ClueChain: A Bayesian Network-Based Mystery Solver Application

Saif Mohammed
2121913042
Department of ECE
North South University
Dhaka, bangladesh
saif.mohammed@northsouth.edu

Humayra Rahman Nipa
2121128042
Department of ECE
North South University
Dhaka, Bangladesh
humayra.nipa@northsouth.edu

Mainul Kaysar
2011615042
Department of ECE
North South University
Dhaka, Bangladesh
kazi.kaysar@northsouth.edu

Jannatul Islam
2014159642
Department of ECE
North South University
Dhaka, Bangladesh
jannatul.islam@northsouth.edu

Saif Uz Zaman
1931587042
Department of ECE
North South University
Dhaka, Bangladesh
saif.zaman01@northsouth.edu

***Abstract*—Reasoning under uncertainty is critical in complex domains requiring the synthesis of disparate evidence. This paper introduces ClueChain, an interactive application developed to demonstrate the application of probabilistic reasoning for solving fictional crime scenarios. The system utilizes a Discrete Bayesian Network (DBN), implemented using the pgmpy library, to model the relationships between potential suspects and various forms of evidence (e.g., forced entry, alibis, fingerprints). The network structure encodes causal assumptions (Guilty Party → Evidence), and Conditional Probability Distributions (CPDs) quantify these dependencies based on scenario logic. ClueChain features a graphical user interface built with Streamlit, enabling users to input observed evidence dynamically. Upon evidence submission, the system employs the Variable Elimination algorithm to perform Bayesian inference, computing the posterior probability distribution P(GuiltyParty — Evidence). This provides an updated, quantitative assessment of each suspect's likelihood of guilt. Furthermore, the application integrates graphviz to visualize the DBN structure, clarifying the underlying conditional independence assumptions. ClueChain serves as a practical demonstration of Bayesian Network capabilities for evidence integration and belief updating, offering an accessible platform for understanding probabilistic inference in decision support contexts.**

***Index Terms*—Bayesian networks, probabilistic inference, reasoning under uncertainty, conditional probability, variable elimination, decision support systems**

I. Introduction

Reasoning under uncertainty is fundamental to artificial intelligence and decision-making, particularly when synthesizing incomplete or ambiguous evidence [1]. While deterministic methods often fall short, Bayesian Networks (BNs) offer a robust graphical framework for modeling probabilistic relationships and performing inference [2], [3]. BNs enable the representation of conditional dependencies and provide mechanisms based on Bayes' Theorem to update beliefs as new evidence emerges, making them ideal for evidential reasoning tasks. This paper introduces ClueChain, an interactive application demonstrating the use of BNs for solving a fictional mystery scenario. Implemented using pgmpy [4] for the probabilistic model and Streamlit [5] for the user interface, ClueChain models potential suspects and evidence variables within a Discrete Bayesian Network (DBN). The structure encodes assumed causal influences (Guilty Party → Evidence), quantified by Conditional Probability Distributions (CPDs). Users input clues, and the system performs Bayesian inference via Variable Elimination [6], [7] to compute the posterior probability P(GuiltyParty — Evidence), visualizing the network structure with graphviz [8]. ClueChain serves as a pedagogical tool illustrating the translation of uncertain problems into probabilistic models and the dynamics of evidence integration through Bayesian inference. While using a fictional case, the demonstrated principles apply broadly to real-world evidence aggregation and assessment problems. The subsequent sections detail the BN model, application implementation, usage examples, and conclusions.

II. Literature Review

The ClueChain system builds upon established research in probabilistic reasoning, specifically Bayesian Networks (BNs), and leverages contemporary software tools for implementation and visualization. This review highlights key literature, emphasizing relevant conference contributions.

A. Probabilistic Reasoning and Bayesian Networks

Addressing uncertainty is a cornerstone of modern Artificial Intelligence [1]. While early AI focused on logic, the need to handle probabilistic information led to significant developments. Bayesian Networks, introduced and formalized through

seminal works often presented at conferences like Uncertainty in Artificial Intelligence (UAI) and the American Association for Artificial Intelligence (AAAI) [2], provide a graphical framework for representing and reasoning with probabilistic dependencies. A BN comprises a Directed Acyclic Graph (DAG), where nodes are random variables and edges denote direct influence, alongside Conditional Probability Distributions (CPDs) quantifying these influences [3]. This structure enables a compact representation of the joint probability distribution by encoding conditional independence assumptions [2], [3]. Pearl's foundational work established BNs as a principled method for evidential reasoning [2]. The utility of BNs has been demonstrated across diverse application domains presented in various conferences, ranging from medical diagnosis systems discussed at AMIA (American Medical Informatics Association) or AI in Medicine (AIME) conferences [e.g., [9]] to system monitoring and diagnostics presented at engineering or reliability conferences. These applications showcase the ability of BNs to integrate multiple sources of uncertain evidence for diagnostic or predictive assessments [10].

The implementation of ClueChain relies on specialized software libraries. The pgmpy library, introduced at the Scientific Computing with Python (SciPy) conference [4], provides essential tools for defining BN structures, specifying CPDs, and executing inference algorithms like VE within the Python ecosystem. For the interactive user interface, Streamlit [5] offers a framework for rapidly developing data-centric web applications, though typically cited via its documentation rather than a specific conference paper. Finally, visualizing the BN structure, crucial for model understanding and validation, is achieved using Graphviz, a graph visualization toolset whose foundational concepts were presented in software engineering venues [8]. ClueChain integrates these theoretical principles and software tools to create an interactive platform for exploring Bayesian evidence integration.

