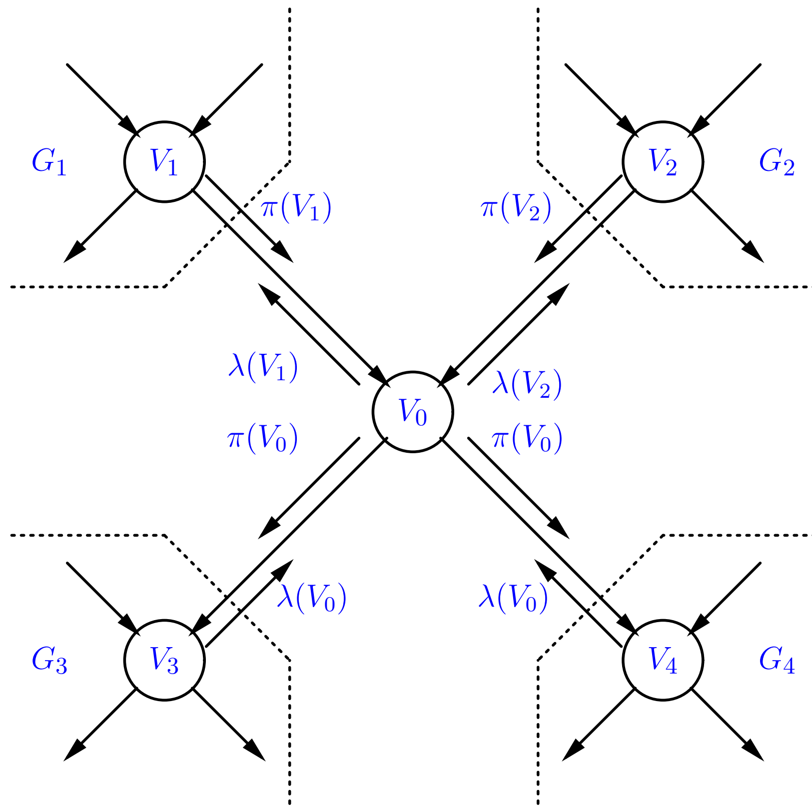
Inference in Bayesian Networks

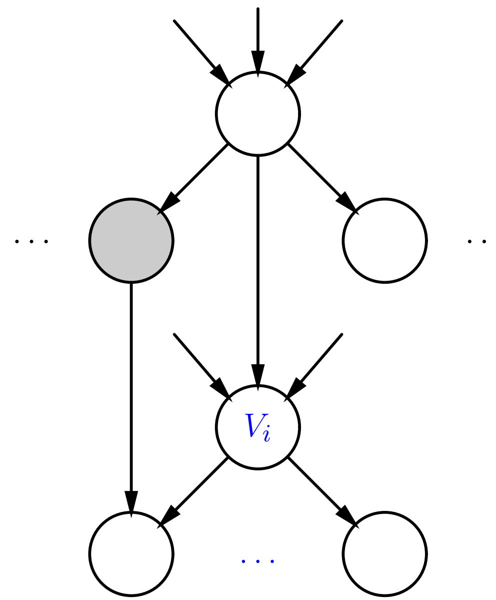
Algorithms for general DAGs

Basic idea of Pearl's algorithm



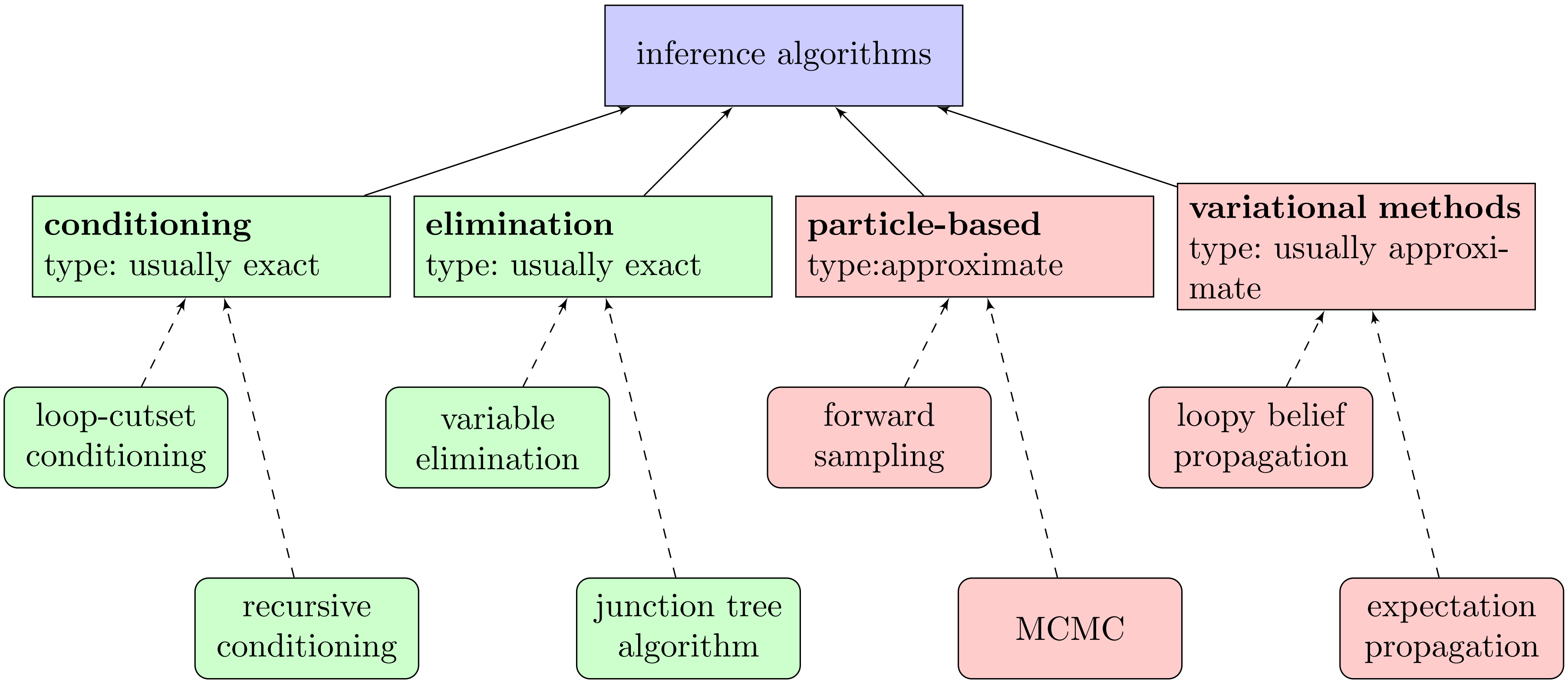
- Object-oriented approach: vertices are objects, which have local information and carry out local computations
- Updating of probability distribution by message passing: arcs are communication channels

