

GL2U: A Python Implementation for Approximate Inference on Credal Nets using Generic Loopy 2U

SUN Yi
IDSIA
Galleria 2, CH-6928 Manno (Lugano)
Switzerland
yi@idsia.ch

January 25, 2008

1 Introduction

Credal networks (CNs) [3] have been revealed as one of the most powerful tools in modeling stochastic systems with imprecise probabilities [4]. Unlike Bayesian networks (BNs), the inference problems on CNs are far more complicated. It has been proved that the complexity of computing exact posterior probability intervals is generally NP [5] (except for some extremely simple cases [6]), which makes approximate inference inevitable.

Currently, one of the most effective approximate inference algorithms on CNs is the loopy 2U (L2U) algorithm [7], which works on the class of separately specified, binary credal nets. To broaden the usage of L2U, we propose the generic loopy 2U (GL2U) algorithm, which is capable of solving inference problems for any type of CNs. GL2U works by first converting input CNs into separately specified, binary CNs through local specification [1] and binarization [2] processes, then solving the converted models using L2U.

The GL2U algorithm is implemented in the package `GL2U`¹, which is an open source python package. The algorithm will be overviewd in the next section, while the software package would be introduced in section 3.

¹To avoid ambiguity, we use the typewriter font `GL2U` for the software package, and normal font GL2U for the algorithm itself.

2 Algorithms

2.1 2U

2U [6] is a polynomial algorithm for exact inference on singly-connected, separately specified binary credal nets. Given sets of observations, 2U generates the upper and lower (conditional) probabilities for each non-observed variables. The complexity of 2U is $O(N4^{D_p})$, where N is the number of variables and D_p is the maximum number of parents among all variables (width of the polytree).

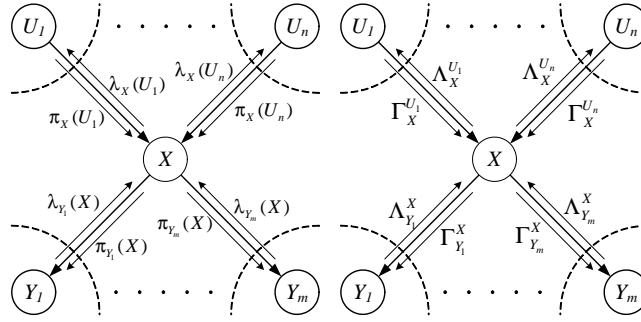


Figure 1: Illustration of Pearl's belief propagation (left) and the modified propagation (right)

The propagation of 2U is based on a slightly different version of Pearl's belief propagation [10]:

$$\begin{aligned}
 \Pr(X|\mathbf{e}) &= \alpha \pi(X) \lambda(X) \\
 \pi(X) &= \sum_{\mathbf{U}} \Pr(X|\mathbf{U}) \prod_{U_i \in \mathbf{U}} \pi_X(U_i) \\
 \lambda(X) &= \prod_{Y_j \in \mathbf{Y}} \lambda_{Y_j}(X) \\
 \pi_{Y_j}(X) &= \pi(X) \prod_{Y_k \in \mathbf{Y} \setminus \{Y_j\}} \lambda_{Y_k}(X) \\
 \lambda_X(U_i) &= \beta \sum_X \lambda(X) \sum_{\mathbf{U} \setminus \{U_i\}} \Pr(X|\mathbf{U}) \prod_{U_k \in \mathbf{U} \setminus \{U_i\}} \pi_X(U_k)
 \end{aligned}$$

Taking advantage of all variables are binary, we can remove the coefficient α, β and reformulate the propagation as:

