User's Guide for the LinkConnectionStrength Package (Version 1.0) –

A PNL Package for the Calculation and Vizualization of Entropy, Link Strengths and Connection Strengths in Discrete Bayesian Networks

by Imme Ebert-Uphoff (imme@users.sourceforge.net)

January 25, 2006

Note: This may or may not be the most recent version of this document. The newest version is always available at www.DataOnStage.com.

Abstract

The LinkConnectionStrength package provides functions to calculate and vizualize entropy, connection strengths and link strengths for discrete Bayesian Networks. The package is implemented for Intel's Open Source Probabilistic Network Library (PNL).

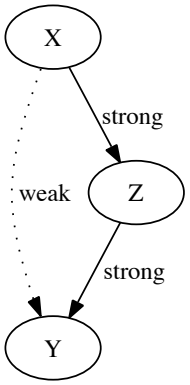
The vizualization component relies on the GraphViz package to provide the actual picture of the graph. Within the graph varying gray scales of the links indicate link strengths and varying shades of the nodes indicate connection strengths relative to a specific node, while the actual numbers are provided as the labels of the links or nodes.

The following measures are implemented:

- Entropy is used to measure the uncertainty in a single node.
- Mutual information is used to measure connection strength.
- Two measures derived from mutual information are available to measure link strength (*True Average Link Strength* and *Blind Average Link Strength*).
- In addition, mutual information *percentage* and link strength *percentage* are provided to measure the *percentage* of the existing uncertainty that has been removed.

Complete definitions of all of the above terms are given in this document.

Contents



$$
\begin{aligned}
\mathbf{X}: \quad & P(X = True) = 0.5 \\
\mathbf{Z}: \quad & P(Z = True|X = True) = 0.9 \\
& P(Z = True|X = False) = 0.1 \\
\mathbf{Y}: \quad & P(Y = True|X = True, Z = True) = 0.9 \\
& P(Y = True|X = False, Z = True) = 0.89 \\
& P(Y = True|X = True, Z = False) = 0.1 \\
& P(Y = True|X = False, Z = False) = 0.11
\end{aligned}
$$

Figure 1: Example 1 - Sample BN with weak link from X to Y, but strong links from X to Z and from Z to Y.

1 Introduction

The LinkConnectionStrength package provides functions to calculate and vizualize entropy, connection strengths and link strengths, for discrete Bayesian Networks. The package is implemented for Intel's Open Source Probabilistic Network Library (PNL).

The vizualization component relies on the GraphViz package to provide the actual picture of the graph. Within the graph varying gray scales of the links indicate link strengths and varying shades of the nodes indicate connection strengths relative to a specific node, while the actual numbers are provided as the labels of the links or nodes. (All source files for the package and additional documentation are available at www.DataOnStage.com)¿

1.1 The Difference Between Link Strength and Connection Strength

The concepts of link strength and connection strength for discrete Bayesian Networks were introduced formally by Boerlage in 1992 [1]. In [1] *connection strength* is defined to apply to any pair of nodes (adjacent or not) and measures the strength between the nodes taking any possible path between them into account. In contrast *link strength* applies to a specific edge between two adjacent nodes and measures the strength of connection only along that single edge.

To demonstrate the difference between link strength and connection strength consider the network shown in Figure 1. Each of the three nodes only has two states, $True$ and $False$. The values for $X = False$, etc., are omitted in Figure 1, since they follow immediately from the values provided.

Let us focus on the connection between nodes X and Y. For this sample network the *direct* link from X to Y is weak[1] , while the indirect link from X to Y through Z is very strong. According to the above (vague) concept definitions, the connection strength between X and Y is strong, but the link strength of the edge $X \rightarrow Y$ is weak:

$$
\begin{aligned}
\mathrm{CS}(X, Y) &= \text{strong}, \\
\mathrm{LS}(X \rightarrow Y) &= \text{weak}.
\end{aligned}
$$

[1]This can easily be seen in the probabilities in Figure 1, because the state of X has little effect on the value of $P(Y = True|X, Z)$.

Any pair of measures for link strength and connection strength should yield this result for the considered example.

1.2 Which measures are implemented here?

The most popular measures for link strength and connection strength are based on entropy and mutual information and those are implemented here:

- Entropy is used to measure the uncertainty in a single node.
- Mutual information is used to measure connection strength.
- Two measures derived from mutual information are available to measure link strength (*True Average Link Strength* and *Blind Average Link Strength*).
- In addition, mutual information *percentage* and link strength *percentage* are provided to measure the *percentage* of the existing uncertainty that has been removed.

Complete definitions of all of the above terms are given in Section 3.

Both entropy and mutual information were defined already by Shannon [2] in the 1940s in the context of communication theory. Pearl [3] was the first to propose the use of mutual information to measure connection strentgth in Bayesian Networks.

There are drawbacks to using entropy as a measure of uncertainty and thus as basis for connection strength and link strength measures. Some of those drawbacks are pointed out by Pearl [3], others by Uffink [4]. However, since no better alternative has yet emerged, those measures are still the most common choice. Nevertheless, an accompanying document (available soon at www.dataonstage.com) discusses the limitations of these measures so that users are aware of what they can and what they cannot do and use the measures accordingly.

There is much less literature on the definition of link strength than on connection strength and it appears to be harder to measure. Boerlage [1] defined measures for both link strength and connection strength. However, those only apply to two-state variables and are not used here. Nicholson and Jitnah [7] derived expressions for link strength based on mutual information for the purpose of efficient approximate inference. Variations of those expressions are used here as measures for link strength. Thus [7] forms the basis for the two link strength measures implemented here.

Several commercial BN software packages also offer some measures for link strength and/or connection strength, but it is often difficult to get a hold of the precise definitions of those. Finally, several other measures and vizualization techniques have been proposed in literature, see for example the work by Nicholson and Jitnah [8], Lacave and Diez [9,10] and Zapata-Rivera et al. [11].

1.3 How to Use the Package

To use the package, download the following files and compile them along with your own code: PNLtoGraphviz.hpp, PNLtoGraphviz.cpp, LinkConnectionStrengths.hpp, LinkConnectionStrengths.cpp. You should then be able to use any of the functions listed in this document in your own code. To see an example, download and compile the files SampleUseLCS.cpp, models.h and models.cpp.

For more detailed installation instruction and information on how to turn the resulting *dot* graph files into rendered graphs, see the other documentation available at www.DataOnStage.com.

2 Organization of this Document

The remainder of this document is organized as follows. Section 3 describes all the available functions to calculate entropy, connection strengths and link strengths. Section 5 describes predefined functions that provide a plain text output for groups of information, e.g. the entropy of all nodes, the link strengths of all links in the network or the mutual information of all nodes relative to a specific target node. Section 6 describes extensions of the PNLtoGraphviz interface that write a description of the network graph along with the information of entropy, connection strengths or link strengths to a file. In addition to providing the actual numbers, link strength is vizualized by gray scale of the *arcs* and connection strength is vizualized by gray scales of the nodes. The file is in the GraphViz format and can be read by GraphViz' dot-command to generate the actual graph picture.

3 Functions for Calculating Entropy, Connection Strengths and Link Strengths

All functions are so far defined *only for discrete Bayesian Networks*!

3.1 Notation

The following notation is used throughout this document. $U(X)$ denotes the uncertainty of a single discrete variable X (measured by its entropy) and is defined as follows:

$$U(X) \;=\; -\sum_{x_i} P(x_i)\log_2\left(P(x_i)\right) \;=\; \sum_{x_i} P(x_i)\log_2\left(\frac{1}{P(x_i)}\right)$$

$U(Y|X)$ is the expected uncertainty in Y if X is known and is calculated by averaging $U(Y|x_i)$ over all possible states x_i of X:

$$U(Y|X) = \sum_{x_i} P(x_i)U(Y|x_i) \;=\; \sum_{x_i} P(x_i)\sum_{y_j} P(y_j|x_i)\log_2\left(\frac{1}{P(y_j|x_i)}\right)$$

Similarly $U(Y|X,Z)$ is the expected uncertainty in Y if both X and Z are known:

$$U(Y|X,Z) \;=\; \sum_{x,z} P(x,z)U(Y|x,z) \;=\; \sum_{x,z} P(x,z)\sum_{y} P(y|x,z)\log_2\left(\frac{1}{P(y|x,z)}\right),$$

where the summations are over all possible states x, z, y of variables X, Z, Y, respectively.

Note that just as in the last formula above, the indices (e.g. i of x_i) are dropped from now on for better readability.

3.2 Entropy

A function is provided to calculate the entropy of any specific node:

```
double Entropy( int X_index, pnlw::BayesNet & BNet );
```

Input variables:

- X_index: index of considered node (X)

- BNet: BayesNet to which X belongs

Functionality: This function returns the entropy of node X_index, according to the formula:

$$U(X) \;=\; \sum_x P(x)\log_2\left(\frac{1}{P(x)}\right)$$

where the summation is over all discrete states of variable X.
Question addressed: *How much uncertainty (in the entropy sense) is there in X if no evidence is given for any of the nodes?*

3.3 Mutual Information

The main function to calculate *mutual information* and *mutual information "percentage"* is the following:

```
void MutualInformation_with_perc( int X_index, int Y_index,
                                  pnlw::BayesNet & BNet,
                                  double & MI, double & MI_perc );
```

Input variables:

- Y_index: node whose uncertainty we wish to analyze (Y is the target node)
- X_index: node whose influence on Y we wish to analyze (X is the evidence node)
  (Role of X and Y is interchangeable for MI, but not for MI%!)
- BNet: BayesNet to which X and Y belong.

