**D-Separation**

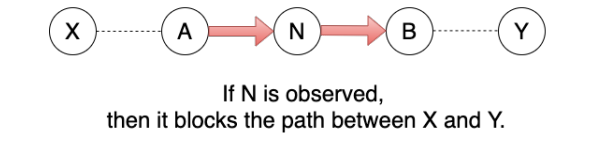
**Definition** (D-Separation). A set of variables E d-separates variables X and Y if E blocks every un-directed path between X and Y in the network.

To determine whether X and Y are independent given the observed variables E, we can verify whether E d-separates X and Y. If d-separation holds, then the independence relationship holds as well. To verify d-separation, we need to consider every path between X and Y. On each path, there could be multiple nodes between X and Y. The path is blocked if at least one node blocks the path. When a path is blocked it is considered closed. When a path is not blocked it is considered open.

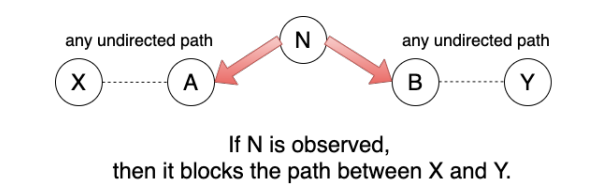
To better understand blocking one needs to think of a bayesian network as a network system of information channels, where each node is a valve that is either active (open) or inactive(closed) and the valves are connected by noisy information channels. The information flow can pass through an active valve but not an inactive one. When all the valves (nodes) on one undirected path between two nodes are active, we say this path is open. If any one valve in the path is inactive(blocked), we say the path is closed. A valve becomes inactive when observed or is a part of an inactive path.

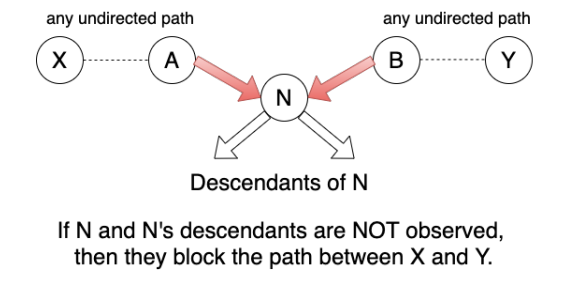
There are three basic scenarios :

**Scenario 1**



**Scenario 2:**



**Scenario 3**:

Based on these 3 scenarios we can determine whether a path is blocked or not and as a sequence whether two nodes are (Conditionally independent or not).

**The Greedy Thick Thinning algorithm**

The Greedy Thick Thinning algorithm constructs Bayesian networks by analyzing dependency relationships among nodes. The dependency relationships are measured by using some kind of conditional independence (CI) test based on the flow of information between two nodes. The amount of information flow between two nodes can be measured by using mutual information, when no nodes are instantiated, or Kullback-Leibler cross entropy, when some other nodes are conditioned upon.

In information theory, the **mutual information** of two nodes is defined as:

**Kullback-Leibler cross entropy** is defined as:

where are two nodes and is a set of nodes.

The Greedy Thick Thinning algorithm uses cross entropy as CI test to measure the average information between two nodes when the statuses of some valves are changed by the condition-set . When is smaller than a certain threshold value are considered d-separated by the condition-set , and conditionally independent. This algorithm also makes the following assumptions:

(1) The database attributes have discrete values and there are no missing values in all the records.

(2) The volume of data is large enough for reliable CI tests.

(3) The ordering of the attributes is available before the network construction, i.e., a node's parents nodes should appear earlier in the order.

This algorithm has three phases: drafting, thickening and thinning. In the first phase, this algorithm computes mutual information of each pair of nodes as a measure of closeness, and creates a draft based on this information. In the second phase, the algorithm adds arcs when the pairs of nodes cannot be d-separated. The result of Phase II is an independence map (I-map) of the underlying dependency model. In the third phase, each arc of the I-map is examined using CI tests and will be removed if the two nodes of the arc can be d-separated. The result of Phase III is the minimal I-map.

