# An Information Theory based Approach to Structure Learning in Bayesian Networks

Gopalakrishna Anantha

9th October 2006

Committee

Dr.Xue wen Chen (Chair)

Dr. John Gauch

Dr. Victor Frost

# Publications

- **“An effective structure learning method for constructing gene networks”**
  - Xue wen Chen, Gopalakrishna Anantha and Xinkun Wang
    - Published in the Bioinformatics Journal, Vol 22. Pg 1367 – 1374.

# Presentation Overview

- Introduction
- Structure Learning in Bayesian Networks
- K2 Algorithm
- Proposed Method
- Results and Analysis
- Conclusions and Future Work

# Introduction-Definition and Key Concepts

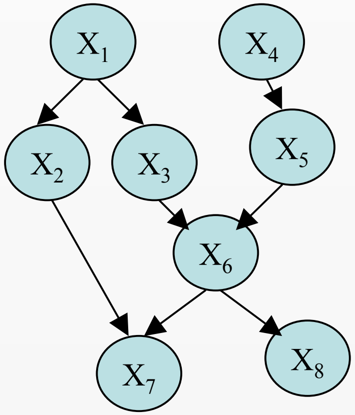
- Bayesian networks - Special type of probabilistic graphical model representing the conditional dependency relationships between random variables.

- Components of the model
  - Random variables which represent the nodes of the graph.
  - Directional edges between pairs of nodes which represent the dependencies among the random variables.

- Representation of the form '*X* → *Y*' in graph structure indicates that node *X* is the parent of node Y and has a direct influence on node *Y*.

- Graphical Models popular with Statistical and Artificial Intelligence communities.

- Applications
  - Finance
  - Robotics
  - Data Mining
  - Bioinformatics
  - Weather Forecasting
  - Medicine

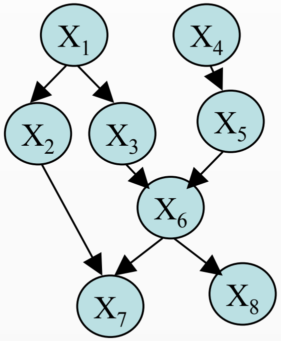


# Definition (Contd.)

- **Two components** are needed to specify a Bayesian network
  - **Directed Acyclic Graph (Structure)**: Represented as an adjacency matrix to specify the graph structure of the Bayesian Network.

    E.g.

$$Model = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



  - **Conditional Probability Distribution (Parameters)**: Parameter to be specified for every node in the network. This parameter depends on the type of nodes in the network. For discrete nodes, this distribution is specified as a table for every node in the network – Conditional Probability Table (CPT).

    E.g **CPT for node $X_1$**

| P($X_1$=0) | P($X_1$=1) |
|---|---|
| 0.5 | 0.5 |

**CPT for node $X_2$**

| $X_2$ | $X_1$ | P($X_2$\|$X_1$) |
|---|---|---|
| 0 | 0 | 0.25 |
| 0 | 1 | 0.6 |
| 1 | 0 | 0.75 |
| 1 | 1 | 0.4 |

# Key Concepts

- **Conditional Independence**: Two random variables *A* and *B* are conditionally independent given another variable *C* if

  P(A| C) = P(A| B,C)

- **D-Separation**: Conditional Independence in the graphs represented by the property of D-Separation.
  - Two nodes *X* and *Y* are *d*-separated in the graph, given specified evidence nodes, if and only if variables *X* and *Y* are conditionally independent given the corresponding evidence variables.
- **Markov Independence**: States that

  "In a graph representing the Bayesian network, a random variable is independent of its non-descendants given its parents in the graph".

  E.g. Consider random variable *B* in the graph. Its parent set is {*A*, *E*}. Its children set is {*C*}.

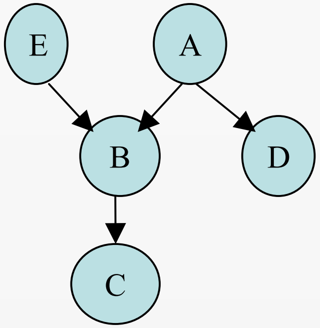
  Non-descendant set is {D}. Based on the Markov condition:

$$P(B \mid A,E) = P(B \mid A,E,D)$$

  This is represented as *I(B; D| A, E)*

  Similarly for the other nodes, the Markov independence relations are

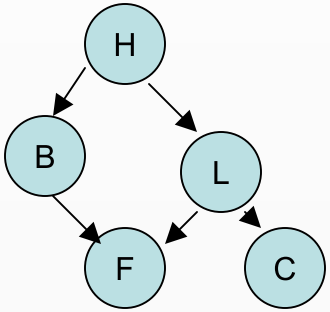
  *I(A; E); I(C; A, E, D| B); I(D; E, B, C | A); I(E;A, D)*



# Inference & Learning in Bayesian Networks

- **Inference**: Given a Bayesian network, determining the probabilities of particular types of events is known as Inference.

  E.g. This network illustrates the joint probability distribution of smoking history (H), bronchitis (B), lung cancer (L), fatigue (F), Chest X-ray (C).



From this network, using the probabilities of the CPT's of the nodes, we can infer the probability of events like If a person has a smoking history and a positive X-ray, what is the probability of that patient having lung cancer i.e. P(L| H,C).
This type of computation is known as inference.

- **Learning**: Given a database of state values sampled independently from the above network, determining the network structure or the network parameters is called learning.

$$Data = \left[\begin{array}{ccccc} H & B & L & F & C \\ 1 & 2 & 1 & 2 & 1 \\ 2 & 2 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 & 1 \\ 2 & 2 & 2 & 2 & 1 \end{array}\right]$$

Data samples sampled randomly from a BN. Two types of learning

- *Parameter Learning*: Given the structure (dependency model), learning the conditional probability tables
- *Structure Learning*: Learning the structure (graphical model or the adjacency matrix of the graph)

- The main focus of this thesis is Structure learning in Bayesian networks.

# Structure Learning in Bayesian Networks

- Problem statement for the Structure Learning of Bayesian Networks:
  - Given (Input):
    - A set of random variables $X_1, X_2 \ldots\ldots X_n$
    - A dataset of complete cases generated from some joint probability distribution P($X_1, X_2 \ldots\ldots X_n$)
  - Result (Output): The network structure (adjacency matrix of the graph) of the Bayesian Network that most likely generated the observed data.
- Two approaches to structure learning:
  - Constraint Based Methods:
    - Starts off with a fully connected graph and removes edges if conditional independencies are measured in the data.
    - Methods cannot handle missing data.
    - Repeated statistical tests tend to lose their statistical power especially in case of larger networks.
  - Search and Score Methods:
    - Searches the space of all possible DAG's and uses a score function to evaluate each stage of the search process.
    - Drawback of this method is that the size of the search space is super-exponential in the number of variables in the network.
    - This method can be tweaked with a few assumptions to reduce the super-exponential search spaces.
    - The BIC (Bayesian Information Criterion) score and Minimum Descriptor Length (MDL) score are the most popularly used scoring criterion.

