Design Document: Bayesian Network Inference Engine

Taron Moe-Stull
taronmoe@icloud.com

October 2025

System Overview

This document describes the design of a Bayesian Network inference system that supports both exact and approximate inference. The system is designed to load Bayesian Networks from BIF files, apply evidence consistently, and produce marginal distributions over a requested set of variables.

Two inference strategies are implemented behind a shared interface:

- **Variable Elimination (VE)** for exact inference using factor algebra.
- **Gibbs Sampling** for approximate inference using Markov blanket conditionals.

The system is built with an emphasis on correctness, traceability, and clean separation of concerns: parsing, evidence handling, inference, metrics, and output formatting are modular and independently testable.

System Requirements

The system runs from a single parameter configuration (e.g., a notebook parameter cell or a configuration block) and executes one algorithm on one chosen network under a specified evidence scenario.

Inputs and Parameters

A run is defined by the following inputs (mirroring the configuration in the original design)

- `GROUP_ID` (used as an identifier in filenames; for portfolio usage this can be a project tag)
- `ALGORITHM` $\in \{$`ve`, `gibbs`$\}$
- `NETWORK_NAME` (path to a `.bif` file, e.g., `/networks/child.bif`)
- `REPORT` (semicolon-separated list of query variables)
- `EVIDENCE_LEVEL` (None, Little, Moderate)
- `EVIDENCE` (semicolon-separated list of `var=value` pairs)

Notes:

- `REPORT` and `EVIDENCE` are semicolon separated strings and are parsed into arrays/maps.
- Evidence values are validated against variable domains read from the network.

Outputs

After inference, the system writes a CSV output file containing normalized marginal distributions. The naming convention is:

`[GROUP_ID]_[ALGORITHM]_[NETWORK_NAME]_[EVIDENCE_LEVEL].csv`

Each reported variable is written using a two line format:

- header line containing variable name and value labels
- probability line containing the corresponding marginal probabilities

The output writer validates that each distribution sums to 1.0 (within tolerance) before saving.

System Architecture

The system is structured as an end-to-end pipeline driven by a top level runner ("Notebook" conceptually), which orchestrates parameter loading, network parsing, evidence application, inference execution, and output writing.

Major Components

Notebook (Driver): Top-level coordinator that executes the full pipeline from parameters to a saved CSV. Major responsibilities include: `run()`, `loadParams()`, `loadNetwork()`, `selectAlgorithm()`, `applyEvidence()`, `runInference()`, `writeOutputs()`.

Parameters: A structured holder for configuration values: `GROUP_ID`, `ALGORITHM`, `NETWORK_NAME`, `REPORT`, `EVIDENCE_LEVEL`, `EVIDENCE`. Includes `validate()` to confirm supported algorithm choice, valid file path, and legal evidence assignments.

BIF Parser: Reads `.bif` files and builds an in-memory Bayesian Network. Major methods: `parse()`, internal node parsing, and `validate()`.

Bayesian Network: Stores the network structure and CPTs, and provides accessors needed for both inference strategies: `childrenOf`, `parentsOf`, `markovBlanket`.

Factor: Core data structure for VE implementing factor operations: `restrict(var,value)`, `multiply(other)`, `sumOut(var)`, `normalize()`.

Evidence Manager: Single source of truth for parsing/validating evidence and applying it consistently for both VE and Gibbs. Produces a canonical `Map <var,value>` and supports clamping behavior for sampling.

Inference Algorithm (Interface): Common contract for inference implementations: `infer(bn, queryVars, evidenceMap)` → `InferenceResult`.

Variable Elimination (Exact): Builds initial factors, restricts by evidence, chooses an elimination order (supports arbitrary orders), then eliminates hidden variables via factor multiplication and summation.

Gibbs (Approximate): Initializes a complete state consistent with evidence, samples non-evidence variables from Markov blanket conditionals, and estimates marginals from tallies. Supports burn-in and thinning.

Inference Result: Standard container for marginal distributions and label orderings so output formatting is uniform across algorithms.

Results Writer: Formats and writes the required CSV output. Validates normalization.

Metrics: Records counters and summary statistics used to compare behavior across algorithms:

- VE: factor multiplications, variables eliminated
- Gibbs: samples drawn, convergence statistics / deltas

ScenarioRunner (Optional): Runs predefined scenarios and compares against known expected distributions for sanity checking.