III. Methodology

A. Overview of Bayesian Networks and Probabilistic Inference

Bayesian networks (BNs) are probabilistic directed acyclic graphical (DAG) models [2], [3]. They use nodes to represent random variables, arcs to signify direct dependencies between linked nodes, and conditional probability tables (CPTs), often referred to as Conditional Probability Distributions (CPDs) in the context of discrete variables, to quantify these dependencies.

For n random variables $X_1, X_2, \ldots, X_n$ and a DAG $G = (V, E)$ with n nodes (where node j corresponds to X_j), the BN represents the joint distribution of $X_1, X_2, \ldots, X_n$ as:

$$P(X_1, X_2, \ldots, X_n) = \prod_{j=1}^{n} P(X_j \mid \text{Pa}(X_j)), \tag{1}$$

where $\text{Pa}(X_j)$ denotes the parents of X_j in the graph G (i.e., all variables X_i such that there is a directed edge $(X_i, X_j) \in E$).

1) Conditional Independence and Chain Rule: The BN's structure encodes conditional independence assumptions. Specifically, a variable is conditionally independent of its non-descendants, given its parents [2]. This allows the joint probability distribution over the set of variables $V = \{X_1, \ldots, X_n\}$ to factorize according to the structure of the graph, as shown in Eq. 1. This factorization is often referred to as the chain rule for Bayesian networks.

2) Marginalization: Marginalization involves computing the probability distribution of a subset of variables $Q \subseteq V$ by summing (or integrating, for continuous variables) the joint probability distribution over the remaining variables $V \setminus Q$:

$$P(Q) = \sum_{V \setminus Q} P(X_1, \ldots, X_n), \tag{2}$$

where the summation is over all possible configurations of the variables not in Q.

3) Probabilistic Inference: Given a BN, probabilistic inference aims to compute the conditional probability distribution of a set of query variables $Q \subseteq V$, given that another set of evidence variables $E \subseteq V$ have been observed to take on specific values $\mathbf{e}$.

Input:

- A Bayesian network representing $P(X_1, \ldots, X_n)$.
- A set of evidence variables $E \subseteq V$ and their observed values $\mathbf{e}$.
- A set of query variables $Q \subseteq V$, where typically $Q \cap E = \emptyset$.

Output:

- The posterior probability distribution $P(Q \mid E = \mathbf{e})$.

The posterior probability is calculated using the definition of conditional probability:

$$P(Q = \mathbf{q} \mid E = \mathbf{e}) = \frac{P(Q = \mathbf{q}, E = \mathbf{e})}{P(E = \mathbf{e})} \tag{3}$$

where the joint probabilities $P(Q, E)$ required in the numerator and the denominator (for normalization) are computed by marginalizing the full joint distribution derived from Eq. 1:

$$P(Q = \mathbf{q}, E = \mathbf{e}) = \sum_{V \setminus (Q \cup E)} P(X_1, \ldots, X_n) \tag{4}$$

Inference algorithms like Variable Elimination [6], [7] perform these conditioning and marginalization steps efficiently by exploiting the network structure encoded in the factorization (Eq. 1).

B. Dataset Collection

A defining aspect of the methodology employed for the ClueChain project is that it did not involve the collection or utilization of a traditional empirical dataset. The primary objective was to develop an illustrative application demonstrating Bayesian Network reasoning principles within a controlled, narrative-driven context, rather than learning model parameters from real-world observations. Consequently, standard procedures such as sourcing raw data, data sampling, or

compiling case records were not performed. The foundational information used to construct the model was derived entirely from the qualitative description of the fictional "Case of the Missing Manuscript" scenario (detailed in Section III.D). This scenario description served as the surrogate for empirical data, providing the basis for identifying relevant variables and, crucially, for the knowledge engineering process used to elicit the prior and conditional probabilities (CPDs) that parameterize the Bayesian Network (Section III.F). Therefore, the 'dataset' in this context was the curated narrative itself, and the 'collection' process involved formalizing the elements and logical relationships described within that narrative into the structure and parameters of the probabilistic model. This approach contrasts significantly with data-driven modeling where statistical patterns are extracted from large datasets to estimate model parameters.

C. System Architecture

ClueChain is implemented as a modular Python application leveraging distinct libraries for specific functionalities, organized into two primary components. The core probabilistic model and inference engine reside in a dedicated backend module (e.g., 'mystery_solver.py'). This module utilizes the pgmpy library [4] to define the Bayesian Network structure (nodes and edges), specify the Conditional Probability Distributions (CPDs), and provide functions for model validation and inference. The frontend user interface and interaction logic are managed by the main application script (e.g., 'main.py'), built using the Streamlit framework [5]. This script handles user input for evidence through interactive widgets, invokes the inference engine from the backend module using the Variable Elimination algorithm provided by pgmpy [6], [7], and presents the prior and posterior probabilities using pandas DataFrames and Streamlit's charting capabilities. Visualization of the network structure is integrated via the graphviz library [8], conditionally rendered if the library and its system dependencies are available. This modular architecture separates the probabilistic reasoning core from the presentation layer, facilitating maintainability and potential future extensions.