# Search and Score Methods

- **Exhaustive Search Method**
  - Employed for small Bayesian networks
  - Score function used to search exhaustively over the entire search space.
- **Hill Climbing**
  - Starts off with a particular point in the search space (an undirected graph)
  - All the nearest neighbors of this point are considered.
  - Nearest neighbors of a particular graph (A) are the graph structure which differ from A by only a single edge addition, edge deletion or an edge reversal.
  - The algorithm applies a score function to each of these neighbors and chooses the neighbor with the highest possible score as the next iterating point.
  - The process stops when no neighbor has a score higher than the previous iterating point.
  - Extremely prone to falling into local maxima.
- **K2 Algorithm**
  - **Reduces the complexity associated with the search techniques by requiring a prior ordering of nodes.**
  - **Algorithm found to be extremely efficient in determining Bayesian network structures.**
  - **Starting point of our proposed method.**

# K2 Algorithm

- **Proposed by Cooper and Herskovits in 1992.**
- **K2 algorithm searches for a DAG that approximates maximizing score *$score_B(d, G)$*.**
- **Input – Output Relationship – K2 Algorithm**
  - Given inputs
    - Node ordering in which most of the parent nodes in the ordering appear before their respective children.
    - Upper bound on the number of parents each node can have. (typically '*n-1*')
    - Input Data.
  - Output
    - Structure of the Bayesian network that most likely generated the observed data.
- **Node ordering Assumption**
  - The node ordering is such that if a node $X_i$ precedes the node $X_j$ in the ordering an arc from the node $X_j$ to $X_i$ is not allowed.
  - Stated otherwise, for each node in the ordering, the nodes that occur downstream of it cannot be one in its parent set.
  - *Pred* $(X_i)$ is a set that is computed for every node during the algorithm and it includes the nodes that precede a node $X_i$ in the ordering.
- **Functional Description:**
  - ***Network_Structure = learn_K2 (Data, Node_Sizes, Node_Order);***

# K2 Algorithm

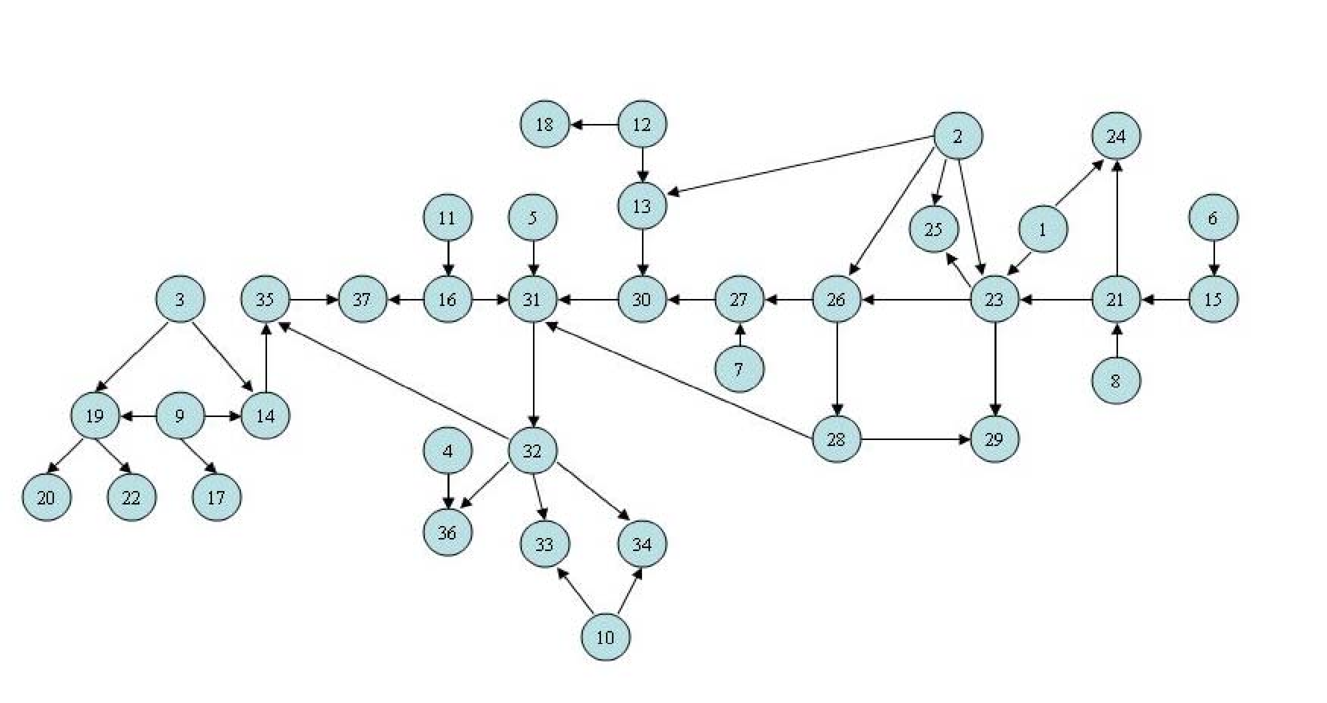
- **Working**
  - The parent set $PA_i$ of node $X_i$ is initially set to an empty set.
  - Using the scoring function the network score, $score_B(d, X_i, PA)$ is computed.
  - The nodes in the ordering are now visited one at a time in sequence.
  - For each node visit, Pred ($\mathrm{X}_\mathrm{i}$) is computed and represents the set of potential parents of node $\mathrm{X}_{\mathrm{i.}}$
  - The node in Pred ($X_i$) which most increases the network score is greedily added to the parent set of node $X_{i.}$
  - The addition of the parents continue until
    - The maximum number of parents for that particular node has been reached.
    - There are no more legal parents to add.
    - No parent addition improves the score.
  - This algorithm terminates when all the nodes in the node ordering have been visited once.
- **Deficiencies**
  - Algorithm performance greatly dependent on the input node ordering.
  - Only when domain knowledge from an expert is available can a correct input ordering be determined.
  - Using all possible node ordering combinations is computationally impossible.

# Proposed Method – Features and Overview

- Focus of this thesis was to develop a method which would uncover this node ordering from the data and be as accurate as possible in adhering to the fact that most parent nodes in this uncovered ordering should appear upstream to their children.
- The prime reason to undertake such an approach is to use the inherent efficiency displayed by the K2 algorithm in determining Bayesian network structures from data.
- The proposed method uses concepts from
  - Information Theory: Mutual Information, Conditional Entropy etc.
  - Bayesian Network Theory: Conditional Independence, D-Separation etc.
  - Graph Theory: Path Matrices, Connectivity Structures etc.

# Method Overview

- The following section illustrates the various stages of the proposed method.
- The ALARM network is used as an example to illustrate the algorithm development.
- ALARM – Bayesian network with 37 discrete random variables and each variable taking on 2, 3 or 4 discrete states.



# Phase I – Mutual Information Stage

- Determine an undirected structure (skeleton structure) from the data using the MI parameter.
- The mutual information between two random variables is defined as the amount of information shared between the two variables. Mathematically,

$$I(X;Y) = H(X) - H(X|Y)$$

where I(X;Y) : MI shared between variables X and Y

H(X) : Entropy of the random variable X

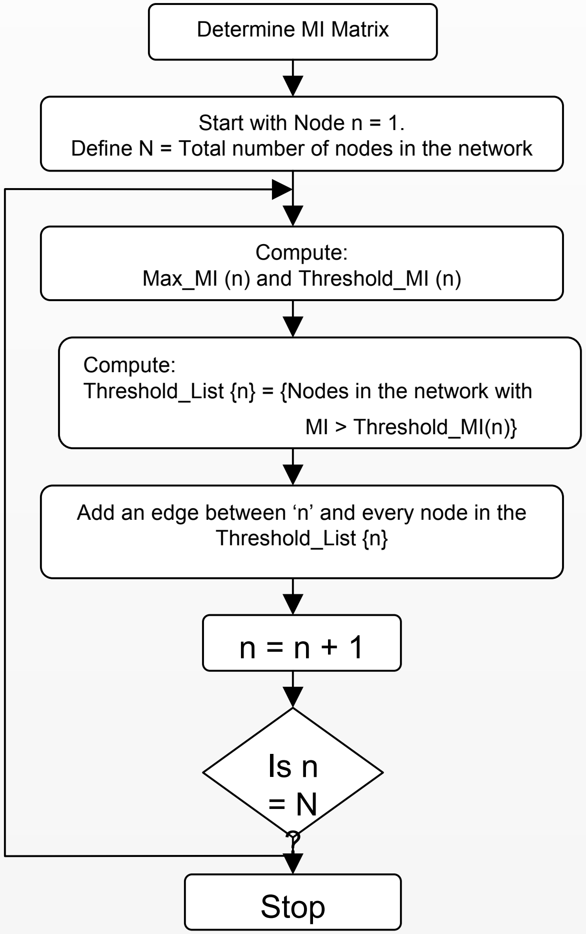
H(X|Y) : Conditional Entropy of the random variable X given variable Y.