Multiply-connected networks inference



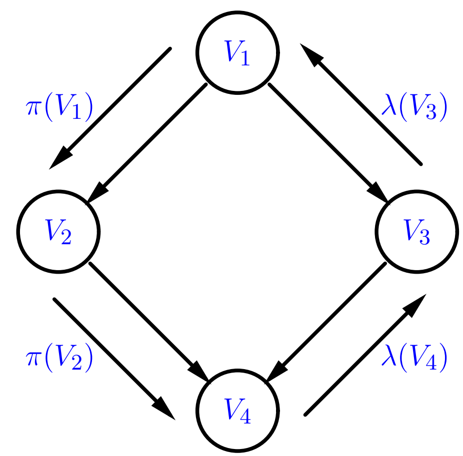
- At least two nodes are connected by more than one path (in the underlying undirected path)
- Thus, some variables can influence another through more than one causal mechanism
- And same evidence counted more than once

Bayesian network inference algorithms



Loopy belief propagation

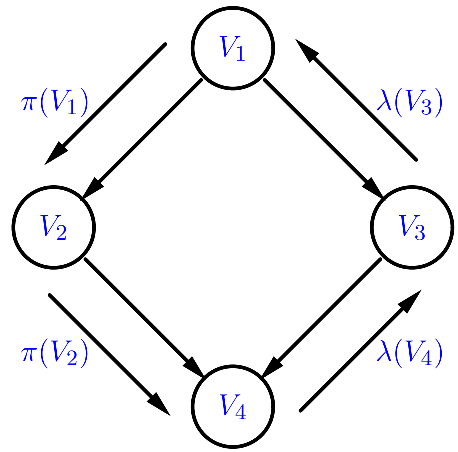
- Apply Pearl's propagation algorithm to multiply-connected networks
- In (undirected) cycles, messages may circle indefinately



- Solutions:
  - Stop after fixed number of iterations
  - Stop when there are no significant changes in the beliefs
- If it converges, it is usually a good approximation

Problems with loopy belief propagation

- It may not converge
- Cycling error: old information is mistaken for new



Suppose V_4 observed and gets new information from V_2:
  - V_4 sends V_3 a message with information about itself and node V_2
  - V_3 passes that on to V_1 which in turn sends it to V_2
  - V_2 misinterprets its own information for new and includes it into its distribution
- Convergence error: the propagation algorithm assumes independence of the parents

Conditioning methods

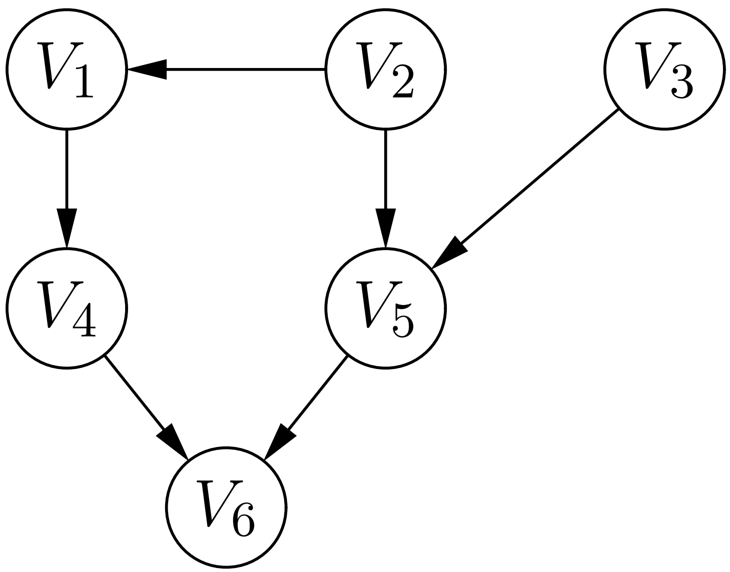
The main ideas of these algorithms are as follows:

1. Find a cutset: if these nodes were instantiated, the network behaves as if it were singly-connected
2. Compute the posterior probability distributions e.g. using Pearl's algorithm for every instantiation
3. Marginalisation/conditioning yields the requested distribution

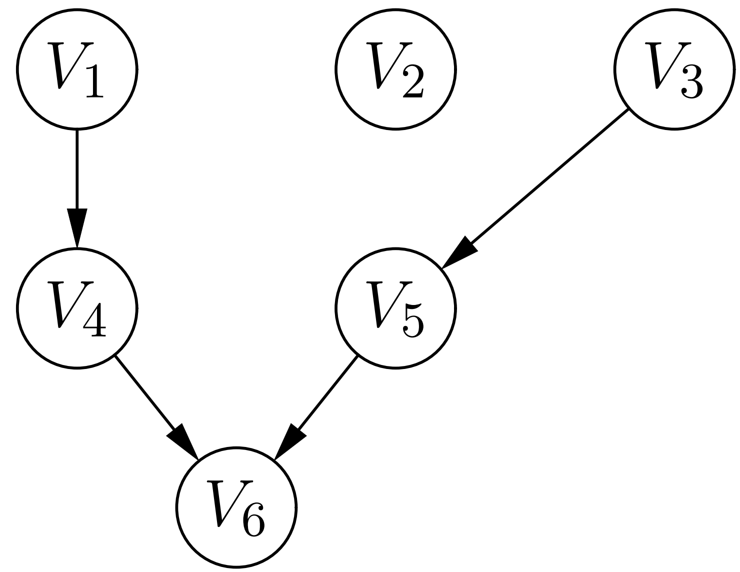
A set is called a loop cutset if every cyclic chain contains three consecutive nodes X_1, X_2, X_3 such that X_2 is part of the cutset and either:

- $X_1 \leftarrow X_2$ and $X_2 \rightarrow X_3$, or
- $X_1 \rightarrow X_2$ and $X_2 \rightarrow X_3$