D. Fictional Mystery Scenario Description

To provide a concrete context for applying Bayesian Network modeling and inference, the ClueChain application is centered around a specific fictional narrative: "The Case of the Missing Manuscript." This scenario posits that a rare and valuable manuscript has been stolen overnight from a library room that was securely locked. The investigation focuses on three primary suspects, each presenting different potential motives and means. Suspect A, designated "The Scholar," had legitimate access to the library and possessed a known professional rivalry with the manuscript's owner, suggesting a possible motive. Suspect B, "The Butler," held keys to the room, granting access, but claims to have heard nothing unusual during the night. Suspect C, "The Cat Burglar," is a known professional thief operating in the vicinity, whose motive would presumably be the manuscript's monetary value, but who lacks an obvious prior connection to the specific premises. The scenario implies the potential availability of various forms of evidence, including whether forced entry occurred, the verifiability of each suspect's alibi for the time of the theft, the discovery (or lack thereof) of identifiable fingerprints matching one of the suspects, and the potential usefulness of security footage. This narrative framework, with its defined suspects, potential evidence types, and inherent uncertainties, served as the basis for defining the random variables, network structure, and conditional probabilities within the Bayesian Network model detailed in the subsequent section.

E. Data Representation

Within the ClueChain application, data is represented in several distinct forms that correspond to the Bayesian Network structure, its parameters, user-provided runtime evidence, and the resulting inferences. The fundamental model structure, implemented as a Directed Acyclic Graph (DAG), is represented internally using the `pgmpy` library's [4] `DiscreteBayesianNetwork` class. This object maintains the network's nodes as string variables (e.g., `'GuiltyParty'`, `'ForcedEntry'`) and the directed edges as a list of dependency tuples (e.g., `[('GuiltyParty', 'ForcedEntry'), ...]`), perfectly aligning with the formal graph definition $G = (V, E)$. For visualization purposes, this structure is translated into a `graphviz` object [8] that renders the nodes and their dependency relationships.

The probabilistic parameters (Θ) that quantify the relationships within the network are encapsulated using `pgmpy`'s `TabularCPD` objects. Each `TabularCPD` instance represents the conditional probability distribution $P(X_j \mid \text{Pa}(X_j))$ for a single variable, storing the variable name, its cardinality, parent variables' information, and the probability values in a multi-dimensional array format. A crucial feature is the `state_names` dictionary that maps numerical indices (0, 1, 2...) to meaningful state labels (e.g., `0: 'Yes', 1: 'No'`), maintaining consistency between numerical computations and conceptual understanding. The complete set of these `TabularCPD` objects forms the parameter set Θ for the Bayesian network $\mathcal{B} = (G, \Theta)$.

At runtime, user-provided evidence is handled through a Python dictionary structure where keys correspond to variable names (e.g., `'ForcedEntry'`, `'Fingerprints'`) and values represent the observed states (e.g., `'Yes'`, `'A'`), matching the predefined `state_names` in the corresponding CPDs. Variables marked as 'Unknown' are intentionally omitted from this dictionary, allowing the inference algorithm to properly marginalize over these unspecified states. This dictionary format directly interfaces with `pgmpy`'s inference methods through their `evidence` parameter requirement.

The inference results, representing the posterior probability distribution over the `GuiltyParty` variable, are initially generated as a `pgmpy` `DiscreteFactor` object. For user presentation and interaction within the Streamlit interface [5], this object is converted into a `pandas` DataFrame with two

primary columns: `Suspect` (containing the state names 'A', 'B', 'C' of the `GuiltyParty` variable) and `Probability` (showing the computed likelihood values). This tabular format enables efficient sorting operations, percentage formatting of probability values, and serves as the foundation for generating comparative visualizations such as bar charts that clearly illustrate the relative likelihoods of each suspect.

F. Bayesian Network Design

Applying the principles outlined in Section III-A, we designed a Discrete Bayesian Network (DBN) to model the "Case of the Missing Manuscript" scenario. The core random variables V include a hypothesis variable *GuiltyParty* (GP) with states $\{A, B, C\}$ representing the potential culprits (Scholar, Butler, Cat Burglar respectively), and six evidence variables: *ForcedEntry* (FE), *AlibiA* (AA), *AlibiB* (AB), *AlibiC* (AC), *Fingerprints* (FP), and *SecurityFootage* (SF), each with discrete states (e.g., FE: {Yes, No}, FP: {A, B, C, None}). The network structure $G = (V, E)$ was constructed with *GuiltyParty* as the sole root node, connecting to all evidence variables via directed edges $E = \{(\text{GP}, X_j) \mid X_j \in \{\text{FE, AA, AB, AC, FP, SF}\}\}$, encoding the conditional independence assumption that all evidence variables are mutually independent given the *GuiltyParty*. This structure, depicted in Fig. 1, was visualized using Graphviz [8] for application interface integration.

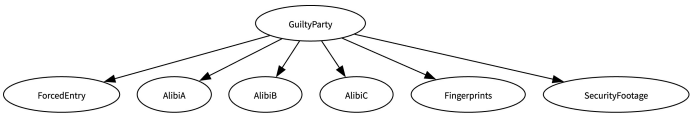

Fig. 1. Bayesian Network Model for ClueChain. 'GuiltyParty' is the root node, influencing all evidence variables.