- MI determines the degree of proximity between random variables.
- If the mutual information between variables *X* and *Y* is zero, it implies that the two variables are independent of each other and that variable *X* does not contain any information about the variable *Y* and vice-versa.
- Higher the mutual information between two variables *X* and *Y*, more closely they are related.
- MI is a metric that is symmetric in nature i.e. *I(X; Y) = I(Y; X)*
- Matrix visualization of a dataset

$$AlarmData = \begin{bmatrix} 1 & 1 & 3 & . & . \\ 2 & 2 & 4 & . & . \\ . & 4 & 3 & . & . \\ . & . & . & . & . \\ 37 & 1 & 2 & . & . \end{bmatrix}$$

# Phase I (Contd.)

- MI for each node in the network is computed from the data (w.r.t every other node) and a Mutual Information matrix is computed.
- Flowchart to obtain the undirected network structure from the MI parameter.



Determine MI Matrix

Start with Node n = 1.
Define N = Total number of nodes in the network

Compute:
Max_MI (n) and Threshold_MI (n)

Compute:
Threshold_List {n} = {Nodes in the network with MI > Threshold_MI(n)}

Add an edge between ‘n’ and every node in the Threshold_List {n}

n = n + 1

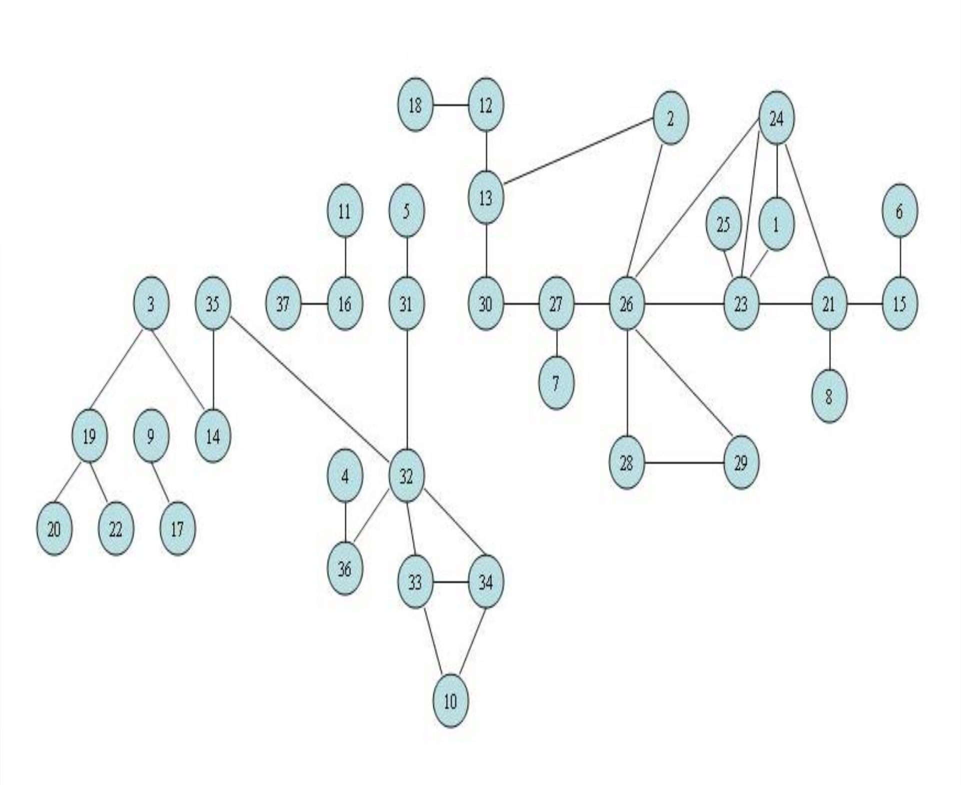
Is n = N ?

Stop

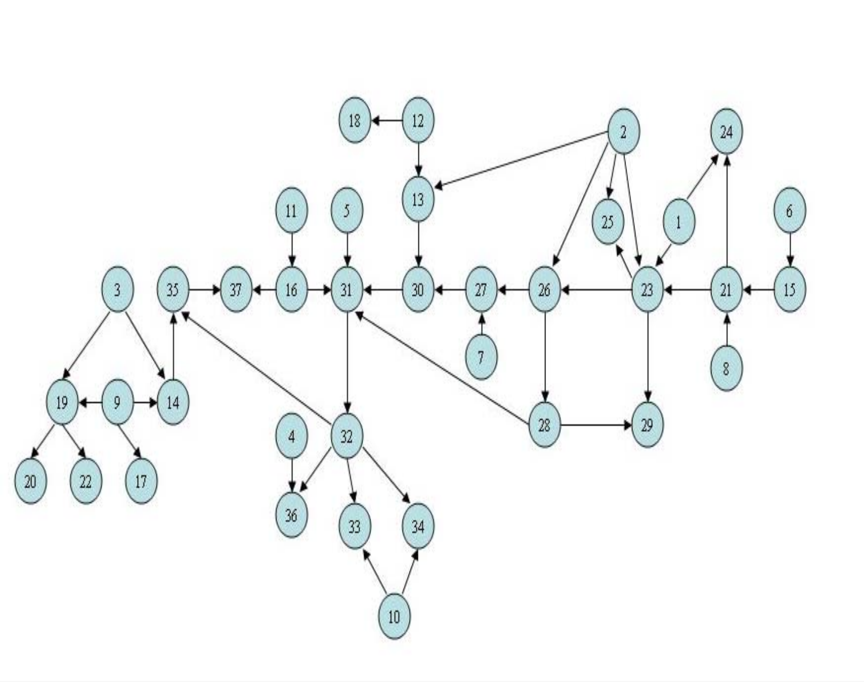
$$MI\_MAT = \begin{bmatrix} I(1,1) & I(1,2) & I(1,3) & ..\ ..\ ..\ .. & I(1,37) \\ I(2,1) & I(2,2) & I(2,3) & ..\ ..\ ..\ .. & I(2,37) \\ I(3,1) & I(3,2) & I(3,3) & ..\ ..\ ..\ .. & I(3,37) \\ .. & .. & .. & ..\ ..\ ..\ .. & .. \\ .. & .. & .. & ..\ ..\ ..\ .. & .. \\ .. & .. & .. & ..\ ..\ ..\ .. & .. \\ .. & .. & .. & ..\ ..\ ..\ .. & .. \\ I(37,1) & I(37,2) & I(37,3) & ..\ ..\ ..\ .. & I(37,37) \end{bmatrix}$$

# Phase I (Contd.)

- **Undirected network structure**

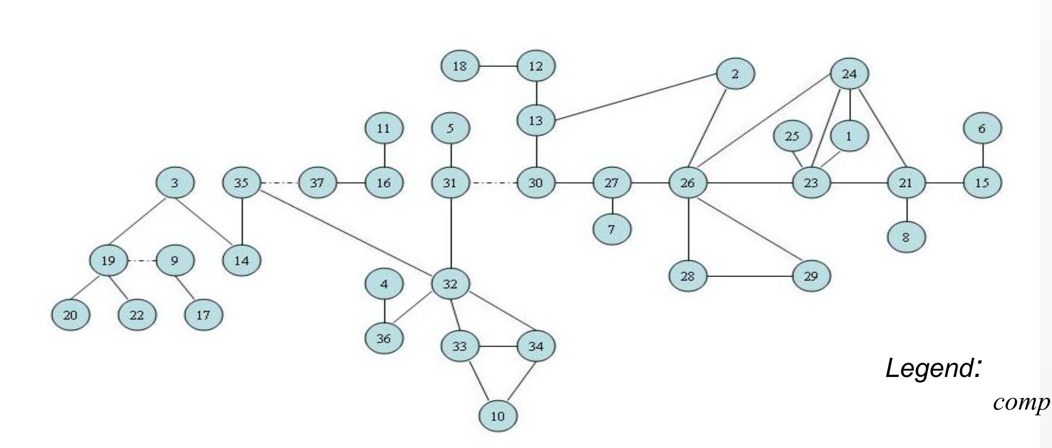


**Original ALARM Network**



# Phase I (Contd.)

- Undirected network structure obtained is not completely connected. Few isolated nodes remain. We obtain a completely connected structure by connecting these isolated nodes back into the network.
- Steps to obtain a completely connected network structure:
  - Sort the computed pair-wise mutual information in descending order.
  - Create a list containing the node pairs sorted in descending order of mutual information.
  - Add an edge between the nodes in the node-pairs if any one of the following conditions are satisfied:
    - The target node is not connected to any other node in the network.
    - The source node is not connected to any other node in the network.
    - Starting from the source node, the target node is not reachable through any other path of nodes of the network structure obtained until that step
  - Stop the processing of the node pairs when the complete connectivity criterion is satisfied.



