Towards an Efficient Simulation-Based Anytime Inference in Subjective Bayesian Networks

Han Jun Yoon*, Shou Matsumoto†, Paulo Costa†*Senior Member, IEEE*, Jin-Hee Cho**Senior Member, IEEE*,
*Department of Computer Science, Virginia Tech, Falls Church, VA 22043 USA
†C5I Center, George Mason University, Fairfax, VA 22030 USA

***Abstract*—Subjective Bayesian networks (SBN) integrate Bayesian Networks (BN) with Subjective Logic, enabling the representation of second-order uncertainty, denoting the uncertainty surrounding the probability distribution of an event. Although prior research predominantly centers on exact inference within the SBN framework, there is a notable dearth of exploration into the realm of approximate inference in SBN. Our work is specifically geared towards addressing this gap, focusing on the application of diverse sampling methodologies (i.e., forward and Gibbs sampling) for approximate inference in SBN. The primary contribution of this work lies not only in the introduction of approximate inference in SBN but also in the formulation of an "anytime" SBN inference algorithm. This implies that a best inference estimate can be obtained at any given moment, given trade-offs in the precision. Moreover, the allocation of computational resources is a customizable and potentially optimizable process. Through a rigorous series of experiments, we empirically demonstrate that the number of iterations to convergence decreases as we provide more samples for both forward and Gibbs sampling. Furthermore, we discover the difference between approximate and exact inference in belief (δ_{belief}) and uncertainty ($\delta_{uncertainty}$) mass of subjective opinion becomes more unpredictable as the error gets large in BN probability. Lastly, in our experiments, we demonstrate the number of BN samples has a greater impact on δ_{belief} than the number of SBN iterations. These findings indicate that the family of greedy algorithms (based on local graded changes – such as gradients) can be a promising approach for finding optimal allocations of computational resources in this framework. The software assets produced and used in this work will be made available as an open source Python library.**

***Index Terms*—Subjective Bayesian Network, Bayesian Network, Sampling**

I. Introduction

A Bayesian Network (BN) serves as a graphical representation of a factored joint probability distribution [1]. BNs have demonstrated efficacy across diverse domains for modeling uncertain knowledge [2, 3], with extensions proposed to enhance their expressive capabilities [4, 5]. This highlights the potential of BNs as robust tools for high-level information fusion under uncertainty.

Traditional BN approaches, reliant on abundant data, excel in effectively characterizing probability distributions. However, in contexts where data is scarce and expert insights are constrained, standard techniques may struggle to adequately capture the full spectrum of uncertainties, especially concerning probability values. In such scenarios, higher-order uncertainty offers a solution by introducing additional layers of uncertainty beyond mere point estimations found in classical probability representations. This capability becomes crucial in scenarios characterized by sparse data and limited expert knowledge.

These additional layers allow for a more nuanced and comprehensive understanding of uncertainty, enabling decision-makers to account for probabilities not only in the outcomes themselves but also in the uncertainties associated with those probabilities. This added depth in uncertainty modeling is essential for making informed decisions and managing risks effectively in domains where uncertainty is prevalent and significant, such as military operations and cybersecurity.

For instance, the concept of second-order uncertainty involves probability distributions over probabilities, representing uncertainty about the likelihood of events rather than uncertainty about the events themselves [6, 7]. SBNs extend traditional BNs by representing second-order uncertainty as opinions.

Inference within Subjective Bayesian Networks (SBNs) entails the utilization of opinions established in Subjective Logic to depict second-order uncertainty [8]. These opinions encompass belief mass distributions across potential states, uncertainty masses denoting evidential voids, and base rate distributions for scenarios of complete ignorance. By propagating these opinions, SBNs enable reasoning under uncertainty, thereby empowering decision-makers to formulate informed inferences despite constrained data availability and expert insights.

One weakness of exact inference in SBNs is its computational complexity. For instance, exact inference approaches based on variable elimination or belief propagation [9, 10] may become computationally infeasible for complex networks due to the exponential growth in computational requirements as its treewidth increases. This can result in long computation times and memory limitations, hindering the scalability of exact inference approaches.

To address this, approximate inference in SBNs shall offer a practical solution to address the computational challenges associated with exact inference. These methods, such as sampling-based approaches such as Monte Carlo Simulation [11] provide efficient approximations to the posterior distribution without requiring accurate computations over a strict mathematical model. Approximate inference methods are often faster and more scalable than exact inference methods.

In this work, we propose an *anytime* approximate SBN inference algorithm that allows for termination at any point,

yielding the best possible inference obtained so far. The term "anytime" in the context of this paper refers to the flexibility of the approximate SBN inference algorithm to terminate at any point during its execution, allowing for the retrieval of the best solution achieved thus far.

In summary, we have a clear objective of developing a computationally efficient "anytime" SBN inference algorithm, and this paper is a basis for/towards this objective.

II. Related Works

Both BNs and SBNs have been utilized as inference tools across various domains, spanning from cybersecurity to mission operations [12, 13]. Within the framework of BNs, researchers have extensively explored different methods for computing probabilities, encompassing both exact and approximate inference techniques. [14] employed exact inference, specifically the belief propagation algorithm, within a BN to infer the mission outcome of an object detection mission system given the available evidence. Approximate inferences of BNs are indeed available in the literature [15, 16].

