# An Optimized Quantum Circuit Representation of Bayesian Networks

Walid Fathallah[1,2(✉)], Nahla Ben Amor[1], and Philippe Leray[2]

[1] LARODEC, University of Tunis, Tunis, Tunisia
walidfathallah34@outlook.com
[2] Nantes Université, École Centrale Nantes, CNRS, LS2N UMR 6004, 44000 Nantes, France

**Abstract.** In recent years, there has been a significant upsurge in the interest surrounding Quantum machine learning, with researchers actively developing methods to leverage the power of quantum technology for solving highly complex problems across various domains. However, implementing gate-based quantum algorithms on noisy intermediate quantum devices (NISQ) presents notable challenges due to limited quantum resources and inherent noise. In this paper, we propose an innovative approach for representing Bayesian networks on quantum circuits, specifically designed to address these challenges. Our aim is to minimize the required quantum resource needed to implement a Quantum Bayesian network (QBN) on a quantum computer. By carefully designing the sequence of quantum gates within the dynamic circuit, we can optimize the utilization of limited quantum resources while mitigating the impact of noise. Furthermore, we present an experimental study that demonstrates the effectiveness and efficiency of our proposed approach. Through simulations and experiments on NISQ devices, we show that our dynamic circuit representation significantly reduces the resource requirements and enhances the robustness of QBN implementation. These findings highlight the potential of our approach to pave the way for practical applications of Quantum Bayesian networks on currently available quantum hardware.

**Keywords:** Bayesian networks · Quantum circuit · Qiskit

## 1 Introduction

Quantum algorithms are typically expressed in terms of quantum circuits, which describe a computation as a sequence of elementary quantum logic gates acting on qubits. There are many ways of implementing a given algorithm with an available set of elementary operations, and it is advantageous to find an implementation that uses the fewest resources especially on near-term device (NISQ machine) [5,10]. The width of the quantum circuit is key for evaluating the potential of its successful execution on that particular machine. Optimizing this metric when implementing quantum Bayesian Networks will be the aim of our work.


Z. Bouraoui and S. Vesic (Eds.): ECSQARU 2023, LNAI 14294, pp. 160–171, 2024.
https://doi.org/10.1007/978-3-031-45608-4_13

The first tentative to define Quantum Bayesian networks were introduced by Tucci [14] as an analog to classical Bayesian networks. He proposed that the conditional probabilities in a classical Bayesian networks can be represented using quantum complex amplitudes. Tucci argued that there could be infinite possible quantum Bayesian networks for a given classical Bayesian network. Following Tucci ideas, Moreira & Wichert [7] proposed quantum-like Bayesian networks, where the marginal and conditional probabilities were represented using quantum probability amplitudes. To determine the parameters of a quantum Bayesian network, a heuristic method was used that considered the similarity between two dimensional vectors corresponding to the two states of the random variables. In 2014 [6] discussed the principles of quantum circuit design to represent a Bayesian network with discrete nodes that have two states, and also discussed the circuit design for implementing quantum rejection sampling for inference and recently, Borujeni et al. [1], proposed Compositional Quantum Bayesian Network (C-QBN) to represent a discrete Bayesian network and discuss the decomposition of complex gates using elementary gates, so that they can be implemented on available quantum computing platforms. In this paper, we optimize the circuit construction of Compositional Quantum Bayesian network by reducing the width with mid-circuit hardware measurement. We reuse the qubit that represents a variable from the Bayesian network once it doesn't step in the calculation of another event in the chain rule.

This paper is organized as follows: we will first introduce Quantum computing. Then we will moves to present classical and quantum Bayesian networks mainly the work of Borujeni et al. on Quantum Bayesian networks and her approach named (C-QBN) and finally we will detail the proposed method for optimizing a quantum circuit to represent a Bayesian network.

## 2 Basic Quantum Computation

Quantum computers can solve some computational problems exponentially faster than classical computers, which may lead to several applications in field of machine learning. To store and manipulate the information, they use their own quantum bits also called 'Qubits' unlike other classical computers which are based on classical computing that uses binary bits 0 and 1 individually.