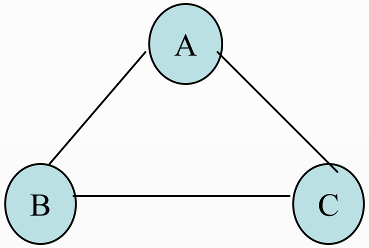
Legend: → *Edges added to establish complete connectivity*

# Phase II

- **Summarizing Phase I**
  - Obtain an undirected network structure using the MI parameter.
  - Complete the connectivity of this structure to remove isolated nodes in the network structure.
  - Network is still undirected at this stage.
  - Structure is called MIUDG structure at this stage.
  - About eighty five percent of the edges are correctly detected; directional orientations are not considered at this stage
- **Phase II: Elimination of triangular loops in the network structure**
  - Triangular loops in the network structure formed due to erroneous connections between two nodes in the network.
  - They are formed due to a high value of MI between the nodes.
  - Consequently these erroneous edges need to be eliminated at this stage of the algorithm to prevent them from further propagating through the remaining phases.
  - Another reason is to eliminate the loops in the network structure thus satisfying the acylic property of the Bayesian networks.
  - Elimination is done in two stages:
    - Common Edge elimination : Elimination of edges that are commonly involved in two or more triangular loops
    - Elimination by conditional independence tests

# Elimination by CI Tests

- Isolated triangular loops are eliminated by this method.
- The working of this method is illustrated with an example triangular loop.
- Markov independence used to eliminate the erroneous edge.
- Edges eliminated one at a time and the conditional independence is tested after the elimination.
- If the equality holds then that particular edge can be deleted and the triangular loop is broken.



$$P(A \mid C) = P(A \mid B, C) \quad \textit{Test after deleting } A - B$$

$$P(A \mid B) = P(A \mid C, B) \quad \textit{Test after deleting } A - C$$

$$P(B \mid A) = P(B \mid C, A) \quad \textit{Test after deleting } B - C$$

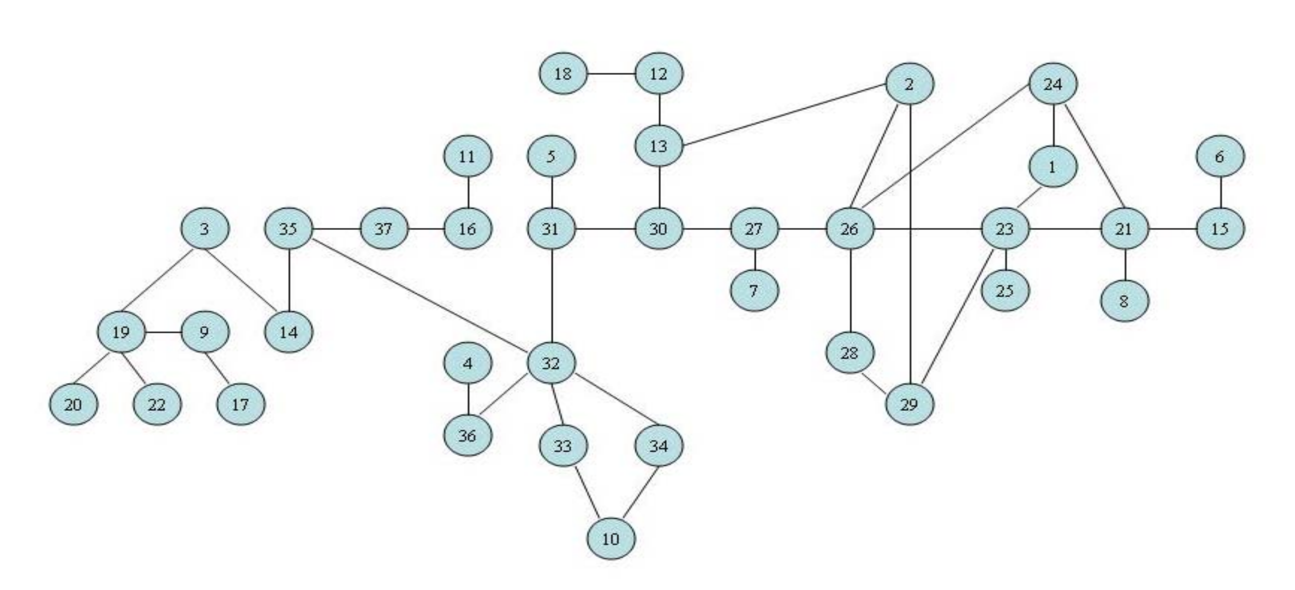
- If none of the CI tests hold, additional nodes are added in the equality criterion to test for the conditional independence. These nodes are obtained from the MI lists and the equalities are tested using these new rules.
- For e.g. the following equality is tested after the deletion of edge A-B

$$P(A \mid C, X) = P(A \mid B, C, X) \quad \textit{Deleting the edge between } A - B$$

- Here X is a node set obtained from the MI matrix.
- If the CI tests still fail to produce a triangular loop free structure, an exhaustive search is performed using the score function of all the sub-structures involved in the triangular loop to eliminate them.

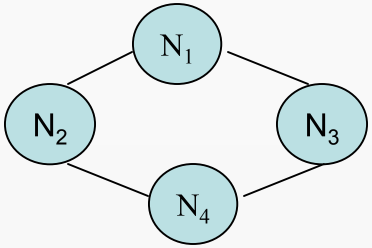
# Phase II (Contd.)

- Elimination using local sub-structures
  - In this method, a sub-graph consisting of all nodes connected to the concerned isolated triangular loop are considered for the evaluation.
  - Each edge of the triangular loop is eliminated one at a time and the score function is used to compute the score of all the sub-structures resulting from the deletion.
  - The maximum sub-structure score is determined and the edge deletion corresponding to this maximum score is finalized to be the edge deletion that needs to be performed on the isolated triangular loop
- After the elimination of all triangular loops, the example ALARM network is:



# Phase III

- After phase II we have an undirected structure that is free of triangular loops.
- We need to assign directional orientations to the edges in the network and then sort topologically to obtain a node ordering that can be used as an input to the K2 algorithm.
- Even after the elimination of triangular loops in the network structure, loops with four nodes and four edges are distinctly possible.
- Phase III of our algorithm aims at detecting these loops and assigning directions to them.
- Determination of four edge loops
  - Obtain the square of the adjacency matrix of the undirected graph structure (structure after the elimination of triangular loops).
  - For each node in the network, determine the entries in the corresponding row of the squared matrix that are equal to two.

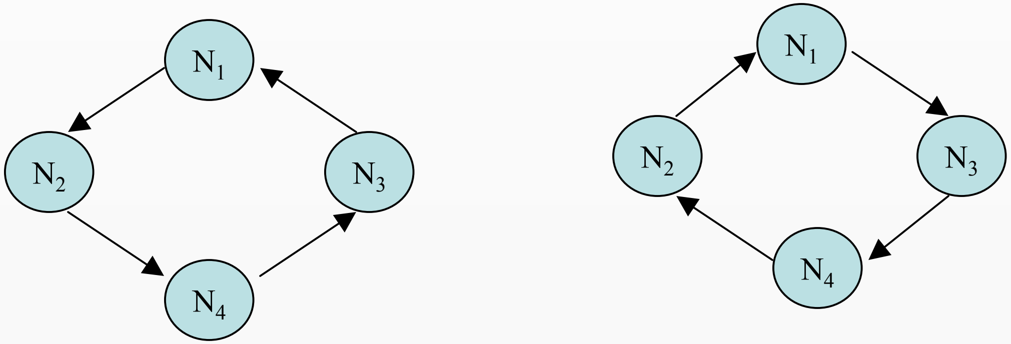


$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad B = A^2 = \begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix}$$

  - Initially, look at the entries in row one (corresponding to node *N1*) that are equal to two. Only the column four (node *N4*) has an entry equal to two (exclude the entry two in column one, this corresponds to node *N1*). This means that node *N4* can be reached from node *N1* in two paths of path length two. These are:
    - ***$N_1 – N_2 – N_4$***
    - ***$N_1 – N_3 – N_4$***
  - Consequently nodes *$N_1 – N_2 – N_3 – N_4$* are a part of a four edge loop.