The parameter set Θ was specified through careful probability assignments based on the narrative logic. We initialized the root node with a uniform prior $P(GP) = \{P(GP = A) = 1/3, P(GP = B) = 1/3, P(GP = C) = 1/3\}$, representing unbiased initial beliefs about the suspects. For each evidence node X_j, we defined conditional probability distributions $P(X_j \mid \textit{GuiltyParty})$ through knowledge engineering, translating scenario descriptions into quantitative values. For instance, reflecting the Cat Burglar's typical method, we set $P(\textit{ForcedEntry} = \text{Yes} \mid \textit{GuiltyParty} = C) = 0.9$. Conversely, considering the Scholar's legitimate access, we set $P(\textit{ForcedEntry} = \text{Yes} \mid \textit{GuiltyParty} = A) = 0.1$. Similar reasoning was applied to define probabilities for alibis (e.g., $P(\textit{AlibiA} = \text{No} \mid \textit{GuiltyParty} = A) = 0.8$, indicating a higher chance the guilty party lacks an alibi) and fingerprints (e.g., $P(\textit{Fingerprints} = A \mid \textit{GuiltyParty} = A) = 0.7$). These distributions were systematically specified for all relevant variable-state combinations.

Implementation utilized the `pgmpy` library's [4] `TabularCPD` class to numerically represent these probability distributions. Each evidence variable's CPD was stored as a multidimensional array encoding its dependence on the different states of *GuiltyParty*. The complete collection of these CPDs, along with the prior distribution for *GuiltyParty*, constitutes the parameter set Θ for the ClueChain BN $\mathcal{B} = (G, \Theta)$. This design enables efficient probabilistic inference while maintaining interpretability through the clear correspondence between network parameters and the underlying mystery scenario narrative.

G. Joint Distribution Representation & Marginalization Calculation

The Bayesian Network structure and associated Conditional Probability Distributions (CPDs), as designed in Section III-F, implicitly define the full Joint Probability Distribution (JPD) over all variables V in the ClueChain model, according to the factorization specified in Eq. 1. Crucially, the methodology avoids explicitly constructing and storing the complete JPD table, which would be computationally infeasible due to the exponential growth in entries with the number of variables (in this case, $3 \times 2 \times 2 \times 2 \times 2 \times 4 \times 2 = 384$ states, manageable but demonstrates the principle). Instead, the JPD is represented implicitly through the collection of individual `TabularCPD` objects associated with each node within the `pgmpy` model structure [4]. This factorized representation enables efficient computation while maintaining probabilistic consistency.

Marginalization, the process of summing out variables to obtain distributions over smaller subsets (Eq. 2), is fundamental to probabilistic inference but is handled computationally rather than via explicit JPD manipulation. While prior marginal probabilities (e.g., $P(\text{FE})$) could conceptually be calculated by summing Eq. 1 over all other variables, the primary application occurs during probabilistic inference when computing posterior probabilities $P(\text{GP} \mid E = \mathbf{e})$ using the Variable Elimination (VE) algorithm (Section III-H). VE systematically eliminates nuisance variables $Z = V \setminus (\{\text{GP}\} \cup E)$ by performing summations over intermediate factors derived from the CPDs [6], [7]. This operational marginalization, embedded within the VE algorithm, ensures that the resulting posterior distribution for *GuiltyParty* correctly accounts for the uncertainty associated with any unobserved evidence variables (Eq. **??**). The `pgmpy` implementation optimizes these summation and multiplication operations by leveraging the network structure to minimize redundant calculations.

H. Probability Distribution Calculation Using Probabilistic Inference

The Bayesian Network designed for ClueChain (Section III-F) implicitly defines the full Joint Probability Distribution (JPD) over all variables $V = \{\text{GP, FE, AA, AB, AC, FP, SF}\}$ through the chain rule factorization in Eq. 1, based on both the network structure G and specified parameters Θ. This representation combines the prior distribution $P(\text{GP})$ for the root node with the Conditional Probability Distributions (CPDs) $P(X_j \mid \text{GP})$ for each evidence variable X_j, stored as `TabularCPD` objects in `pgmpy` [4]. The core computational task involves probabilistic inference: determining the posterior probability

distribution $P(\text{GP} \mid E = \mathbf{e})$, which quantifies the updated beliefs about each suspect's guilt given the specific evidence $\mathbf{e}$ provided by the user through the interface (Section III-I).