Output variables:

- MI: mutual information of X and Y = reduction of uncertainty in Y by knowing X;
- MI%: *percentage* reduction of uncertainty in target node Y by knowing X.

MI and MI% are defined as follows:

1. **MI**: Mutual information of X and Y

$$\begin{aligned} MI(X,Y) &= U(Y) - U(Y|X) = U(X) - U(X|Y) && (1)\\ &= \overbrace{\sum_{x,y} P(x,y)\log_2\left(\frac{1}{P(y)}\right)}^{U(Y)} - \overbrace{\sum_{x,y} P(x,y)\log_2\left(\frac{P(x)}{P(x,y)}\right)}^{U(Y|X)} && (2)\\ &= \sum_{x,y} P(x,y)\log_2\left(\frac{P(x,y)}{P(x)P(y)}\right), \end{aligned}$$

   where the summation is over all discrete states x of X and y of Y. Equation (2) is actually used in the implementation, because it gives separate access to $U(Y)$ and $U(Y|X)$, which is used to calculate MI Percentage below.
   Note that MI(X,Y) is symmetric in X and Y, thus the order of X and Y in the function call does not matter for this.
   Questions addressed: *By how much is the uncertainty in Y reduced by knowing the state of X? By how much is the uncertainty in X reduced by knowing the state of Y?*

2. **MI%**: *Percentage* reduction of uncertainty in target node Y by knowing X

$$MI\%(X,Y) \;=\; \frac{U(Y)-U(Y|X)}{U(Y)}\cdot 100 \;=\; \frac{MI(X,Y)}{U(Y)}\cdot 100.$$

Note that MI%(X,Y) is *not* symmetric in X and Y, thus the order of X and Y in the function call *does* matter for this!

Question addressed: *By how many percentage points is the uncertainty in Y reduced by knowing the state of X?*

A second function is available for convenience if one wants only one of the values, MI *or* MI%:

```
double MutualInformation( int X_index, int Y_index, pnlw::BayesNet & BNet,
                          bool want_percentage);
```

returns MI, if want_percentage = false
returns MI%, if want_percentage = true.

3.4 Link Strength

Link Strength measures *how strongly* two adjacent nodes are connected *along the connecting edge*. Connectivity along other paths should *not* be included in link strength.

All link strengths measures provided here are derived from the concept of mutual information. The main function to calculate link strengths for *two adjacent discrete nodes* of a Bayesian Network is as follows:

```
void LinkStrength_with_perc( int index1, int index2, pnlw::BayesNet & BNet,
                             const std::string & formula,
                             double & TotalValue, double & PercentageValue);
```

Input variables:

- index1: index of first node
- index2: index of second node (must be adjacent to first node!)
- BNet: BayesNet to which both nodes belong
- formula: name of formula to be used for calculation.
  Current options for formula are: "TrueAverage" and "BlindAverage".

Output variables:

- TotalValue: Value of link strengths according to formula;
- PercentageValue: Percentage value of link strength according to formula (see def. below).

Functionality:
The function first establishes which node of index1, index2 is the parent and which one is the child and assigns:

X = parent, $\quad Y$ = child.

Then it determines the set, Z, of all *other* parents of Y, i.e. all parents of Y except node X:

$$Z = \text{other parents(Y)} = \text{parents(Y)} - \{X\}.$$

Note that Z may represent several variables, but for simplicity it is represented here by a single letter (which can be read as the Cartesian product of all of those variables). Likewise any possible state combination of Z is denoted by a single letter, z.

PercentageValue is defined for LinkStrength equivalently to to the way the percentage value was defined for mutual information (MI%).

TotalValue and **PercentageValue** are calculated according to the following formulas:

1. **True Average Link Strength:**

$$LS^{true}(X \to Y) \;=\; U(Y|Z) - U(Y|X,Z),$$

where by definition

$$U(Y|X,Z) \;=\; \sum_{x,z} P(x,z) \sum_{y} P(y|x,z) \log_2 \frac{1}{P(y|x,z)} \tag{3}$$

and

$$U(Y|Z) \;=\; \sum_{z} P(z) \sum_{y} P(y|z) \log_2 \frac{1}{P(y|z)} = \sum_{x,z} P(x,z) \sum_{y} P(y|x,z) \log_2 \frac{1}{P(y|z)}. \tag{4}$$

The expression for $U(Y|Z)$ on the very right in Equation (4) may look more complicated than necessary, but it is just as simple to calculate because it reuses many terms from the calculation of $U(Y|X,Z)$ and it is the one used in the implementation. $P(y|z)$ is calculated from existing terms as

$$P(y|z) = \sum_{x} \frac{P(y|x,z)P(x,z)}{P(z)}.$$

Combining the results above one can write

$$LS^{true}(X \to Y) \;=\; \sum_{x,z} P(x,z) \sum_{y} P(y|x,z) \log_2 \frac{P(y|x,z)}{P(y|z)},$$

but the actual implementation uses Equations (3) and (4) for $U(Y|Z)$ and $U(Y|X,Z)$ to facilitate the calculation of the True Average Percentage below.

Question addressed: *By how much is the uncertainty in Y reduced by knowing the state of X, if the states of all other parent variables are known (averaged over the parent states using the actual frequency of occurrence of the parent states)?*

Comment: This may be the most meaningful measure for link strength, if only a single measure is to be used. Proposed by Nicholson and Jitnah [7] for a similar purpose.

2. **True Average Percentage:**

$$\begin{aligned} LS\%^{true}(X \to Y) &= \frac{U(Y|Z) - U(Y|X,Z)}{U(Y|Z)} \cdot 100 \\ &= \frac{LS^{true}(X \to Y)}{U(Y|Z)} \cdot 100 \end{aligned} \tag{5}$$

Question addressed: *By how many percentage points is the uncertainty in Y reduced by knowing the state of X, if the states of all other parent variables are known (averaged over the parent states using the actual frequency of occurrence of the parent states)?*

3. **Blind Average Link Strength:**
This measure is derived from True Average Link Strength, but disregards the actual frequency of occurrence of the parent states by assuming X, Z are independent and all uniformly distributed, i.e.

$$\hat{P}(x,z) = P(x)P(z), \quad \hat{P}(x) = \frac{1}{\#(X)}, \quad \hat{P}(z) = \frac{1}{\#(Z)}, \tag{6}$$

where $\#(X)$ denotes the number of discrete states of X, etc.
This creates a local measure that depends only on the child node and its conditional probability table, but nothing else in the network.
This yields a much simpler formula:

$$LS^{blind}(X \to Y) = \hat{U}(Y|Z) - \hat{U}(Y|X,Z),$$

where

$$\begin{aligned} \hat{U}(Y|X,Z) = &= \sum_{x,z} \hat{P}(x,z) \sum_{y} P(y|x,z) \log_2 \frac{1}{P(y|x,z)} \\ &= \frac{1}{\#(X)\#(Z)} \sum_{x,y,z} P(y|x,z) \log_2 \frac{1}{P(y|x,z)} \end{aligned} \tag{7}$$

$$\begin{aligned} \hat{U}(Y|Z) &= \sum_{x,z} \hat{P}(x,z) \sum_{y} P(y|x,z) \log_2 \frac{1}{\hat{P}(y|z)} \\ &= \frac{1}{\#(X)\#(Z)} \sum_{x,y,z} P(y|x,z) \log_2 \frac{1}{\hat{P}(y|z)}, \end{aligned} \tag{8}$$

where

$$\hat{P}(y|z) = \sum_{x} \frac{P(y|x,z)\hat{P}(x,z)}{\hat{P}(z)} = \sum_{x} \frac{P(y|x,z)\frac{1}{\#(X)\#(Z)}}{\frac{1}{\#(Z)}} = \frac{1}{\#(X)} \sum_{x} P(y|x,z)$$

and thus

$$LS^{blind}(X \to Y) = \frac{1}{\#(X)\#(Z)} \sum_{x,y,z} P(y|x,z) \log_2 \left(\frac{P(y|x,z)}{\frac{1}{\#(X)} \sum_x P(y|x,z)} \right),$$

where $P(y|x,z)$ is given by the conditional probability table of Y and *no* inference is required. Again, Equations (7) and (8) are actually used in the implementation, to obtain $\hat{U}(Y|Z)$ and $\hat{U}(Y|X,Z)$ to facilitate the calculation of the Blind Average Percentage below.
Question addressed: *By how much is the uncertainty in Y reduced by knowing the state of X, if the states of all other parent variables are known (averaged over the parent states assuming all parents are independent of each other and uniformly distributed)?*
Comment: This is the simplest and computationally least expensive measure. It is also a local measure, taking only the child and its conditional probability table into account, thus allowing for isolated analysis of child and parents, regardless of the rest of the network.