In the context of SBNs, [13] implemented a partially approximate inference within an SBN to conduct a mission impact assessment of Unmanned Aerial Vehicles transporting critical items between hospitals. [17] used Monte Carlo Simulation to approximate subjective logic operators used in message passing in SBNs. However, to the best of our knowledge, there exists a dearth of research about the complete solution of anytime approximate inference in SBNs.

In terms of computing budget allocation for efficient anytime inference, [18] introduced an Optimal Computing Budget Allocation (OCBA) algorithm designed to identify optimal budget allocation within the constraints of a highly limited computing budget. [19] proposed a method for distributing the number of simulation replications for constrained optimization. Our proposed general pseudocode for allocating the computing budget is provided in Listing 2.

III. Anytime Approximate simulation-based SBN inference Algorithm

A key distinction between an *anytime* approximate SBN inference algorithm and a general approximate SBN inference algorithm in Figure 1 lies in their termination conditions. Anytime SBN algorithm allows for termination at any point, yielding the best possible solution achieved thus far. Conversely, an arbitrary approximate SBN inference algorithm may not necessarily have this stopping condition. In this context, "best" refers to the culmination of implementation efforts, rather than an assurance of achieving the theoretical best solution. Consequently, there exists no guarantee that the obtained inference represents the utmost achievable inference within the same time frame or total number of iterations. In our future work, we plan to utilize our idea of adjusting the computing budget in Section III-E to develop a more efficient anytime approximate SBN inference algorithm, yielding inference close to the theoretical best solution.

Note that the training phase precedes the inference process as an optional step. Using training data samples D, we populate the tables of absolute counts for each node in the trained SBN. These tables of absolute counts are commonly referred to as the count table or absolute frequency table. This is analogous to the concept of Conditional Probability Tables (CPTs) of a node in the trained BN.

Just like a BN is defined as a tuple $\{\text{V, A, } \Phi\}$ where V is the set of random variables (nodes), A is the set of arcs, and Φ is the set of CPTs of each node given its parents, a SBN can be defined as a tuple $\{\text{V, A, } \Psi\}$, where V denotes the set of random variables (nodes), A represents the set of arcs, and Ψ is the set of conditional opinions of each node given its parents.

It is crucial to note that the inference stage can operate independently of how the tuple $\{\text{V, A, } \Psi\}$ was elicited. Once a tuple has been fully specified, it does not matter whether it was trained using data or manually constructed by a Subject Matter Expert (SME). The training phase may be viewed as optional since equivalent outcomes (i.e., a functional SBN model suitable for applying inference algorithms) can be achieved by SMEs without resorting to training data. That's one of the purposes of having opinions in SL because these would come from human knowledge. However, given our utilization of training data to train the model, we provide detailed procedures undertaken in the next paragraph.

In our context, we utilize the count tables (i.e., table of absolute frequency) as a representation of Ψ, aligning with the training data D. There are many other alternative ways to represent the second-order conditional probability Ψ. One representation of Ψ could be a table of belief, disbelief, and uncertainty given combinations of states of parents; and another could be a simple script (e.g., a collection of if-then-else clauses/statements which maps the conditions/states of the parents to opinions or any other parameter of higher-order probabilities). In our implementation, we use the count tables because it's simple and easily traceable to the training data D.

Consequently, these count tables form an integral part of the trained SBN model, and if we exclude these count tables, we technically don't have an SBN anymore. Once we have the full tuple $\{\text{V, A, } \Psi\}$, the training data is out of the scope/concern.

In the context of parameter learning, before and during the training phase, we have the training data D, the nodes V, and the arcs A, but we do not have Ψ yet. After the training phase, we have the full tuple $\{\text{V, A, } \Psi\}$ and the training data should be discarded from the scope afterward. Entities within this scope include the SBN $\{\text{V, A, } \Psi\}$ and the training data D.

After the optional training phase, we delineate the scopes of our proposed anytime SBN inference algorithm into distinct phases including 1) Bijective mapping between subjective opinions and Beta/Dirichlet parameters; 2) The outer SBN inference phase; 3) The inner BN inference phase and 4) The aggregation phase in the outer SBN inference. We discuss the algorithm as a series of interconnected blocks executed sequentially, each with clearly defined boundaries or interfaces. This approach facilitates a more structured separation

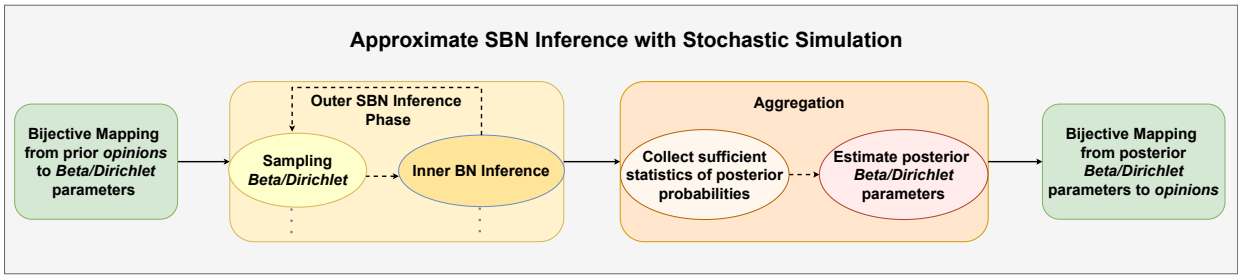


Fig. 1. Operational view of the Approximate SBN Stochastic Simulation inference algorithm.

of concerns, allowing for a focused examination of individual sub-phases rather than treating them as a single entity. An abstract pseudocode of the proposed algorithm can be found in Listing 1. We describe each phase below.