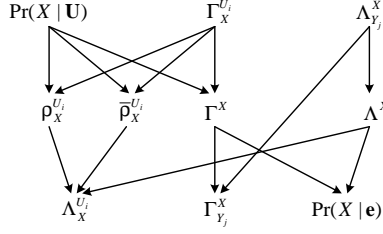


Figure 2: Dependency relations of variables in the modified propagation

$$\begin{aligned}
\Gamma^X &= \sum_{\mathbf{u} \in \Omega_{\mathbf{U}}} \Pr(x|\mathbf{u}) \prod_{u_i \in \mathbf{u}} \theta(u_i, \Gamma_X^{U_i}) \\
\Lambda^X &= \prod_{Y_j \in \mathbf{Y}} \Lambda_{Y_j}^X \\
\rho_X^{U_i} &= \sum_{\mathbf{U} \setminus \{U_i\}} \Pr(x|\mathbf{u} \setminus \{U_i\}, u_i) \prod_{U_k \in \mathbf{U} \setminus \{U_i\}} \theta(u_k, \Gamma_X^{U_k}) \quad \text{for } i = 1 \dots n \\
\bar{\rho}_X^{U_i} &= \sum_{\mathbf{U} \setminus \{U_i\}} \Pr(x|\mathbf{u} \setminus \{U_i\}, \neg u_i) \prod_{U_k \in \mathbf{U} \setminus \{U_i\}} \theta(u_k, \Gamma_X^{U_k}) \quad \text{for } i = 1 \dots n \\
\Pr(x|\mathbf{e}) &= \left(1 + \left(\frac{1}{\Gamma^X} - 1 \right) \frac{1}{\Lambda^X} \right)^{-1} \\
\Gamma_{Y_j}^X &= \left(1 + \left(\frac{1}{\Gamma^X} - 1 \right) \frac{1}{\prod_{Y_k \in \mathbf{Y} \setminus \{Y_j\}} \Lambda_{Y_k}^X} \right)^{-1} \quad \text{for } j = 1 \dots m \\
\Lambda_X^{U_i} &= \frac{\rho_X^{U_i} + (\Lambda^X - 1)^{-1}}{\bar{\rho}_X^{U_i} + (\Lambda^X - 1)^{-1}} \quad \text{for } i = 1 \dots n
\end{aligned}$$

where $\Gamma_X^{U_i}, \Lambda_{Y_j}^X$ are the input messages, and $\Gamma_{Y_j}^X, \Lambda_X^{U_i}$ are the output messages.

The 2U propagation is derived by computing upper and lower value of each messages and conditional probabilities in the modified propagation:

$$\begin{aligned}
\overline{\Gamma^X} &= \max_{\Gamma_X^{U_i} \in \{\underline{\Gamma}_X^{U_i}, \overline{\Gamma}_X^{U_i}\}} \sum_{\mathbf{u} \in \Omega_{\mathbf{U}}} \overline{\Pr(x|\mathbf{U})} \prod_{u_i \in \mathbf{u}} \theta(u_i, \Gamma_X^{U_i}) \\
\underline{\Gamma^X} &= \min_{\Gamma_X^{U_i} \in \{\underline{\Gamma}_X^{U_i}, \overline{\Gamma}_X^{U_i}\}} \sum_{\mathbf{u} \in \Omega_{\mathbf{U}}} \underline{\Pr(x|\mathbf{U})} \prod_{u_i \in \mathbf{u}} \theta(u_i, \Gamma_X^{U_i}) \\
\overline{\Lambda^X} &= \prod_{Y_j \in \mathbf{Y}} \overline{\Lambda_{Y_j}^X} \\
\underline{\Lambda^X} &= \prod_{Y_j \in \mathbf{Y}} \underline{\Lambda_{Y_j}^X}
\end{aligned}$$

$$\begin{aligned}
\overline{\Pr(x|\mathbf{e})} &= \left(1 + \left(\frac{1}{\overline{\Gamma^X}} - 1 \right) \frac{1}{\overline{\Lambda^X}} \right)^{-1} \\
\underline{\Pr(x|\mathbf{e})} &= \left(1 + \left(\frac{1}{\underline{\Gamma^X}} - 1 \right) \frac{1}{\underline{\Lambda^X}} \right)^{-1}
\end{aligned}$$

$$\overline{\Gamma_{Y_j}^X} = \left(1 + \left(\frac{1}{\overline{\Gamma^X}} - 1 \right) \frac{1}{\prod_{Y_k \in \mathbf{Y} \setminus \{Y_j\}} \overline{\Lambda_{Y_k}^X}} \right)^{-1}$$

$$\underline{\Gamma_{Y_j}^X} = \left(1 + \left(\frac{1}{\underline{\Gamma^X}} - 1 \right) \frac{1}{\prod_{Y_k \in \mathbf{Y} \setminus \{Y_j\}} \underline{\Lambda_{Y_k}^X}} \right)^{-1}$$

$$\overline{\rho_X^{U_i}(\Gamma_X^{\mathbf{U}})} = \sum_{\mathbf{U} \setminus \{U_i\}} \overline{\Pr(x|\mathbf{u} \setminus \{U_i\}, u_i)} \prod_{U_k \in \mathbf{U} \setminus \{U_i\}} \theta(u_k, \Gamma_X^{U_k})$$