For performing exact inference in ClueChain, we implemented the Variable Elimination (VE) algorithm [6], [7]. This choice was motivated by the network's structure; being a polytree (a DAG where the underlying undirected graph has no cycles), VE provides an efficient and exact inference method [7]. We utilized `pgmpy`'s [4] implementation available through the `VariableElimination` class. The inference process, invoked when the user requests a solution, proceeds through conceptually distinct stages:

1) **Evidence Instantiation:** The algorithm incorporates the user's observations $\mathbf{e}$ by restricting the factors (derived from the CPDs) to the rows/columns corresponding to the observed states. For variables marked 'Unknown' by the user, no evidence is set, and they are treated as unobserved.
2) **Factor Multiplication and Summation (Elimination):** VE operates on the set of factors (initial CPDs and prior). It chooses an elimination ordering for the non-query, non-evidence variables $Z = V \setminus (\{\text{GP}\} \cup E)$. For each variable Z_k to be eliminated, VE multiplies all factors involving Z_k and then sums out Z_k from the resulting product factor. This step corresponds to the marginalization process.
3) **Result Computation:** After eliminating all variables in Z, the remaining factors involve only the query variable (GP) and possibly evidence variables (if any were parents of GP, which is not the case here). These factors are multiplied together to yield the joint probability $P(\text{GP}, E = \mathbf{e})$.
4) **Normalization:** The resulting factor representing $P(\text{GP}, E = \mathbf{e})$ is normalized by dividing by $P(E = \mathbf{e}) = \sum_{\text{gp}} P(\text{GP} = \text{gp}, E = \mathbf{e})$, to obtain the final posterior distribution $P(\text{GP} \mid E = \mathbf{e})$.

The algorithm outputs the posterior distribution $P(\text{GP} \mid E = \mathbf{e})$ as a discrete probability vector over the states $\{A, B, C\}$, representing the system's updated belief about the likelihood of each suspect being the culprit, given the evidence. This distribution is then passed to the user interface for presentation (Section III-I). The VE approach guarantees exact probability computations for this network topology while maintaining computational efficiency suitable for interactive use, with its complexity being exponential only in the treewidth of the graph, which is small for polytrees [7].

The choice of the Variable Elimination (VE) algorithm for inference was motivated by the structure and scale of the defined Bayesian Network. Given that the ClueChain network is relatively small (7 variables) and possesses a simple polytree structure (specifically, a naive Bayes structure where evidence nodes only depend on the class node), VE provides an efficient method for computing exact posterior probabilities. Its computational complexity is manageable for this topology ($O(n \cdot d^2)$ where n is number of variables and d is max domain size for naive Bayes), avoiding the need for approximate inference techniques (like sampling methods) which might be necessary for larger, more densely connected networks but would introduce approximation errors. Using VE ensures that the probabilities presented to the user are the precise results dictated by the model's structure, CPDs, and the provided evidence, aligning with the goal of clearly demonstrating Bayesian inference principles without the confounder of approximation error.

I. User Interaction and Interface

To facilitate user engagement with the Bayesian Network model and inference process, we developed an interactive web-based interface using the Streamlit library [5]. The interface was designed with two primary objectives: (1) providing intuitive evidence input mechanisms and (2) clearly visualizing the prior and posterior probability distributions over the suspects. The application layout typically consists of a sidebar panel used for user inputs and displaying introductory text or scenario descriptions, while the main panel is dedicated to displaying the Bayesian Network structure (if available) and the inference results. Evidence collection is implemented using a series of Streamlit widgets, specifically `st.selectbox`, presented in the sidebar. One selectbox is generated for each evidence variable (e.g., *ForcedEntry*, *AlibiA*, *Fingerprints*, *SecurityFootage*, etc.). Each selectbox allows the user to choose the observed state for that variable from a predefined list corresponding to the variable's domain (e.g., ['Yes', 'No', 'Unknown'] for *ForcedEntry*; ['A', 'B', 'C', 'None', 'Unknown'] for *Fingerprints*). A critical design feature is the consistent inclusion of an 'Unknown' option in each selectbox. When 'Unknown' is selected for a variable, it signifies that no observation is available for that piece of evidence. This selection is explicitly handled during the evidence preparation stage before inference.

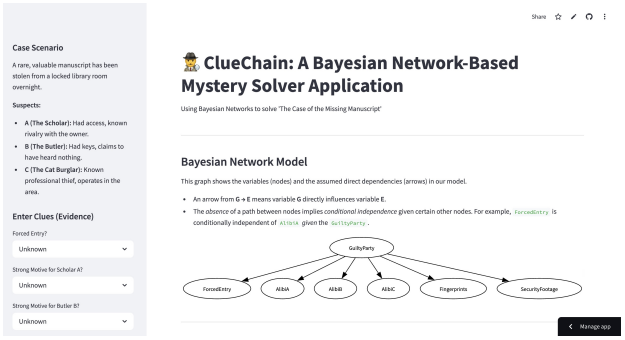

Fig. 2. ClueChain App UI Interface: Sidebar for Inputting Evidence ('Unknown' option allows omitting evidence).

Upon user interaction, typically by clicking a button like "Solve Mystery" (using `st.button`), the application triggers the inference process. This involves several key operations executed within the Streamlit script: First, it gathers the current state selected in each evidence selectbox. It constructs a Python dictionary to represent the evidence set $E = \mathbf{e}$ required

by the `pgmpy` inference engine (described in Section III-H). This dictionary maps variable names (strings) to their observed states (strings), crucially filtering out any variables for which the user selected 'Unknown'. Only variables with a specific observed state are included in the evidence dictionary passed to the inference function. The system then invokes the backend inference function, which executes the Variable Elimination algorithm (Section III-H) using the constructed evidence dictionary. The resulting posterior probability distribution $P(\text{GP} \mid E = \mathbf{e})$ is returned, typically as a `pgmpy` factor or a pandas DataFrame. Finally, the results are presented in the main panel of the Streamlit app through multiple synchronized visualizations. A formatted table (generated using `pandas` and displayed with `st.dataframe` or `st.table`) lists each suspect ('A', 'B', 'C') and their corresponding posterior probability of guilt, often formatted as percentages for readability. Alongside the table, a bar chart (generated using Streamlit's native charting, e.g., `st.bar_chart`) visually compares the relative likelihoods of the suspects based on the computed posterior probabilities. If the `graphviz` library [8] and its system dependencies are installed, the interface additionally renders the Bayesian Network's DAG structure (Fig. 1) using `st.graphviz_chart`, helping users understand the underlying model's probabilistic dependencies.