A. Bijective Mapping Between Subjective Opinions and Beta/Dirichlet Parameters

An opinion of a categorical (discrete) random variable in the SBN is defined as a tuple $\omega = \{\overline{b}, u, \overline{A}\}$ [8]. In this representation, the $\overline{b}$, u, and $\overline{A}$, represents respectively the belief masses (for each possible state of the random variable), the uncertainty mass, and the priors over the belief masses. The belief masses plus the uncertainty must add up to 1.

The Beta distribution is the Bayesian conjugate of a binomial distribution. Similarly, The Dirichlet distribution is the Bayesian conjugate of a multinomial distribution. As such, it is commonly used in Bayesian machine learning as the probability distribution that is derived from data [8, 20] (details about its density function are not in the scope of this paper, though). For instance, the Dirichlet distribution associated with some discrete random variable can be specified in terms of its shape/concentration parameters $\overline{\alpha}$, which denote the amount of evidence for each possible state (*e.g.*, absolute frequency).

The literature can also define the Dirichlet distribution in terms of its shape parameters α, β. The following equation relates these shape parameters with the count parameters r and s:

$$\alpha = r + aW, \quad \beta = s + (1-a)W \tag{1}$$

In this case, the mean μ_i and variance σ_i^2 of the probability density function associated with the i-th state/category of a Dirichlet distribution with a total of K categories (*i.e.*, associated with a multinomial variable with K possible states) can be defined as:

$$\mu_i = \frac{\alpha_i}{\sum_{j=1}^{K} \alpha_j}; \quad \sigma_i^2 = \frac{\mu_i(1-\mu_i)}{1+\sum_{j=1}^{K} \alpha_j} \tag{2}$$

An opinion ω and the Beta distribution $Beta(r, s, a, W)$ are tied by the following bijective mapping rule [8]:

$$\omega = \begin{cases} b = \frac{r}{W+r+s} \\ d = \frac{s}{W+r+s} \\ u = \frac{W}{W+r+s} \end{cases} \tag{3}$$

Given a random variable with n states and r_i the absolute frequency for the i-th state (with $1 \leq i \leq n$), the non-binary multinomial opinion ω can be estimated with the Equation 5. Equation 3 can be extended to the multinomial case as well. An opinion $\omega = (\overline{b}, u, \overline{A})$ and the Dirichlet distribution $Dirichlet(\overline{\alpha})$ are tied by the following mapping rule [8]:

$$\omega = \begin{cases} b_i = \frac{r_i}{W+\sum_{j=1}^{n} r_j} \\ u = \frac{W}{W+\sum_{j=1}^{n} r_j} \end{cases} \tag{4}$$

$$\omega = \begin{cases} b_i = \frac{\alpha_i - A_i W}{W+\sum_{j=1}^{K} \alpha_j - A_j W} \\ u = \frac{W}{W+\sum_{j=1}^{K} \alpha_j - A_j W} \end{cases} \tag{5}$$

Where the index i is associated with the i-th possible state of the discrete/categorical variable; $1 \leq i \leq K$; and K is the total number of possible states/categories. Traditionally (in the Bayesian machine learning literature), we assume W to be equal to the number of possible states for the random variable (*e.g.*, $W = 2$ for binary variables) and $\overline{A}$ being uniformly distributed by default (hence, $A_i W = 1$).

B. Outer SBN Inference Phase

The transition to the outer SBN inference phase is now possible. At this point, the process entails deriving $\{V_1, A_1, \Phi_1\}$, $\{V_2, A_2, \Phi_2\}$, $\{V_3, A_3, \Phi_3\}$, ..., $\{V_n, A_n, \Phi_n\}$ from {V, A, Ψ}. In other words, this signifies the sampling of "n" BNs from SBN.

Practically speaking, both the network structure of the SBN and the BNs remain identical. Hence, V, V_1, V_2, V_3,..., V_n are all equivalent, as are A, A_1, A_2, A_3,..., A_n. The primary focus thus lies on acquiring Φ_1, Φ_2, Φ_3,..., Φ_n based on Ψ.

Therefore, at this point, we read the set of count tables (i.e., Ψ) in the SBN as parameters or the Beta/Dirichlet distribution, and then sample these Beta/Dirichlet distributions to obtain the set of CPTs (i.e., Φ_i). This is the scope of one iteration in the outer sampling, where Φ_i denotes prior conditional probabilities. Note that sampling of all CPTs (i.e., one CPT for each node) must be completed before starting the inner inference phase.

Entities encompassed within this outer inference phase include the SBN {V, A, Ψ}, the BNs $\{V_1, A_1, \Phi_1\}$, $\{V_2, A_2, \Phi_2\}$, $\{V_3, A_3, \Phi_3\}$, ..., $\{V_n, A_n, \Phi_n\}$, and the Beta/Dirichlet sampling procedure which facilitates the translation of Ψ into Φ_i.

C. Inner BN Inference Phase

At this inner BN inference phase, having already sampled a Beta/Dirichlet distribution, the tuple $\{V_i, A_i, \Phi_i\}$ has been obtained, which fully characterizes a single BN. With Φ_i now fixed, it's essential to disregard Beta/Dirichlet within the inner inference phase, given that we already have Φ_i.