$$\underline{\rho_X^{U_i}(\Gamma_X^{\mathbf{U}})} = \sum_{\mathbf{U} \setminus \{U_i\}} \underline{\Pr(x|\mathbf{u} \setminus \{U_i\}, u_i)} \prod_{U_k \in \mathbf{U} \setminus \{U_i\}} \theta(u_k, \Gamma_X^{U_k})$$

$$\overline{\bar{\rho}_X^{U_i}(\Gamma_X^{\mathbf{U}})} = \sum_{\mathbf{U} \setminus \{U_i\}} \overline{\Pr(x|\mathbf{u} \setminus \{U_i\}, \neg u_i)} \prod_{U_k \in \mathbf{U} \setminus \{U_i\}} \theta(u_k, \Gamma_X^{U_k})$$

$$\underline{\bar{\rho}_X^{U_i}(\Gamma_X^{\mathbf{U}})} = \sum_{\mathbf{U} \setminus \{U_i\}} \underline{\Pr(x|\mathbf{u} \setminus \{U_i\}, \neg u_i)} \prod_{U_k \in \mathbf{U} \setminus \{U_i\}} \theta(u_k, \Gamma_X^{U_k})$$

$$\Lambda_X^{U_i}(\Gamma_X^{\mathbf{U}})^+ = \frac{\overline{\rho_X^{U_i}(\Gamma_X^{\mathbf{U}})} + (\Lambda^X - 1)^{-1}}{\underline{\bar{\rho}_X^{U_i}(\Gamma_X^{\mathbf{U}})} + (\Lambda^X - 1)^{-1}}$$

$$\Lambda_X^{U_i}(\Gamma_X^{\mathbf{U}})^- = \frac{\underline{\rho_X^{U_i}(\Gamma_X^{\mathbf{U}})} + (\Lambda^X - 1)^{-1}}{\overline{\bar{\rho}_X^{U_i}(\Gamma_X^{\mathbf{U}})} + (\Lambda^X - 1)^{-1}}$$

$$\overline{\Lambda_X^{U_i}(\Gamma_X^{\mathbf{U}}, \Lambda^X)} = \begin{cases} \Lambda_X^{U_i}(\Gamma_X^{\mathbf{U}})^+, & \Lambda^X > 1 \\ 1, & \Lambda^X = 1 \\ \Lambda_X^{U_i}(\Gamma_X^{\mathbf{U}})^-, & \Lambda^X < 1 \end{cases}$$

$$\underline{\Lambda_X^{U_i}(\Gamma_X^{\mathbf{U}}, \Lambda^X)} = \begin{cases} \Lambda_X^{U_i}(\Gamma_X^{\mathbf{U}})^-, & \Lambda^X > 1 \\ 1, & \Lambda^X = 1 \\ \Lambda_X^{U_i}(\Gamma_X^{\mathbf{U}})^+, & \Lambda^X < 1 \end{cases}$$

$$\overline{\Lambda_X^{U_i}} = \max_{\Gamma_X^{U_k} \in \{\overline{\Gamma_X^{U_k}}, \underline{\Gamma_X^{U_k}}\}, U_k \in \mathbf{U} \setminus \{U_i\}} \left\{ \max_{\Lambda^X \in \{\overline{\Lambda^X}, \underline{\Lambda^X}\}} \overline{\Lambda_X^{U_i}(\Gamma_X^{\mathbf{U}}, \Lambda^X)} \right\}$$

$$\underline{\Lambda_X^{U_i}} = \max_{\Gamma_X^{U_k} \in \{\overline{\Gamma_X^{U_k}}, \underline{\Gamma_X^{U_k}}\}, U_k \in \mathbf{U} \setminus \{U_i\}} \left\{ \max_{\Lambda^X \in \{\overline{\Lambda^X}, \underline{\Lambda^X}\}} \underline{\Lambda_X^{U_i}(\Gamma_X^{\mathbf{U}}, \Lambda^X)} \right\}$$

2.2 Loopy 2U

Despite the excellent speed and accuracy, the class of credal nets where 2U is directly applicable is limited. Loopy 2U [7] extends 2U to work on a wider class of multiple-connected, separately specified binary credal nets, using similar idea behind loopy belief propagation [9].