Instead of using high and low voltages to represent the 1's and 0's of binary data, we generally use the two spin states of an electron, $|1\rangle$ and $|0\rangle$ [3,12].

Any measurement made on this states will always yield one of the two states with no way of knowing which one. If we prepare an ensemble of identical systems then quantum mechanics will assure that we will observe the result 1 with probability $|\alpha|^2$ and the result 0 with probability $|\beta|^2$. Normalization of the state to unity guarantees:

$$|\alpha|^2 + |\beta|^2 = 1$$

Information stored in a 2-states quantum system is called a quantum bit or qubit: besides storing classical 1 and 0 information there is also the possibility of storing information as a superposition of 1 and 0 states.

To represent the state of a qubit, we can use the Bloch sphere. For instance, if we have a qubit that is initially prepared in state $|1\rangle$ and then apply the NOT operator (also known as the Pauli-X gate), we will find the qubit in state $|0\rangle$. This operation corresponds to a rotation of the qubit state vector by 180° around the X-axis of the Bloch sphere.

The reversible logic gates used in classical computing (such as AND, OR, and NOT) have quantum analogues that are implemented using unitary operators that act on the basis states of a qubit. These quantum gates are also reversible and can be used to perform quantum computations. The basic quantum gates include:

- The *Hadamard gate*, which creates a superposition of the $|0\rangle$ and $|1\rangle$ states.
- The *Pauli gates* which have four different types: $R_X$, $R_Y$ and $R_Z$ gates corresponding to the three axes of the Bloch sphere ($X$, $Y$, and $Z$), and the identity gate. The $R_X$ gate, also known as the NOT gate, flips the value of a qubit from $|0\rangle$ to $|1\rangle$or vice versa. The $R_Y$ gate is similar to the $R_X$ gate, but also introduces a phase shift around the Y-axis.
- The *CNOT gate*, which entangles two qubits and flips the second if the first is in the $|1\rangle$ state.
- The *Measurement gate*, which is used to extract classical information from a quantum state by collapsing a qubit to one of its possible classical states.

These gates form the basis for constructing more complex quantum circuits. The impact of hardware on quantum algorithms is significant, as the performance of a quantum algorithm is ultimately limited by the quality and capabilities of the underlying quantum hardware. These hardware limitations can affect the performance of quantum algorithms in several ways that can be summarized as follows:

- **Number of qubits and the available gate set** on the hardware can limit the size and complexity of the quantum circuits that can be implemented efficiently. Certain quantum algorithms require a large number of qubits or a specific gate set to perform optimally. If the hardware lacks the required number of qubits or gate set, the algorithm may not be implementable or may produce suboptimal results.
- **Coherence time** of the qubits determines how long they can maintain their quantum state before they decohere and become classical. Longer coherence times are generally better for implementing quantum algorithms, as they allow for more operations to be performed before the quantum state is lost. If the coherence time is too short, the algorithm may not be able to be implemented or may perform poorly.
- **Connectivity** of the qubits on the hardware determines how easy it is to implement certain types of quantum circuits, such as those involving entanglement. If the qubits are not well-connected, it may be difficult or impossible to implement certain algorithms efficiently.
- **Error rates** of the gates and measurements on the hardware can limit the accuracy and reliability of the quantum computation. High error rates can lead to a loss of coherence and errors in the final result of the algorithm.

Therefore, as quantum hardware continues to improve, it is expected that the performance and applicability of quantum algorithms will also improve. This is why the development of high-quality and scalable quantum hardware is one of the key challenges in the field of quantum computing. Meanwhile one of the techniques to cushion the impact of hardware on quantum algorithm is to reduce the size of quantum circuit.

## 3 Quantum Bayesian Networks

In this section, we first introduce classical Bayesian networks and then their most recent quantum representation proposed by Borujeni et al. [1].