Assuming i = 1, the focus narrows solely to $\{V_1, A_1, \Phi_1\}$, thereby excluding Ψ (i.e., the count tables in the SBN) and other Φ_2, Φ_3, ..., Φ_n from consideration. Consequently, we run m iterations of forward [21] or Gibbs sampling [22] for the same BN, thus utilizing only Φ_1 while disregarding the remaining Φ_i.

Subsequently, completion of the m iterations of the forward or Gibbs sampling derives P(V1 | E), the posterior marginal probabilities of V1 given evidence E. E can be empty. If E is empty, then we are deriving the prior marginal probabilities.

For instance, if V_1 = {X, Y, Z} (i.e., (i.e., if the BN contains the nodes X, Y, and Z) and evidence E = {Z = false}, then P(V1 | E) is the set of probabilities {P(X=true | Z=false), P(Y=true | Z=false), P(Z=true | Z=false), P(X=false | Z=false), P(Y=false | Z=false), P(Z=false | Z=false)}. Upon obtaining P(V1 | E), the m forward or Gibbs samples can be discarded.

It's crucial to discern that P(V1 | E) pertains to posterior marginal probabilities, distinct from the prior conditional probabilities derived from Beta/Dirichlet (i.e., the Φ_i sampled from Beta/Dirichlet).

Entities in this inner BN inference phase encompass the BN $\{V_1, A_1, \Phi_1\}$, evidence E, the m BN samples (i.e., forward or Gibbs), and the posterior marginal probabilities P(V1 | E) derived from the m samples.

D. Aggregation Phase in the Outer SBN Inference

Upon completing the inner inference for $\{V_1, A_1, \Phi_i\}$, we've obtained the set of posterior marginal probabilities, P(V1 | E). In other words, near the end of the first iteration in the outer inference method, 1 set of posterior marginal probabilities is obtained. For instance, if $V_1 = V$ = {X, Y, Z}, we obtained 1 set with 6 entries {P(X=true | E), P(Y=true | E), P(Z=true | E), P(X=false | E), P(Y=false | E), P(Z=false | E)}.

In the subsequent stages of the "outer" inference, this process iterates n times, yielding n tuples P(V_1 | E), P(V_2 | E), P(V_3 | E), ..., P(V_n | E). It's worth noting that the outer inference adopts a Monte Carlo method that includes an aggregation phase. Within this aggregation phase, means are computed as (P(V_1 | E) + P(V_2 | E) + ... + P(V_n | E)) / n, with similar procedures for variance calculations. Subsequently, once means and variances are determined, opinions can be derived using bijective mapping [8].

Upon acquiring the means and variances through bijective mapping, Equation 6 offers an estimation for the *posterior* marginal Beta parameters using the sample mean and variance gathered as follows:

$$\alpha_i^{posterior} = \left(\frac{\mu_i(1-\mu_i)}{{\sigma_i}^2} - 1 \right) * \mu_i \tag{6}$$

In this context, μ_i represents the collected sample mean, while σ_i^2 denotes the sample variance.

Crucially, it's important to emphasize that the outer inference remains unaware of the m forward or Gibbs samplings conducted. These samples are already discarded upon the calculation of P(V_1 | E).

Entities relevant within this aggregation phase in the "outer" inference encompass the SBN {V, A, Ψ}, the posterior marginals P(V_1 | E), P(V_2 | E), P(V_3 | E), ..., P(V_n | E), as well as means, variances, and the derived opinions, as applicable.

Synopsis of the Proposed Anytime SBN Algorithm.

1) The training (parameter learning) generates conditional opinions Ψ, which happens to be absolute frequencies based on the training data samples *D*, but that's only because we implemented Ψ that way.
2) The beginning of the outer inference generates prior conditional probabilities (i.e., the CPTs Φ_i) from Ψ, for each node, by sampling Beta/Dirichlet distributions.
3) Now that you have a fixed Φ_i, the inner inference generates posterior marginal probabilities 1 time. We may run m forward or Gibbs samples, but the result is 1 set of P(V_i | E). Each m run of forward or Gibbs samples should use the same Φ_i because we are running forward/Gibbs sampling on a particular BN.
4) Since the outer inference of item 2 repeats the inner inference of item 3 for n times, we end up with n posterior marginal probabilities P(V_1 | E), P(V_2 | E), P(V_3 | E), ..., P(V_n | E).
5) At the aggregation phase (end) of the "outer" inference, means and variances shall be calculated from the P(V_1 | E), P(V_2 | E), P(V_3 | E), ..., P(V_n | E) of item 4. Note that the types of probabilities handled by the outer and inner inferences are different. The outer inference is responsible for Φ_i (CPTs sampled with Beta/Dirichlet), means, and variances. The inner inference is responsible only for P(V_i | E) given fixed CPTs.

E. Computing Budget Allocation

By balancing the number of iterations to apply in the outer versus inner inference, we can have efficient convergence rates under limited computational resources. Since this allocation of the number of iterations implies the allocation of computational resources or budget, by adjusting the allocation of the number of iterations between inner and outer iterations in a way that the convergence rate is highest, we can have better or best allocation of computational resources. An example of a generalized abstract pseudocode for adjusting the computing budget can be found in Listing 2.