L2U works as follows:

1. Choose arbitrarily a sequence S being on a path P in the network;
2. Initialize nodes without parents and evidence nodes as in 2U;
3. Initialize messages for edges not in P ;

4. Repeat updating $\overline{\Lambda}_X^{U_i}, \underline{\Lambda}_X^{U_i}, \overline{\Gamma}_{Y_j}^X, \underline{\Gamma}_{Y_j}^X$ for each variable X using equations in 2U until convergence or time out;
5. Compute $\overline{\Pr}(x|\mathbf{e}), \underline{\Pr}(x|\mathbf{e})$ for each un-observed variable from the final message $\overline{\Lambda}_X^{U_i}, \underline{\Lambda}_X^{U_i}, \overline{\Gamma}_{Y_j}^X, \underline{\Gamma}_{Y_j}^X$.

When L2U converges, it gives approximations of the conditional upper and lower probabilities. Currently, there is little theoretical result concerning convergence behavior or accuracy of L2U algorithm. However, empirical study shows that L2U converges fast and achieves satisfying accuracy in almost all cases [8].

2.3 Local specification

When using 2U & L2U, it is required that the input credal nets must be separately specified, where the local credal sets under different parent configurations should be defined independently. However, there are various classes of credal nets which do not fit into this category. Fortunately, according to [1], any credal net can be converted into an equivalent separately specified credal net in which all the conditional probabilities are either precise or vacuous.

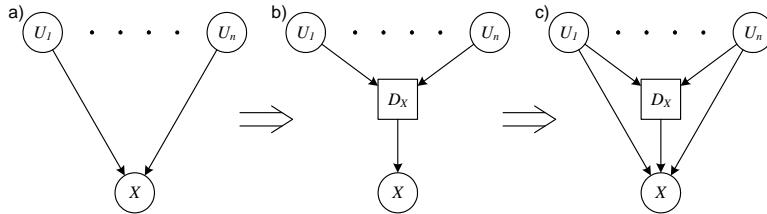


Figure 3: Local specifications for separately specified credal nets. a) the original credal net, b) transformation using method defined in [1], c) transformation in GL2U package

Currently, GL2U only implemented the local specification for separately specified credal nets. The transformation is described originally in [1] as:

For each variable X :

1. Let \mathbf{U} be the parent set of X , and $K(X|\mathbf{u})$ be the credal set of $\Pr(X|\mathbf{U} = \mathbf{u})$. Create a decision variable D_X , whose domain is the union of all credal sets $K(X|\mathbf{u})$ under all possible parent configurations, i.e.

$$\Omega_{D_X} = \bigcup_{\mathbf{u} \in \Omega_{\mathbf{U}}} K(X|\mathbf{u})$$

Since Ω_{D_X} is finite, we can give each possible (conditional) distribution of X an index, namely

$$\Omega_{D_X} = \{f_i(X) | f_i(X) \in K(X|\mathbf{u}) \text{ for some } \mathbf{u} \in \Omega_{\mathbf{U}}\}$$

2. Break all edges between X and U_i ;
3. Set \mathbf{U} as parent set of D_X ;
4. Set D_X as the only parent of X ;
5. Let η_n be a vector of length $|\Omega_{D_X}|$, with the i th entry $\eta_n [i] = \delta (i - n)$. Set the credal set for D_X given $\mathbf{U} = \mathbf{u}$ to be:

$$K (D_X | \mathbf{u}) = \{ \eta_n | f_n (X) \in K (X | \mathbf{u}) \}$$

6. Set the precise conditional distribution of X given D_X to be

$$\Pr (X | D_X = i) = f_i (X)$$

Using the above transformation, the decision variable D_X has extremely large cardinality bounded by $|\Omega_{pa(X)}| \cdot |K_m|$, where K_m is the maximum cardinality among all credal sets for X . In the package GL2U, we use an improved transformation, which significantly reduces the cardinality of D_X . The new transformation is defined as:

For each variable X :

1. Create a decision variable D_X of cardinality $K_m = \max_{\mathbf{u} \in \Omega_{\mathbf{U}}} |K (X | \mathbf{u})|$;
2. Add edges from \mathbf{U} to D_X ;
3. Add edge from D_X to X ;
4. Update the credal set for D_X given $\mathbf{U} = \mathbf{u}$ to be:

$$K (D_X | \mathbf{u}) = \{ \eta_1, \dots, \eta_{|K(X|\mathbf{u})|} \}$$

5. Update the precise conditional distribution of X to be $\Pr (X | \mathbf{U} = \mathbf{u}, D_X = i) = f_i (X | \mathbf{u})$ where $f_i (X | \mathbf{u})$ is the i th vertex of the credal set $K (X | \mathbf{u})$.

2.4 Binarization

Algorithm 2U & L2U work only on binary credal nets. To perform inference on non-binary credal nets, the input model should be transformed into an equivalent binary one. An approximate binarization method is described in [2], which translate the original models into an interval credal nets.