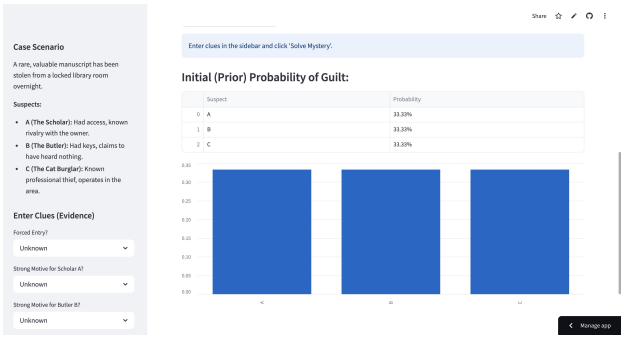

Fig. 3. ClueChain App UI Interface: Main Panel Displaying Posterior Probability of Guilt (Table and Bar Chart).

This interface design achieves several important methodological goals: It maintains a clear correspondence between the theoretical Bayesian Network model (Sections III-F, III-H) and the user's interactions. It provides immediate, intuitive visual feedback on how accumulating or changing evidence impacts the levels of suspicion associated with each suspect, thereby demonstrating the concept of Bayesian belief updating. It allows users to easily explore different hypothetical scenarios by changing evidence inputs and observing the resultant shifts in probability, facilitating an interactive learning or investigative experience grounded in rigorous probabilistic reasoning principles. The implementation leverages Streamlit's [5] reactive programming model, ensuring that the display automatically updates whenever input widgets change or the inference button is pressed, creating a seamless and responsive user experience.

IV. Result

The ClueChain application successfully implements and demonstrates Bayesian inference for a fictional mystery scenario through an interactive web interface powered by Streamlit [5]. Upon launching the application, the system correctly initializes and displays the prior probability distribution for the *GuiltyParty* (GP) variable. As designed, with no initial evidence, each suspect (A: Scholar, B: Butler, C: Cat Burglar) is assigned an equal likelihood of guilt, $P(\text{GP}) = \{A : 0.333, B : 0.333, C : 0.333\}$. This uniform prior is presented numerically in a pandas DataFrame table and visually via a bar chart in the main interface panel, establishing a clear baseline belief state. Concurrently, if Graphviz [8] is configured, the interface renders the Directed Acyclic Graph (DAG) of the Bayesian Network (as shown in Fig. 1), visually confirming the model structure where *GuiltyParty* is the parent node influencing all six evidence variables (FE, AA, AB, AC, FP, SF). This initial state effectively communicates the model's starting assumptions and structure before any evidence is introduced.

The core functionality, driven by user interaction via the sidebar evidence selection widgets (Fig. 2), performs as expected. When users select specific states for one or more evidence variables (e.g., setting *ForcedEntry*='Yes', *AlibiC*='No', *Fingerprints*='C') and trigger the inference by clicking the "Solve Mystery" button, the application correctly interfaces with the `pgmpy` [4] backend. The Variable Elimination algorithm (Section III-H) executes efficiently to compute the exact posterior probability distribution $P(\text{GP} \mid E = \mathbf{e})$ based on the provided evidence $\mathbf{e}$. The system correctly handles the 'Unknown' state for evidence variables by excluding them from the evidence dictionary passed to the inference engine, ensuring appropriate marginalization over unobserved variables as per Bayesian principles.

The application dynamically updates the main panel (Fig. 3) to reflect the computed posterior probabilities. Both the numerical table and the bar chart are refreshed instantly, providing immediate feedback on how the entered evidence shifts the belief about each suspect's guilt. Testing across various evidence combinations confirmed the system's ability to produce logically consistent results based on the predefined CPDs:

A. Scenario Examples

To illustrate the system's behavior under different evidence patterns, consider two distinct scenarios based on the fictional "Case of the Missing Manuscript":

Scenario 1: Strong Evidence Against the Cat Burglar (Suspect C) Suppose the user inputs the following evidence:

- Forced Entry (FE): Yes
- Alibi A (AA): Yes
- Alibi B (AB): Yes
- Alibi C (AC): No
- Fingerprints (FP): C
- Security Footage (SF): Obscured (treated as 'Unknown' or a state indicating no useful info)

Upon clicking "Solve Mystery", ClueChain processes this evidence. The presence of forced entry strongly supports Suspect C (P(FE=Yes—GP=C)=0.9). The lack of an alibi for C (P(AC=No—GP=C) is high, e.g., 0.8) and the presence of C's fingerprints (P(FP=C—GP=C) is high, e.g., 0.7) further incriminate C. Conversely, confirmed alibis for A and B significantly reduce their likelihood (P(AA=Yes—GP=A) and P(AB=Yes—GP=B) are low, e.g., 0.2). The inference algorithm combines these pieces of evidence according to Bayes' theorem. The resulting posterior probability distribution displayed would be heavily skewed towards Suspect C, likely showing $P(\text{GP} = C \mid \mathbf{e}) > 0.95$, while $P(\text{GP} = A \mid \mathbf{e})$ and $P(\text{GP} = B \mid \mathbf{e})$ would be very close to zero. The bar chart would visually emphasize C as the overwhelmingly likely culprit.