Instantiated network



G



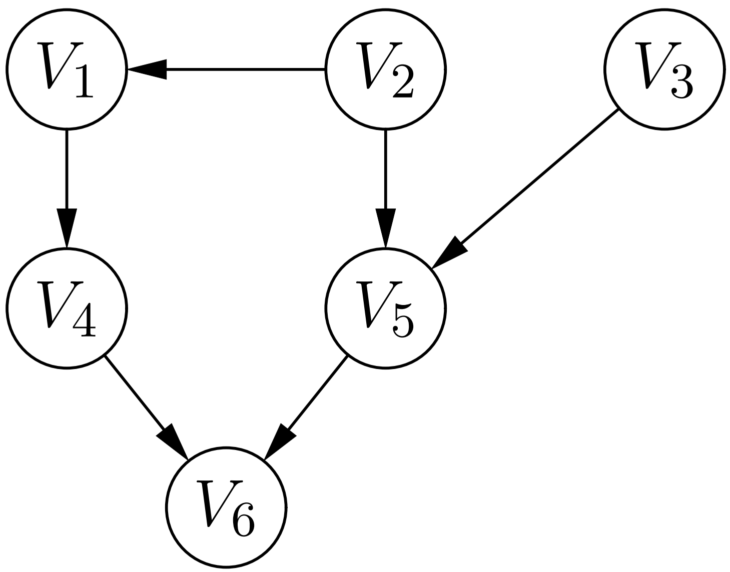
structure of G^{v_2}

Suppose V_2 is cutset, then G can be instantiated by (e.g.) v_2

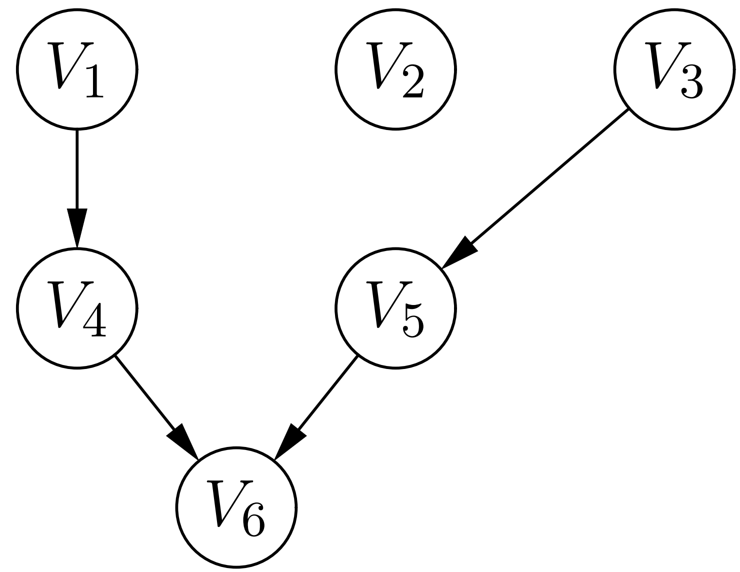
- Outgoing edges of V_2 can be deleted
- CPTs are updated, e.g. $P_{G^{v_2}}(V_5 \mid V_3) = P_G(V_5 \mid v_2, V_3)$
- It holds that:

$$P_G(V_i, v_2) = P_{G^{v_2}}(V_i, v_2)$$

Cutset conditioning: general idea



G

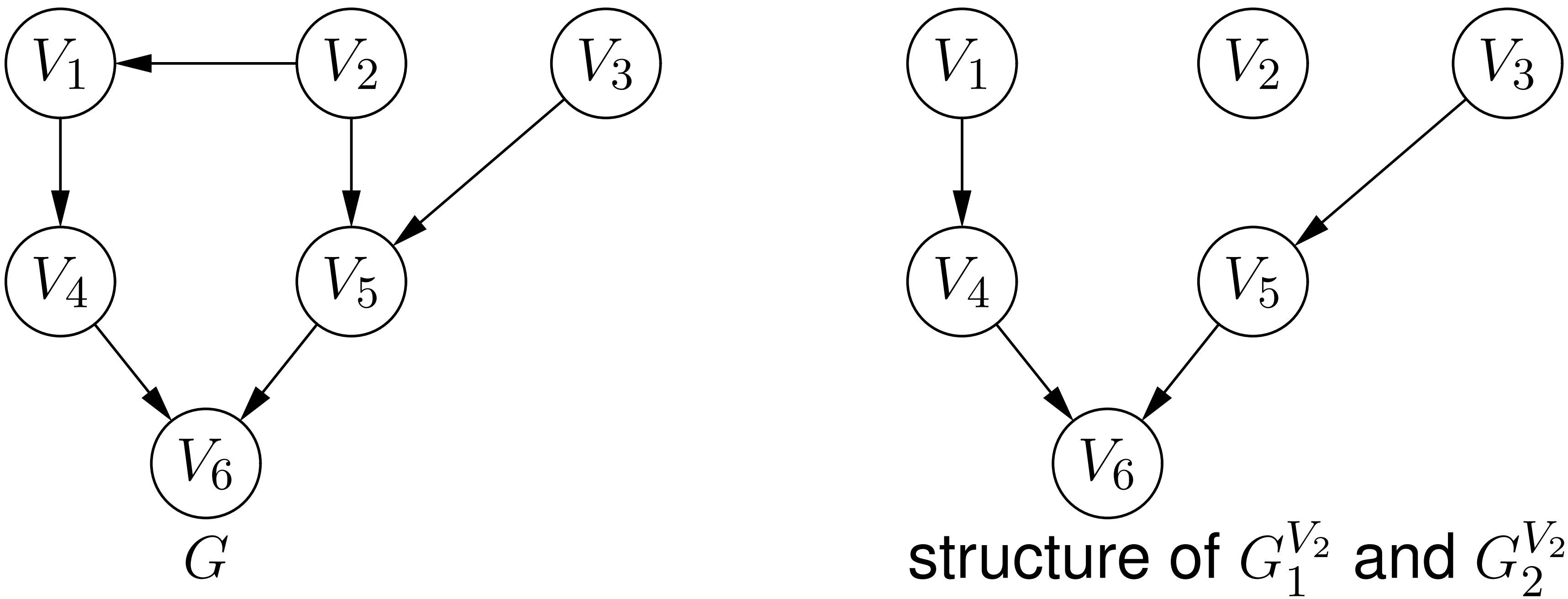


structure of G^{v_2}

Suppose we would like to compute $P(v_6)$:

- Find a cutset, e.g. $\{V_2\}$ (which others are there?)
- Delete edges: get singly-connected networks
- Compute $P_{G^{v_2}}(v_6 \mid v_2)$ and $P_{G^{\neg v_2}}(v_6 \mid \neg v_2)$
- $P(v_6) = P_{G^{v_2}}(v_6 \mid v_2)P(v_2) + P_{G^{\neg v_2}}(v_6 \mid \neg v_2)P(\neg v_2)$

Recursive conditioning: general idea



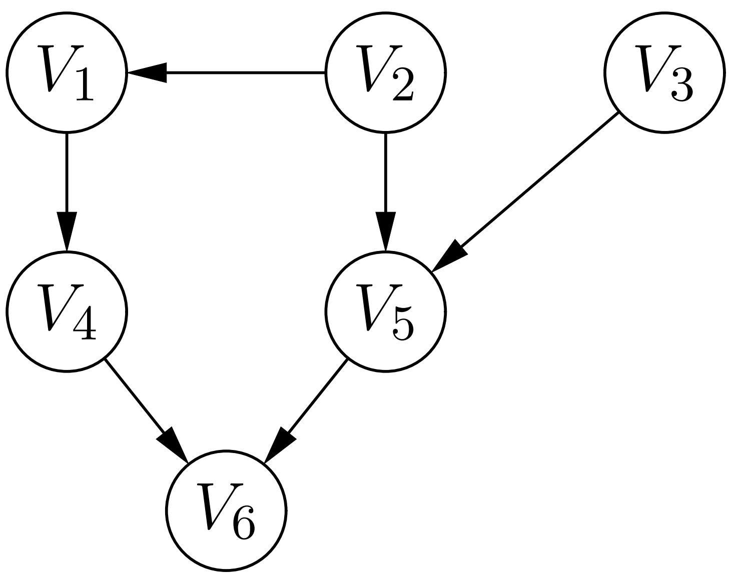
structure of $G_1^{V_2}$ and $G_2^{V_2}$