For instance, a concrete example of the pseudocode of Listing 2 which adjusts the number of samples in the inner inference accordingly to the previous outer iteration (i.e., the outer iteration virtually samples a network, but the number of inner iterations to run for this network is adjusted based on feedback from the previous outer iteration) would be:

Listing 1. Approximate SBN stochastic simulation algorithm (abstract pseudocode).

```
[Translating SBN to Dirichlet distributions]
For each node N in the network:
    Retrieve parents Pa(N) of N;
    Obtain conditional opinion table φ(N|Pa(N));
    Translate the opinions in φ(N|Pa(N)) to
        Dirichlet parameters α(N|Pa(N)), based on
         the bijective relation;
[Belief propagation]
While running Outer Inference:
    For each node N in the network:
        Retrieve the parents Pa(N) of N;
        Sample the Dirichlet distribution
            α(N|Pa(N)) to obtain the conditional
            probability P(N|Pa(N))
        This should result in a classic BN;
    Insert evidence E to observed nodes: simply
        provide no evidence when calculating the
         prior distribution;
    Run Inner Inference (e.g., forward or Gibbs
        sampling) with the above sampled BN;
    For each node N in the network:
        Store the marginal probability of N, to
            calculate the mean and variance
            later;
[Retrieving the marginal opinions]
For each node N in the network:
    For each state S in N:
        Calculate the mean/average marginal
            probability μ of S;
        Calculate the variance σ^2 of the marginal
             probability of S;
    Solve a system of equations of μ and σ^2 to
        estimate the posterior marginal
        Dirichlet parameters;
    Use the bijective relationship to translate
        the Dirichlet parameters to opinion ω;
    For each state S in N:
        Extract the opinion belief b from ω,
            respective to the state S;
        Overwrite the marginal probability of S
            with the extracted opinion belief b;
    Append the 'uncertainty' state in N;
    Set the marginal of the 'uncertainty' state
        with the uncertainty value in ω;
Return or print the marginal probabilities of all
     nodes in the network;
```

Listing 2. A generalized abstract pseudocode for allocating the computing budget.

```
Based on historical data or heuristics,
    initialize M as the number of iterations/
    samples to perform in inner inference;
Begin outer inference;
For N number of outer iterations/samples to run
    in this experiment block, do:
    Run M number of inner (BN) samples;
    Collect inner convergence metrics;
Adjust M based on the convergence metrics;
Repeat the process until the outer inference's
    stop condition is reached;
```

1) Set M = 100 (an arbitrary number given by the user).
2) Begin outer inference.
3) Run M number of inner (BN) iterations/samples.
4) Check if BN converged before M (Note: the convergence metric here is therefore just a Boolean true/false value).
5) Decrement M (for next iteration) if converged before M. Otherwise increment M.
6) Repeat the process until the maximum number of iterations in outer inference is reached

Another potential instance of Listing 2 would be to run the optimal budget allocation algorithms of [18, 19]. In this case, the N would be the number of parallel models being compared, and the adjustment steps could include a gradient-based optimization, such as [23–25].

IV. Experimental setup

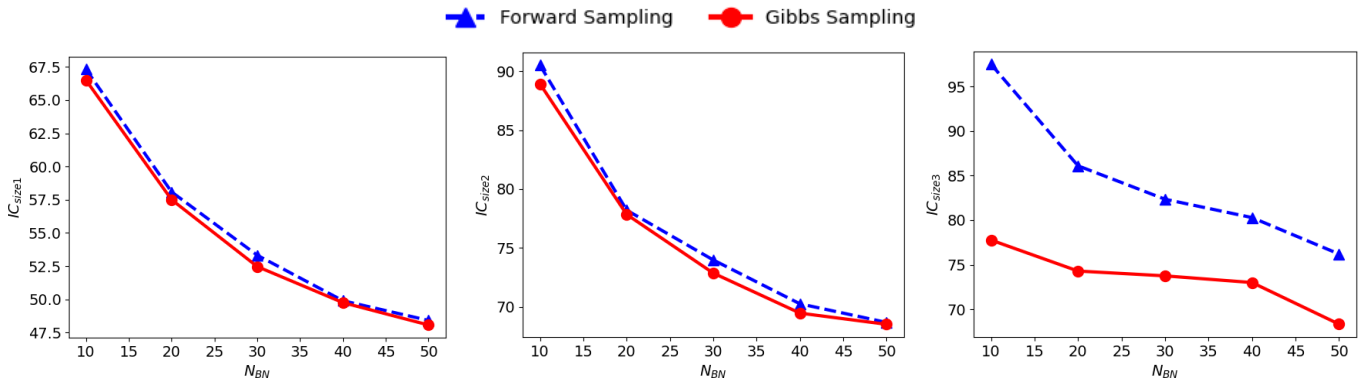
In contrast to the algorithm proposed in Section III, our experimental implementation restricts itself to binary variables, each of which takes state values of either 1 (i.e., True) or 0 (i.e., False). We opt for count tables, reflecting absolute frequency, to represent Ψ, primarily due to their simplicity and direct traceability to the training data set D. We employ the *PGMPY* library [26], renowned for its capabilities in learning, inference, and simulations within Bayesian Networks. Leveraging this Python library enables us to facilitate various sampling methods and facilitate both learning (i.e., parameter) and inference processes. We summarize the design parameters and their meaning in Table I.

Network Configuration. We explore three distinct Bayesian network configurations characterized by their varying sizes and structural complexities. Network size 1, the most diminutive in our analysis, utilizes the preexisting sprinkler dataset. Conversely, for network sizes 2 and 3, we construct Bayesian networks of varying sizes by generating random networks with a predetermined number of variables and edge probabilities between them.