Architecture Diagram

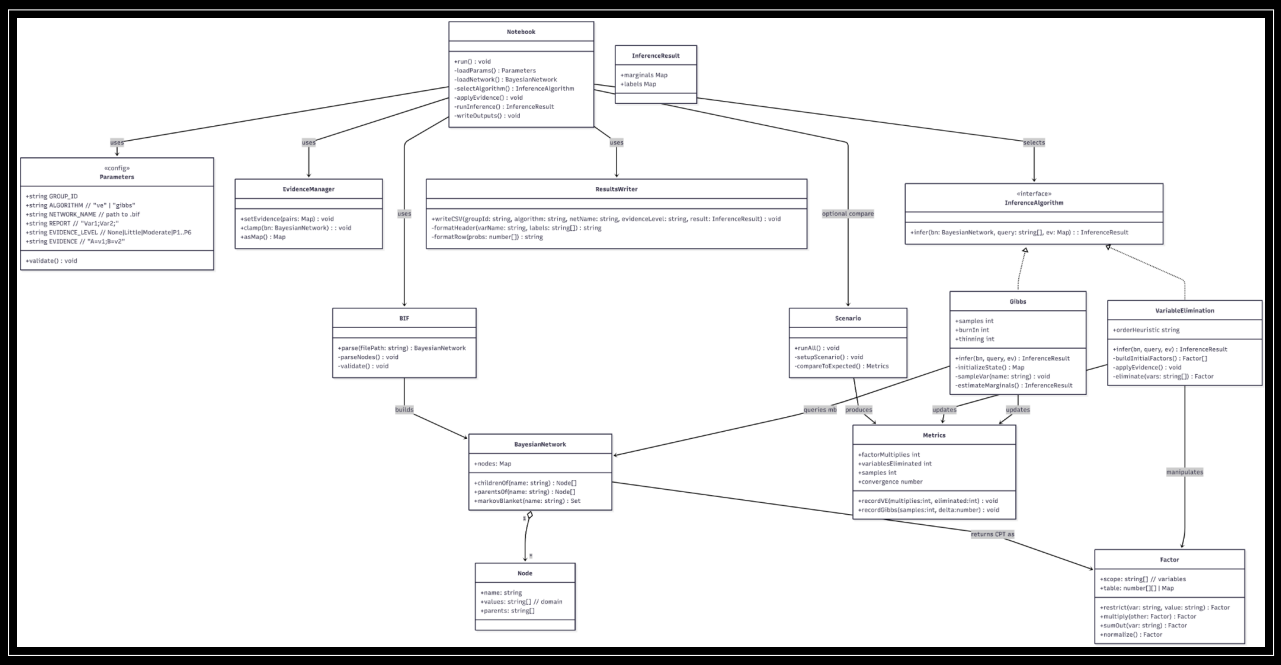

Figure 1: System architecture and class relationships for the Bayesian Network inference engine (VE + Gibbs).

System Flow

The inference engine runs as a single pipeline from configuration to CSV output:

1. **Parameter Loading:** Read configuration values and validate: network path exists, algorithm is supported, and evidence assignments match variable domains.

2. **Network Parsing:** Parse the selected `.bif` file to extract nodes, value domains, parent relationships, and CPTs, then build an in-memory `Bayesian Network`.
3. **Evidence Integration:** Parse `EVIDENCE` and `EVIDENCE_LEVEL` into a canonical evidence map. Apply evidence consistently:
   - VE: restrict factors prior to elimination.
   - Gibbs: clamp evidence variables during sampling.
4. **Algorithm Selection:** Select VE or Gibbs based on `ALGORITHM`.
5. **Inference Execution:**
   - VE: build initial factors, apply restrictions, choose elimination order, multiply and sum-out hidden variables, normalize marginals.
   - Gibbs: initialize a state, iterate sampling of non-evidence variables, apply burn-in and thinning, estimate marginals from sample frequencies.
6. **Normalize and Validate Results:** Ensure each returned distribution is normalized and label ordering is consistent.
7. **Output:** Write the CSV file following the naming convention and two-line distribution format.
8. **Metrics:** Record inference behavior counters and (when applicable) compare distributions against expected scenario outputs.

Test Strategy

Testing is structured to validate requirements from configuration to output:

- **Parameter validation tests:** reject invalid algorithms, missing networks, illegal evidence assignments.
- **Parser validation:** confirm node counts, edge structure, and CPT normalization for each provided network.
- **Evidence correctness:** ensure VE restrictions and Gibbs clamping are applied identically from the same evidence map.
- **VE correctness:** compare exact marginals against known expected outputs for multiple evidence scenarios.
- **Gibbs behavior:** verify convergence trends and error reduction as sample count increases; confirm stable estimates under burn-in/thinning schedules.
- **Output validation:** CSV formatting, filename correctness, and row-sum sanity checks.

A practical testing approach is to maintain a matrix across networks, evidence levels, and algorithms that records: parsing success, evidence integration correctness, VE exact match status, and Gibbs approximation error statistics.

Roadmap

Development can be staged without coupling implementation to any one environment:

1. Build and validate `BIF` parsing and `BayesianNetwork` structure.
2. Implement `Factor` operations and validate algebra (restrict/multiply/sum-out/normalize).
3. Implement `VariableElimination` with pluggable elimination ordering and exact marginal checks.
4. Implement `Gibbs` sampling with evidence clamping and verify behavior under increasing sample sizes.
5. Integrate the full pipeline and harden output formatting and normalization validation.

References

1. Russell, S., & Norvig, P. *Artificial Intelligence: A Modern Approach.* Pearson, 4th ed., 2020.
2. Mermaid Live Editor. Diagramming tool used to draft architecture and class diagrams.
3. GeeksforGeeks. Background reading on Bayesian Networks, exact inference, and Gibbs sampling.