# Lossless Bayesian Network Implementation

Shyamal Chandra

2025

**Abstract**

This document describes the implementation of a lossless Bayesian network in C++. The implementation provides exact inference capabilities using variable elimination, maintaining all probability information without approximation. The system supports directed acyclic graphs (DAGs) with conditional probability tables (CPTs) and provides a complete API for network construction, inference, and serialization.

## Contents

# 1 Introduction

Bayesian networks are probabilistic graphical models that represent a set of variables and their conditional dependencies via a directed acyclic graph (DAG). This implementation provides a *lossless* representation, meaning that all probability computations are performed exactly without approximation, preserving the full precision of the probability distributions.

## 1.1 Key Features

- **Lossless Representation**: All probabilities stored and computed exactly
- **Exact Inference**: Variable elimination algorithm for precise inference
- **DAG Validation**: Automatic cycle detection and topological sorting
- **Flexible Structure**: Support for arbitrary DAG structures
- **CPT Management**: Efficient storage and access of conditional probability tables
- **File I/O**: Network serialization and loading capabilities

# 2 Architecture

## 2.1 Core Components

The implementation consists of three main components:

### 2.1.1 Node Class

The `Node` class represents a variable in the Bayesian network. Each node has:

- A unique identifier and name
- A set of possible states
- Parent relationships (for DAG structure)
- Fast state lookup via index mapping

### 2.1.2 ConditionalProbabilityTable Class

The `ConditionalProbabilityTable` class stores conditional probabilities in a multi-dimensional array format. Key features:

- Efficient multi-dimensional indexing using stride calculations
- Automatic normalization of conditional distributions
- Validation of probability distributions
- Lossless storage of all probability values

#### 2.1.3 BayesianNetwork Class

The `BayesianNetwork` class is the main interface for working with Bayesian networks. It provides:

- Network construction (adding nodes and edges)
- DAG validation and topological sorting
- Exact inference using variable elimination
- Joint probability computation
- File I/O operations

# 3 Mathematical Foundation

## 3.1 Bayesian Network Definition

A Bayesian network is a pair $(G, P)$ where:

- $G = (V, E)$ is a directed acyclic graph with vertices $V$ (variables) and edges $E$ (dependencies)
- $P$ is a set of conditional probability distributions, one for each variable given its parents

The joint probability distribution factorizes as:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \mathrm{Pa}(X_i))$$

where $\mathrm{Pa}(X_i)$ denotes the parents of $X_i$ in the graph.

## 3.2 Inference

Given evidence $E = e$, we compute the posterior probability:

$$P(Q|E = e) = \frac{P(Q, E = e)}{P(E = e)} = \frac{\sum_{\mathbf{H}} P(Q, E = e, \mathbf{H})}{\sum_{\mathbf{Q},\mathbf{H}} P(Q, E = e, \mathbf{H})}$$

where $Q$ is the query variable, and $\mathbf{H}$ are hidden variables.

## 3.3 Variable Elimination

Variable elimination is an exact inference algorithm that:

1. Eliminates variables one at a time by summing them out
2. Maintains factors (functions over subsets of variables)
3. Computes the exact posterior distribution

# 4 Implementation Details

## 4.1 Data Structures

### 4.1.1 Multi-dimensional Array Indexing

The CPT uses a flat array with stride-based indexing. For dimensions $[d_0, d_1, \ldots, d_{n-1}]$, the stride for dimension $i$ is:

$$\text{stride}_i = \prod_{j=i+1}^{n-1} d_j$$

The flat index for multi-dimensional indices $[i_0, i_1, \ldots, i_{n-1}]$ is:

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

### 4.1.2 Topological Sorting

The network uses Kahn's algorithm for topological sorting:

1. Compute in-degrees for all nodes
2. Initialize queue with nodes having in-degree 0
3. Repeatedly remove nodes from queue and update in-degrees
4. Detect cycles if queue becomes empty before all nodes are processed

## 4.2 Inference Algorithm

The variable elimination algorithm:

1. Generate all possible assignments for query variables
2. For each query assignment, sum over all hidden variables
3. Normalize the resulting distribution

# 5 Usage Examples

## 5.1 Basic Network Construction

```
BayesianNetwork network;

// Add nodes
network.addNode("Disease", "Disease", {"None", "Cold", "Flu"});
network.addNode("Symptom", "Fever", {"No", "Yes"});

// Add edge
network.addEdge("Disease", "Symptom");

// Create and set CPT
std::vector<size_t> dims = {3, 2};
ConditionalProbabilityTable cpt(dims);
cpt.setProbability({0}, 0, 0.9);  // P(Fever=No | Disease=None) = 0.9
cpt.setProbability({0}, 1, 0.1);   // P(Fever=Yes | Disease=None) = 0.1
// ... set other probabilities
cpt.normalize();
network.setCPT("Symptom", cpt);
```

### 5.2 Performing Inference

```
// Set evidence
std::map<std::string, std::string> evidence;
evidence["Symptom"] = "Yes";

// Query
std::vector<std::string> queryNodes = {"Disease"};
auto results = network.variableElimination(queryNodes, evidence);

// Display results
for (const auto& pair : results) {
    std::cout << "P(Disease=" << pair.first.at("Disease")
              << ") = " << pair.second << std::endl;
}
```

## 6 File Format

The network can be saved to and loaded from files. The format includes:

- Node definitions (ID, name, states)
- Edge definitions (parent -> child)
- CPT data (dimensions and probabilities)

## 7 Performance Considerations

- **Time Complexity**: Variable elimination is exponential in the number of variables in the worst case
- **Space Complexity**: CPT storage is exponential in the number of parents
- **Optimization**: Topological ordering minimizes computation during inference

## 8 Error Handling

The implementation includes comprehensive error handling:

- Cycle detection when adding edges
- Validation of probability values (must be in [0, 1])
- Normalization checks for CPTs
- Missing node/state validation

## 9 Conclusion

This implementation provides a complete, lossless Bayesian network system with exact inference capabilities. The design emphasizes correctness and precision, making it suitable for applications requiring exact probabilistic reasoning.

## 10 References


- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann.
- Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models.* MIT Press.
- Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach.* Pearson.