- Network Size 1: We use the existing sprinkler dataset. This dataset is characterized by its simplicity, comprising four variables (i.e., Sprinkler, Cloudy, Rain, Wet Grass) each of which can take on values of either 1 or 0.
- Network Size 2: We incorporate 10 variables interconnected with a 50% edge probability, yielding a network structure comprising 21 edges.

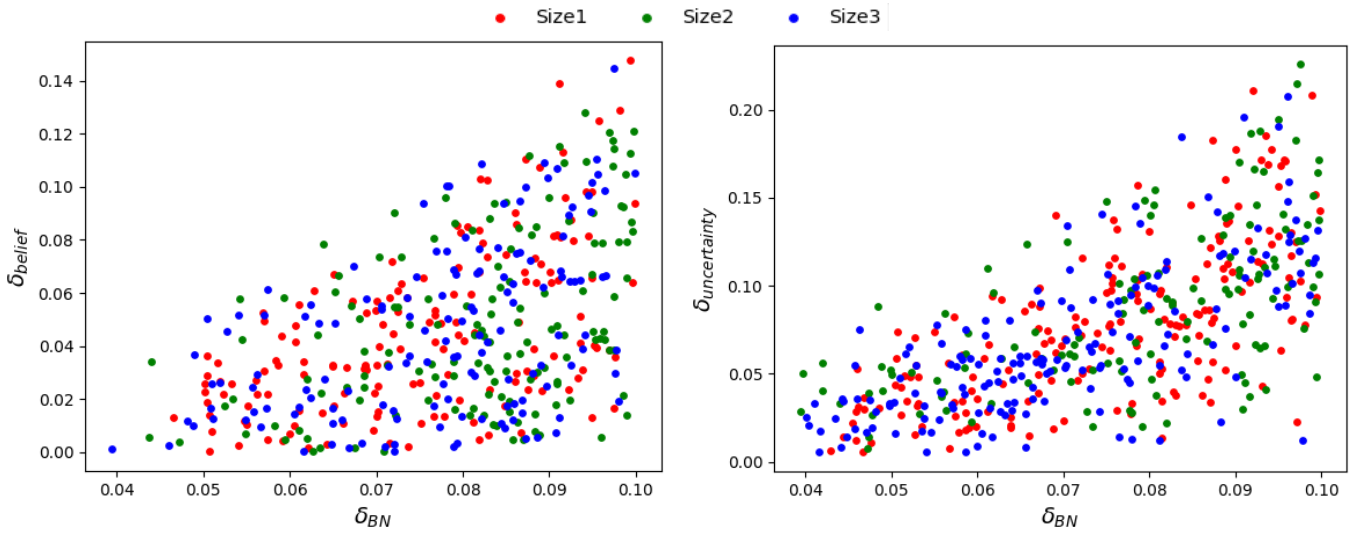
TABLE I
Design parameters, and their meaning

Par.	Meaning
N_{BN}	Number of forward or Gibbs Samples
N_{SBN}	Number of out SBN iterations
δ_{BN}	Discrepancy in probability of BN level between exact and approximate inference
δ_{belief}	Discrepancy in belief mass of SBN level between exact and approximate inference
$\delta_{uncertainty}$	Discrepancy in uncertainty mass of SBN level between exact and approximate inference
ϵ	Threshold value defining the acceptable level of closeness or precision in convergence



(a) N_{BN} vs. $\mathcal{IC}_{size1}$ (b) N_{BN} vs. $\mathcal{IC}_{size2}$ (c) N_{BN} vs. $\mathcal{IC}_{size3}$

Fig. 2. Effect of different number of BN samples (N_{BN}) with respect to varying sampling methods (i.e., forward, Gibbs) in terms of the number of outer SBN iterations to convergence (IC).



(a) δ_{BN} vs. δ_{belief} (b) δ_{BN} vs. $\delta_{uncertainty}$

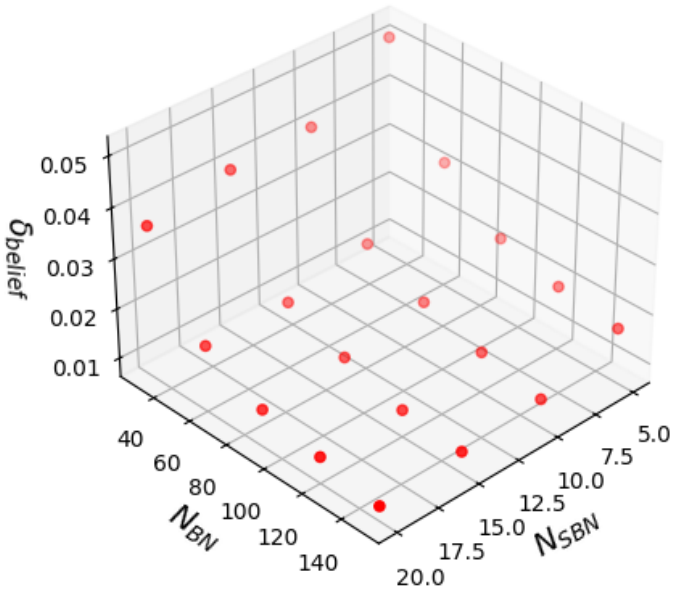
Fig. 3. Effect of errors in BN level (δ_{BN}) with respect to varying network sizes in terms of the errors in SBN level (δ_{belief}, $\delta_{uncertainty}$).

- Network Size 3: We incorporate 20 variables with a 70% edge probability, resulting in a total of 129 edges within the network structure.