4. **Blind Average Percentage:** Same expression as (6), but using the independence and uniformity assumptions of (6). This yields:

$$\begin{aligned} LS\%^{blind}(X \to Y) &= \frac{\hat{U}(Y|Z) - \hat{U}(Y|X,Z)}{\hat{U}(Y|Z)} \cdot 100 \\ &= \frac{LS^{blind}(X \to Y)}{\hat{U}(Y|Z)} \cdot 100 \end{aligned} \tag{9}$$

Question addressed: *By how many percentage points is the uncertainty in Y reduced by knowing the state of X, if the states of all other parent variables are known (averaged over the parent states assuming all parents are independent of each other and uniformly distributed)?*

A second function is available for convenience if one only wants one of the values, LS *or* LS%:

```
double LinkStrength( int index1, int index2, pnlw::BayesNet & BNet,
                     const std::string & formula, bool want_percentage);
```

returns LS, if want_percentage = false
returns LS%, if want_percentage = true.

3.5 Scaling Function

The following function is useful to scale the values of entropy, mutual information or link strength to a value between 0 and 1. This is useful for example for vizualization purposes, namely to select a gray value for printing of the links, nodes, etc. The function

```
double Entropy_bound( pnlw::BayesNet * BNet_p );
```

first determines the node in the network with the largest number of states and then returns its logarithm:

$\log_2$(largest # of states per node).

Entropy, Mutual Information (MI) and Link Strength (LS) all return values between 0 and Entropy_bound. Thus dividing Entropy, MI or LS by *Entropy_bound* yields a value between 0 and 1 that can be converted to gray values:

$$\frac{Entropy}{Entropy\_bound} \in [0,1], \qquad \frac{MI}{Entropy\_bound} \in [0,1], \qquad \frac{LS}{Entropy\_bound} \in [0,1].$$

Note that in contrast the percentage values, namely MI% and LS%, yield a value between 0 and 100, thus should be scaled by 100 to yield a value between 0 and 1:

$$\frac{MI\%}{100} \in [0,1], \qquad \frac{LS\%}{100} \in [0,1].$$

4 Computational Issues

4.1 Handling Special Cases (Division by zero, etc.)

When calculating entropy, mutual information and link strengths there can always be degenerate terms. For example, what is the result for expression $P(x) \cdot \log_2 \left(\frac{1}{P(x)}\right)$ in the entropy calculation if $P(x) = 0$? It turns out that the above expression converges towards 0 if $P(x)$ converges towards zero. Thus this implementation simply tests whether $P(x) < \epsilon$, where here we use $\epsilon = 10^{-10}$, and in that case simply drops the entire expression, i.e. sets it to zero.

Considering the formulas for mutual information and link strengths turns up a variety of similar degenerate cases that would lead to division by zero, calculating the logarithm of zero, or calculating an undefined expression such as $P(y|x)$ for $P(x) = 0$. Fortunately, careful analysis shows that in all of those cases the expressions in question converge towards zero when approaching the degenerate case and thus a similar similar simple procedure can be used: whenever certain probabilities are smaller than ϵ the corresponding expression is treated as zero.

4.2 Inference Used and Computational Complexity

The computation with the highest computational complexity in all of the connection strength and link strength functions is probably the inference used to calculate all the required probabilities. Thus the inference steps required are discussed for each function.

- **Entropy:**
  Calculating the entropy of a single node, X, requires to calculate its marginal distribution, $P(X)$.
  Thus function $GetJPD(\text{“}X'')$ is called once for each node using the junctionTree method for inference.

- **Mutual Information:**
  Calculating mutual information of a node pair (X, Y) requires to calculate its joint distribution, $P(X, Y)$.
  Thus function $GetJPD(\text{“}X\ Y'')$ is called once for each node pair using the junctionTree method for inference.

- **Blind Average Link Strength (including Percentage):**
  Calculating the Blind Average Link Strength for an arc $X \rightarrow Y$ does *not* require any inference. Only the conditional proabilities $P(Y|$ all parents of $Y)$ are needed.
  Those are obtained through calls to $GetPTabular(\ldots)$ which simply read values from the existing conditional probability tables.

- **True Average Link Strength (including Percentage):**
  To calculate the True Average Link Strength for an arc $X \rightarrow Y$ requires calculating the joint distribution of the parents of Y: $P(\text{all parents of } Y)$.
  Thus $GetJPD(\text{“}parent1\ parent2\ \ldots'')$ is called once for each arc $X \rightarrow Y$ using the junctionTree method for inference[2].

[2] One also needs the same calls to *GetPTabular* as in the case of Blind Average Link Strength, but in comparison to the call to *GetJPD* those are assumed to be negligeable.

One may notice that the same joint distribution, P(all parents of Y), may potentially be calculated many times, when calculating all the true average link strengths in a network, which is quite wasteful. For example, if X_1 and X_2 are both parents of Y, then P(all parents of Y) is calculated once again for arc $X_1 \to Y$ and once for $X_2 \to Y$.

However, computational complexity so far does not appear to be an issue, and the current implementation has the advantage of keeping the calculation of $LS(X_1 \to Y)$ *independent* of $LS(X_1 \to Y)$. Nevertheless, if computational complexity turns out to be an issue for larger networks, pre-calculating and storing P(all parents of Y) for all nodes in the networking may be a good way to cut down complexity by a factor of about the average number of parents per node in the network.

5 Functions for Plain Text Output

5.1 Available Functions

The following functions provide various information by printing the results on the screen (rather than writing them to a graph file as is the case for the functions in the following section). These functions simply call the various functions of the previous section for all nodes or all links, etc., and print the result.

An example of the use and output of most of these functions is given in Section 5.2.

Input variables used in the functions below:

- net: pointer to discrete BN.
- target_node, target_node_name: index or name of target_node relative to which Mutual Information (Percentage) of all other nodes is calculated.
- formula: name of formula to be used for link strength calculation.
  Current options are "TrueAverage" and "BlindAverage"
- want_percentage: True or False; If "True" return absolute value (of link or connection strength), otherwise return percentage value.

Available Functions:

1. **Graph Structure**

   ```
   void Print_Graph_Structure( pnlw::WGraph & myGraph );
   ```

   Functionality: Output the graph structure by printing each node name, followed by a list of its parents.

2. **Entropy**

   ```
   void Print_Entropy( pnlw::BayesNet & BNet);
   ```

   Functionality: Print entropy for all nodes of the network.

3. **Link Strengths**

```
void Print_Link_Strengths( pnlw::BayesNet & BNet, const std::string & formula,
                           bool want_percentage);
```

Functionality: Print desired type of link strength for all arcs of the network

4. **Connection Strengths**

```
void Print_Mutual_Information_For_Single_Node( pnlw::BayesNet & BNet,
                                               String & target_node_name,
                                               bool want_percentage);
```

Functionality: Print Mutual Information of all nodes relative to target_node (where target_node is denoted by its name).

```
void Print_Mutual_Information_For_Single_Node( pnlw::BayesNet & BNet,
                                               int target_node,
                                               bool want_percentage);
```

Functionality: Same as above, but target_node is denoted by its index.

```
void Print_Mutual_Information_For_All_Nodes( pnlw::BayesNet & BNet,
                                             bool want_percentage);
```

Functionality: Call Print_Mutual_Information_For_Single_Node for all nodes of network, i.e. in turn each node is used as target_node (one after the other)

5. **Summary Report**

```
void Print_Summary_Report( pnlw::BayesNet & BNet );
```

Functionality: Print the following information by calling the above functions: Graph structure, Entropy, Blind Average Link Strength + Percentage, True Average Link Strength + Percentage, Mutual Information + Percentage for all nodes as target nodes.