The algorithm is described as follows:

1. For each variable X , replace it with a new variable \hat{X} with cardinality being power of 2;

2. Replace conditional probability $\Pr(X|\pi_X)$ with $\Pr(\hat{X}|\pi_{\hat{X}})$. If the parent configuration $\pi_{\hat{X}}$ is compatible with π_X , then

$$\Pr(\hat{X} = i|\pi_{\hat{X}}) = \begin{cases} \Pr(X = i|\pi_X), & \text{if } i < |\Omega_X| \\ 0, & \text{if } i \geq |\Omega_X| \end{cases}$$

Otherwise, use random distribution of \hat{X} for $\Pr(\hat{X} = i|\pi_{\hat{X}})$;

3. For each \hat{X} , generate $N_X = \log_2 |\Omega_{\hat{X}}|$ binary variables, denoted as $B_X = \{B_X^0, \dots, B_X^{N_X-1}\}$. The combination of B_X uniquely indicates the state of \hat{X} ;
4. Set $\pi_B^X = \bigcup_{U \in \pi_{\hat{X}}} B_U$, rewrite $\Pr(\hat{X}|\pi_{\hat{X}})$ as $\Pr(B_X|\pi_B^X)$, and form the credal set $K(B_X|\pi_B^X)$ by rewriting all vertices in $K(\hat{X}|\pi_{\hat{X}})$;
5. Define $B_X^{a,b}$ as $\{B_X^j | a \leq j < b\}$. For each B_X^i , set its parents $\pi_{B_X^i}$ to be

$$\pi_B^X \cup B_X^{0,i}$$

And define the upper and lower conditional probability:

$$\overline{\Pr(B_X^i|\pi_B^X; B_X^{0,i})} = \max_{f(B_X^{i,N_X}) \in K(B_X^{i,N_X}|\pi_B^X; B_X^{0,i})} \sum_{B_X^{i+1,D_X}} f(B_X^{i,N_X})$$

$$\underline{\Pr(B_X^i|\pi_B^X; B_X^{0,i})} = \min_{f(B_X^{i,N_X}) \in K(B_X^{i,N_X}|\pi_B^X; B_X^{0,i})} \sum_{B_X^{i+1,D_X}} f(B_X^{i,N_X})$$

6. For each state i of each original variable X , define binary indicator variables I_X^i . Set the parent set of I_X^i to be B_X , with the precise conditional probability:

$$\Pr(I_X^i = 1|B_X) = \begin{cases} 1, & \text{if } B_X \text{ corresponds to } X = i \\ 0, & \text{otherwise} \end{cases}$$

We use the upper and lower probabilities of event $I_X^i = 1$ as approximations of the upper and lower probabilities of event $X = i$;

The interval credal nets generated by the algorithm are superset of the original models, in the sense that they include the complete credal sets in the original models. However, except in some extreme cases, the generated models also include vertices not in the original models, which makes the algorithm an approximate transformation. If using an exact inference algorithm, the posterior probability intervals computed from the binarization will contain the intervals computed from original models.

2.5 Local specification & binarization in one pass

In GL2U, we implement an exact algorithm converting separately specified non-binary credal nets into separately specified binary credal nets, where all the conditional probabilities are either precise or vacuous. The algorithm is exact in the sense that the transformed credal sets have one to one correspondence to the original credal sets, and if using an exact inference algorithm, the probability interval computed from both models are exactly the same.

The algorithm is defined as follows:

1. For each variable X , replace it with a new variable \hat{X} with cardinality being power of 2;
2. Replace conditional probability $\Pr(X|\pi_X)$ with $\Pr(\hat{X}|\pi_{\hat{X}})$. If the parent configuration $\pi_{\hat{X}}$ is compatible with π_X , then

$$\Pr(\hat{X} = i|\pi_{\hat{X}}) = \begin{cases} \Pr(X = i|\pi_X), & \text{if } i < |\Omega_X| \\ 0, & \text{if } i \geq |\Omega_X| \end{cases}$$

Otherwise, use random distribution of \hat{X} for $\Pr(\hat{X} = i|\pi_{\hat{X}})$;

