

An optimized Quantum circuit representation of Bayesian networks

Walid Fathallah^{1,2}, Nahla Ben Amor¹, Philippe Leray²

¹LARODEC, University of Tunis, Tunis, Tunisia

²Nantes Université, École Centrale Nantes, CNRS, LS2N UMR 6004, F-44000 Nantes, France
walid.fathallah@outlook.com, nahla.benamor@gmx.fr, philippe.leray@ls2n.fr

Abstract

Recent years have seen a surge of interest in Quantum machine learning, methods to harness the power of quantum technology in solving high complexity problems are quickly developed in a variety of domains. Implementing a gate-based quantum algorithm on a noisy intermediate quantum device (NISQ) presents several challenges related to the fact that these devices have limited quantum resources and there are noisy. In this paper, we propose a new way to represent Bayesian networks on quantum circuit. The proposed approach will reduce depth and size of the quantum circuit necessary to implement Quantum Bayesian network (QBN) on a quantum computer. Experimental study showing the efficiency of the proposed approach is also presented.

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) (Preskill 2018)(Leymann and Barzen 2020). The depth and width of the quantum circuit is key for evaluating the potential of its successful execution on that particular machines. Optimizing those two metrics when implementing quantum Bayesian Networks will be the aim of our

work. take into account that the depth represent the number of gates to be performed sequentially, and the width is the number of qubits it actually manipulates.

The first tentative to define Quantum Bayesian networks were introduced by Tucci (Tucci 1995) 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 (Moreira and Wichert 2016) 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 which considered similarity between two dimensional vectors corresponding to the two states of the random variables. In 2014 (Low, Yoder, and Chuang 2014) 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. (Borujeni et al. 2021), proposed Compositional Quantum Bayesian Network (C-QBN) to represent a discrete Bayesian network and discuss the decomposition of complex gates using elementary gates such 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 and the depth. We reuse the qubit that represent a variable from the Bayesian network once it doesn't step in the calculation of another event in the chain rule also we will eliminate the redundancy of not gate in the circuit.

This paper is organized as follows : we will first introduce Quantum computing then the impact of hardware on quantum algorithms. Then it 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 details the proposed method for optimizing a quantum circuit to represent a Bayesian network.

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 (Hey 1999).

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 $|0\rangle$.

The time evolution of a quantum system is usually approximated by the Schroedinger equation. The superposition property of quantum mechanics means that the general state may be written as a superposition of eigenstates (Schrödinger 1926). In the case of the 2-states quantum system the general state may be written as :

$$|\Psi\rangle = \alpha|0\rangle.\beta|1\rangle \quad (1)$$

According to the standard interpretation of quantum mechanics, any measurement (of spin or polarization) made on this state will always yield one of the two eigenvalues with no way of knowing which one. If we prepare an ensemble of identical systems then quantum mechanics assures us that we will observe result 1 with

probability $|\alpha|^2$ and 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 degrees around the X-axis of the Bloch sphere (Figure 1).

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 selection of gates for a quantum circuit depends on the specific problem being solved and the algorithm being employed.

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 limita-

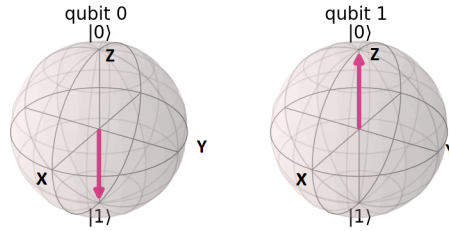


FIGURE 1 – Bloch sphere

tions 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. Additionally, the design of new and efficient quantum algorithms that can take advantage of the unique properties of the hardware is also an active area of research.

One of the techniques to cushion the impact of hardware on quantum algorithm is to reduce the size of quantum circuit (depth and width). The depth refers to the number of layers of gates that must be applied sequentially to transform the input state into the desired output state and the width refers to the number of qubits or quantum bits that are involved in the computation.

The relationship between depth and width of a quantum circuit depends on the specific computation being performed. In general, increasing the width of a quantum circuit tends to increase its depth. This is because more qubits typically require more gates to interact with each other, and this interaction often takes place over multiple layers.

Due to the multiple factor explained earlier, the depth d and width w must be controlled. In practice, the following formula from (Willsch et al. 2017) approximates the limits of executing a quantum algorithm on a given quantum computer where ϵ is the error rate of the quantum computer :

$$d.w < \frac{1}{\epsilon} \quad (2)$$

As the formula indicates, the depth d or the width w must be small. For example, if an algorithm requires 7 qubits ($w = 7$) and it should run on a quantum computer (ibmq-jakarta) with an error rate of about $\epsilon \simeq 8.220e-3$, then d must be less than 18 , i.e. the algorithm must terminate after no more than 18 sequential steps, otherwise the result would be

far too inaccurate.

Quantum Bayesian networks

In this section, we first introduce classical Bayesian networks and then their most recent quantum representation proposed by Borujeni et al. (Borujeni et al. 2021).

Classical Bayesian networks

Bayesian networks (Pearl 1988a), 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 the behavior of those systems, even in the presence of missing or noisy data.