Suppose we would like to compute $P(v_6)$:

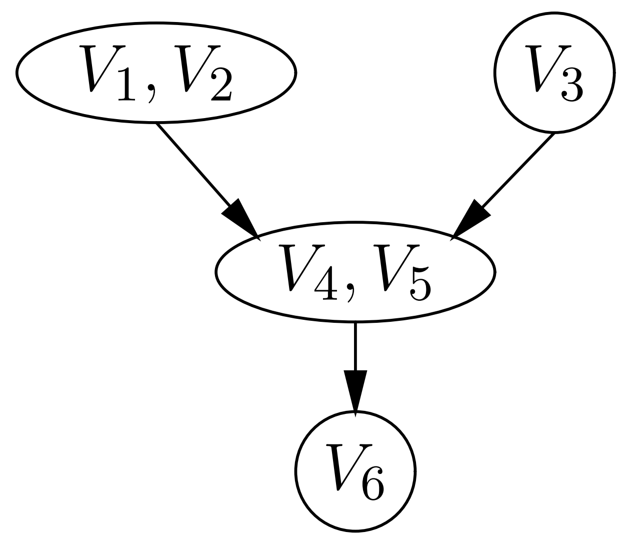
- If we use a cutset $\{V_2\}$ it will decompose G into two parts: G_1 (with V_2) and G_2 (with V_6)
- It holds: $P_{G^{v_2}}(v_2, v_6) = P_{G_1^{v_2}}(v_2) P_{G_2^{v_2}}(v_6)$
- So: $P^{v_2}(v_6) = P_{G_1^{v_2}}(v_2) P_{G_2^{v_2}}(v_6) + P_{G_1^{\neg v_2}}(\neg v_2) P_{G_2^{\neg v_2}}(v_6)$

Clustering inference algorithm

- Transform a BN into an equivalent polytree by merging nodes
  - Removal of multiple paths between nodes
  - New node has as states all possible instantiations of combined nodes
- Probabilities updating on transformed polytree



(a)



(b)

Junction tree algorithm: overview

Efficient method for clustering: junction tree algorithm

- Junction trees

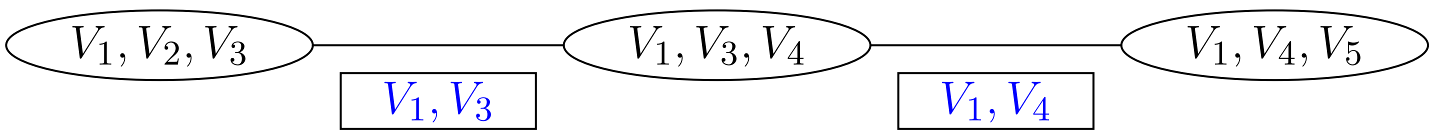
- Constructing the junction tree
  - moralisation
  - triangulation
  - clustering nodes into a tree

- Computing parameters of the junction tree

- Using a message passing algorithm to compute probabilities

Junction tree

- (Maximal) clique: a (maximal) complete subset of nodes of an undirected graph
- A junction tree represents a tree of maximal cliques



- Separator sets: variables shared by neighbours
- A junction tree factorises as:

$$P(V) = \frac{\prod_C \varphi_C(V_C)}{\prod_S \varphi_S(V_S)} \text{ with } C \text{ cliques and } S \text{ separators}$$

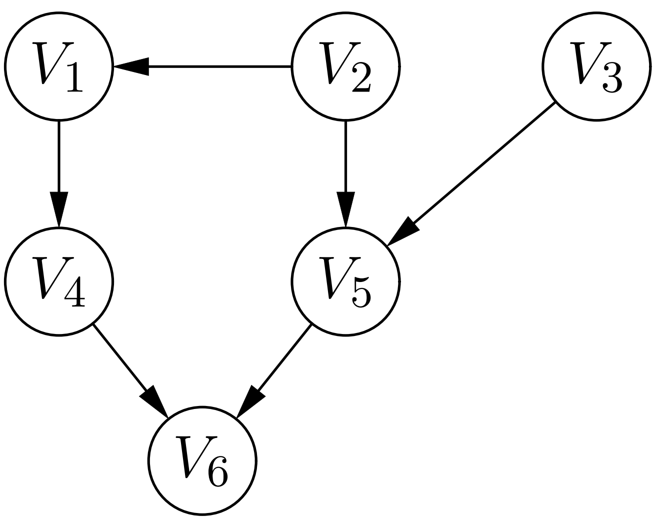
- For all pair of cliques C_1 and C_2, all nodes on the path between C_1 and C_2 contain $C_1 \cap C_2$ (running intersection property)

Moralisation

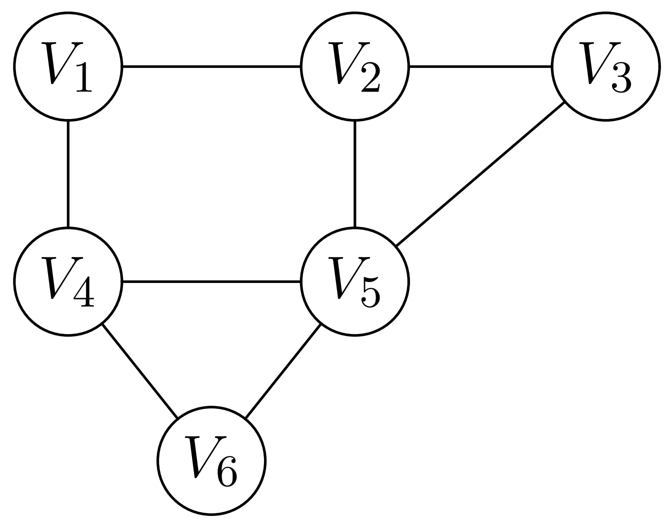
Let G be an acyclic directed graph, its associated undirected moral graph G^m can be constructed by moralisation:

1. add lines to all non-connected vertices, which have a common child, and
2. replace each arc with a line in the resulting graph

Proposition: if G is an I-map, then so is G^m



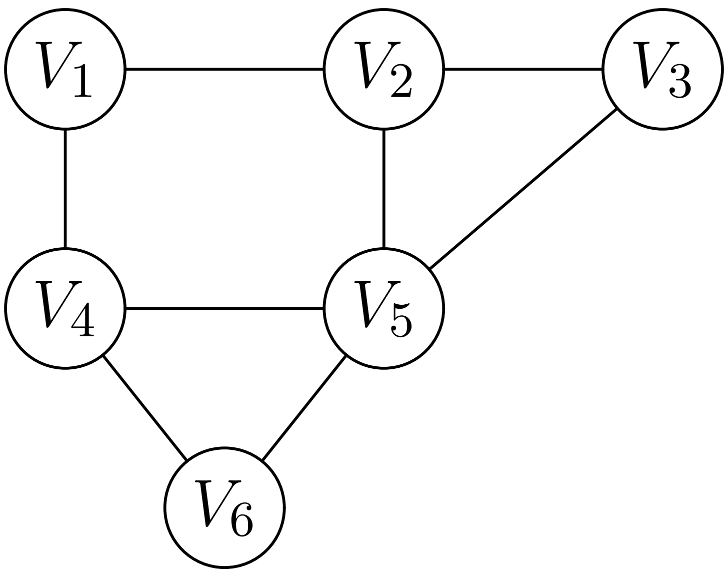
(a)