### 3.1 Classical Bayesian Networks

Bayesian networks [8], are among the most powerful probabilistic graphical models representing knowledge and reasoning under uncertainty. Bayesian networks are widely used in artificial intelligence, machine learning, and decision analysis for tasks such as diagnosis, prediction, and decision-making under uncertainty. They can be used to model complex systems and make predictions about their behavior, even in the presence of missing or noisy data.

Formally, a Bayesian network $BN = \langle G, P \rangle$ has two components:
(i) The *graphical component* composed of a Directed Acyclic Graph (DAG) $G = (V, E)$, where $G$ is a DAG with nodes (or vertices $V$) representing variables and edges $E$ representing the dependencies between variables.
(ii) The *numerical component* $P$ composed of a set of conditional probability distributions $P_{X_i} = P(X_i \mid Pa(X_i))$ for each node $X_i \in V$ in the context of its parents $Pa(X_i)$. The set of all these conditional probability tables $P$ is used to define the joint probability distribution over all variables in the network using a chain rule expressed as:

$$P(X_1.....X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i)) \tag{1}$$

**Example 1.** *Figure 1 shows an example of a Bayesian network with four binary nodes $V = \{A, B, C, D\}$ that we will use in the rest of the article.*

Inference is a crucial task in Bayesian networks that involves calculating probabilities of interest based on observations or evidences. The two most common types of inference are computing marginal probabilities of a subset of variables and conditional probabilities of a subset of variables given evidence about another subset of variables. Inference is an optimization problem that involves manipulating the joint probability distribution of the Bayesian network, which can be computationally expensive for large and complex networks. It has been proven that this problem is NP-hard [2]. The problem of inference in Bayesian networks has been an active research area for decades, leading to many proposed algorithms and techniques for efficient computation of probabilities [4,9].

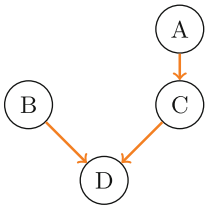


**Fig. 1.** A 4-nodes Bayesian network

### 3.2 Compositional Quantum Bayesian Networks C-QBN

Recently, Borujeni et al. [1] introduced a systematic method for designing a quantum circuit to represent a discrete Bayesian network. This method (outlined by Algorithm 1) is mainly based on mapping each variable in a Bayesian network to one or more qubits (depending on its cardinality). Then, it computes associated gates (via the *Gates* function) by first calculating the probability amplitudes of the qubit states from conditional probabilities, following by obtaining the probability amplitudes of the associated quantum states through the application of rotational gates. In this representation four gates are used: Hadamard gates $X$ (green), *Pauli gates* $R_Y$ (purple), *CNOT gates* and *measurement gates* (black).

Note the use of extra qubits (ancilla bits) that are not part of the input or output of a quantum circuit but are instead used to perform intermediate computations that help to improve the efficiency and accuracy of quantum algorithms. The use of ancilla bits is a common technique in quantum computing.

**Example 2.** *To illustrate the transformation procedure (Algorithm 1), we reconsider the Bayesian network of Fig. 1. This generates a five-qubit circuit represented in Fig. 2. Qubits* $q_0$*,* $q_1$*, and* $q_2$ *and* $q_3$ *are associated to A, B, C, D, respectively, while* $q_4$ *is the ancilla qubit associated to the decomposition on the rotation gate relative to the node D which has 2 parents.*

The resulting quantum circuit can then be used to compute the joint probability of any configuration, or the marginal probability of a subset of variables, by assigning the corresponding values as input of the quantum circuit.

In the proposed method, each node $X$ in the Bayesian network is mapped onto a qubit in a quantum circuit. As mentioned earlier, qubits are a scarce resource in quantum computing, and reducing the number of qubits required to represent the network can provide a significant advantage in representing more complex networks and performing more sophisticated analyses. This is the main idea that we propose in the following section.

## 4 Optimized Representation of Quantum Bayesian Networks

In this section, we present an optimized Algorithm that reduces the size of a given quantum Bayesian network circuit compared to Algorithm 1.