Scenario 2: Ambiguous Evidence Now, consider a scenario with less definitive clues:

- Forced Entry (FE): No
- Alibi A (AA): No
- Alibi B (AB): Yes
- Alibi C (AC): Yes
- Fingerprints (FP): None
- Security Footage (SF): Unknown

In this case, the evidence is mixed. No forced entry might slightly favor insiders A or B over C (P(FE=No—GP=A/B) ¿ P(FE=No—GP=C)). Suspect A lacking an alibi increases suspicion towards A (P(AA=No—GP=A) is high), while B and C having alibis reduces suspicion towards them (P(AB=Yes—GP=B) and P(AC=Yes—GP=C) are low). The absence of identifiable fingerprints ('None') provides little discriminatory information, perhaps slightly favoring those less likely to leave prints or those who wore gloves (this depends heavily on the P(FP=None—GP) values set in the CPDs). When ClueChain processes this evidence, the posterior probabilities will likely be more distributed than in Scenario 1. Suspect A's probability would increase from the prior due to the lack of alibi, but the absence of forced entry might temper this increase or slightly favor B. Suspect B might retain some probability due to access (keys) despite the alibi. Suspect C's probability would likely decrease significantly due to the lack of forced entry and confirmed alibi. The final output might show, for example, $P(\text{GP} = A \mid \mathbf{e}) \approx 0.60$, $P(\text{GP} = B \mid \mathbf{e}) \approx 0.30$, $P(\text{GP} = C \mid \mathbf{e}) \approx 0.10$. The visualization would reflect this remaining uncertainty, highlighting A as the most likely, but B as a non-negligible possibility.

These examples demonstrate the system's core capability: integrating multiple, potentially conflicting pieces of evidence according to the defined probabilistic model (structure and CPDs) to produce a reasoned, quantitative assessment of likelihoods, dynamically updating from the prior based on the specific evidence pattern provided.

V. Discussion

ClueChain successfully demonstrates the practical application of Bayesian Networks as an interactive reasoning tool, effectively bridging abstract probabilistic concepts with tangible user experiences. The system's design translates theoretical elements – prior probabilities, conditional probability distributions (CPDs), evidence observation, and posterior probability calculation via inference (Eq. **??**) – into an intuitive interface where users directly manipulate inputs and observe real-time updates to the belief state $P(\text{GP} \mid E = \mathbf{e})$. This implementation, leveraging the capabilities of `pgmpy` [4] for the probabilistic modeling and inference, Streamlit [5] for the rapid web application development, and optionally Graphviz [8] for model visualization, serves as both a functional prototype and a valuable pedagogical instrument. The visualization of the network structure (Fig. 1) is particularly useful for clarifying the model's underlying conditional independence assumptions (e.g., evidence items are independent given the guilty party). The immediate feedback loop between evidence selection and the resulting shifts in posterior probabilities (Fig. 3) effectively illustrates the core Bayesian principles of belief updating based on evidence, making complex probabilistic reasoning more accessible and understandable, even for users without a deep technical background in statistics or AI.

However, the interpretation and applicability of the model's outputs must be considered within the context of its design choices and inherent limitations. The posterior probabilities generated by ClueChain are fundamentally conditioned not only on the observed evidence $\mathbf{e}$ but also critically on the predefined network structure G and the manually specified CPD parameters Θ. These parameters were derived through knowledge engineering based on the logic of the fictional narrative, rather than learned from empirical data. This highlights a key challenge in many practical Bayesian Network applications: the potential subjectivity and difficulty in accurately eliciting or estimating these probabilities, especially in domains where reliable statistical data is scarce [10]. While the model correctly computes the mathematically sound consequences of its assumptions and the provided evidence using exact inference (VE [6]), the resulting probabilities represent beliefs conditional on the model itself, rather than objective statements of fact. Furthermore, the current model adopts a significant simplification by assuming conditional independence between all evidence variables given the guilty party (a naive Bayes structure). Real-world investigative scenarios often involve complex interdependencies between different pieces of evidence that are not captured here. The scope, limited to three suspects and six evidence types within a single fixed scenario, intentionally trades realism for pedagogical clarity. These limitations are acceptable given the project's goal as an educational demonstrator but underscore the gap between this prototype and a system suitable for complex, real-world forensic analysis.