5.2 Plain Text Output for Example 1

The following sample code demonstrates how to generate plain text output for Example 1 (see Figure 1 in Section 1.1):

Sample Code for Plain Text Output of Example 1:

```
BNet_p = Strong_Weak_example();

Print_Graph_Structure( BNet_p->Net().Graph() );
Print_Entropy( *BNet_p );
Print_Link_Strengths( *BNet_p, "TrueAverage", false);
Print_Link_Strengths( *BNet_p, "TrueAverage", true);
Print_Link_Strengths( *BNet_p, "BlindAverage", false);
Print_Link_Strengths( *BNet_p, "BlindAverage", true);
Print_Mutual_Information_For_All_Nodes( *BNet_p, false);
Print_Mutual_Information_For_All_Nodes( *BNet_p, true);
```

Here is the corresponding output:

Output for Sample Code Above:

```
Printing Graph Structure
 Node 0: X
   Parents:

 Node 1: Y
   Parents:   X   Z

 Node 2: Z
   Parents:   X

Entropy for all nodes:
 Node 0: X    1.000
 Node 1: Y    1.000
 Node 2: Z    1.000

Link Strengths for all arcs using  True Average
 X -> Y      LS = 0.000
 Z -> Y      LS = 0.204
 X -> Z      LS = 0.531

Link Strengths for all arcs using  True Average Percentage
 X -> Y      LS = 0.0%
 Z -> Y      LS = 29.7%
 X -> Z      LS = 53.1%

Link Strengths for all arcs using  Blind Average
 X -> Y      LS = 0.000
 Z -> Y      LS = 0.516
 X -> Z      LS = 0.531

Link Strengths for all arcs using  Blind Average Percentage
 X -> Y      LS = 0.0%
 Z -> Y      LS = 51.6%
 X -> Z      LS = 53.1%

Mutual Information for Target Node: X
 MI(Y,X) =   0.311
 MI(Z,X) =   0.531

Mutual Information for Target Node: Y
 MI(X,Y) =   0.311
 MI(Z,Y) =   0.515

Mutual Information for Target Node: Z
 MI(X,Z) =   0.531
 MI(Y,Z) =   0.515

Mutual Information Percentage for Target Node: X
 MI%(Y,X) =   31.1%
```

```
 MI%(Z,X) =   53.1%

Mutual Information Percentage for Target Node: Y
 MI%(X,Y) =   31.1%
 MI%(Z,Y) =   51.5%

Mutual Information Percentage for Target Node: Z
 MI%(X,Z) =   53.1%
 MI%(Y,Z) =   51.5%
```

6 Functions for Creating Graphs showing Entropy, Link and Connection Strengths

In most instances graphs are much more helpful than plain text to vizualize all the information provided by entropy, connection strengths and link strengths. Thus the measures defined in the LinkConnectionStrength package are now combined with the PNLtoGraphviz package which can generate graph files for Bayesian Networks. Four new functions have been added to PNLtoGraphviz to vizualize entropy, mutual information and link strength for discrete Bayesian Networks.

If you have successfully downloaded and used the files of the LinkConnectionStrength package, then you *already* also have all the files for PNLtoGraphviz, since they come together in the PNLVizualize package. The key thing to know about PNLtoGraphviz is that it generates a description of a graph in a 'dot'-file, which must then still be converted to a rendered graph using the *dot* command of the GraphViz package. The GraphViz package is a widely used open source graph visualization software available at http://www.graphviz.org/. It is very common and available for almost any platform. It is also *very* easy and fast to install. So, *please* don't be concerned about this additional step - the results will be worth your effort (and you'll probably find many other uses for GraphViz, too).

For additional information on the PNLtoGraphViz package and how to use it, please see the manual available at www.dataonstage.com.

6.1 Available Functions

The PNLtoGraphViz package now makes four new graph printing routines available for discrete Bayesian Networks. These are described below. As mentioned above the code generates only the 'dot'-file for each graph which is then converted to a rendered graph using the *dot* command of the GraphViz package.

Input variables used in the functions below:

- net: pointer to BayesNet
- filename: name of output file
- target_node_index: index of node relative to which Mutual Information (Percentage) of all other nodes is calculated.
- formula: name of formula to be used for link strength calculation.
  Current options are "TrueAverage" and "BlindAverage".

- want_percentage: True or False; Calculates absolute value of link or connection strength if "True", otherwise percentage value.
- customized_node_shape: optional parameter that allows one to change the look of the nodes in resulting graph (see PNLtoGraphviz documentation for its use).

Output variable:

- Each function *should* return "1" if the file was created successfully and "0" otherwise. However, this features hasn't been tested extensively.

Available Functions:

1. **Entropy Graph for Discrete BN**

```
int PNLtoGraphviz_with_Entropy ( BayesNet * net,
                                 const std::string & filename);
```

Functionality: Creates graph including entropy for each node. The entropy is shown in the graph as a number below the node name.

2. **Graph with Connection Strengths for Discrete BN**

```
int PNLtoGraphviz_with_MI ( BayesNet * net, const std::string & filename,
                            int target_node_index, bool want_percentage);
```

Functionality: Creates graph including mutual information (or mutual information percentage) relative to target_node. The target node is indicated by an octagonal node shape and its entropy included underneath the node name. Connection Strength of all other nodes relative to this one is displayed by (1) number underneath the node name and (2) gray scale of node.

```
int PNLtoGraphviz_with_MI ( BayesNet * net, const std::string & filename,
                            const String target_node_name,
                            bool want_percentage);
```

Functionality: Same as above, but using Node Name instead of Node Index for target_node.

3. **Graph with Link Strengths for Discrete BN**

```
int PNLtoGraphviz_with_LS ( BayesNet * net, const std::string & filename,
                            const std::string & formula,
                            bool want_percentage,
                            std::map<std::string,std::string>
                            customized_node_shape=
                            std::map<std::string, std::string>() );
```

Functionality: Create graph with link strengths (True Average or Blind Average formula, absolute value or percentage). Link Strength of each arc is displayed by (1) number next to the arrow and (2) gray scale of arrow. If an arrow is very weak and would be almost invisible, it is replaced by a dashed arrow.

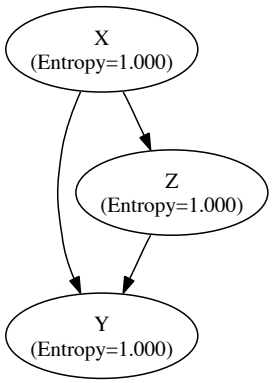


Figure 2: Entropy Graph for Example 1.

6.2 Graph Output for Example 1

The following sample code demonstrates how to generate graphs for Example 1 (see Figure 1 in Section 1.1):

Sample Code to Generate Graphs for Example 1:

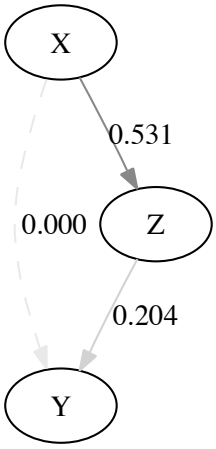
```
BNet_p = Strong_Weak_example();

PNLtoGraphviz_with_Entropy ( BNet_p, "Entropy.dot" );

PNLtoGraphviz_with_LS ( BNet_p, "LS_True.dot", "TrueAverage", false );
PNLtoGraphviz_with_LS ( BNet_p, "LS_True_P.dot", "TrueAverage", true );

PNLtoGraphviz_with_LS ( BNet_p, "LS_Blind.dot", "BlindAverage", false );
PNLtoGraphviz_with_LS ( BNet_p, "LS_Blind_P.dot", "BlindAverage", true);

PNLtoGraphviz_with_MI ( BNet_p, "MI_0.dot", 0, false );
PNLtoGraphviz_with_MI ( BNet_p, "MI_1.dot", 1, false );
PNLtoGraphviz_with_MI ( BNet_p, "MI_2.dot", 2, false );

PNLtoGraphviz_with_MI ( BNet_p, "MI_P_0.dot", 0, true );
PNLtoGraphviz_with_MI ( BNet_p, "MI_P_1.dot", 1, true );
PNLtoGraphviz_with_MI ( BNet_p, "MI_P_2.dot", 2, true );
```

The above code was used to generate all graphs shown in this subsection. As mentioned above the code generates only the 'dot'-file for each graph which is then converted to a rendered graph using the *dot* command of the GraphViz package.

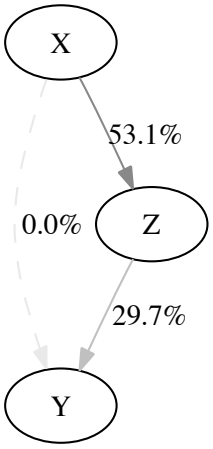
These graphs are now discussed one by one in the order they were generated by the code. Note that all graphs (except for the entropy graph) automatically receive a descriptive caption that precisely describes what is being displayed in the graph.

Figure 2 shows the entropy of all nodes. The entropy of nodes X and Z equals exactly one, while the value for node Y is actually 0.99995 (rounded to 1.000 due to the limited number of decimals).

Figure 3 shows the link strengths for all links using True Average and True Average Percentage. Figure 4 shows the same information using the Blind Average formula instead. Note that the link

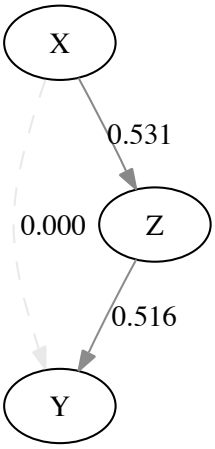


Link Strengths using True Average

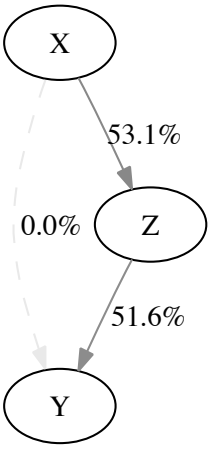


Link Strengths using True Average Percentage

Figure 3: True Average Link Strength (left) and Percentage (right) for Example 1.



Link Strengths using Blind Average



Link Strengths using Blind Average Percentage

Figure 4: Blind Average Link Strength (left) and Percentage (right) for Example 1.