**Algorithm 1:** Transformation of a BN into a QC

**Input** : $BN =< G = (V, E), P >$
**Output:** A quantum circuit $QC$
$QC \leftarrow$ an empty quantum circuit
**for** *each* $X$ *in topological order of* $V$ **do**
  Create a qubit $q_X$
  $A_X \leftarrow$ empty
  **for** *each* $Y$ *in* $Pa(X)$ **do**
    Create ancilla_qubit $a_x$
    Add $a_x$ to $A_X$
  **end**
  **for** $i$ *in* $|Dom(Pa(X))| - 2$ **do**
    Create ancilla_qubit $a_x$
    Add $a_x$ to $A_X$
  **end**
  $G_X \leftarrow Gates(X, P_X, q_X, A_x)$
  Add $G_X$ to $QC$
**end**
**for** *each* $X$ *in* $V$ **do**
  Measure$(q_X, QC)$
**end**

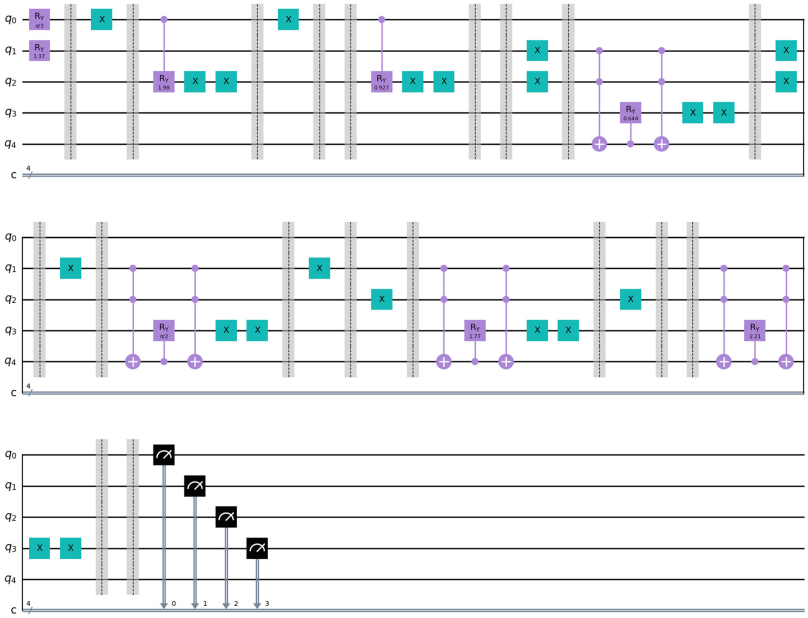


**Fig. 2.** Quantum circuit of Bayesian network of Fig. 1 (Algorithm 1)

The idea is to take advantage of the structure of the DAG in the given Bayesian network to measure and reuse the qubit that represents a node midway through, by using the standard measurement gate before applying further quantum gates. This allows us to reuse a qubit after computing the probability amplitude of all its child nodes.

The optimized version, outlined by Algorithm 2, starts by initializing an empty quantum circuit QC and creating a list $Pa_{list}$ that contains all the parents of each node. This list will serve as an indicator to know if there are still nodes that have not been mapped to the circuit and that require the presence of their parents to calculate their probability amplitude. Otherwise, the qubit relative to this node will be measured and added to $Available_{list}$ to be reused for another variable. Then, it iterates over the nodes in $V$ in a topological order[1]. For each node $X$, it computes the number of extra qubits $n$ needed by the quantum gates to represent its probability distribution. It also computes the number of qubits $k$ required to represent the probability distribution even with the reuse of the reinitialized qubits in $Available_{list}$. If the node has no parents, it creates a new qubit for it and computes the quantum gates that implement the node's probability distribution $G_X(X, P_X, q_X, A_X)$. Then If $Pa(X)$ is already in $Pa\_list$, the algorithm checks if $Available\_list$ is empty. Then it creates a new qubit and build its gates. Or it uses a qubit from $Available\_list$.