Formally, a Bayesian network $BN = \langle G, P \rangle$ has two components :

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.

The *numerical component* P composed of a set of conditional probability distributions $P_{X_i} = P(X_i | Pa(X_i))$ 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 :

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i | Pa(X_i)) \quad (3)$$

Example 1 *Figure 2 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 evidence. The two most common types of inference are computing marginal probabilities of a subset of variables and conditional probabilities of a subset

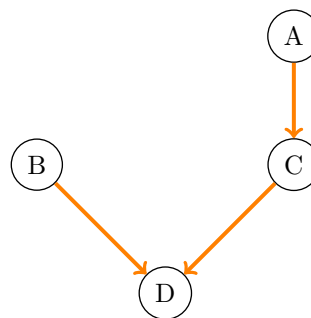


FIGURE 2 – A 4-nodes Bayesian network

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 (Cooper 1990).

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 (Pearl 1988b)(Lauritzen and Spiegelhalter 1988) .

Compositional Quantum Bayesian networks C-QBN

Recently, (Borujeni et al. 2021) 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, and then 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.

Algorithm 1: Transformation of a BN into a QC

Input : $BN = \langle G = (V, E), P \rangle$
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

Example 2 To illustrate Algorithm 1, we reconsider the Bayesian network of Figure 2. This generates a five-qubit circuit represented in Figure 4. Qubits q_0, q_1, 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.

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.

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. 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 Figure 2, 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 ($P(C | A)$) 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.

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 4. This will act on global width of the final circuit by reducing it from 5 qubits using Algorithm 1 to 4 qubits using our algorithm.

Experiments

To evaluate the effectiveness of our algorithm, we analyze the Bayesian network shown in Figure 5, which consists of 10 binary nodes. It is worth noting that this network was previously used in (Borujeni et al. 2021), where Algorithm 1 required 12 qubits to transform

Algorithm 2: Optimized transformation of a BN into a QC

Input : $BN = \langle G = (V, E), P \rangle$
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**
 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
end
for each not measured X in V **do**
 | Measure(q_X, QC)
end