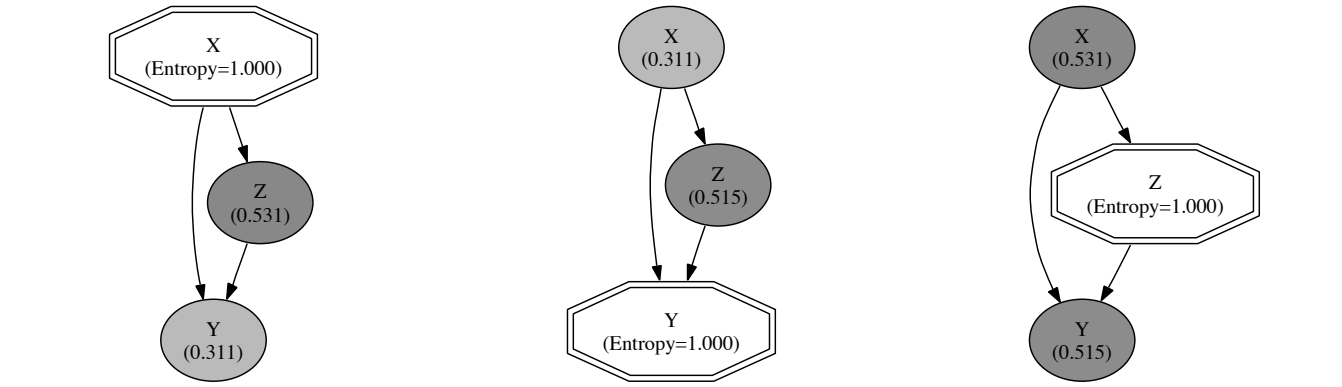


Figure 5: Connection Strength (Mutual Information) relative to node X (left), Y (center) and Z (right) for Example 1.

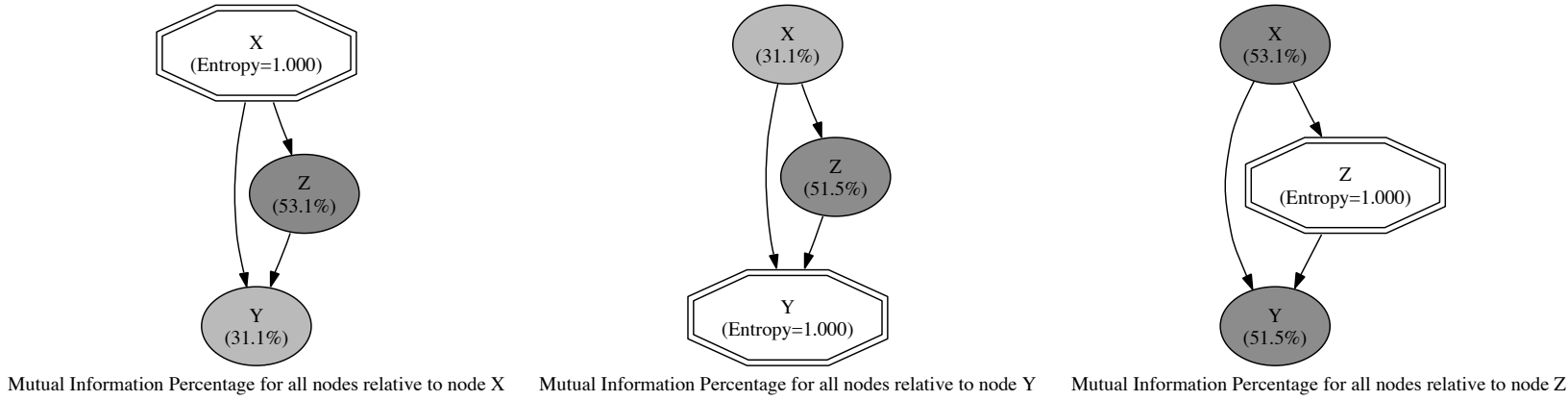


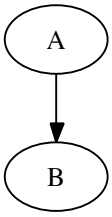
Figure 6: Connection Strength (Mutual Information) Percentage relative to node X (left), Y (center) and Z (right) for Example 1.

strengths are indicated in two ways: as a number next to the arc and by the gray scale of the arc's arrow. In cases where the link strength is *relatively* weak[3] and the arrow would be nearly invisible, the arrow is *dashed* instead in a light gray. Thus a dashed line always indicates a link strength (percentage) that would be below the treshold for visibility.

Figure 5 shows the mutual information using a separate plot for for each node. For example the figure on the left of Figure 5 shows the mutual information relative to node X (X is thus indicated by an octagonal node shape). The numbers underneath Y and Z indicate the connection strength of those nodes relative to X. Furthermore, the gray scale of the nodes also indicates the strength of influence: a darker node is more strongly connected to X than a lighter node.

For completeness, we also include the graphs showing Mutual Information *percentage* here in Figure 6, although those do not provide any new information in this case and are rather boring.

[3]See Section 7.2 on a discussion of which strengths to consider to be weak.



$$\begin{aligned} \mathbf{A}: \quad & P(A = True) = a \\ \mathbf{B}: \quad & P(B = True|A = True) = b1 \\ & P(B = True|A = False) = b2 \end{aligned}$$

Figure 7: Example 2 - Two-Node Network with parameters describing all probabilities.

7 Properties and Interpretation of the Measures

This section provides results for several different types of Bayesian Network models and uses them to illustrate properties of the link strength and connection strength measures.

7.1 Demonstrating Difference Between Link Strength and Connection Strength

Let us revisit the graph output for Example 1 provided in Section 6.2 and compare those results to the desired properties for link strength and connection strength outlined in Section 1.1:

- **Connection Strength:** Figure 5 shows that the connection strength is as expexted. Each pair of nodes, (X, Y), (X, Z) and (Y, Z), is strongly connected. In particular, the pair of nodes (X, Y) receives a strong connectivity value, because they are strongly connected through the chain $X \to Z \to Y$.
- **Link Strength:** The results for the link strengths (Figure 3 and 4) are also consistent with the expectation in Section 1.1: no matter which formula is used (True Average or Blind Average), the link strengths of the arcs from X to Z and from Z to Y are significant, while the strength of the arc from X to Y vanishes.

Thus the link strength and connection strength measures defined here behave for this example as specified in Section 1.1.

Two more details are noteworthy: Firstly, the link strength from X to Z is identical for both formulas, which makes sense since Z only has a single, uniformly distributed parent, X, thus the assumptions imposed for Blind Average Link Strengths are satisfied anyway.

Secondly, the difference between the two formulas for the arc from Z to Y shows that the results for those formulas can generally vary significantly. Thus one must carefully choose which formula to use. This issue is discussed in more detail in Section 7.3 below.

7.2 What Threshold Should be Used to Indicate a "Strong" Relationship?

This question will not be fully answered here, but we will try to shed some light on it by considering the simple example in Figure 7. Nodes A and B in the network in Figure 7 are both binary with states $True$ and $False$. The parameters $a, b1, b2$ describe all probabilities, since all other values easily follow from them.

First let us note some very interesting property.

Property 1: For any node, Y, in a Bayesian Network with only a single parent, X, mutual

b	0.0	0.01	0.02	0.05	0.1	0.2	0.3	0.4	0.5
$MI(A,B) = LS^{true/blind}(A \rightarrow B)$	1.0	0.919	0.859	0.714	0.531	0.278	0.119	0.029	0
$MI\%(A,B) = LS\%^{true/blind}(A \rightarrow B)$	100	91.9	85.9	71.4	53.1	27.8	11.9	2.9	0.0

Table 1: Connection and Link Strengths for varying b in Example 2a.

information and True Average Link Strength yield the same value. Mutual information Percentage and True Average Link Strength Percentage also coincide in this case:

$$\text{If Parents}(Y) = \{X\}: \qquad MI(X,Y) \quad = LS^{true}(X \rightarrow Y)$$
$$\text{If Parents}(Y) = \{X\}: \qquad MI\%(X,Y) = LS\%^{true}(X \rightarrow Y)$$

Proof: If Y only has a single parent, then Z is the empty set in the definition of Link Strength:

$$LS^{true}(X \rightarrow Y) \;\; = \;\; U(Y|Z) - U(Y|X,Z) \;\; = \;\; U(Y) - U(Y|X) = MI(X,Y).$$

The equality of the percentages follows in the same way.

Example 2a: Now let us consider the special case of Example 2, where $a = 0.5$, $b1 = b$ and $b2 = 1 - b$. Thus node A is uniformly distributed and by varying b we can influence how much the state of A affects the probability of states of B. For example, for $b = 1.0$ ($b = 0.0$) we have perfect certainty for B knowing A: B is $True$ if and only if A is $True$ ($False$). Furthermore, uncertainty is maximal for $b = 0.5$ and the amount of uncertainty is identical for b and for $b' = 1 - b$.

The question for which we seek to gain intuition with this example is how do mutual information and link strength "scale" for this example, i.e. what values do they result in for varying b?

Table 1 shows the result for $MI(A,B)$ for a variety of values of b. Note that these also represent the results obatined for both link strength formulas: From Property 1 we know that $MI(A,B) = LS^{true}(A \rightarrow B)$ for this example. Furthermore, it is $LS^{true}(A \rightarrow B) = LS^{blind}(A \rightarrow B)$ for Example 2a, since A is uniformly distributed.