**The Algorithm**

**Phase 1**

1) Initiate a graph G(V, E) where V={all the nodes of a data set}, E={ }. Initiate two empty ordered set S, R.

2) For each pair of nodes (vi, vj) where vi, vj ∈ V, compute mutual information I(vi, vj). For the pairs of nodes that have mutual information greater than a ceftain small value ε, sort them by their mutual information from large to small and put them into an ordered set S.

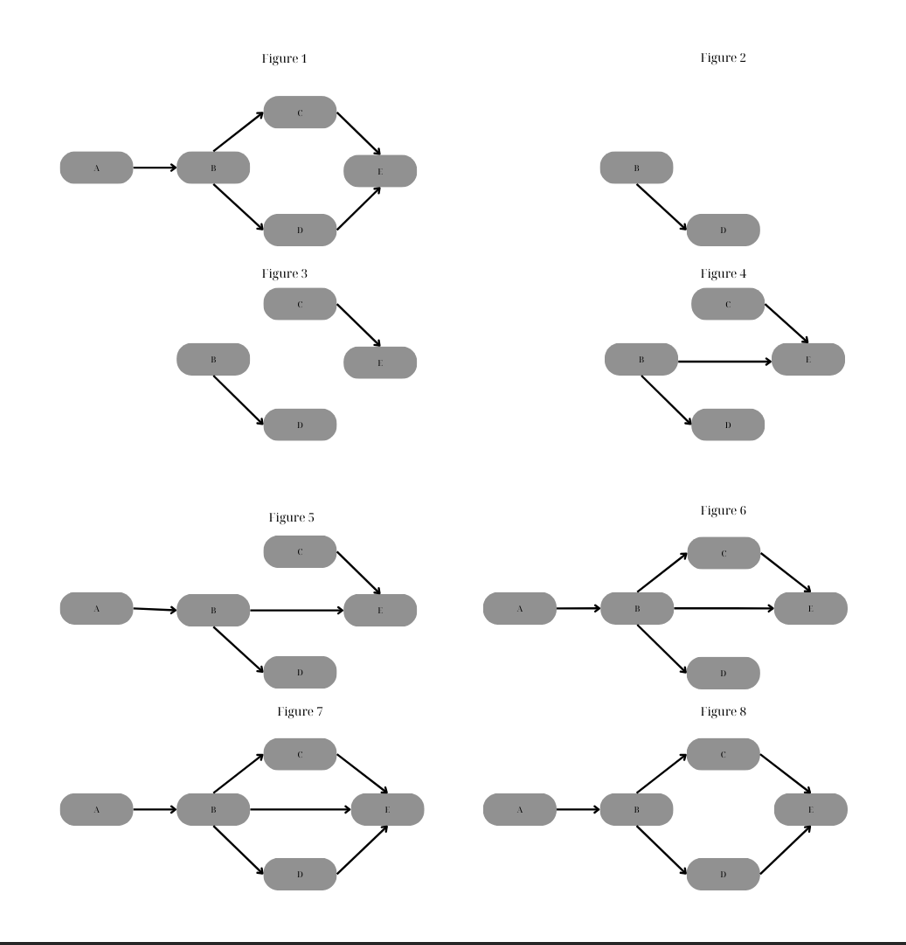
3) Get the first two pairs of nodes in S and remove them from S. Add the corresponding arcs to E. (the direction of the arcs in this algorithm is determined by the previously available nodes ordering.)

4) Get the first pair of nodes remained in S and remove it from S. If there is no open path between the two nodes (these two nodes are d-separated given empty set), add the corresponding arc to E. Otherwise, add the pair of nodes to the end of an ordered set R.

5) Repeat step 4 until S is empty.

Suppose we have a database that has an underlying Bayesian network as in Figure 1, we also have a nodes' order as and by computing mutual information we get

Sequentially nodes and corresponding arcs will be added. A step by step graphical representation of the procedure is shown bellow. For the pairs at fig2 at fig3 at fig4 at fig5 at fig 6 will be added. At fig 6 will be ommitted due to D-separation.