# Phase III (Contd.)

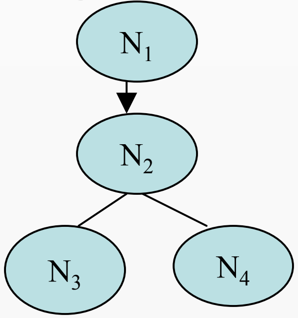
- Assignment of edge orientations to four-edge loops
  - Exhaustive search and score method is used to assign directions to the edges in the cyclic loops.
  - Acyclic loops eliminated while using the search and score method.
  - Structure size of the exhaustive search space for four-edge loops = $2^4 – 2 = 14$



**Structures to be omitted from the exhaustive search space while assigning directions to the edges in four-edge loops.**

# Phase IV

- After Phase III
  - Only the edges involved in four-edge loops are oriented.
  - We need to find a method to orient the remaining edges in the network.
- In Phase IV, we orient the remaining edges in the network using
  - Conditional Independence Tests (CI Tests)
  - Graph sub-structure splitting
- Assignment of edge orientations using CI tests
  - CI Tests performed to assign edge orientations.
  - Consider,



  - The edge orientation from $N_1$ → $N_2$ has already been set during Phase III.
  - Node $N_2$ has two edges (to node $N_3$ and $N_4$) that do not have any directional orientation at this stage.
  - We are interested in determining the orientations of these edges using CI tests.

# Phase IV (Contd.)

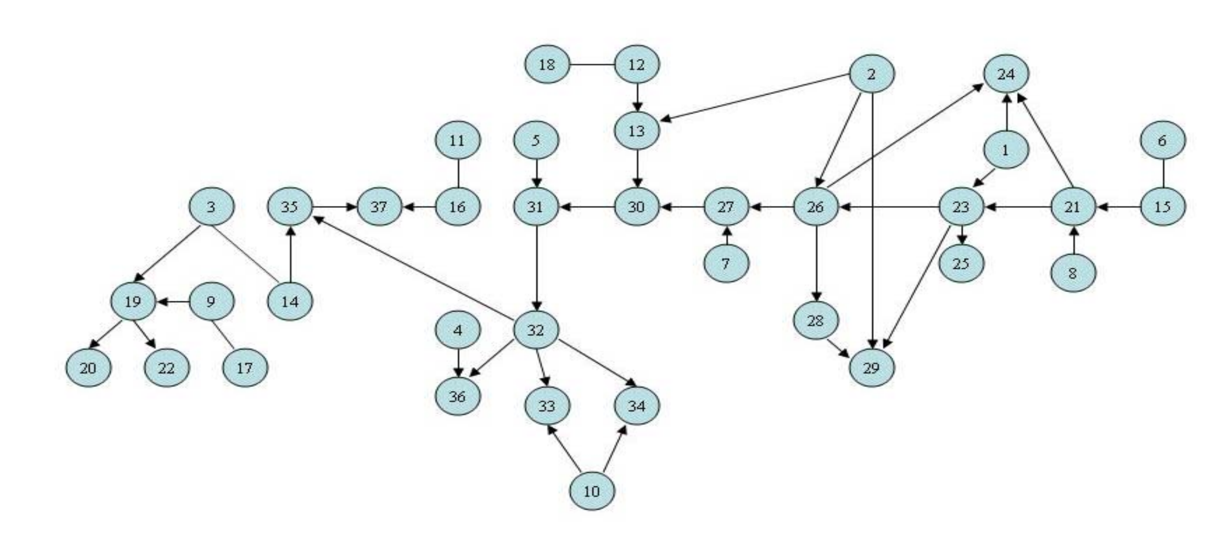
- Assignment of edge orientations using CI tests
  - Assume we are interested in assigning an orientation to the edge $N_2 – N_3$.
  - We test the equality,

$$P(N_3 \mid N_2) = P(N_3 \mid N_2, N_1)$$

  - If the equality holds then $N_3$ is conditionally independent of $N_1$ given $N_2$.
    - Using the rules of D-separation we can assign the direction $N_2$ → $N_3$ in this case.
  - If the equality does not hold then $N_3$ and $N_1$ are not independent conditional on $N_2$. Two reasons might exist for this inequality:
    - $N_3$ and $N_1$ are truly dependent. In this case, the edge orientation assignment can be made from $N_3$ → $N_2$ using the rules of D-separation.
    - There exists another path from *N3* to *N1*. We add the highest ranking common potential parents to nodes $N_1$ and $N_3$ at this stage using the MI metric.
- Steps to assign edge orientations using CI tests
  - After setting the edge orientations for the edges in the loops, sort the nodes in descending order of incoming connections.
  - For each node in this list, determine the edges that have no directional orientation.
  - Use the CI tests for each node in the list to assign edge orientations (as shown in the example above).
  - This is a dynamic process as after every assignment the network structure is changing. Iterate this process around ten times so that the edge orientations propagate through the network structure.

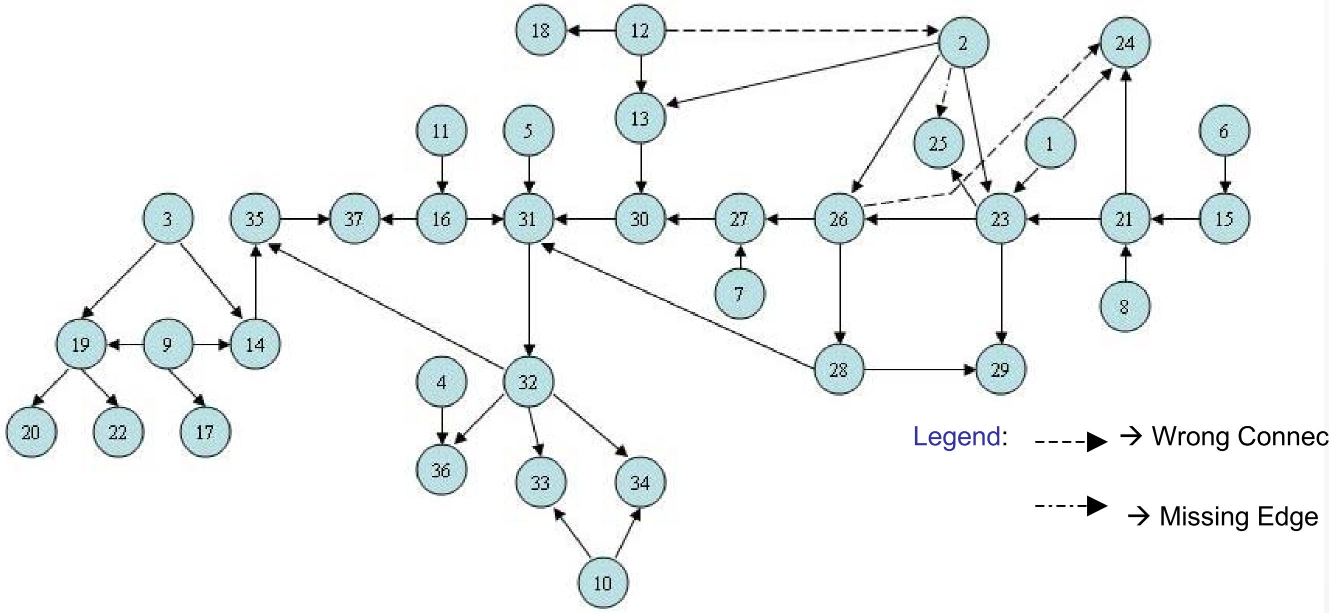
# Phase IV (Contd.)