In Table 1 notice *how* quickly $MI(A,B)$ decreases when increasing b from zero. For example, for $b = 0.1$ we know that in 90% of cases B is $True$ if and only if A is $False$. However, the connection/link strength value is only 0.531 with a percentage value of 53.1%. Similarly, even for $b = 0.4$ we know that A still has a *significant* effect on B, but the percentage value of removed uncertainty is only 2.9%.

The lesson from this is that while the values of the measures increase monotonously when uncertainty is reduced, the *scale* of the actual values is *not* linear and *not* intuitive. This needs to be considered when choosing a treshold for when a connection is considered "strong". This threshold must be chosen carefully and probably relatively low. Furthermore, Table 1 should be kept in mind when making statements about the relative importance of one arc over another: A higher number indicates a higher significance, but the scale is not linear.

It is a question of further research whether a transformation function should be applied to the values of the link and connection strength to yield a more intitive result. However, most likely such research would more likely yield a different function to be used to measure uncertainty, i.e. choosing an elementary function $U(X)$ different from entropy, but to date it seems no other function has shown more promise. Nevertheless, the non-intuitive scale is probably the property of these measures that most begs for improvement and should be considered in the future.

7.3 The Difference Between True Average and Blind Average Link Strength

The difference between True Average Link Strength and Blind Average Link Strength is that the True Average formula measures the strength of an arc in the context of the entire network, while the *Blind Average formula takes only the conditional probability table of the child into account.* Thus Blind Average is a local measure that isolates the impact of the arc from everything else happening in the network.

To demonstrate the difference between the True Average and Blind Average Link Strength let us revisit Example 2 (Figure 7), but with different values for $a, b1, b2$.

Example 2b: Consider Example 2 in Figure 7 for the values $a = 0.99$, $b1 = 0.5$ and $b2 = 0.99$. This means that A is almost always True. If A is $True$, nothing is known about the likely state of B. On the other hand, if A is $False$, B is very likely to be $True$, but the former rarely ever occurs.

Thus, if one asks how much information about B is likely gained by knowing the state of A, the answer is "very little", since most likely A is $True$ and then we do not learn anything about B. This type of question is answered by the True Average Link Strength, since it considers the entire picture, i.e. it also takes the likelihood of the states of A into account. It yields for this case:

$$\begin{aligned} LS^{true}(A \rightarrow B) &= 0.009, \\ LS\%^{true}(A \rightarrow B) &= 0.9\%, \end{aligned}$$

which is low even considering the type of scale in Table 1.

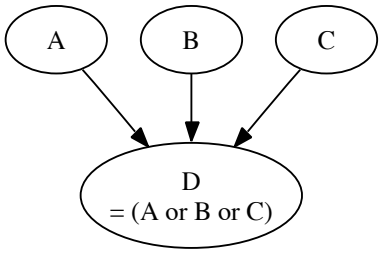
In contrast, Blind Average Link Strength is a function of only the conditional probability table of the child, B, thus it ignores the joint probability of the parents (only one parent in this case), assuming instead conditionally independent, uniformly distributed parents. In the case of Example 2b the probabilities of A are thus replaced by $\hat{P}(A = True) = 0.5$, resulting in

$$\begin{aligned} LS^{blind}(A \rightarrow B) &= 0.279, \\ LS\%^{blind}(A \rightarrow B) &= 34.0, \% \end{aligned}$$

which by all standards is quite high.

One may ask: Why would anyone want to use the Blind Average Link Strength if it ignores important information of the network? The answer is that in some cases one may want to *isolate* the evaluation of an arc from everything else going on in the network. Consider for example a large network, where node Y is somewhere in the middle and has several parents. Changing *any* probability distribution for *any* node far away in the network may affect the joint probability of the parents of Y. *Should* that affect the link strength evaluation of arcs between Y and its parents, even if *nothing* has changed about the direct relationship from Y to its parents? Some may argue, no, the link strength should not change. Then Blind Average Link Strength is the answer, since it effectively cuts out Y and its parents from the rest of the network and thus would not be affected by any of changes to other nodes. For all other cases, True Average Link Strength delivers the answer taking the change in the actual frequency of occurrence of the parent states into account.

Other Reasons for Considering Blind Average Link Strength:
A different reason for considering Blind Average Link Strength, rather than True Average Link Strength, is computational complexity, since the Blind Average formula only requires a few calculations and the True Average formula requires inference. However, so far this has not been a deciding factor, since (1) computational complexity has not been an issue for the networks tested so far and (2) one could use approximate inference (rather than JunctionTree) to reduce complexity.



$$
\begin{aligned}
\mathbf{A}: &\quad P(A = True) = 0.5 \\
\mathbf{B}: &\quad P(B = True) = 0.5 \\
\mathbf{C}: &\quad P(C = True) = 0.5 \\
\mathbf{D}: &\quad P(D = True|A, B, C) = 0.0 \quad \text{if } A = B = C = False \\
&\quad P(D = True|A, B, C) = 1.0 \quad \text{otherwise}
\end{aligned}
$$

Figure 8: Example 2: Deterministic Network where $D = A$ or B or C.

Finally, the last paragraph in Section 7.6 may provide additional motivation for considering Blind Average Link Strength, *in addition* to True Average Link Strength. Nevertheless, clear guidelines on when exactly to use True Average Link Strength, when to use Blind Average Link Strength, or when to use both, are still to be developed.

7.4 Detecting Deterministic Relationships *or* Why to Use Link Strength Percentages

This section illustrates an interesting property of the Link Strength *Percentages* for deterministic functions. By deterministic function we mean that the state of a child is completely known if the states of all of its parents are known, i.e. there is *no* uncertainty involved. While this may seem unusual for a Bayesian Network (and in fact shows that the child node is redundant), it may nevertheless occur in practice and be interesting to detect, e.g. if a network was learned from data. Furthermore, degenerate cases often make it easier to demonstrate certain properties, as is the case here.

Property 2: For a deterministic child the True Average Link Strength Percentage to *any* of its parents is 100%. The same statement holds for the *Blind* Average Strength Percentage.[4]
Proof: True Average Percentage is defined (Section 3.4) for a child node Y and parent X as

$$LS\%^{true}(X \to Y) \quad = \quad \frac{U(Y|Z) - U(Y|X,Z)}{U(Y|Z)} \cdot 100,$$

where Z is the set of all other parents of Y. If the child, Y, has a deterministic relationship to its parents then by definition it is $U(Y|X,Z) = 0$. Thus we get for *any* parent X of Y:

$$LS\%^{true}(X \to Y) \quad = \quad \frac{U(Y|Z) - 0}{U(Y|Z)} \cdot 100 \;=\; 100.$$

Let us consider a specific example to see the usefulness of this property.

Example 3: Figure 8 shows a network with nodes A, B, C, D, where each variable has only two possible states, $True$ and $False$. Parent nodes A, B, C are all uniformly distributed and D is a deterministic child of its parents, defined by the relationship $D = A$ or B or C.

Figure 9 shows the entropy for Example 2. The only noteworthy item is that D has a much lower

[4]Vice versa,I suspect that the reverse statement also holds (I didn't get around to prooving that yet though): If the True Average Link Strength (or the Blind Average Link Strength) for a child to *all* of its parents is 100%, then the child is a deterministic child of its parents.

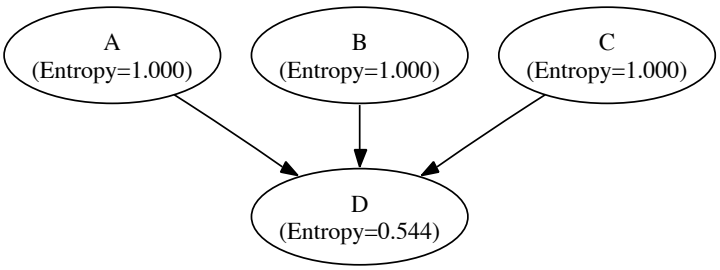


Figure 9: Entropy Plot Deterministic OR.

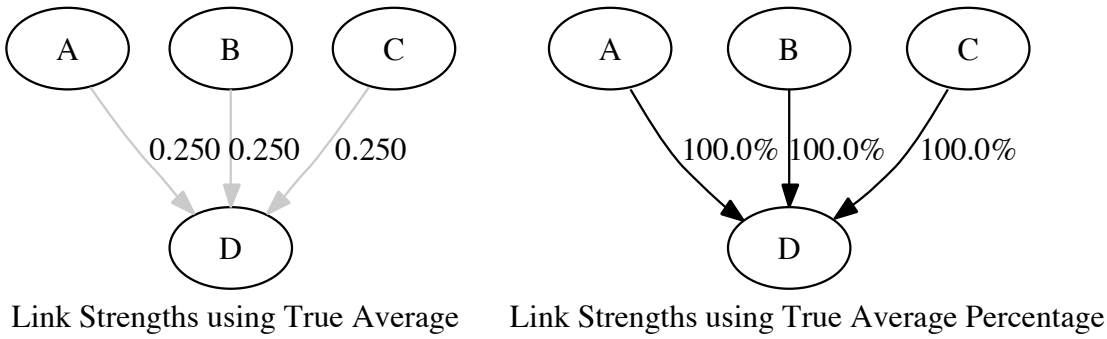


Figure 10: True Average Link Strengths (left) and Percentages (right) for Deterministic OR.

entropy than A, B or C, because $D = True$ with high probability ($P(D = True) = 7/8, P(D = False) = 1/8$). Thus D has much lower uncertainty than the other nodes.