After that, we update the $Pa_{list}$ and the $Available\_list$, and perform mid-circuit measurement if needed, based on the requirements of the not-yet-built nodes, with regard to the presence of their parent nodes, to compute their probability amplitude. Finally, we measure all the qubits that have not yet been measured in $V$ and add these measurements to $QC$. The resulting quantum circuit can be used to compute the joint probability distribution of the Bayesian network BN.

**Example 3.** *Given the BN in Fig. 1, let us consider the following topologically order* $[A, B, C, D]$. *We have* $Pa\_list = [Pa(A), Pa(B), Pa(C), Pa(D)] = [A, B, C]$. *We start by considering node A, which is binary and root node. Since A has no parents, we allocate only one qubit, denoted* $q_0$, *to build its gates and add them to the quantum circuit. We do not make any modifications to the available Parent list Pa_list. Next, we move to the variable B. Similar to A, we allocate one qubit* $q_1$ *and build its gates.*

*Then, we handle variable C which is a parented node with* $Pa(C) = A$. *To compute its gates, we need the values from A gates because the values expressed by its conditional probability table* $P(C \mid A)$ *are based on A. After computing the probability amplitudes of the qubits* $q_2$ *and adding the gates to the circuit, we delete A from Pa_list and add its relative qubit to Available_list because no further nodes in the topological list are dependent on it. This allows us to apply a measurement gate to* $q_0$ *then a reset gate, enabling its reuse to map another variable and reducing the global width of the circuit.*

[1] A numbering of the vertices of a DAG such that every edge from a vertex numbered $i$ to a vertex numbered $j$ satisfies $i < j$ i.e. ancestors before descendants.

*Finally, we move to the last node D, which has two parents. This requires the use of an extra ancilla qubit ($q_3$), which is added to the node itself, and only one qubit from Available_list is added to the circuit of Fig. 3. This will act on global width of the final circuit by reducing it from 5 qubits using Algorithm 1 to 4 qubits using our optimized algorithm.*

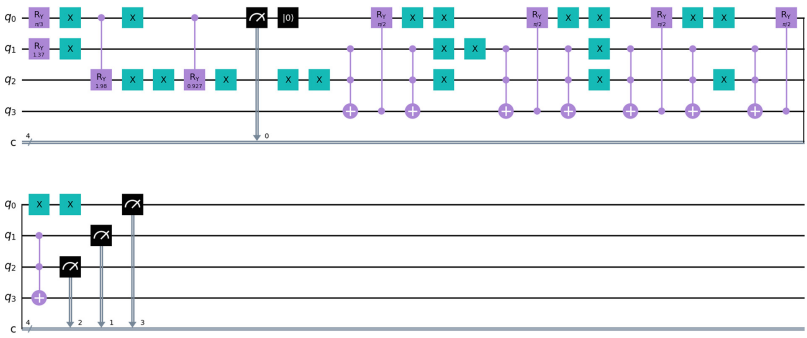


**Fig. 3.** Optimized Quantum circuit of BN in Fig. 2 (Algorithm 2)

## 5 Experiments

To evaluate the effectiveness of our algorithm, we analyze the Bayesian network shown in Fig. 4, which consists of 10 binary nodes. It is worth noting that this network was previously used in [1], where Algorithm 1 required 12 qubits to transform it into a quantum circuit, while our optimized version only needs 6 qubits. Our main objective is to assess the accuracy of our model in terms of marginal probabilities. To achieve this, we compare the results obtained through an exact inference algorithm applied to the original Bayesian network with those obtained by measuring the quantum circuits generated by Borujeni's algorithm and our optimized approach. We use the mps method from Qiskit Aer, an open-source quantum circuit simulator [11], to simulate the quantum circuits and also execute them on a real quantum machine with 7 qubits (IBM_Perth).

We ran the Bayesian network circuit five times on the simulatorwithout any hardware noise and on a real quantum computer, each with 20,000 shots.