A. Limitations

Despite its utility as an educational demonstrator, ClueChain possesses several inherent limitations stemming primarily from its design focus on pedagogical clarity over real-world complexity. The model employs a significantly simplified naive

Bayes structure, assuming conditional independence between all evidence variables given the guilty party, thereby ignoring potentially crucial interdependencies often present in actual investigations [10]. Furthermore, the conditional probability distributions (CPDs) and prior probabilities were established through subjective knowledge engineering based on the fictional narrative, rather than empirical data, making the model's outputs highly sensitive to these specific parameter choices and lacking objective grounding [10]. The scope is narrow, confined to a single scenario with a small, fixed set of suspects and evidence types, limiting generalizability and preventing validation against real-world case data. Additionally, the reliance on discrete variable states struggles to capture the nuances and uncertainties often associated with real evidence, and the model itself is static, lacking mechanisms for learning or adaptation beyond processing the predefined inputs. These constraints collectively underscore that ClueChain functions primarily as a tool for illustrating Bayesian principles, not as a robust system for practical forensic analysis or complex decision support.

VI. Future Work

Future enhancements for ClueChain could significantly increase its realism and utility by addressing current limitations. Firstly, the model's structure could be advanced beyond the naive Bayes assumption to incorporate dependencies between evidence variables, reflecting more complex real-world scenarios, although this would necessitate more sophisticated causal reasoning and potentially more complex CPD elicitation [3]. Secondly, the reliance on subjective, manually defined parameters could be mitigated by exploring data-driven approaches; if relevant data were available, techniques like Maximum Likelihood or Bayesian parameter estimation could be implemented using libraries such as `pgmpy` [4], [7] to learn CPDs empirically, coupled with sensitivity analysis to assess the model's robustness to parameter uncertainty [10]. Thirdly, to handle potentially larger and more complex networks where exact inference via Variable Elimination [6] becomes computationally infeasible, future versions could implement and allow comparison of approximate inference algorithms like MCMC or variational methods, thereby also educating users on the trade-offs between computational cost and accuracy [7]. Finally, the user interface could be augmented to visualize these richer models, display outputs from approximate inference (e.g., confidence intervals), and potentially allow greater user interaction with the model's structure or parameters.

VII. Conclusion

This paper presented ClueChain, an interactive web application designed to demonstrate the principles and application of Bayesian Networks for reasoning under uncertainty within the context of a fictional mystery-solving scenario. By employing a Discrete Bayesian Network implemented with the `pgmpy` library [4], the system effectively models the probabilistic dependencies between potential suspects (*GuiltyParty*) and various forms of evidence. Users can interactively input observed clues through a graphical user interface built with Streamlit [5]. Upon receiving evidence, the application utilizes the Variable Elimination algorithm [6], [7] to perform exact Bayesian inference, computing and visualizing the updated posterior probability distribution over the suspects. The optional integration of Graphviz [8] further aids user understanding by displaying the network's underlying structure and conditional independence assumptions.

ClueChain successfully illustrates the core strengths of Bayesian methods: the ability to formally integrate multiple, disparate pieces of evidence and to dynamically update beliefs in a quantitative manner as new information becomes available. While the current implementation is subject to limitations inherent in its pedagogical focus—namely, a simplified model structure and subjective, manually defined parameters—it serves as an effective and accessible tool for educating users on the fundamentals of probabilistic reasoning and belief updating. It highlights the potential utility of Bayesian Networks as a framework for building decision support systems in domains characterized by uncertainty and the need for evidence synthesis. Future work could focus on enhancing the model's realism by incorporating more complex dependencies, exploring data-driven parameter learning techniques, investigating approximate inference algorithms for scalability, and enriching the user interface to provide deeper insights into the reasoning process.

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Hoboken, NJ, USA: Pearson Education, Inc., 2020.

[2] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[3] F. V. Jensen and T. D. Nielsen, *Bayesian Networks and Decision Graphs*, 2nd ed. New York, NY, USA: Springer Science+Business Media, LLC, 2007.

[4] A. Ankan and A. Panda, "pgmpy: Probabilistic Graphical Models using Python," in *Proceedings of the 14th Python in Science Conference (SciPy)*, Austin, TX, USA, Jul. 2015, pp. 91–96.

[5] Streamlit Inc., "Streamlit Documentation," 2023. [Online]. Available: https://streamlit.io/

[6] R. Dechter, "Bucket elimination: A unifying framework for probabilistic inference," in *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence (UAI'96)*, E. Horvitz and F. Jensen, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996, pp. 211–219.

[7] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA, USA: The MIT Press, 2009.

[8] E. R. Gansner and S. C. North, "An open graph visualization system and its applications to software engineering," *Software: Practice and Experience*, vol. 30, no. 11, pp. 1203–1233, Sep. 2000.

[9] L. C. van der Gaag, S. Renooij, C. L. M. Witteman, B. M. P. Aleman, and B. G. Taal, "Probabilities for a probabilistic network: a case study in oesophageal cancer," *Artificial Intelligence in Medicine*, vol. 25, no. 2, pp. 123–148, Jun. 2002.

[10] K. B. Korb and A. E. Nicholson, *Bayesian Artificial Intelligence*, 2nd ed. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group, 2010.

[11] Stanford University, "Bayesian Networks 1 - Inference — Stanford CS221: AI (Autumn 2019)," YouTube, Oct. 2019. [Online]. Available: https://youtu.be/U23yuPEACG0?si=ifV8mUMcd-86-rmL. [Accessed: 24-Feb-2025].

[12] "Evidence in Context: Bayes' Theorem and Investigations," YouTube, [Online]. Available: https://youtu.be/EC6bf8JCpDQ?si=vzAoYaC-O6bckkdI. [Accessed: Feb. 24, 2025].