3. For each \hat{X} , generate $N_X = \log_2 |\Omega_{\hat{X}}|$ binary variables, denoted as $B_X = \{B_X^0, \dots, B_X^{N_X-1}\}$. The combination of B_X uniquely indicates the state of \hat{X} ;
4. For each credal set $K(\hat{X}|\pi_{\hat{X}})$, generate decision variables $D_X = \{B_X^0, \dots, B_X^{M_X-1}\}$, where $M_X = \lceil \max_{\pi_{\hat{X}}} \log_2 |K(\hat{X}|\pi_{\hat{X}})| \rceil$, which is the logarithm of the maximum credal set size;
5. Extend every credal set $K(\hat{X}|\pi_{\hat{X}})$ to size of 2^{M_X} by repeating some vertices;
6. Set $\pi_B^X = \bigcup_{U \in \pi_{\hat{X}}} B_U$, rewrite $\Pr(\hat{X}|\pi_{\hat{X}})$ as $\Pr(B_X|\pi_B^X)$, and form the credal set $K(B_X|\pi_B^X)$ from $K(\hat{X}|\pi_{\hat{X}})$;
7. Localize $K(B_X|\pi_B^X)$ by breaking $K(B_X|\pi_B^X)$ into $K(D_X|\pi_B^X)$ and $\Pr(B_X|D_X; \pi_B^X)$, where $K(D_X|\pi_B^X)$ are vacuous credal sets and $\Pr(B_X|D_X; \pi_B^X)$ are precise probabilities. Define $\Pr(B_X|D_X; \pi_B^X)$ to be the k th vertices of credal set $K(B_X|\pi_B^X)$ if the combination of D_X indicates k ;
8. Define as before $B_X^{a,b}$ to be $\{B_X^j | a \leq j < b\}$. For each B_X^i , set its parent set $\pi_{B_X^i}$ to be

$$\pi_B^X \cup B_X^{0,i} \cup D_X$$

And the precise conditional probability

$$\Pr(B_X^i|\pi_B^X; B_X^{0,i}; D_X) = \sum_{B_X^{i+1, D_X}} \Pr(B_X^{i, N_X}|\pi_B^X; B_X^{0,i}; D_X)$$

9. For each state i of each original variable X , define binary indicator variables I_X^i . Set the parent set of I_X^i to be B_X , with the precise conditional probability:

$$\Pr(I_X^i = 1 | B_X) = \begin{cases} 1, & \text{if } B_X \text{ corresponds to } X = i \\ 0, & \text{otherwise} \end{cases}$$

2.6 GL2U

GL2U is an algorithm framework which combines the local specification, the binarization and the loopy 2U inference process. It is capable of dealing with any types of credal net, including non-separately specified and non-binary models. Currently, we only implement algorithms for inference on interval and separately-specified non-binary models, additional algorithms should be added in later updates.

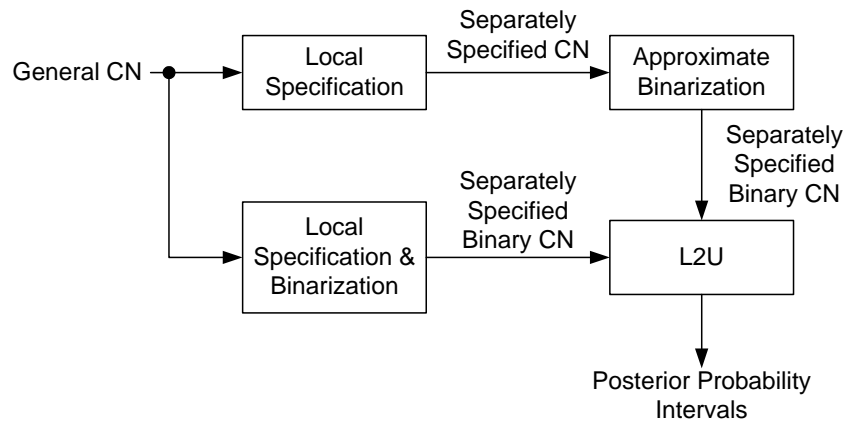


Figure 4: Flow chart of GL2U algorithm. The input model can be converted to separately specified binary CN through either separated local specification and binarization or one pass transformation.

3 GL2U Package

3.1 File Formats

GL2U support two types of model files: the `bif/cif` format and `xbif/xcif` format. For definition of such format, please refer to document of JavaBayes [12]. All parsers are defined in the folder `parsers`. For `bif/cif` files, we use a hand written parser; for `xbif/xcif` format, we use the standard `xml.sax` library, so it is capable of scanning large model description files.