Figure 10 shows the True Average Link Strengths and Percentages. As expected the Percentages are 100% for all links. It may be surprising that the link strengths themselves do *not* add up to one, even if the child is deterministic. However, the value 0.25 is correct and can be explained as follows: Let us consider the influence of node A on D. There is a total of $2^3 = 8$ different parent state combinations. For 6 of those, namely whenever B or C is $True$, it is already known that D is also $True$. Thus the state of A has an effect on the state of D only for 2 out of 8 states, which yields a ratio of $2/8 = 0.25$.

Given the fact that the True Average Link Strength Percentage and the Blind Average Link Strength Percentage both take the value of 100% for all parent-child links of a deterministic child, looking at the percentages may be a good way to identify deterministic and nearly deterministic relationships in a network.

Figure 11 displays the mutual information of all nodes relative to node A. The numbers for B and C are zero, since knowing only variable B or only variable C has no influence on A in this system (for unkonwn D). Knowing the state of D reduces the uncertainty of A by 0.138, i.e. by 13.8% (since $U(A) = 1.0$). Going in the other direction, we see in Figure 12 that knowing A reduces the uncertainty in D also by 0.138, but that represents 25.4% of uncertainty reduction in D (since $U(D) = 0.544$).

Another motivation for looking at the percentage is to determine how important the effect of a parent node is, not only in absolute terms, i.e. how much uncertainty is removed, but also to determine how significant that removed uncertainty is in comparison to the still remaining uncertainty.

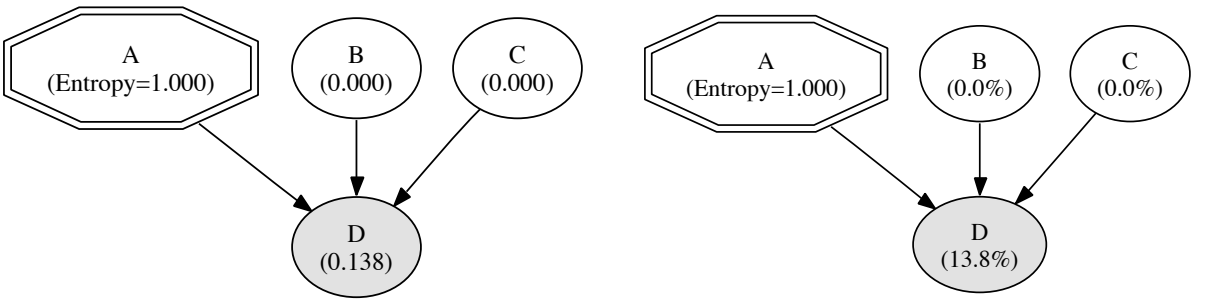


Figure 11: Mutual Information (left) and MI Percentage (right) for all nodes relative to node A.

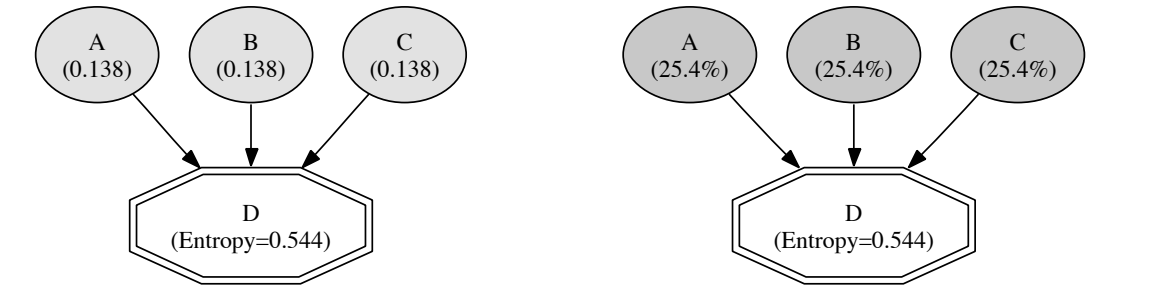


Figure 12: Mutual Information (left) and MI Percentage (right) for all nodes relative to node D.

7.5 Subtracting Link Strength from Mutual Information is like Subtracting Apples from Oranges

While Property 1 in Section Subsection 7.2 showed that Mutual Information and True Average Link Strength coincide in the special case of a node with a single parent, this section shows that generally the two measures act in different realms and thus generally cannot be compared numerically.

Let us revisit the deterministic network from Example 3 and compare the results obtained in Section 7.4 for link strength and connection strength. Because the only causal connection between A and D is a direct link from A to D, one may expect mutual information and link strength to yield the same result of uncertainty reduction for D by knowing A. However, neither the absolute numbers, nor the percentages match: according to Mutual Information only 0.138 units (25.4%) of uncertainty is removed in D by knowing A, while according to link strength 0.25 units (100%) of uncertainty is removed. The difference is easily explained: The uncertainty in D is reduced *less* by learning the state of A *if nothing else is known* (Mutual Information), than by learning the state of A *if* B *and* C *are already known* (Link Strength). In formulas:

$$\begin{aligned} MI(A,D) &= U(D) - U(D|A) = 0.138. \\ LS(A \to D) &= U(D|B,C) - \underbrace{U(D|A,B,C)}_{0} = U(D|B,C) = 0.25. \end{aligned}$$

This difference between assuming nothing else being known (for connection strength), and assuming all other parents are known (for link strength), is a crucial difference that must always be kept in mind when interpreting the results of these two different measures. Furthermore, this

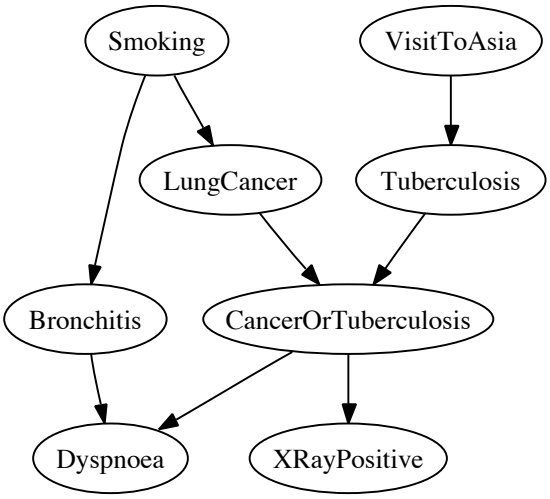


Figure 13: Graph of Asia Model – for the probabilities see either literature or file PNL/high/DEMO/models.cpp of PNL Release 1.0 (with correction described here in footnote).

fact implies that one *cannot* apply direct comparisons between the two. For example one may be tempted to substract $LS(Y \rightarrow Z)$ from $MI(Y, Z)$ to obtain an estimate of the overall strength of all causal connections between Y and Z *excluding* the direct link from Y to Z. However, as the above example clearly demonstrates, that is not feasible.

7.6 A Final Example: The "Visit to Asia" Network

As a final example we use the *Visit to Asia* network[5], which is often used to introduce Bayesian Networks (Figure 13). It was originally introduced by Lauritzen and Spiegelhalter [12] and represents a simplified version of a medical model to demonstrate the general concept of using Bayesian Networks in that context. It should thus *not* be used to draw any conclusions for actual medical decisions.

Figures 14, 15 and 16 show entropy, True Average Link Strength and Blind Average Link Strength, respectively, while Figure 17 shows some selected mutual information graphs.

The following demonstrates what kind of information can be read from the graphs.

- The Link Strengths Percentages in Figures 15 and 16 immediately show that *CancerOrTuberculosis* is a deterministic child of its parents (which, admittedly, in this case could have been guessed from its name, too).
- The mutual information graph in the center of Figure 17 shows that if one wants to determine whether a patient has LungCancer, that the most important *measurable* indicator in an Xray. Whether the person smokes or has Dyspnea (Shortness of breath) is also significant in making the diagnosis. Everything else appears to be of little importance (*according to this simplified model!*).

[5]Note that the AsiaModel in file PNL/high/DEMO/models.cpp of PNL Release 1.0 has a typo. The correct version should read: $P(Bronchitis = True|Smoking = True) = 0.6$, $P(Bronchitis = False|Smoking = True) = 0.4$.

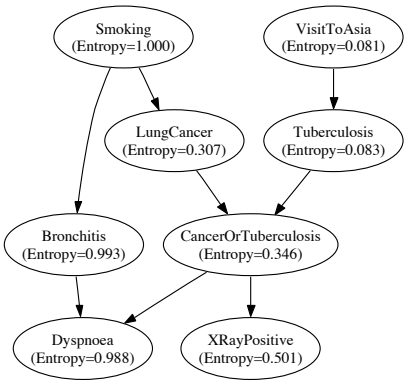


Figure 14: Entropy Graph for Asia Model.