- Graph Sub-structure splitting
  - Performed to orient edges after the CI test stage.
  - Main idea is to split the large network structure into smaller local sub-structures and exhaustively search over these sub-structures.
  - Searching over the local sub-structures is not computationally demanding.
  - Structure with N edges is split into several small structures with ‘n’ edges where n << N.
  - For each of these smaller structures, all possible directional orientations are considered.
  - Scores for these sub-structures are computed using the score function.
  - Directional orientations are set in the graph structure corresponding to the directional orientations of the sub-structure with the maximum score.



# Phase V

- After, Phase IV
  - We have edge orientations for most edges in the network
  - We sort the nodes topologically (i.e. parents before their children) using the edge orientations in the structure obtained to obtain a node order.
  - Using this node order, we apply the K2 algorithm to determine the final Bayesian network structure.



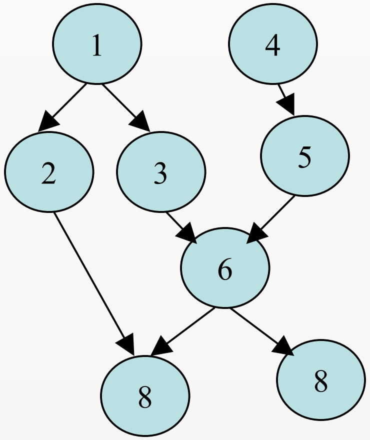
Legend: ----▶ → Wrong Connection (Extra edge)

-·-·▶ → Missing Edge

**Final Learnt Structure of the ALARM network – After the K2 algorithm.**

# Results and Analysis

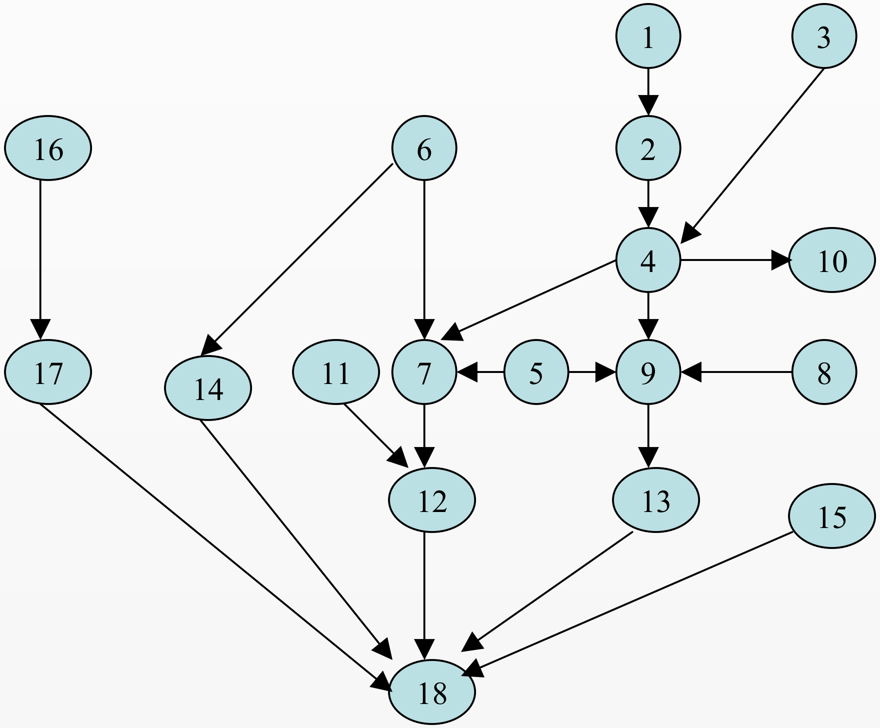
- Parameters considered while testing the algorithm
  - Overall algorithm efficiency: Efficiency judged based on the number of errors in the graph structure or the Hamming distance between the structure discovered and the gold structure.
  - Timing efficiency: Time to learn defined as the time taken to learn the final structure of the Bayesian network from data.
  - Comparison of the algorithm with standard BN structure learning algorithms.
  - Robustness: Determined by how well the algorithm performs for networks of varying sizes.
- Four networks of varying sizes used to test the algorithm performance
  - ASIA: Small Bayesian network – 8 nodes and 8 edges.
  - ASIA is a small Bayesian network that calculates the probability of a patient having tuberculosis, lung cancer or bronchitis respectively based on different factors.



Eight random variables - discrete in nature.

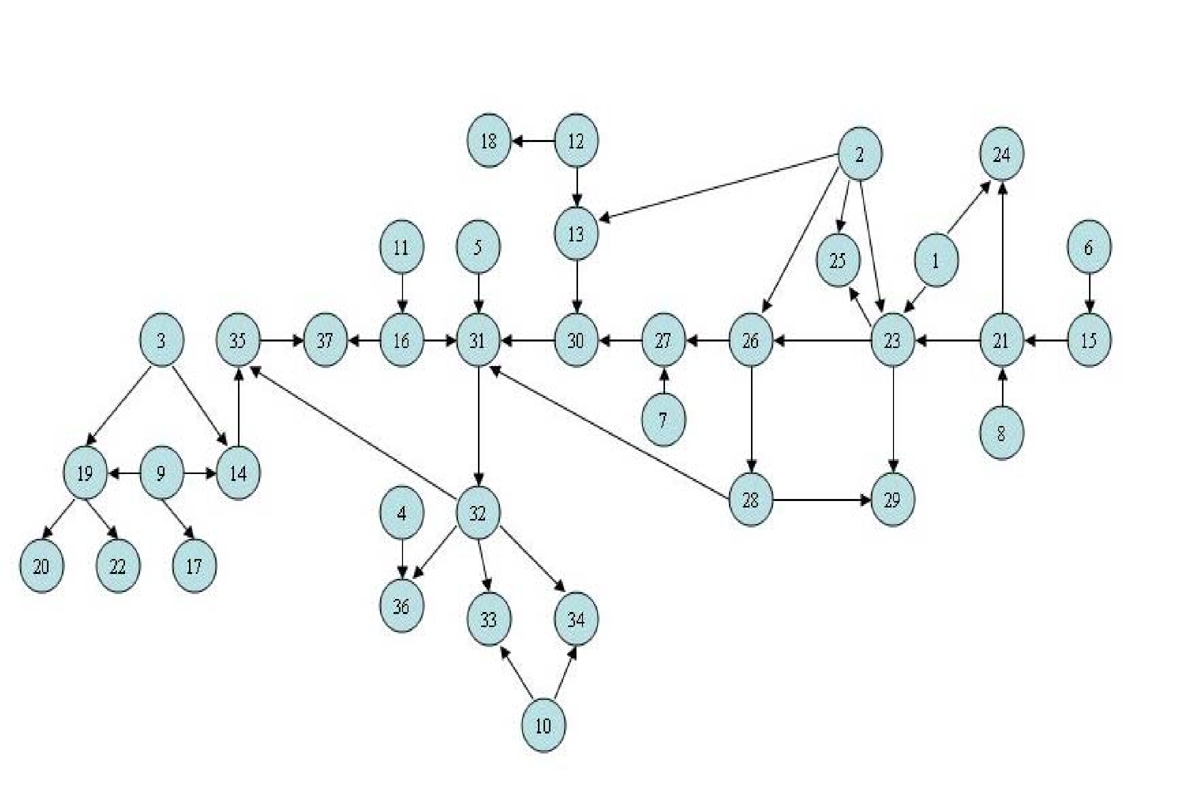
# Testing

- CAR_DIAGNOSIS: Medium sized Bayesian network – 18 discrete nodes and 20 edges.
- The CAR_DIAGNOSIS network is a mid-sized network that is used to diagnose why a car won't start, based on spark plugs, headlights, main fuse, etc.
- The variables in the network can take either two or three states.