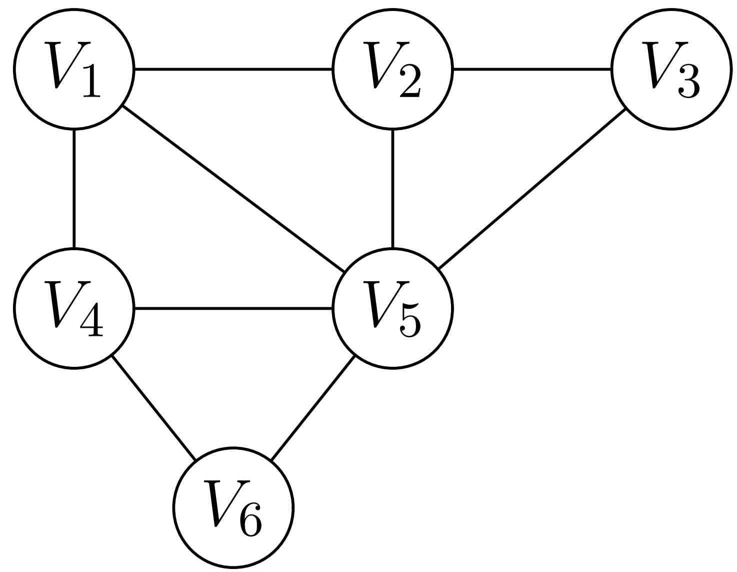


(b)

Triangulation

A chord of a cycle is a pair V_i, V_j of non-consecutive vertices in a cycle such that (V_i, V_j) is an edge in G





An undirected graph G is called chordal or triangulated if every one of its cycles of length ≥ 4 posseses a chord

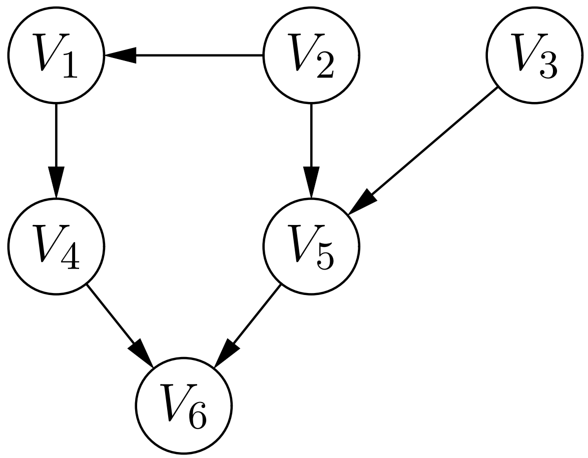
Theorem: every triangulated graph has a junction tree

Constructing the junction tree

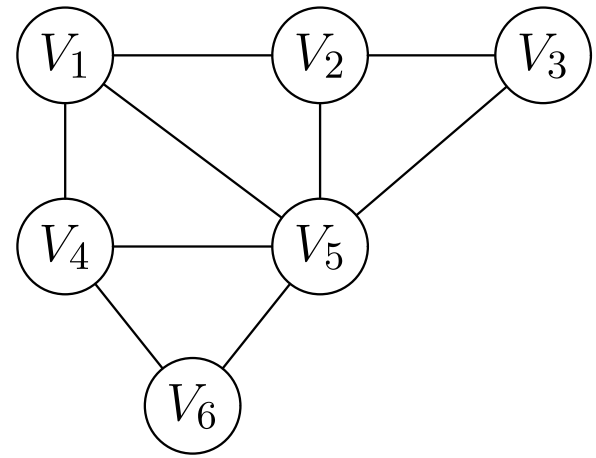
Given a triangulated graph G. A junction tree is obtained using the following steps:

- Find all the cliques, each one becomes a cluster, i.e., a node in the junction tree
- If two clusters have a non-empty intersection, create an edge with the intersection as separator
- If this graph contains a cycle, then all separators on this cycle contain the same variable. Remove the cycle by creating a maximal spanning tree: include as many separators as possible while avoiding a cycle

