Foundations of Science
https://doi.org/10.1007/s10699-021-09781-6

Quantum Bayesian Decision-Making

Michael de Oliveira[1] · **Luis Soares Barbosa**[1]

© The Author(s), under exclusive licence to Springer Nature B.V. 2021

Abstract
As a compact representation of joint probability distributions over a dependence graph of random variables, and a tool for modelling and reasoning in the presence of uncertainty, Bayesian networks are of great importance for artificial intelligence to combine domain knowledge, capture causal relationships, or learn from incomplete datasets. Known as a NP-hard problem in a classical setting, Bayesian inference pops up as a class of algorithms worth to explore in a quantum framework. This paper explores such a research direction and improves on previous proposals by a judicious use of the utility function in an entangled configuration. It proposes a completely quantum mechanical decision-making process with a proven computational advantage. A prototype implementation in Qiskit (a Python-based program development kit for the IBM Q machine) is discussed as a proof-of-concept.

Keywords Bayesian inference · Quantum algorithms · Quantum decision making

1 Motivation

Bayesian reasoning is widely used in machine learning and data science, as a powerful framework for probabilistic analysis, applications ranging from learning processes (Neal 1996) to pragmatic representations (Li et al. 2018). Broadly speaking, machine learning algorithms are able to learn from data, with the purpose of performing some tasks, without requiring explicit programming; in a sense outcomes are directly built by the sampled data. However, the current rate of data creation is almost exponential Al-Jarrah et al. (2015) (going, for example, from 3.5 million text messages per minute in 2016, to

This work is financed by the ERDF–European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation–COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT, within project POCI-01-0145-FEDER-030947. The first author was further supported by project NORTE-01-0145-FEDER-000037, funded by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement.

✉ Michael de Oliveira
michaeldeoliveira@live.com.pt

Luis Soares Barbosa
lsb@di.uminho.pt

[1] INL - Quantum Software Engineering, Universidade do Minho, Braga, Portugal

Published online: 20 March 2021

over 15 million in 2017), a fact that calls for radically new approaches and, most probably, new computational models and hardware to effectively deal with such numbers.

Can quantum computing bring some useful contribution to this state of affairs? On the one hand it is well known that building very large quantum-addressable classical memories is technologically very demanding, and will not be available soon. On the other, at least from a theoretical point of view, the question seems worth to discuss. Actually, even at its present, quite preliminar stage of development, quantum computing allows for a variety of speed ups with respect to classical algorithmic counterparts in e.g. information storage (Giovannetti et al. 2008), pattern recognition (Biamonte et al. 2016), and matrix inversion, the latter being a basic ingredient of several machine learning algorithms (Harrow et al. 2009). As a matter of fact, quantum algorithms, as the ones discussed in this paper, suggest radically different ways to approach old problems and to explore complexity boundaries. For example, to know whether for a concrete problem, as the size of the input parameter grows, one may asymptotically go faster with the use of a quantum memory than with purely classical states, is a question underlying many interesting problems from big-data to optimisation, or molecular synthesis.

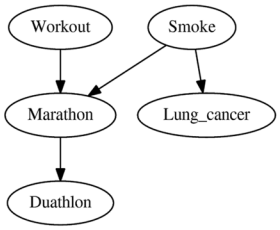
The synergies between research lines in quantum technologies and Bayesian inference, in particular, seem promising. In one direction, a quantum processor can be expressed and studied as a Bayesian Network (Sakkaris 2016). In the reverse one, quantum mechanics can describe naturally probabilistic systems in physical terms. Reference (Mansinghka 2009), for example, describes very promising improvements on the implementation of approximate Bayesian inference routines resorting to physical stochastic logic gates building up hardware implementations of sampling algorithms. Quantum processors are, in a sense, part of such a family.