V. Numerical Results & Analyses

Average Number of Iterations to Convergence in SBN Approximate Inference: Fig. 2 shows the average number of outer SBN iterations required for convergence under varying numbers of BN samples for all three network sizes. Across all network sizes (i.e., 1(a)-(c)), we can observe that the number of iterations required for convergence decreases when providing more BN samples (i.e., forward, Gibbs). As the network structure and size get larger and more complex, it requires more iterations to reach the convergence level (i.e., ϵ). We observe that Gibbs has better performance than forward sampling. This is because Gibbs usually performs better than forward sampling in average cases. Since we consider multiple configurations of evidence, queries and conditional probability table (i.e., CPT) for this experiment, we deal with average cases, resulting in the outperformance of Gibbs over forward sampling. In other words, these results suggest that allocating small number of samples for the inner (BN-level) inference requires higher number of samples for the outer (SBN-level) inference for convergence. Moreover, the negative slope/decay shows that, at a certain point, allocating more samples for the inner inference would not result in decent decrease in the required number of samples/iterations in the outer inference scope – so there is a potential "point of balance" between the number of samples in the inner versus outer scopes.

Correlation Between Errors in BN Level (i.e., δ_{BN}) and Errors in SBN Level (i.e., δ_{belief}, $\delta_{uncertainty}$): Fig. 3 illustrates the correlation between errors in BN level (i.e., δ_{BN}) and errors in SBN Level (i.e., δ_{belief}, $\delta_{uncertainty}$). These errors represent the absolute value of the difference between exact and approximate inference in BN and SBN levels, respectively. While errors in the BN level represent the discrepancy in inference probability for queries given evidence, errors in the SBN level represent the discrepancy between exact and approximate inference in belief and uncertainty mass of subjective opinion. These scatter dots are



(a) N_{BN} vs. N_{SBN} vs. δ_{belief}

Fig. 4. Effect of number of BN and SBN samples (N_{BN}, N_{SBN}) in terms of the errors in SBN level (δ_{belief}).

obtained by performing approximate inferences using forward or Gibbs sampling. In Fig. 3(a), we observe that as δ_{BN} becomes larger, δ_{belief} gets larger and more unpredictable. In the maximum cases, as δ_{BN} becomes larger, δ_{belief} naturally gets larger. This is because, in Beta distribution, when we have large variances in the input, the output will have large variances. However, we also see cases where δ_{belief} is small when δ_{BN} is large. This is due to the operation in the proposed algorithm that performs an automatic average that cancels out large errors, eventually resulting in low δ_{belief}. In Fig. 3(b), we see the same behavior for $\delta_{uncertainty}$ as δ_{BN} increases. We see this correlation across all three network sizes. While Fig. 2 suggested there is a monotonic relationship between the number of samples in the inner inference versus the outer inference *for convergence*, the observation in Fig. 3 suggests that such relationships may not follow a uniform, predictable curve when the precision is considered (i.e., although low number of inner samples require higher number of outer samples to converge, the converged value may or may not be very precise).

Impact of Number of SBN Iterations and BN Samples on Errors in SBN Level (i.e., δ_{belief}): In Fig. 4, it is evident that increasing both N_{SBN} and N_{BN} results in reduced δ_{belief} for network size 1. Specifically, for this network configuration, the impact of varying the number of BN samples is more pronounced in decreasing δ_{belief} compared to adjusting the number of SBN iterations. The convex property of this graph (i.e., the plane comprising the points in Fig. 4 is convex) shows that there is a potentially optimal balance. This indicates that the general idea presented in Section III-E about "optimal" allocation of resources is plausible and feasible. Achieving an optimal balance between N_{SBN} and N_{BN} is crucial, especially when considering constraints such as limited time and computational resources. In our forthcoming research, we intend to employ Optimal Computing Budget Allocation techniques [18, 19] to dynamically determine the most efficient allocation between the number of outer SBN iterations and inner BN samples. Moreover, the topology of Fig. 4 also suggests that optimizations based on the calculation of gradients might also be promising. It is worth citing that the literature offers many examples of such gradient-based optimizations [23–25].

VI. Conclusion

In this paper, we propose an anytime approximate simulation-based Subjective Bayesian Network (i.e., SBN) inference algorithm. We observe that increasing Bayesian Network (i.e., BN) samples reduces convergence iterations across all network sizes. Additionally, we identify a correlation between errors in BN and SBN levels, highlighting the unpredictability of discrepancies in belief and uncertainty mass as BN errors escalate. Notably, the number of BN samples has a greater impact on SBN belief error than the number of iterations. These findings emphasize the importance of optimizing resource allocation between SBN iterations and BN samples for efficient inference in BNs. Furthermore, the convex nature of the graph, as demonstrated by the plane comprising the points in Figure 4, signifies the presence of a potentially optimal balance. This observation suggests that the idea introduced in Section III-E regarding the "optimal" allocation of resources holds promise and viability. Our approach is a step towards a truly anytime SBN inference algorithm, although not a full solution yet. Our algorithm would become a truly efficient anytime algorithm if the system could optimally decide how much computational power to allocate to the outer iteration (the SBN scope) versus the inner iteration (the BN scope) which will be addressed in our future work.

A. Key Findings & Future work

The **key findings** from our study are:

- Increasing sample sizes in both forward and Gibbs sampling methods leads to a decrease in the number of SBN iterations required for convergence.
- Discrepancies between approximate and exact inference in belief (δ_{belief}) and uncertainty ($\delta_{uncertainty}$) mass of subjective opinion become more unpredictable as errors in BN probabilities grow larger.
- The number of BN samples has a more significant impact on δ_{belief} compared to the number of SBN iterations.