To specify evidence, just write a simple text file indicate the variable and observed value, one observation per line. Here is an example:

```
variable0 state0
variable20 state2
variable37 state1
...
```

Sometimes, it is important to specify which variables are we interested in. To do so, we have to write a task file, each line suggests a variable that we are interested in. For example, we can write a task file to compute the posterior probability intervals of `variable0` and `variable2` only:

```
variable0
variable2
```

The reason to use a task file is that it can greatly reduce the indicator variables generated during the binarization process, hence significantly accelerating the computation.

3.2 Algorithms

All the transformation algorithms are implemented in `transform.py`. The main algorithms are:

- `binarize(net)`: perform approximate binarization, the result is an interval model;
- `bin_local_spec(net)`: perform the one pass binarization and local specification, the result is a credal net containing only vacuous and precise probabilities.

GL2U employs LRS [13] package to convert interval definitions into credal set definitions. The LRS package is interfaced in `wrapper.py`. The conversion is defined in `interval.py`, the main functions are:

- `to_bin_ln(net)`: convert a credal set definition into an interval definition.
- `to_credal_lu(net_l, net_u)`: convert to credal definition using lower and upper models.

The loopy 2U propagation is performed by directly calling 2UBayes [11]. For more details, please refer to documentation of 2UBayes.

3.3 Command Line

GL2U can be invoked as a command line tool. The original command line help string is:

```
gl2u [-j] [-n cif | -i head | -s itv | -xn xcif | -xi head
      | -xs xitv] [-e evi] [-t tsk] [-o result] [-bin | -lsb]
```

```

-n      : specify a cif file
-i      : specify file name header for a pair of interval files
-s      : specify a single interval cif file
-xn     : specify a xcif file
-xi     : specify a file name header for a pair of
          interval xcif files
-xs     : specify a single interval xcif file
-j      : read the input file using JavaBayes order
-e      : specify an evidence file
-t      : specify interested variables
          (to avoiding unnecessary indicators)
-o      : specify an result file
-bin    : inference using binarization + L2U
-lsb    : inference using localization + binarization + L2U

cif     : cif file name
head    : header of interval file name, the interval files
          are headLB.bif and headUB.bif
itv     : interval cif file name
xcif    : xcif file name
xhead   : header of interval xbif file name, the interval
          files are xheadLB.xbif and xheadUB.xbif
xitv    : interval xcif file name
evi     : evidence file
tsk     : task file name
result  : result file

```

The command line tool accepting 3 types of input:

- A single, standard `cif` or `xcif` file which contains the vertex definition of each credal sets;
- A pair of `bif` or `xbif` file which shares the same file name header and adds LB and UB to their file name (eg. `alarmLBr.bif` and `alarmUB.bif`, where `alarm` is the common file header). This represents an interval model. The lower file specifies the lower probabilities of all intervals and the upper file specifies the upper probabilities².
- An interval `cif` or `xcif` file, where each probability contains exactly two tables, where the first table specifies the lower values of the probability intervals and the second specifies the upper values.

²Note that the definition of lower and upper files is different from what in 2UBayes, where the lower and upper files are all valid Bayesian networks, and the lower file contains the distributions with smaller first value. In GL2U, the lower and upper files are no longer valid Bayesian networks, and the lower file contains all the lower probabilities of each intervals. It is because the definition of lower and upper files would not work if variables are not binary.

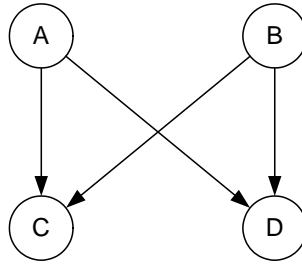


Figure 5: Example credal net

There are two main versions of `bif` and `xbif` format, which employs different table arrangements. For example, the one used in standard JavaBayes suggests that we arrange the probability value of $\Pr(A|B)$ as:

```
table Pr(a|b) Pr(a|$\lnot $b) Pr($\lnot $a|b) Pr($\lnot $a|$\lnot $b)
```

However, in some other places, the `bif` or `xbif` files are provided in a more natural order:

```
table Pr(a|b) Pr($\lnot $a|b) Pr(a|$\lnot $b) Pr($\lnot $a|$\lnot $b)
```

The latter arrangement is the default in GL2U. Add the `-j` option if the file uses the JavaBayes arrangement.