# Testing

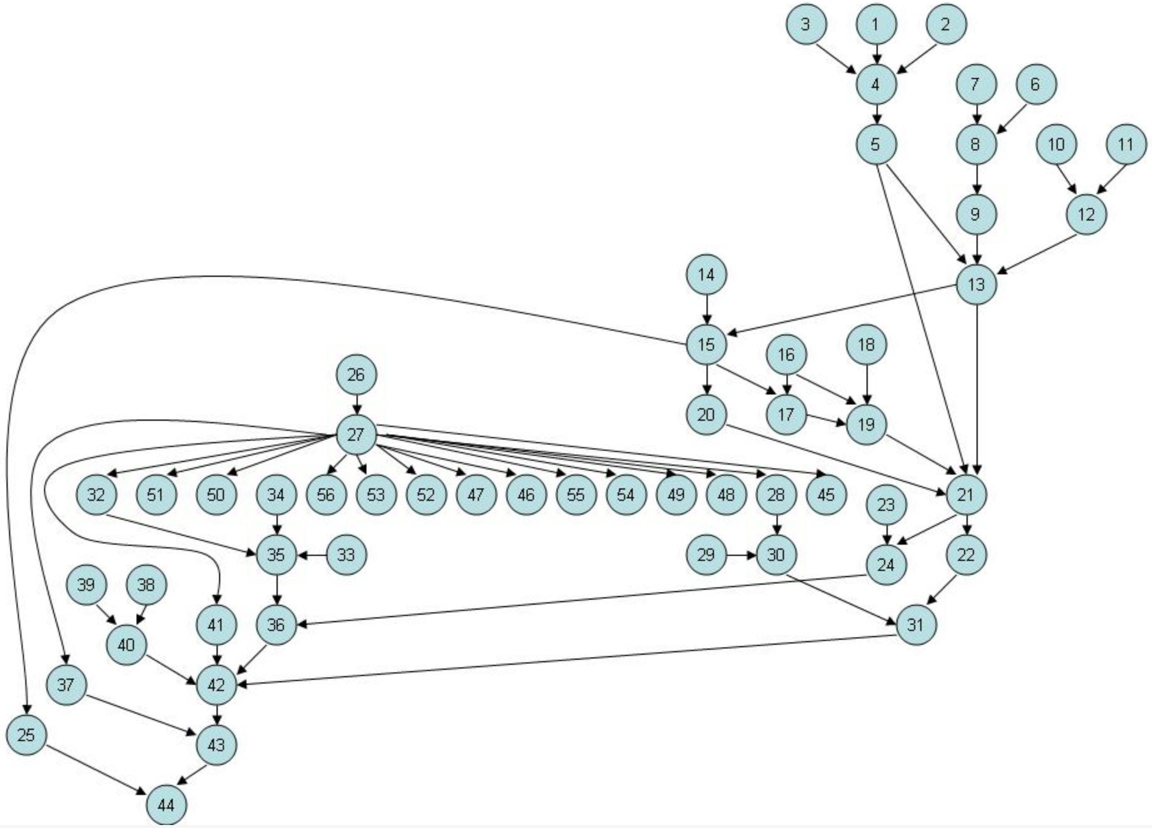
- ALARM: Large Network – 37 nodes and 46 edges
- Stands for ‘A Logical Alarm Reduction Mechanism’ and is a medical diagnostic system used for patient monitoring. Most commonly used network in Bayesian network testing.
- Discrete variables take two, three or four states respectively.



# Testing

- Hailfinder: Large network – 56 nodes and 66 edges
- Complex network used to predict summer weather in North Eastern Colorado.
- The discrete nodes take two, three or four states respectively.

Original Hailfinder Network - I



# Testing

- To obtain a fair idea of algorithm performance, hundred datasets are randomly generated for each network and the algorithm is run on each of these datasets to obtain a mean result of the algorithm performance.
- For comparing the graphical efficiency, a few terms are defined here
  - Correct edges: Edges correctly detected by the algorithm (with the right orientation) in comparison to the Gold network.
  - Missing edges: Edges not picked up by the algorithm in comparison to the gold network.
  - Wrong orientation edges: Edges detected by the algorithm but having the opposite orientation in comparison to the gold network.
  - Wrong connection edges: Edges that are present in the structure detected by the algorithm but not present in the Gold network
  - Graph errors – Hamming distance - Summation of the three types of graph errors mentioned above (Missing edges + Wrong orientation edges + Wrong connection edges).
- Ideally, a structure learning algorithm must uncover the maximum number of Correct edges with a minimum number of errors.

# Results – ALARM Network

- The table below shows the performance of the algorithm using 10000 data samples for the ALARM Network
- A high percentage of correct edges are detected (42.93/46 about 94 %).
- For a ideal dataset, a structure with only 1 error  is detected.
- Results for the ALARM Network

| | Mean results | | Best result |
|---|---|---|---|
| | Mean | SD | |
| Correct edges | 42.93 | 2.18 | 45 |
| Missing edges | 0.9 | 0.3 | 1 |
| WO Edges | 2.17 | 2.13 | 0 |
| WC Edges | 5.87 | 4.86 | 0 |
| Hamming | 8.94 | 6.15 | 1 |
| Score | -94983.4 | 576.3 | -94412.8 |

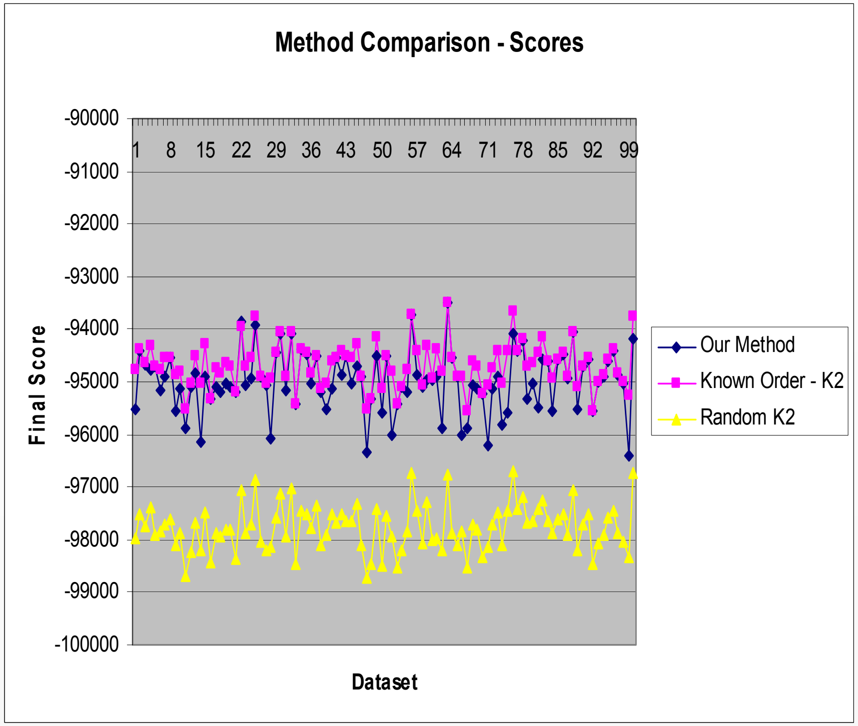
Comparison Table with other methods

| | Our Method | | Random K2 | | Hill Climbing | |
|---|---|---|---|---|---|---|
| | Mean | SD | Mean | SD | Mean | SD |
| Correct | 42.93 | 2.18 | 21.87 | 0.61 | 21 | 4.24 |
| Missing | 0.9 | 0.3 | 2.36 | 0.25 | 2.5 | 0.71 |
| WO | 2.17 | 2.13 | 21.77 | 0.54 | 22.5 | 4.94 |
| WC | 5.87 | 4.86 | 40.78 | 1.45 | 12 | 2.82 |
| Hamming | 8.94 | 6.15 | 64.92 | 1.66 | 37 | 7.07 |
| Score | -94983.4 | 576.3 | -97774.7 | 440.2 | -95450.7 | 457.7 |

- Key points to be noted:
  - The performance of other methods like Random K2 and Hill Climbing are significantly worse than the proposed method both in terms of the overall network score and the graph errors in the learnt structure.
  - For a suitable dataset the number of graph errors in the learnt structure using the proposed method is one.