To investigate the effect of width reduction of QBN circuits using the two approaches described in Algorithm 1 and 2, we computed the root mean square error (RMSE) expressed by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \left( P(X_i = 0) - \hat{P}(X_i = 0) \right)^2}{N}}$$

**Algorithm 2:** Optimized transformation of a BN into a QC

**Input :** $BN =< G = (V, E), P >$
**Output:** A quantum circuit $QC$
$Pa\_list \leftarrow \bigcup_{X \in V} Pa(X)$
$Available\_list \leftarrow \{\}$
$QC \leftarrow$ an empty quantum circuit
**for** *each* $X$ *in topological order of* $V$ **do**
  $n \leftarrow extra\_qubit(Pa(X))$
  $k \leftarrow n - |Available\_list|$
  **if** $k > 0$ **then**
    Create $k$ qubit(s) $A_X$
  **end**
  `/* check if we need to add additional qubits */`
  **if** $Pa(X) \notin Pa\_list$ **then**
    Create a qubit $q_X$
    $G_X \leftarrow$Gates($X$, $P_X$, $q_X$, $A_x$)
  **else**
    **if** $Available\_list = \{\}$ **then**
      Create a qubit $q_X$
      $G_X \leftarrow$ Gates($X$, $P_X$, $q_X$, $A_x$)
    **else**
      $G_X \leftarrow$Gates($X$, $P_X$, $q_X$, $A_x$)
      Delete(Available_list,$q_X$)
    **end**
  **end**
  **if** $Count(Pa\_list, Pa(X)) = 1$ **then**
    Measure($q_{Pa(X)}, QC$)
    Reset($q_{Pa(X)}$,$A_X$)
    Add($Available\_list, q_{Pa(X)}, A_X$)
  **else if** $Count(Pa\_list, Pa(X)) > 1$ **then**
    Delete($Pa\_list, Pa(X)$)
    Reset($A_X$)
    Add($Available\_list, A_X$)
  **else**
    Reset( $A_X$)
    Add($Available\_list, A_X$)
  Add $G_X$ to $QC$
**end**
**for** *each not measured* $X$ *in* $V$ **do**
  Measure($q_X, QC$)
**end**

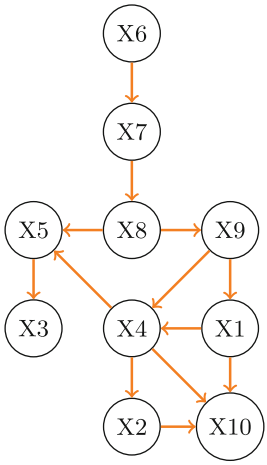


**Fig. 4.** A 10-node Bayesian network [13]

where N is the number of nodes in the Bayesian network, $P(X_i = 0)$ is the exact probability computed from the full joint distribution, and $\hat{P}(X_i = 0)$ is the probability from the quantum circuit. This measure will indicate the extent to which a set of marginal probabilities computed with a simulator and a real quantum computer deviates from the exact values. The results showed that the RMSE of the optimized circuit is lower than the one generated by Algorithm 1 (3% versus 7%). This improvement is particularly noteworthy given the size of the initial network, and is attributed to the efficient reuse of qubits enabled by our approach.

Note that the circuit generated by Algorithm 1 exceeded the 7 qubits available on the real quantum machine we used, and thus could not be executed. Therefore, we only tested the optimized circuit generated by Algorithm 2 on a real quantum computer.

Clearly, the reduction in the width of the quantum circuit has the potential to improve the error rate as it reduces the number of physical qubits required to implement the circuit. This, in turn, minimizes the complexity of the hardware and mitigates some sources of errors (Table 1).