We plan to conduct the following **future work**:

- We will enhance the algorithm's efficiency by dynamically allocating computational resources between outer SBN iterations and inner BN samples. This optimization strategy aims to bring our algorithm closer to achieving truly efficient anytime inference capabilities.

REFERENCES

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

[2] T. Helldin and M. Riveiro, "Explanation Methods for Bayesian Networks: review and application to a maritime scenario," in *Proc. of The 3rd Annual Skövde Workshop on Information Fusion Topics, SWIFT*, 2009, pp. 11–16.

[3] C. Y. Park, K. B. Laskey, P. C. Costa, and S. Matsumoto, "Predictive situation awareness reference model using multi-entity bayesian networks," in *17th International Conference on Information Fusion (FUSION)*. IEEE, 2014, pp. 1–8.

[4] D. Koller and A. Pfeffer, "Object-Oriented Bayesian Networks," in *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 302–313, event-place: Providence, Rhode Island.

[5] K. B. Laskey, "MEBN: A language for first-order bayesian knowledge bases," *Artificial intelligence*, vol. 172, no. 2, pp. 140–178, 2008.

[6] D. Sundgren and A. Karlsson, "Uncertainty Levels of Second-Order Probability," *Polibits*, vol. 48, pp. 5–11, Jul. 2013.

[7] M. Borsotto, W. Zhang, E. Kapanci, A. Pfeffer, and C. Crick, "A Junction Tree Propagation Algorithm for Bayesian Networks with Second-Order Uncertainties," in *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, Nov. 2006, pp. 455–464, iSSN: 2375-0197.

[8] A. Jøsang, *Subjective Logic: A Formalism for Reasoning Under Uncertainty*. Springer, 2016.

[9] M. Chavira and A. Darwiche, "Compiling bayesian networks using variable elimination." in *IJCAI*, vol. 2443. Citeseer, 2007.

[10] S. K. Andersen, K. G. Olesen, F. V. Jensen, and F. Jensen, "Hugin-a shell for building bayesian belief universes for expert systems." in *IJCAI*, vol. 89, no. August, 1989, pp. 1080–1085.

[11] R. L. Harrison, "Introduction to monte carlo simulation," in *AIP Conf. proceedings*, vol. 1204, no. 1. American Institute of Physics, 2010, pp. 17–21.

[12] P. Xie, J. H. Li, X. Ou, P. Liu, and R. Levy, "Using bayesian networks for cyber security analysis," in *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*. IEEE, 2010, pp. 211–220.

[13] S. Matsumoto, J. F. Ferrari, H. J. Yoon, A. R. Thukkaraju, D. Lee, M. K. Ahn, J.-H. Cho, and P. Costa, "Software-friendly subjective bayesian networks: Reasoning within a software-centric mission impact assessment framework," in *2023 26th International Conference on Information Fusion (FUSION)*. IEEE, 2023, pp. 1–8.

[14] A. R. Thukkaraju, H. J. Yoon, S. Matsumoto, J. F. Ferrari, D. Lee, M. K. Ahn, P. Costa, and J.-H. Cho, "Interdependent mission impact assessment of an iot system with hypergame-heoretic attack-defense behavior modeling," in *2023 31st International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2023, pp. 1–8.

[15] C. Li and S. Mahadevan, "Efficient approximate inference in bayesian networks with continuous variables," *Reliability Engineering & System Safety*, vol. 169, pp. 269–280, 2018.

[16] J. Kwisthout, "Approximate inference in bayesian networks: Parameterized complexity results," *International Journal of Approximate Reasoning*, vol. 93, pp. 119–131, 2018.

[17] F. M. Zennaro, M. Ivanovska, and A. Jøsang, "An empirical evaluation of the approximation of subjective logic operators using Monte Carlo simulations," *International Journal of Approximate Reasoning*, vol. 111, pp. 56–77, Aug. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0888613X18305115

[18] G. J. LaPorte, J. Branke, and C.-H. Chen, "Optimal computing budget allocation for small computing budgets," in *Proceedings of the 2012 winter simulation conference (wsc)*. IEEE, 2012, pp. 1–13.

[19] N. A. Pujowidianto, L. H. Lee, C.-H. Chen, and C. M. Yap, "Optimal computing budget allocation for constrained optimization," in *Proceedings of the 2009 Winter Simulation Conference (WSC)*. IEEE, 2009, pp. 584–589.

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003, publisher: JMLR.org.

[21] M. Henrion, "Propagating uncertainty in bayesian networks by probabilistic logic sampling," in *Machine intelligence and pattern recognition*. Elsevier, 1988, vol. 5, pp. 149–163.

[22] T. Hrycej, "Gibbs sampling in bayesian networks," *Artificial Intelligence*, vol. 46, no. 3, pp. 351–363, 1990.

[23] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754–1766, 2020.

[24] G. Zhang, J. Martens, and R. B. Grosse, "Fast convergence of natural gradient descent for over-parameterized neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[25] H. Yuan and T. Ma, "Federated accelerated stochastic gradient descent," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5332–5344, 2020.

[26] A. Ankan and A. Panda, "pgmpy: Probabilistic graphical models using python," in *Proceedings of the 14th Python in Science Conference (SCIPY 2015)*. Citeseer, 2015.