# Results – ALARM Network

- Score Comparison graph
  - The scores of the proposed method mirror the known K2 method (wherein the correct preordering of nodes is supplied as the input).
  - For a few datasets the proposed method performs exactly alike the known K2 algorithm.
  - The performance of the random K2 ordering is comparatively very poor.
- Time Complexity
  - The proposed method performs much better with regards to time to learn the final network structure.
  - These times indicate the time taken to learn a single network structure from observed data.
  - All computations performed on a 1 GB RAM, Pentium Xeon IV system.



| Method | Time to Learn |
|---|---|
| PM | 969 sec |
| Hill Climbing | 54540 sec |

# Results – ASIA Network

- ## Results of the Proposed Method

| | Mean results | | Best result |
|---|---|---|---|
| | Mean | SD | |
| Correct edges | 7.21 | 0.86 | 8 |
| Missing edges | 0.41 | 0.62 | 0 |
| WO Edges | 0.38 | 0.75 | 0 |
| WC Edges | 0.46 | 0.97 | 0 |
| Hamming | 1.25 | 1.69 | 0 |

## Comparison Table with other methods

| | Our Method | | Random K2 | | Hill Climbing | |
|---|---|---|---|---|---|---|
| | Mean | SD | Mean | SD | Mean | SD |
| Correct | 7.21 | 0.86 | 3.41 | 0.2 | 3.75 | 1.7 |
| Missing | 0.41 | 0.62 | 1.32 | 0.4 | 2.5 | 0.65 |
| WO | 0.38 | 0.75 | 3.26 | 0.25 | 3.29 | 1.75 |
| WC | 0.46 | 0.97 | 4.67 | 0.24 | 1.68 | 1.31 |
| Hamming | 1.25 | 1.69 | 9.27 | 0.28 | 5.93 | 2.80 |

- Comparing the tables it can be noted that
  - In comparison with the other methods, the proposed method produces far fewer errors on an average in the learnt structure.
  - The proposed method is able to detect the entire network structure (without any errors) in the case of certain datasets.
- Time Complexity
  - Although, in the case of smaller networks time is not a very constraining factor the proposed method performs better than the Hill climbing method w.r.t. time complexity.

| Method | Time to Learn |
|---|---|
| PM | 24 sec |
| Hill Climbing | 62 sec |

# Results – CAR_DIAGNOSIS

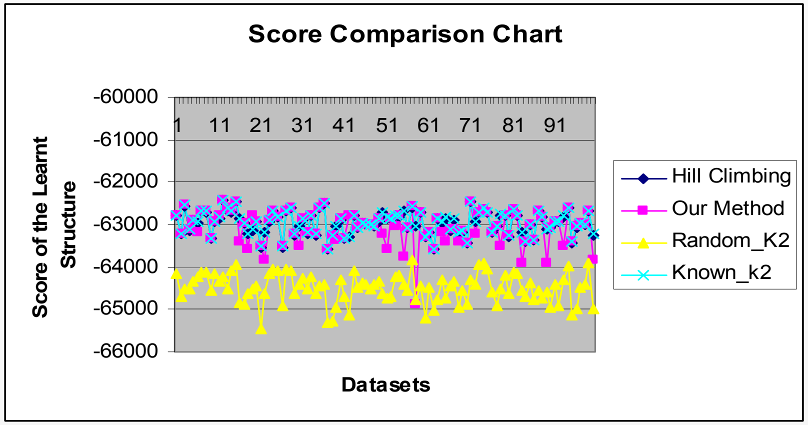
- **Results for the CAR_DIAGNOSIS network**

| | Mean results | | Best result |
|---|---|---|---|
| | Mean | SD | |
| Correct edges | 18.56 | 0.95 | 19 |
| Missing edges | 1.29 | 0.62 | 1 |
| WO Edges | 0.15 | 0.45 | 0 |
| WC Edges | 0.83 | 1.47 | 0 |
| Hamming | 2.27 | 2.26 | 1 |
| Score | -63061 | 397.2 | -62984 |

**Comparison Table with other methods**

| | Our Method | | Random K2 | | Hill Climbing | |
|---|---|---|---|---|---|---|
| | Mean | SD | Mean | SD | Mean | SD |
| Correct | 18.56 | 0.95 | 8.58 | 0.11 | 14.91 | 1.51 |
| Missing | 1.29 | 0.62 | 3.65 | 0.37 | 1.07 | 0.25 |
| WO | 0.15 | 0.45 | 7.75 | 0.33 | 4.02 | 1.46 |
| WC | 0.83 | 1.47 | 16.24 | 1.09 | 1.59 | 1.05 |
| Hamming | 2.27 | 2.26 | 27.65 | 1.07 | 6.68 | 2.43 |
| Score | -63061 | 397.3 | -64498 | 342.4 | -62961 | 266.37 |

- **For a medium sized network**
  - The proposed method performs consistently better than the standard methods with regards to graph error. With regards to score the proposed method performs very close to the known order K2 algorithm. It can be seen from the graph that the Hill Climbing method also performs quite similarly w.r.t score of the learnt structure.



# Results - Hailfinder

- **The Hailfinder network is an extremely large and a complex Bayesian network.**
- **The conditional probability tables of some nodes in this network are filled with absolute values of probabilities (0's and 1's). This makes the network structure determination extremely difficult.**
- **A lack of available literature also makes it difficult to compare the performance of structure learning algorithms for the Hailfinder network.**
- **Cheng's Bayesian network Powerconstructor is considered to be a universally accepted software to construct Bayesian network structures from data.**

**Comparison of the Proposed Method to other structure learning algorithms**

| | Proposed Method | Cheng BNP | Random K2 | Known K2 |
|---|---|---|---|---|
| Correct | 48 | 43.6 | 27.2 | 60 |
| Missing | 11 | 15.4 | 18.2 | 6 |
| WO Edges | 7 | 6 | 20.2 | 0 |
| WC Edges | 8.25 | 6.8 | 32.4 | 10 |
| Hamming | 26.25 | 28.2 | 70.8 | 16 |

- It follows from the above table that while the proposed method does not compare to the known order K2 algorithm, it performs better than other structure learning algorithms like random K2 and the universally accepted BNP software method.
- Methods like Hill Climbing are computationally impossible to run for large and complex networks like Hailfinder.

# Conclusions & Future Work

- **The proposed thesis developed**
  - A time and a graph efficient structure learning algorithm that is robust and can be used to learn the structures of different sizes of networks effectively from data.
  - Uses the inherent efficiency built into the K2 algorithm but does away with the need for prior knowledge of node ordering.
  - Uses key Bayesian network principles like Conditional Independence, Markov Properties and D-separation.
  - Incorporates concepts of Information theory into the learning algorithm.
- **Application Area – Bioinformatics**
  - Use of Bayesian networks in Bioinformatics is a hot research area.
  - The proposed method was applied to discover the structure of the gene regulatory networks using publicly available microarray data.
  - The method was found to be efficient in uncovering the genetic interactions (about 86% and 64% interactions for two groups of tested genes).
  - This work was published as a paper in a journal.
  - The proposed ideas can be extended in general in the field of Bioinformatics.
- **Dynamic Bayesian networks**
  - The Bayesian networks presented in this thesis deal with static Bayesian networks.
  - The ideas presented in this thesis can be extended to Dynamic Bayesian networks as well.

# References

- Neapolitan R.E., 'Learning Bayesian Networks' - Prentice Hall, 2004.
- Cooper G.F. and Herskovits E., 'A Bayesian method for the induction of probabilistic networks from data'. *Machine Learning*, 9: 309 – 347, 1992.
- Proakis J.G. 'Digital Communications'- Mcgraw – Hill, 2001.
- Heckerman D. 'A tutorial on learning in Bayesian networks' - *Technical Report* - Microsoft Research. 1996
- Jensen F.V. 'Introduction to Bayesian Networks' - Springer-Verlag New York Inc,1996
- Murphy K.P. Web reference: http://bnt.sourceforge.net /usage.htm, 2004.
- Chartrand G. and Lesniak L. 'Graphs and Digraphs' - Chapman and Hall, 1996
- Lauritzen S.L. and Spiegelhalter D.J., 'Local computations with probabilties on graphical structures and their application to expert systems' - *J. Royal Statistical society B*, 50:154--227, 1988.
- Beinlich I.A., Suermondt H.J., Chavez R.M. and Cooper G.F., 'The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks.' - *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, 1989
- Jordan M.I., 'Learning in Graphical Models' - Kluwer Academic Publishers, 1996.