**Table 1.** Exact, then mean values of marginal probabilities of the 10 node Bayesian network on the simulator with the two approaches and on IBM_perth quantum computer

| Marginal | Exact probability | Simulator | | Quantum computer IBM_perth Algorithm 2 |
|---|---|---|---|---|
| | | Algorithm 1 | Algorithm 2 | |
| $P(X_1 = 0)$ | 0.431 | 0.441 | 0.455 | 0.651 |
| $P(X_2 = 0)$ | 0.863 | 0.867 | 0.867 | 0.567 |
| $P(X_3 = 0)$ | 0.976 | 0.976 | 0.974 | 0.670 |
| $P(X_4 = 0)$ | 0.570 | 0.563 | 0.549 | 0.576 |
| $P(X_5 = 0)$ | 0.527 | 0.528 | 0.518 | 0.522 |
| $P(X_6 = 0)$ | 0.980 | 0.981 | 0.981 | 0.884 |
| $P(X_7 = 0)$ | 0.977 | 0.977 | 0.978 | 0.899 |
| $P(X_8 = 0)$ | 0.026 | 0.026 | 0.0285 | 0.701 |
| $P(X_9 = 0)$ | 0.956 | 0.956 | 0.955 | 0.507 |
| $P(X_{10} = 0)$ | 0.240 | 0.462 | 0.331 | 0.464 |
| **RMSE** | | **7%** | **3%** | **30%** |

## 6 Conclusion and Perspectives

We have proposed an optimized version to design a quantum circuit to represent Bayesian networks based on C-QBN approach. Our approach takes advantage of the structure of the DAG in Bayesian networks to measure and reuse the qubit that represents a node midway through, by using the standard measurement gate before applying further quantum gates. This allows us to reuse a qubit after computing the probability amplitude of all its child nodes.

This technique has been shown to reduce the width of the quantum circuit even on small networks, as demonstrated by the example with 10 nodes, where it resulted in a reduction of half the number of qubits required to implement the QBN circuit. As a result, the reduction in the number of required qubits leads to a simplification of the hardware and helps to alleviate certain sources of errors.

While our first experiments with two examples showed promising results, further investigation on more complex Bayesian networks is needed to fully evaluate the effectiveness of our technique in reducing the width of quantum circuits. As access to quantum hardware with larger numbers of qubits becomes available under certain conditions, we plan to test our approach on more challenging problems. In addition, we will investigate the potential benefits of reducing the number of qubits required for implementing a quantum circuit, which could provide additional resources for improving the overall reliability of the computation. One approach we will explore is integrating error correction techniques directly into the circuit design, which could further reduce error rates.

## References


1. Borujeni, S.E., Nannapaneni, S., Nguyen, N.H., Behrman, E.C., Steck, J.E.: Quantum circuit representation of Bayesian networks. Expert Syst. Appl. **176**, 114768 (2021)
2. Cooper, G.F.: The computational complexity of probabilistic inference using Bayesian belief networks. Artif. Intell. **42**(2), 393–405 (1990). https://www.sciencedirect.com/science/article/pii/000437029090060D
3. Hey, T.: Quantum computing: an introduction. Comput. Control Eng. J. **10**, 105–112 (1999)
4. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. J. Roy. Stat. Soc. Ser. B-Methodol. **50**, 415–448 (1988)
5. Leymann, F., Barzen, J.: The bitter truth about gate-based quantum algorithms in the NISQ era. Quantum Sci. Technol. **5**(4), 044007 (2020)
6. Low, G.H., Yoder, T.J., Chuang, I.L.: Quantum inference on Bayesian networks. Phys. Rev. A **89**(6) (2014)
7. Moreira, C., Wichert, A.: Quantum-like Bayesian networks for modeling decision making. Front. Psychol. **7**, 1–20 (2016)
8. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, Burlington (1988)
9. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco (1988)
10. Preskill, J.: Quantum computing in the NISQ era and beyond. Quantum **2**, 79 (2018)
11. Qiskit contributors: Qiskit: An open-source framework for quantum computing (2023). https://doi.org/10.5281/zenodo.2573505
12. Schrödinger, E.: An undulatory theory of the mechanics of atoms and molecules. Phys. Rev. **28**, 1049–1070 (1926)
13. Tavana, M., Abtahi, A.R., Di Caprio, D., Poortarigh, M.: An artificial neural network and Bayesian network model for liquidity risk assessment in banking. Neurocomputing **275**, 2525–2554 (2018)
14. Tucci, R.R.: Quantum Bayesian nets. Int. J. Mod. Phys. B **09**(03), 295–337 (1995)