3.4 Example

The following example illustrates how to use the command line tool of GL2U. The credal net is defined by `test4s.cif`, which is an example in [8]:

```
network "test4s" { }
variable "A" { type discrete[2] { "0" "1" }; }
variable "B" { type discrete[2] { "0" "1" }; }
variable "C" { type discrete[2] { "0" "1" }; }
variable "D" { type discrete[2] { "0" "1" }; }
probability ( "A" ) {
  table 0.6 0.4;
  table 0.5 0.5; }
probability ( "B" ) {
  table 0.4 0.6;
  table 0.3 0.7; }
probability ( "C" "A" "B" ) {
  table 0.8 0.2 0.6 0.4 0.4 0.6 0.3 0.7;
  table 0.6 0.4 0.4 0.6 0.2 0.8 0.1 0.9; }
probability ( "D" "A" "B" ) {
  table 0.3 0.7 0.5 0.5 0.7 0.3 0.9 0.1;
  table 0.2 0.8 0.4 0.6 0.6 0.4 0.8 0.2; }
```

The most simple way of invoking GL2U inference is use the following command line:

```
gl2u -n test4s.cif -o test4s.result -bin
```

Since the network is itself binary, the binarization is not performed, and the inference result is in `test4s.result`:

```
Pr(A = 0 | e) : [0.500, 0.600]
Pr(A = 1 | e) : [0.400, 0.500]
Pr(B = 0 | e) : [0.300, 0.400]
Pr(B = 1 | e) : [0.600, 0.700]
Pr(C = 0 | e) : [0.295, 0.544]
Pr(C = 1 | e) : [0.456, 0.705]
Pr(D = 0 | e) : [0.480, 0.640]
Pr(D = 1 | e) : [0.360, 0.520]
```

Now, suppose that we observed $A = 1$, $D = 0$ then we write the evidence file `test4s.evi`:

```
A 1
D 0
```

And the command line:

```
gl2u -n test4s.cif -e test4s.evi -o test4s.result -bin
```

The output now is:

```
Pr(A = 1 | e) : <observed>
Pr(B = 0 | e) : [0.222, 0.368]
Pr(B = 1 | e) : [0.632, 0.778]
Pr(C = 0 | e) : [0.122, 0.337]
Pr(C = 1 | e) : [0.663, 0.878]
Pr(D = 0 | e) : <observed>
```

Further more, suppose that we are only interested in the probability of B , we can write a task file `test4s.tsk` containing a string "B" only, and write the command line:

```
gl2u -n test4s.cif -e test4s.evi -t test4s.tsk -o test4s.result -bin
```

The output is:

```
Pr(B = 0 | e) : [0.222, 0.368]
Pr(B = 1 | e) : [0.632, 0.778]
```

References

- [1] Antonucci, A., Zaffalon, M. (2006). Locally specified credal networks. In Studen? M., Vomlel, J. (Eds), PGM'06: Proceedings of the third European Workshop on Probabilistic Graphical Models, Action M Agency, Prague (Czech Republic), pp. 25–34.
- [2] Antonucci, A., Zaffalon, M., Ide, J. S., Cozman, F. G. (2006). Binarization algorithms for approximate updating in credal nets. In Penserini, L., Peppas, P., Perini, A. (Eds), STAIRS'06: Proceedings of the third European Starting AI Researcher Symposium. IOS Press, Amsterdam (Netherlands), pp. 120–131.
- [3] Cozman, F. G. (2000). Credal networks. *Artificial Intelligence Journal*, vol. 120, pp. 199-233.
- [4] Cozman, F. G. (2005). Graphical models for imprecise probabilities. *Journal of International Journal of Approximate Reasoning*, 39(2-3):167-184.
- [5] Campos, C. P., Cozman, F. G. (2005). The inferential complexity of Bayesian and credal networks. *IJCAI 2005*: 1313-1318.
- [6] Fagioli, E., Zaffalon, M. (1998). 2U: an exact interval propagation algorithms for polytrees with binary variables. *Artificial Intelligence*, 106:77–107.
- [7] Ide, J. S., Cozman, F.G. (2004). IPE and L2U: Approximate algorithms for credal networks, *Second Starting AI Researcher Symposium (STAIRS)*, pp. 118-127, IOS Press.
- [8] Ide, J. S., Cozman, F.G. (2007). Approximate algorithms for credal networks with binary variables. Manuscript for *IJAR*.
- [9] Murphy, K. P., Weiss, Y., Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. In *Conf. on Uncertainty in Artificial Intelligence*, pages 467–475.
- [10] Pearl, J. (1998). *Probabilistic Reasoning in Intelligent Systems*.
- [11] <http://www.pmr.poli.usp.br/ltd/Software/2UBayes/2UBayes.html>
- [12] <http://www.cs.cmu.edu/~javabayes/>
- [13] <http://cgm.cs.mcgill.ca/~avis/C/lrs.html>