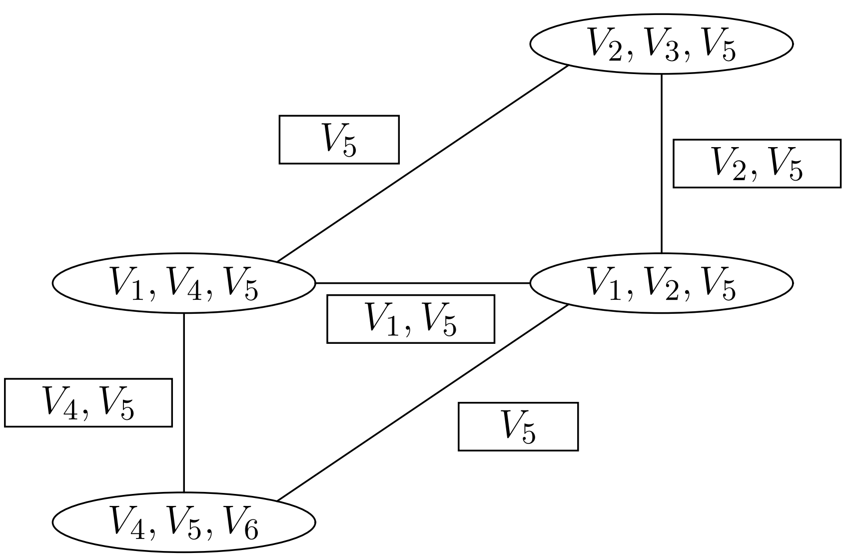
Example



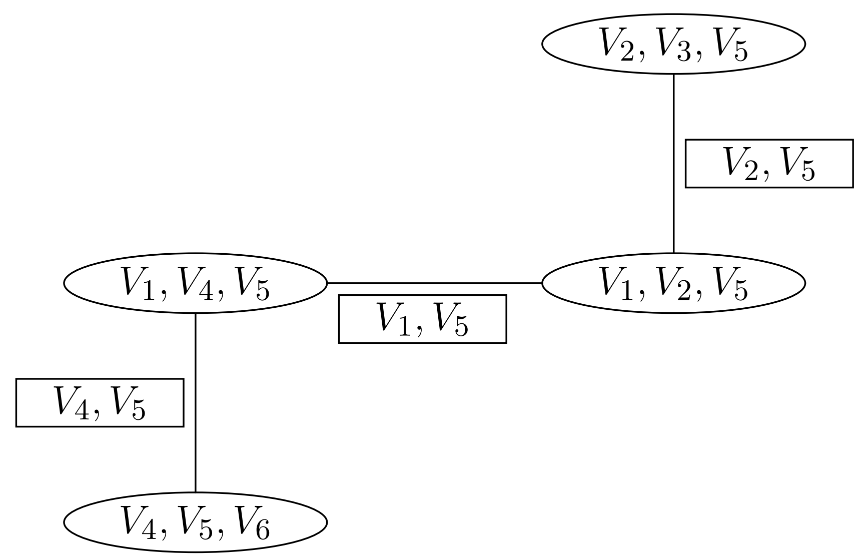
a



b



c



d

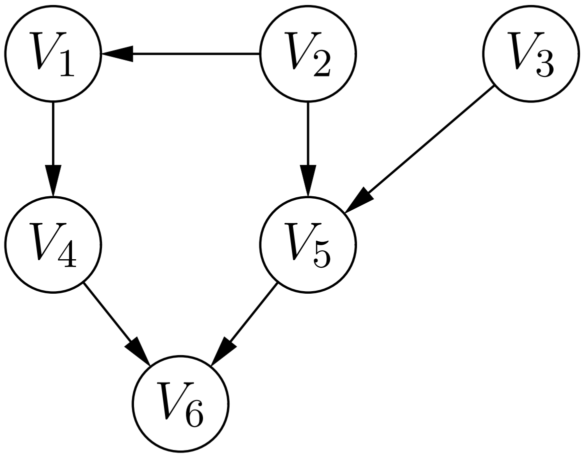
Computing parameters

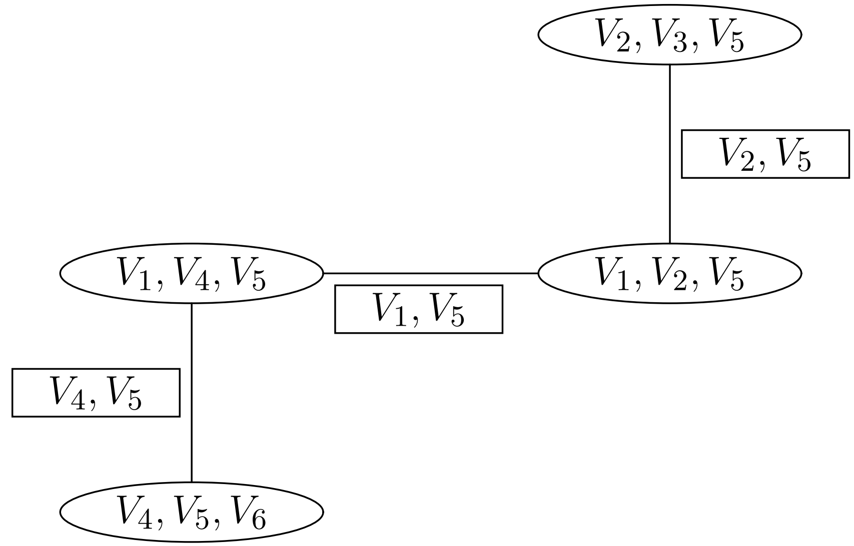
Theorem. Let G be an I-map of a probability distribution P. It holds that G is triangulated iff the probability distribution can be factorised in terms of marginal densities over variables in the cliques of G.

This can be done as follows:

- Choose a node C in the junction tree that contains X and all of X's parents
- Multiply $P(X \mid \text{pa}(X))$ yielding C's table

Example

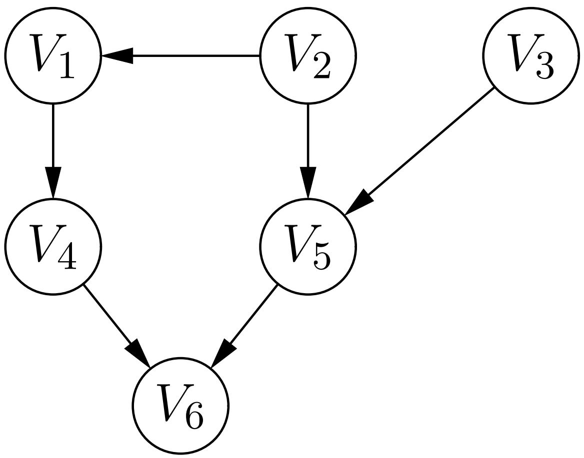


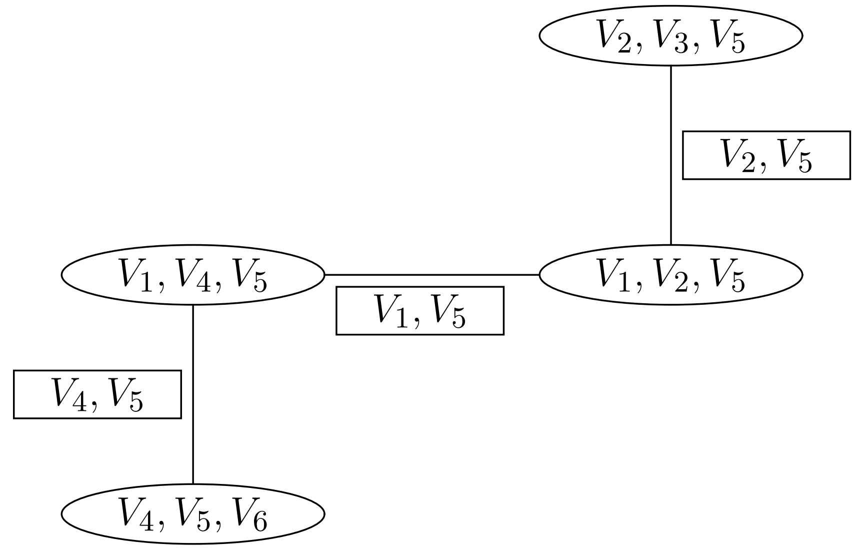


The original graph is factorised as:

$$\begin{aligned} P(V) &= P(V_6 \mid V_4, V_5) P(V_5 \mid V_2, V_3) P(V_4 \mid V_1) \cdots \\ & \quad P(V_3) P(V_2) P(V_1 \mid V_2) \end{aligned}$$

Example



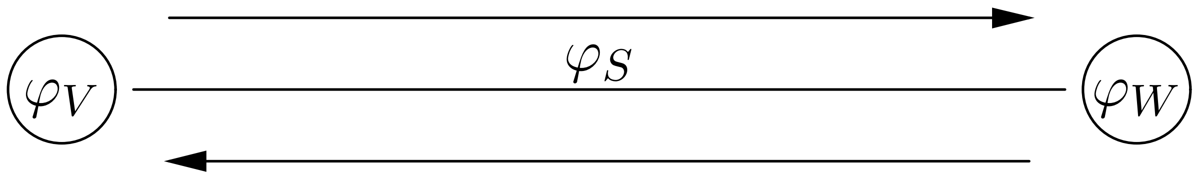


The junction tree has parameters (e.g.):

$$
\begin{aligned}
\varphi(V_2, V_3, V_5) &= P(V_5 \mid V_2, V_3)P(V_3) \\
\varphi(V_1, V_2, V_5) &= P(V_1 \mid V_2)P(V_2) \\
\varphi(V_1, V_4, V_5) &= P(V_4 \mid V_1) \\
\varphi(V_4, V_5, V_6) &= P(V_6 \mid V_4, V_5)
\end{aligned}
$$