**Phase 2 (Thickening)**

6)Get the first pair of nodes in R and remove it from R

7) Find a block set that blocks each open path between these two nodes by a set of minimum number of nodes. (This procedure find\_block\_set (current graph, node1, node2) is given at the end of this subsection.) Conduct a CI test. If these two nodes are still dependent on each other given the block set, connect them by an arc.

8) go to step 6 until R is empty.

In our example, the graph after Phase II is shown in Figure 7. When this algorithm examines the pair of nodes in step 7, it finds that is the minimum set which blocks all the open paths between and . Since the CI test can reveal that and are still dependent given ,arc is added resulting in Figure 7.

**Phase 3 (Thinning)**

9) For each arc in E, if there are open paths between the two nodes besides this arc, remove this arc from E temporarily and call procedure find\_block\_set (current graph, node1, node2). Conduct a CI test on the condition of the block set. If the two nodes are dependent, add this arc back to E; otherwise remove the arc permanently.

In our example arc is removed resulting in figure 8 which is the minimal I-map of the underlying dependency model. In order to avoid large conditioning sets we need to know the minimum d-Separation set. This is achieved with the following algorithm.

**Procedure find\_block\_set** (current graph, node1, node2)

Begin

find all the undirected paths between node1 and node2;

store the open paths in open\_path\_set;

store the closed paths in closed\_path\_set;

do

while there are open paths which have only one node do

store the nodes of each such path in the block set;

remove all the blocked paths by these nodes from the open\_path\_set and closed\_path\_set;

from the closed\_path\_set, find paths opened by the nodes in block set and move them to the open\_path\_set. shorten such paths by removing the nodes that are also in the block set;

end while

if there are open paths do

find a node which can block maximum number of the rest paths and put it in the block set;

remove all the blocked paths by the node from the open\_path\_set and the closed\_path\_set;

from the closed\_path\_set, find paths opened by this node and move them to the open\_path\_set. shorten such paths by removing the nodes t\_hat are also in the block set;

end if

until there are no open path

end procedure.

Implementing this algorithm we can get the underlying structure of the Bayesian Network based only on data an not domain knowledge.

**The Dirichlet Multinomial Bayesian Model**

The **Dirichlet Multinomial Bayesian Model** is a probabilistic model commonly used in Bayesian statistics, particularly for modeling categorical data. It combines the **Dirichlet distribution** and the **Multinomial distribution** to handle scenarios where observations belong to multiple categories, and uncertainty exists about category probabilities.

**Key components**

**1)The Multinomial Distribution**

The **Multinomial distribution** is a generalization of the [binomial distribution](https://en.wikipedia.org/wiki/Binomial_distribution) . **Multinomial Distribution** models the probability of observing counts across multiple categories and it is Suitable for discrete, categorical data .

Formula :

*With*

Where :

* *N* = total count of observations.
* = counts for category .
* = probability for category .

for non-negative integers

The probability mass function can be expressed using the gamma function as:

With

This form shows its resemblance to the Dirichlet distribution, which is its conjugate prior.

**2)The Dirichlet Distribution**

The Dirichlet distribution , often denoted , is a family of continuous multivariate probability distributions parameterized by a vector α of positive reals. It is a multivariate generalization of the beta distribution, hence its alternative name of multivariate beta distribution . Dirichlet distributions are commonly used as prior distributions in Bayesian statistics, and in fact, the Dirichlet distribution is the conjugate prior of the categorical distribution and multinomial distribution.

The Dirichlet distribution of order K ≥ 2 with parameters α1, ..., αK > 0 has a probability density function with respect to Lebesgue measure on the Euclidean space

*Where*

and .

The normalizing constant is the multivariate beta function, which can be expressed in terms of the gamma function:

.

The Dirichlet distribution is the conjugate prior for the Multinomial distribution!

Proof :

**Dirichlet-Multinomial**

Suppose that

(prior)

(likelihood)

The posterior:

*==(1)*

But

Pugging (2) into (1) we have :

= =

= =

=

Choosing hyperparameter vector α with equal weights

() makes the prior uniform.