

Lossless Bayesian Network Implementation

Exact Probabilistic Inference in C++

Shyamal Chandra

2025

Overview

- 1 Introduction
- 2 Architecture
- 3 Algorithms
- 4 Examples
- 5 Implementation
- 6 Features
- 7 Conclusion

What is a Bayesian Network?

- Probabilistic graphical model
- Represents variables and their conditional dependencies
- Directed Acyclic Graph (DAG) structure
- Enables efficient probabilistic reasoning

Joint Distribution Factorization:

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i | \text{Pa}(X_i))$$

What Does "Lossless" Mean?

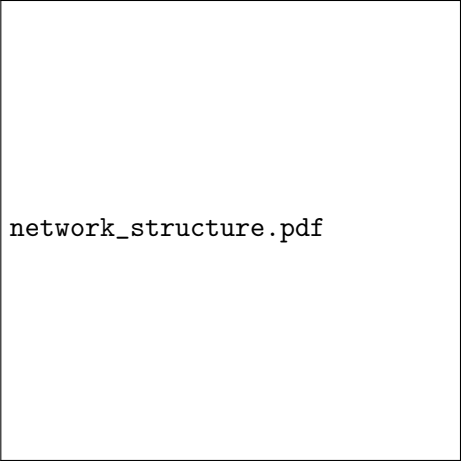
- **Exact computation:** No approximation in probability calculations
- **Full precision:** All probability values maintained exactly
- **Complete information:** No loss of probabilistic information
- **Deterministic results:** Same inputs always produce same outputs

Key Advantage

Unlike approximate methods (MCMC, variational inference), lossless methods provide exact results, crucial for applications requiring precision.

Core Components:

- Node class
 - Variable representation
 - State management
 - Parent relationships
- CPT class
 - Conditional probabilities
 - Multi-dimensional storage
 - Normalization
- BayesianNetwork class
 - Network construction
 - Inference algorithms
 - File I/O



`network_structure.pdf`

(Conceptual diagram)

Node Structure

```
1 class Node {  
2     std::string name;           // Variable name  
3     std::vector<std::string> states; // Possible states  
4     std::set<std::string> parentIds; // Parent nodes  
5     std::map<std::string, int> stateIndexMap; // Fast lookup  
6 };
```

Features:

- Fast state lookup: $O(1)$ via hash map
- Parent tracking for DAG structure
- Flexible state definitions

Conditional Probability Table

```
1 class ConditionalProbabilityTable {  
2     std::vector<double> probabilities; // Flat storage  
3     std::vector<size_t> dimensions;   // Multi-dim sizes  
4     std::vector<size_t> strides;      // Indexing strides  
5 };
```

Multi-dimensional Indexing:

$$\text{index} = \sum_{k=0}^{n-1} i_k \cdot \text{stride}_k$$

Features:

- Efficient storage in flat array
- Automatic normalization
- Validation of probability distributions

Kahn's Algorithm:

- ① Compute in-degrees for all nodes
- ② Initialize queue with nodes (in-degree = 0)
- ③ While queue not empty:
 - Remove node from queue
 - Decrease in-degree of children
 - Add children with in-degree = 0 to queue
- ④ If processed nodes $<$ total nodes: **CYCLE DETECTED**

Purpose

Ensures DAG property and provides ordering for efficient inference.

Variable Elimination

Exact Inference Algorithm:

- 1 **Query:** Variables of interest Q
- 2 **Evidence:** Observed variables $E = e$
- 3 **Hidden:** Variables to sum out H
- 4 **Compute:** $P(Q|E = e) = \frac{\sum_H P(Q, E=e, H)}{\sum_{Q, H} P(Q, E=e, H)}$

Complexity

Exponential in the number of variables, but exact results.

Medical Diagnosis Example

Network Structure:

- Disease \rightarrow Fever
- Disease \rightarrow Cough

Inference: Given: Fever=Yes, Cough=Yes
Query: $P(\text{Disease})$?

Results

- $P(\text{Disease}=\text{None}) = 0.001$
- $P(\text{Disease}=\text{Cold}) = 0.234$
- $P(\text{Disease}=\text{Flu}) = 0.765$

Classic Bayesian Network:

Nodes:

- Burglary
- Earthquake
- Alarm
- JohnCalls
- MaryCalls

Edges:

- Burglary \rightarrow Alarm
- Earthquake \rightarrow Alarm
- Alarm \rightarrow JohnCalls
- Alarm \rightarrow MaryCalls

Inference: Given JohnCalls=True, MaryCalls=True

Query: $P(\text{Burglary}=\text{True})$?

Code Example: Network Construction

```
1 BayesianNetwork network;
2
3 // Add nodes
4 network.addNode("Disease", "Disease",
5                 {"None", "Cold", "Flu"});
6 network.addNode("Symptom", "Fever", {"No", "Yes"});
7
8 // Add edge
9 network.addEdge("Disease", "Symptom");
10
11 // Create CPT
12 std::vector<size_t> dims = {3, 2};
13 ConditionalProbabilityTable cpt(dims);
14 cpt.setProbability({0}, 0, 0.9); // P(No|None) = 0.9
15 cpt.setProbability({0}, 1, 0.1); // P(Yes|None) = 0.1
16 cpt.normalize();
17 network.setCPT("Symptom", cpt);
```

Code Example: Inference

```
1 // Set evidence
2 std::map<std::string, std::string> evidence;
3 evidence["Symptom"] = "Yes";
4
5 // Perform inference
6 std::vector<std::string> query = {"Disease"};
7 auto results = network.variableElimination(query, evidence);
8
9 // Display results
10 for (const auto& pair : results) {
11     std::cout << "P(Disease=" << pair.first.at("Disease")
12               << ")_=_ " << pair.second << std::endl;
13 }
```

Key Features

- **Lossless Representation**

- Exact probability storage
- No approximation errors

- **Exact Inference**

- Variable elimination algorithm
- Deterministic results

- **DAG Validation**

- Automatic cycle detection
- Topological sorting

- **Flexible API**

- Easy network construction
- Comprehensive error handling

- **File I/O**

- Network serialization
- Persistent storage

Summary

- Complete C++ implementation of lossless Bayesian networks
- Exact inference using variable elimination
- Efficient data structures for probability storage
- Comprehensive documentation and examples

Applications

- Medical diagnosis systems
- Risk assessment
- Decision support systems
- Any application requiring exact probabilistic reasoning

Copyright (C) 2025, Shyamal Chandra

Thank you for your attention!

For more information, see the full documentation and reference manual.