$P(V) = \frac{\prod_C \varphi_C(V_C)}{\prod_S \varphi_S(V_S)}$ where the separator potentials $\varphi_S(V_S)$ are set to 1

Message passing



Updating works in 2 passes:

1. Updating from V to W (*forward pass*):

$$\varphi_S^* = \sum_{V \setminus S} \varphi_V \qquad\qquad \varphi_W^* = \frac{\varphi_S^*}{\varphi_S} \varphi_W$$

   This sets the separator to the marginal in φ_V

2. Then, from W to V (*backward pass*):

$$\varphi_S^{**} = \sum_{W \setminus S} \varphi_W^* \qquad\qquad \varphi_V^* = \frac{\varphi_S^{**}}{\varphi_S^*} \varphi_V$$

Note: here the variables are implicit, i.e., $\varphi_V = \varphi_V(V)$

Message passing: soundness

The update procedure

$$\varphi_S^* = \sum_{V \setminus S} \varphi_V \qquad\qquad \varphi_W^* = \frac{\varphi_S^*}{\varphi_S} \varphi_W$$

is *sound*, i.e., after an update the probability distribution is the same (after step 1)

Proof. Note that nothing happens with φ_V, so define $\varphi_V^* = \varphi_V$. Then:

$$P^*(V \cup W) = \frac{\varphi_V^* \varphi_W^*}{\varphi_S^*} = \frac{\varphi_V \varphi_S^* \varphi_W}{\varphi_S^* \varphi_S} = \frac{\varphi_V \varphi_W}{\varphi_S} = P(V \cup W)$$

Exercise. Proof that after both passes (local consistency):

$$\sum_{V \setminus S} \varphi_V^* = \sum_{W \setminus S} \varphi_W^*$$

Global consistency junction trees

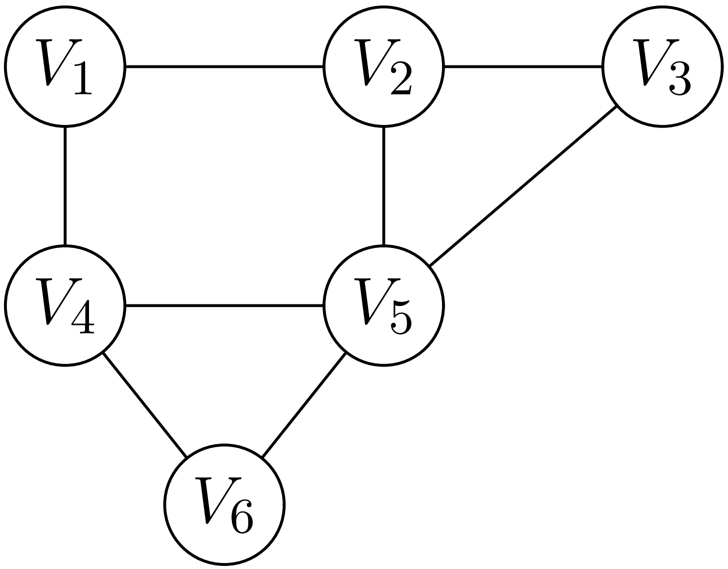
- Global consistency: a cluster tree is globally consistent if for any nodes V and W with intersection I we have:

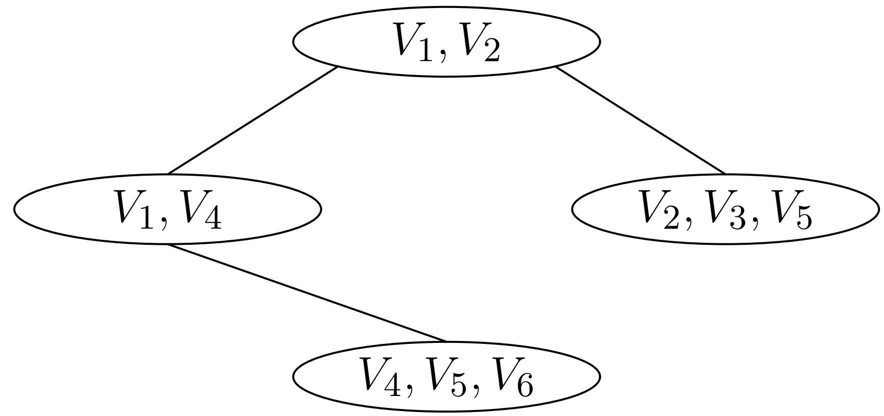
$$\sum_{V \setminus I} \varphi_V = \sum_{W \setminus I} \varphi_W$$

- Junction trees are after the message passing globally consistent:
  Proof.
  By induction on distance of the path between V_0 and V_n. Suppose they are neighbours: then by local consistency. Otherwise, we have consistency of length k. Since I will be in the separator between V_k and V_{k+1} (running intersection property), local consistency can again be applied, so the property follows for $k+1$.

The need for triangulation

Suppose we would not triangulate:





V_5 appears in two non-neighbouring cliques. There is no guarantee that marginal V_5 should be equal, i.e.,

$$\sum_{V_2, V_3} \varphi_{235}(V_2, V_3, V_5) = \sum_{V_4, V_6} \varphi_{456}(V_4, V_5, V_6)$$

Reasoning in Junction Trees

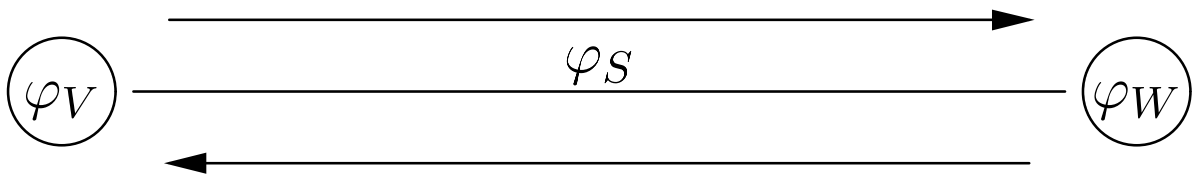
- If a variable V_E is observed, we modify the potential φ which includes V_E such that the marginal is V_E and otherwise the same
- A node C_1 can send a message to another node C_2 if it has received a message from all its other neighbours
- Choose an arbitrary node as root and collect and distribute messages to and from this node
- Afterwards, it holds:

$$\varphi_C(V_C) = P(V_C \mid V_E)$$

- Moreover:

$$\varphi_S(V_S) = P(V_S \mid V_E)$$

Proof sketch correctness



Recall that after the update procedure it holds that $\sum_{V \setminus S} \varphi_V(V) = \varphi^*_S(S)$. So then:

$$
\begin{aligned}
P(W) &= \sum_{V \setminus S} P(V \cup W) \\
&= \sum_{V \setminus S} \frac{\varphi_V(V)\varphi_W(W)}{\varphi_S(S)} \\
&= \frac{\sum_{V \setminus S} \varphi_V(V)\varphi_W(W)}{\varphi_S(S)} = \varphi^*_W(W)
\end{aligned}
$$