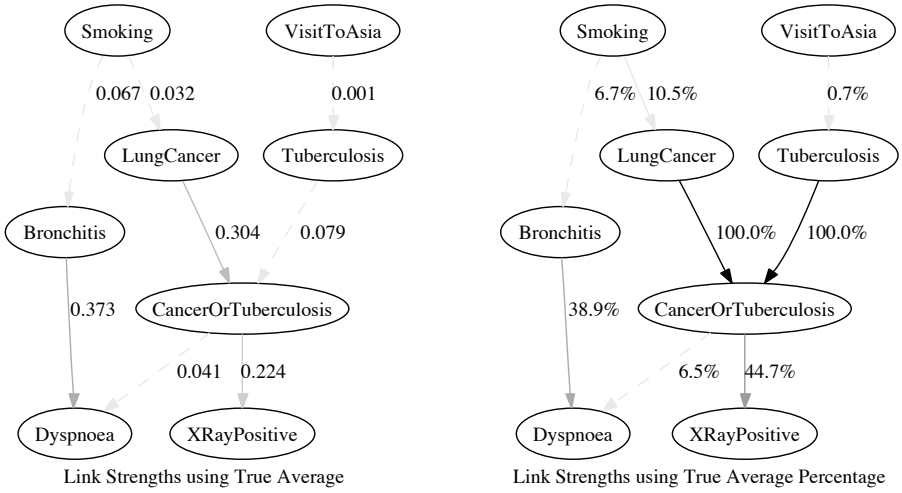


Figure 15: True Average Link Strength (left) and Percentage (right) for Asia Model.

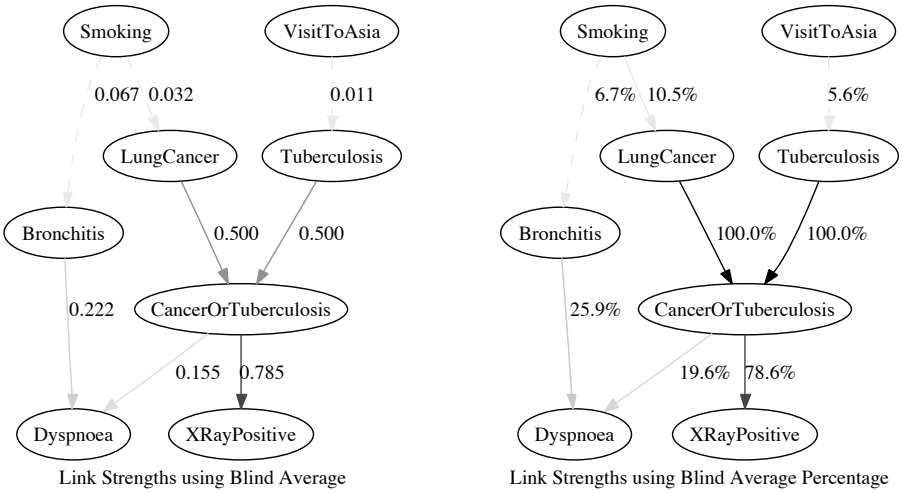


Figure 16: Blind Average Link Strength (left) and Percentage (right) for Asia Model.

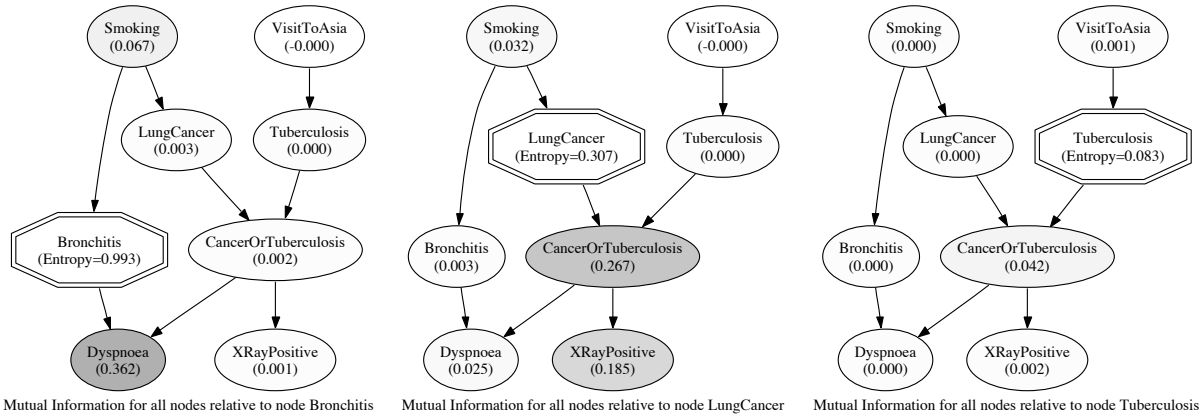


Figure 17: Connection Strength (Mutual Information) relative to node *Bronchitis* (left), *LungCancer* (center) and *Tuberculosis* (right) for Asia Model.

- As indicated by the True Average Percentages on the right of Figure 15 most links are quite strong. Keeping the comments on scale from Section 7.2 in mind all connections except for the one from *Visit to Asia* to *Tuberculosis* can be classified as significant.

  The Blind Average Percentage for *Visit to Asia* is much higher though, indicating that the reason for the low True Average Value is the low frequency of occurrence of the state $True$ for *Visit to Asia*. (Side comment: Nevertheless, since the cost of obtaining the information from a patient is small (no test, just a question) and the network is very simple anyway, there is no reason to eliminate the variable *Visit to Asia*.)

The last observation above concerning the *Visit to Asia* variable may indicate that it may be good practice to look not only at Mutual Information and True Average Link Strength, but also at Blind Average Link Strength, before deciding to eliminate a variable due to apparent low significance. In a nutshell, one could say that in this example True Average Link Strength only considers the benefit of the information of variable *Visit to Asia* for the *average* patient. In contrast Blind Average Link Strength consideres all patient categories equally – in this case the low number of patients actually having traveled to Asia are given equal weight tho those not having traveled there – and thus gives more attention to special cases (small groups) and the value of information of variable *Visit to Asia* for that special group.

8 Conclusions

Much work remains to be done to develop guidelines for the use of the measures presented here. There is also a need for developing additional measures that have a more intuitive scale. Nevertheless, we hope that the implementation of the measures in PNL along with this document will help to restart the discussion on measuring connection and link strength.

Finally, any feedback or thoughts would be very much appreciated! When working on a voluntary project like this one, there is nothing more rewarding than hearing from people who actually use it. Even constructive criticism is much better than no feedback at all... So, *please* send me your comments and questions!

9 References

[1] Boerlage, B., "Link Strengths in Bayesian Networks", M.S. Thesis, Dept. of Computer Science, The University of British Columbia, October 1992.

[2] Shannon, C.E., and Warren, W., "The Mathematical Theory of Communication", University of Illinois Press, Urbana and Chicago, 1949.

[3] Pearl, J., "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Interference", Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[4] Uffink, J., "Can the maximum entropy principle be explained as consistency requirement?", Studies in History and Philosophy of Modern Physics 26B (1995): 223-261.

[5] Klir, G.J., "Uncertainty and Information Measures for Imprecise Probabilities: An Overview", 1st International Symposium on Imprecise Probabilities and Their Applications, Ghent, Belgium, 29 June - July 2, 1999.

[6] Klir, G.J., and Smith, R.S., "On measuring uncertainty and uncertainty-based information: Recent developments", Annals of Mathematics and Artificial Intelligence, vol. 32, pp. 5-33, Kluwer Academic Publishers, 2001.

[7] Nicholson, A.E., and Jitnah, N., "Using Mutual Information to determine Relevance in Bayesian Networks", Pacific Rim International Conference on Artificial Intelligence", pp. 399-410, 1998.

[8] Nicholson, A.E., and Jitnah, N., "Treenets: A framework for anytime evaluation of belief networks.", First International Joint COnference on Qualitative and Quantitative Practical Reasoning, ECSQARU-FAPR'97, 1997. (Lecture Notes in Articifical Intelligence, Springer Verlag.)

[9] Lacave, C. and Diez, F.J., "The Elvira GUI: a tool for generating explanations for Bayesian networks" (submitted journal paper, in review), 2004.

[10] Lacave, C., "Explicacion en redes bayesianas causales. Aplicaciones medicas." Ph.D. Thesis, Dept. Artificial Intelligence, UNED, Madrid, Spain, Dec. 2002. (Available in Spanish at http://www.inf-cr.uclm.es/www/clacave/public/Publicaciones.htm.)

[11] Zapata-Rivera, J.D., Neufeld, E. and Greer, J. (1999). Visualization of Bayesian Belief Networks. IEEE Visualization 1999 Late Breaking Hot Topics Proceedings, pp. 85-88.

[12] Lauritzen, S. L., and Spiegelhalter, S.J., "Local computations with probabilities on graphical structures and their application to expert systems" in J. Royal Statistics Society B, 50(2), 157-194, 1998.