PNLtoGraphviz and LinkConnectionStrength (Version 1.0)–
Overview, Installation and Quick Command Reference

by Imme Ebert-Uphoff (imme@users.sourceforge.net)
January 21, 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.

1 Overview

This package contains the following two components:

1. **PNLtoGraphviz**
   - Is an interface from PNL to Graphviz[1] (Open-Source Graph Vizualization Software) that facilitates the vizualization of PNL network graphs.
   - It applies to any PNL network of type BayesNet, DBN or LIMID.
   - It generates a graph description for the network and writes it to a file that can be read by Graphviz. The Graphviz package can then be invoked to generate the actual graph rendering.
2. **LinkConnectionStrength**
   - Is a package for the Calculation and Vizualization of Entropy, Link Strengths and Connection Strengths
   - It currently applies only to Discrete Bayesian Networks.
   - It provides functions for the calculation of
     - entropy,
     - mutual information to measure connection strength between two nodes,
     - certain derivatives of mutual information to measure link strength of any directed arc.
   - Several output routines allow the user to print the resulting values to the screen in a formatted way.
   - Another set of output routines uses the PNLtoGraphviz interface to generate a picture of the graph, employing varying gray scales of links and nodes to indicate link strengths and connections strengths.

[1] Available for most platforms at http://www.graphviz.org

2 Installation Instructions

2.1 Step 1: Download Four C++ Files

To use *either* component of the PNLVizualize package, namely PNLtoGraphviz or LinkConnectionStrength, you must you must download *all* of the following four files:

- PNLtoGraphviz.hpp
- PNLtoGraphviz.cpp
- LinkConnectionStrength.hpp
- LinkConnectionStrength.cpp

(All available at www.DataOnStage.com.) Placing these four files in the same directory as your own PNL code and linking them to your code will allow you to use all functions of the two packages.

2.2 Step 2: Download Optional Sample Files

You may also want to download some examples for the use of the packages. These should be placed in the same directory as the four files above:

- SampleUsePTG.cpp: Main file for Demonstrating use of PNLtoGraphviz
- SampleUseLCS.cpp: Main file for Demonstrating use of LinkConnectionStrength
- models.h, models.cpp: Various network models used in the two SampleUse files above.

2.3 Step 3: Install GraphViz Package for Graph Layout and Rendering

If you *only* want to use the text output of the measures of the LinkConnectionStrength package, skip this section. However, most users will want to generate graphs, too. As mentioned before this package generates graph representations for PNL networks as *dot*-files, which can then be read by the GraphViz (open source visualization software) to generate the layout and rendering of the graph. This step creates images in most standard output formats (including jpg, gif, imap, mif, mp, pcl, ps, ps2, png, svg, vrml and many others).

Thus, if not already installed, get the GraphViz package (available at http://www.graphviz.org for most platforms) which is *very* easy to install and use. (You will probably find many other uses for GraphViz, too.)

3 How to Use the Package

The use of the PNLtoGraphviz interface is in two steps:

1. Within your PNL code call any of the PNLtoGraphviz functions for your network. This creates a graph file in DOT language (plin text description of the graph).

2. Outside of PNL: Open the DOT-file from Graphviz to generate automatic layout and rendering. A single command is generally sufficient. For example, to generate a postscript image on a Linux system from the file "graph.dot" generated by Step 1 one simply types in the command line:

   ```
   dot -Tps graph.dot -o graph.ps
   ```

   Using the DOT command on Windows and other platforms is just as easy. Please see the documention at http://www.graphviz.org for more information.

For more information, see [2].

The use of the LinkConnectionStrength functions is straight forward, especially when using the plain text interface. When generating graphs the process is the same as described above. The tricky part in this case is to interpret the results, e.g. what does it mean to have a link strength of 0.06? Is that strong or not? See [3] for a discussion on that topic.

4 Which User's Guides Should You Read?

For detailed information, please see the separate User's Guides for PNLtoGraphviz [2] and LinkConnectionStrength [3].

1. If you only want to use the PNLtoGraphviz functions to generate graphs *without* any information on entropy, link strength or connection strength:
   Only read the PNLtoGraphviz User's Guide [2].

2. If you only want to use the LinkConnectionStrength functions *without* generating graphs:
   Only read the LinkConnectionStrength User's Guide [3].

3. If you want to generate graphs that contain information on entropy, link strength or connection strength:
   Read the LinkConnectionStrength User's Guide [3] and at least glance over the PNLtoGraphviz User's Guide [2].

5 To Get Started

The easiest way to get started with either package is to try out the sample programs (SampleUsePTG and SampleUseLCS) and to paly around by changing commands in those files. This should give you a good feeling for how to use the files.

6 References

[1] Gansner, E., Koutsofios, E. and North, S., "Drawing graphs with *dot*", Feb. 2002. Available at http://graphviz.org/ by clicking on "Documentation" and selecting "User's Guide: dot".

[2] Ebert-Uphoff, I., "User's Guide for the PNLtoGraphviz Package (Version 1.0)". Available at www.DataOnStage.com.

[3] Ebert-Uphoff, I., "User's Guide for the LinkConnectionStrength Package (Version 1.0) – A PNL Package for the Vizualization of Entropy, Link Strength and Connection Strenghts in Discrete Bayesian Networks". Available at www.DataOnStage.com.

A Quick Reference of Commands

This section provides a quick reference only for the *syntax* of all available commands. For complete information on *what the commands do*, please see the documentation of PNLtoGraphviz [2] and LinkConnectionStrength [3].

A.1 Calculating Entropy, Connection Strength and Link Strength

Complete definitions of the formulas used to calculate entropy, connection strength and link strength are given in [3].

1. **Entropy**

   Input variables:

   - X_index: index of considered node (X)
   - BNet: BayesNet to which X belongs

   Output Variable: Returns the entropy of node X_index.

   One available function:

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

2. **Connection Strength**

   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.

   Two available functions:

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

   Calculates both *mutual information* and *mutual information "percentage"*.

   ```
   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. **Link Strength**

   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).

   Two available functions:

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

   Calculates both True/Blind Average Link Strength and Percentage.

```
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.

4. **Scaling Function**

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

   Determines the node in a discrete Bayesian Network with the largest number of states and returns its logarithm.

A.2 Plain Text Output Functions

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.

A.3 Graph Output

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. **Standard Graph for BayesNet**

```
int PNLtoGraphviz ( BayesNet * net, const string & filename,
                    map<string,string>
                    customized_node_shape=map<string, string>() );
```

Functionality: Graph showing nodes with node names, connected by arrows. Default node shape is ellipse.

2. **Standard Graph for DBN**

```
int PNLtoGraphviz ( DBN * net, const string & filename,
                    map<string,string>
                    customized_node_shape=map<string, string>() );
```

Functionality: Graph showing nodes of slices 0 and 1 with node names, connected by arrows. Nodes of each slice are grouped into a cluster, bounded by a box. Default node shape is ellipse.

3. **Standard Graph for LIMID**

```
int PNLtoGraphviz ( LIMID * net, const string & filename,
                    map<string,string>
                    customized_node_shape=map<string, string>() );
```

Functionality: Graph showing nodes with node names, connected by arrows. Default node shape is ellipse for chance nodes, rectangles for decision nodes and diamonds for value nodes.

4. **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.

5. **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.

6. **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.