In general, factors are eliminated one by one

Sampling

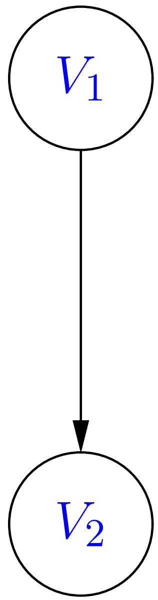
Logic sampling traverses the tree from the root nodes:

- Initialise $Count(x, e) = 0$ and $Count(e) = 0$
- Randomly choose a variable from the root nodes, weighted by the priors
- Repeat: randomly choose values for the children, weighted by the conditional probability given the known values of the parents
- If e is in the assignment, then increase $Count(e)$ by 1
- If both x and e are in the assignment, then increase $Count(x, e)$ by 1

After many iterations:

$$P(x \mid e) = \frac{Count(x, e)}{Count(e)}$$

Example



$$P(v_1) = 0.8, P(\neg v_1) = 0.2$$

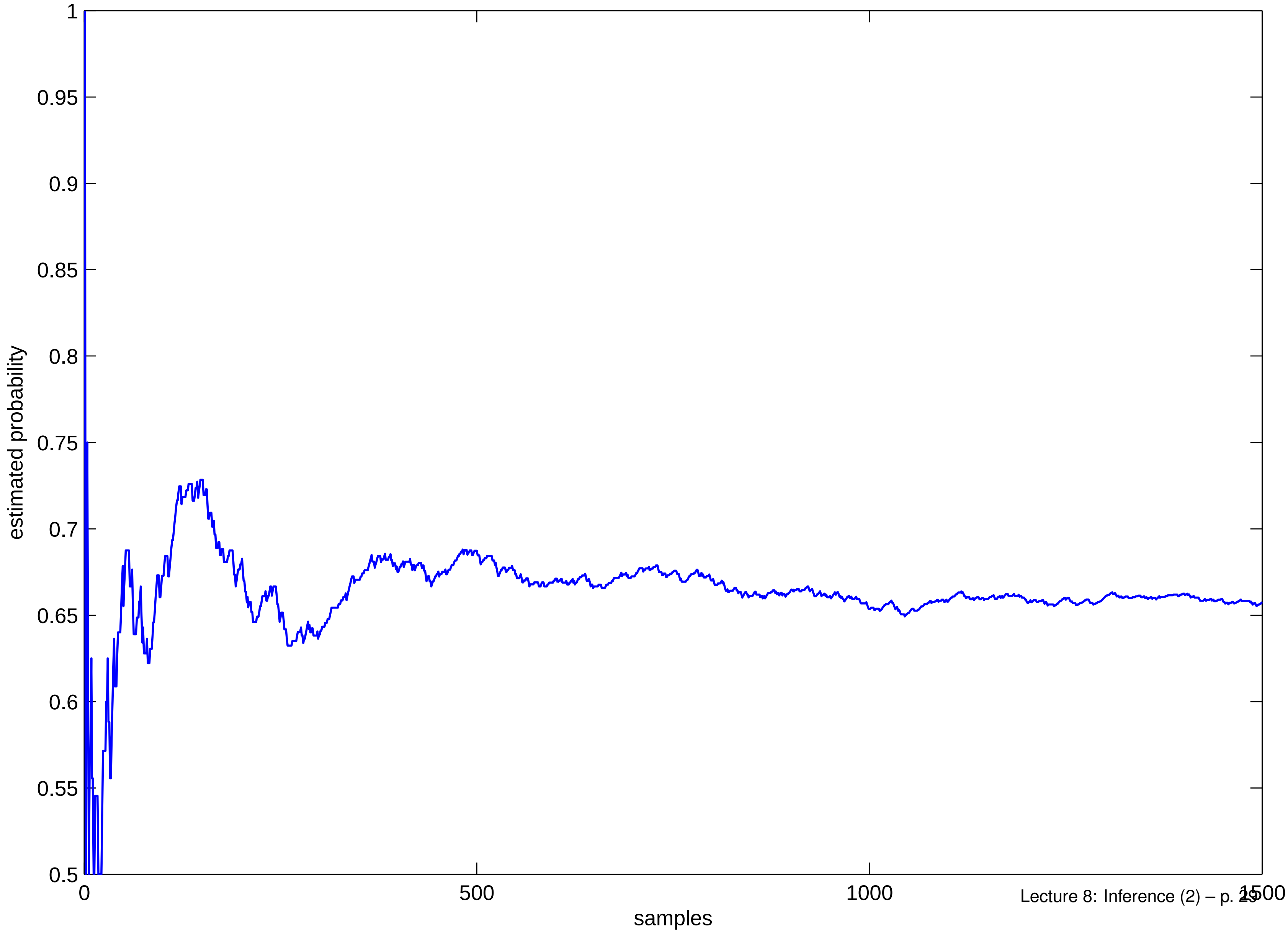
$$P(v_2|v_1) = 0.4, P(\neg v_2|v_1) = 0.6$$
$$P(v_2|\neg v_1) = 0.9, P(\neg v_2|\neg v_1) = 0.1$$

To calculate $P(v_1 \mid v_2)$ sample e.g. 1000 instantiations

$$Count(v_1, v_2) \simeq 320 (= 0.8 \cdot 0.4 \cdot 1000)$$
$$Count(\neg v_1, v_2) \simeq 180 (= 0.2 \cdot 0.9 \cdot 1000)$$
$$Count(v_2) \simeq 500$$

So $P(v_1 \mid v_2) \simeq 320/500 = 0.64$

Convergence



Complexity

- Computational complexity loop cutset conditioning: $O(n \cdot d \cdot 2^{d+l})$, where n is the number of vertices, d is the maximal in-degree and l is the number of vertices in the cutset
- Computational complexity junction tree algorithm: $O(n \cdot 2^{c})$ where c is the number of vertices in the largest clique
- However, generally probabilistic inference within an arbitrary network is NP-hard
  - Also for approximate inference!
- Empirically: sometimes approximate inference can be used to compute marginals if exact methods fail

References

- Darwiche, A. (2000) *Recursive conditioning* Artificial Intelligence 126(1-2) pp. 5–41.
- Pearl, J. (1988) *Probabilistic Reasoning in Intelligent Systems.* Morgan Kauffman. ISBN 0-934613-73-7
- Spiegelhalter, D. J. and Lauritzen, S. L. (1990) Sequential updating of conditional probabilities on directed graphical structures. *Networks*, **20**, pp. 579-605
- Max Henrion: Propagating uncertainty in Bayesian networks by probabilistic logic sampling. *UAI, 1986*, pp. 149-164

Ideas for seminar topics

- Inference by weighted model counting
- Inference by arc reversal
- Loopy belief propagation
- Lifted belief propagation
- Inference with continuous variables

$\Rightarrow$ and many more!