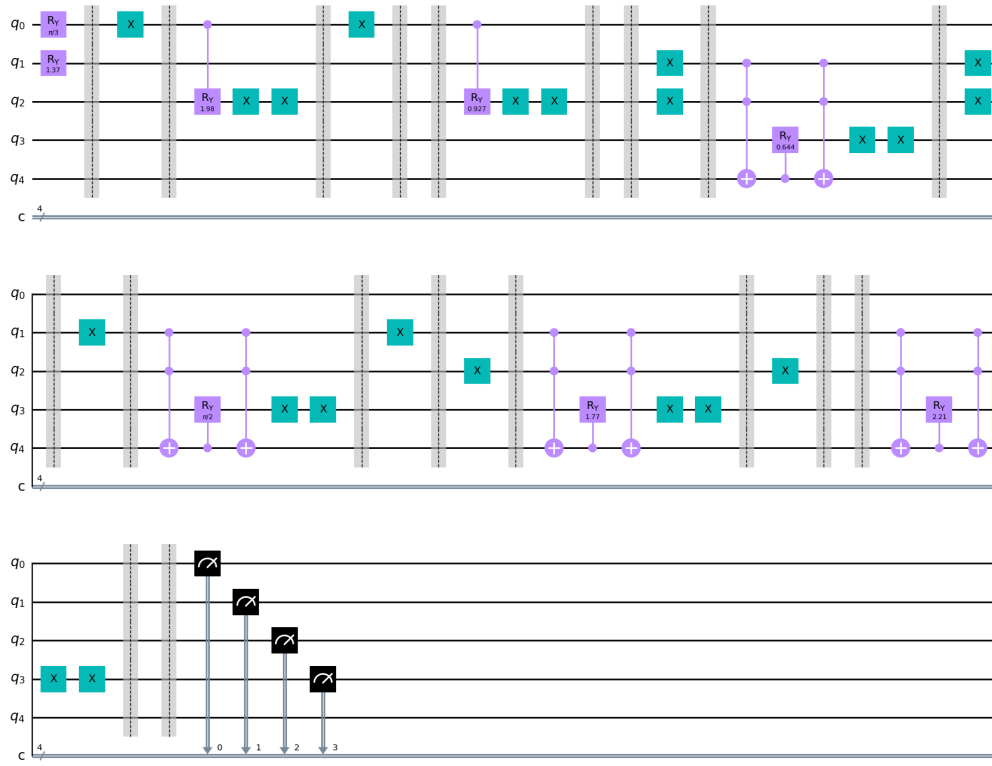


FIGURE 3 – Quantum circuit of Bayesian network of Figure 2

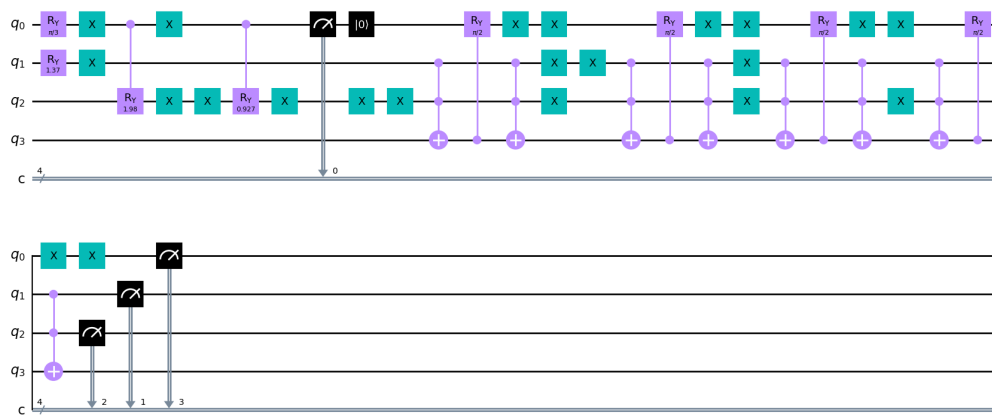


FIGURE 4 – Optimized Quantum circuit of Bayesian network of Figure 2

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 (Qiskit contributors 2023), to simulate the quantum circuits and also execute them on a real quantum machine with 7 qubits (IBM_Perth).

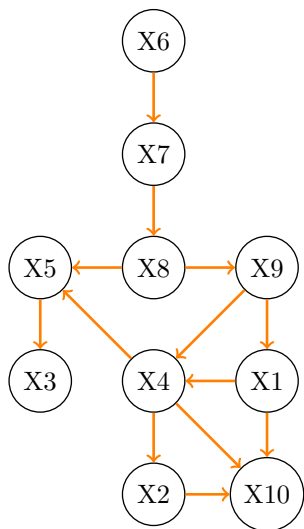


FIGURE 5 – A 10-node Bayesian network (Tavana et al. 2018)

We ran the Bayesian network circuit five times on a simulator without any hardware noise and on a real quantum computer, each with 20,000 shots. The mean values of a selection of marginal probabilities are presented in Table 1. 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 :

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

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.

Table 1 shows 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.

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.

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

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 Algorithm2
		Algorithm 1	Algorithm 2	
$P(X_1 = 0)$	0.431	0.44085	0.4549	0.6506
$P(X_2 = 0)$	0.863	0.86655	0.86755	0.56695
$P(X_3 = 0)$	0.976	0.97615	0.9736	0.69965
$P(X_4 = 0)$	0.570	0.56265	0.5487	0.5758
$P(X_5 = 0)$	0.527	0.5285	0.51815	0.52205
$P(X_6 = 0)$	0.980	0.9806	0.981	0.8842
$P(X_7 = 0)$	0.977	0.97685	0.9784	0.89905
$P(X_8 = 0)$	0.026	0.02565	0.02845	0.7006
$P(X_9 = 0)$	0.956	0.9564	0.95545	0.5072
$P(X_{10} = 0)$	0.240	0.46165	0.3312	0.46385
RMSE		7%	3%	30%

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.

Références

- [1] Borujeni, S. E. ; Nannapaneni, S. ; Nguyen, N. H. ; Behrman, E. C. ; and Steck, J. E. 2021. Quantum circuit representation of bayesian networks. *Expert Systems with Applications* 176 :114768.
- [2] Cooper, G. F. 1990. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence* 42(2) :393–405.
- [3] Hey, T. 1999. Quantum computing : An introduction. *Computing & Control Engineering Journal* 10 :105 – 112.
- [4] Lauritzen, S. L., and Spiegelhalter, D. J. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the royal statistical society series b-methodological* 50 :415–448.
- [5] Leymann, F., and Barzen, J. 2020. The bitter truth about gate-based quantum algorithms in the nisq era. *Quantum Science and Technology* 5(4) :044007.
- [6] Low, G. H. ; Yoder, T. J. ; and Chuang, I. L. 2014. Quantum inference on bayesian networks. *Physical Review A* 89(6).
- [7] Moreira, C., and Wichert, A. 2016. Quantum-like bayesian networks for modeling decision making. *Frontiers in Psychology* 7 :11.
- [8] Pearl, J. 1988a. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann.
- [9] Pearl, J. 1988b. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc.

- [10] Preskill, J. 2018. Quantum computing in the NISQ era and beyond. *Quantum* 2 :79.
- [11] Qiskit contributors. 2023. Qiskit : An open-source framework for quantum computing.
- [12] Schrödinger, E. 1926. An undulatory theory of the mechanics of atoms and molecules. *Phys. Rev.* 28 :1049–1070.
- [13] Tavana, M.; Abtahi, A.-R.; Di Caprio, D.; and Poortarigh, M. 2018. An artificial neural network and bayesian network model for liquidity risk assessment in banking. *Neurocomputing* 275 :2525–2554.
- [14] Tucci, R. R. 1995. Quantum bayesian nets. *International Journal of Modern Physics B* 09(03) :295–337.
- [15] Willsch, D.; Nocon, M.; Jin, F.; De Raedt, H.; and Michielsen, K. 2017. Gate-error analysis in simulations of quantum computers with transmon qubits. *Phys. Rev. A* 96 :062302.