This paper is a step in this direction. Our starting point is a quantum version of a Bayesian inference algorithm introduced by Low et al. (2014) based on a square-root quantum speedup to rejection sampling on a Bayesian network, which avoids the use of an oracle. Note that an oracle-based version appeared previously in Ozols et al. (2013). This approach, revisited in Sect. 3, was implemented by us on Qiskit—the IBM open-source platform for quantum computing. Our main contribution, presented in Sect. 4, extends Low et al. algorithm to a decision-making setting: this incorporates an utility function which is applied before any observation of the quantum state which encodes the Bayesian network. The computational effort for the proposed solution and a simpler quantum solution are determined in Sect. 5. A proof-of-concept implementation Qiskit is discussed in Sect. 6. Finally, Sect. 7 concludes and points out a number of issues for future work. A background section—Sect.2—recalls the Bayesian inference problem and provides a brief overview of the basic intuitions underlying decision making.

2 Background

2.1 Bayesian Inference

Bayesian inference is used to update the posterior probability distribution of some query variables given the value of the observed variables, also known as evidence variables (Russel and Norvig 2010). The conditional probability is given by



Workout	P(Workout)
True	0.7
False	0.3

Smoke	P(Smoke)
True	0.6
False	0.4

Smoke	Workout	Marathon	P(Marathon | Workout,Smoke)
True	True	True	0.2
True	True	False	0.8
True	False	True	0.05
True	False	False	0.95
False	True	True	0.4
False	True	False	0.6
False	False	True	0.1
False	False	False	0.9

Smoke	Lung cancer	P(Lung cancer | Somke)
True	True	0.02
True	False	0.98
False	True	0.005
False	False	0.995

Marathon	Duathlon	P(Duathlon | Marathon)
True	True	0.7
True	False	0.3
False	True	0
False	False	1

Fig. 1 Bayesian network over 5 variables

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{1}$$

These joint probabilities can be stored in a distribution table, but note that the dimension of the latter grows exponentially with the number of variables. This means that for most applications the table would be too large to be stored computationally. Alternatively, Bayesian networks, as in Fig. 1, allow for a compact representation of joint probability distributions (Darwiche 2008) as a directed acyclic graph structure. The advantage is that the space complexity of the representation can be made much smaller than in the general case, by exploiting conditional dependencies in the distribution, through the association to each graph node of a conditional probability table for each random variable, with directed edges representing conditional dependencies. For this reason, they are largely used in industrial applications. However, inference via a Bayesian Networks is still a NP-problem. Figure 1 depicts a toy Bayesian network relating a few variables encoding different sorts of activities and the possibility of a lung cancer diagnosis.

2.1.1 Inference

Algorithms that infer over a Bayesian network compute joint probabilities using the following equation:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | Parents(Xi)) \tag{2}$$

The computational effort to determine this value is low because the number of values selected is linearly bounded by the number of variables. The envisaged joint probability, however, may not be defined for all variables. If such is the case, it is necessary to sum out over all undefined variables as in

$$P(X_1 | x_2, x_3) = \sum_{x_4} \sum_{x_5} P(X_1, x_2, x_3, x_4, x_5) \tag{3}$$

Consequently, the number of values to sum out grows exponentially with the number of undefined variables. A well known algorithm for variable elimination algorithm works exactly in this way. Approximate algorithms, treading off consumption of computational resources for precision, are typically used to tackle this problem. Solutions are found faster

but may not be precise. The literature documents a bunch of approximate algorithms. In this paper we will focus on *rejection sampling* because the first part of the quantum inference algorithm discussed in the next section is a quantum analog to it. Rejection sampling is a popular method first systematised by von Neumann (1951), who curiously enough also developed the Hilbert space formalization of quantum mechanics and its logic.

Rejection sampling generates samples resorting to the probability distributions defined by the conditional probability tables as depicted in Fig. 1. The consequence of this generation process is that a certain configuration of values for the variables is only sampled with the associated probability:

$$P(Sample < X_1 = true, X_2 = false >) = P(X_1 = true, X_2 = false) \tag{4}$$

Under those circumstances, a conditional probability can be determined by:

$$P(X_1 = true | X_2 = false) \approx \frac{\#Samples(X_1 = true, X_2 = false)}{\#Samples(X_2 = false)} \tag{5}$$

Clearly, the precision of the query grows with the number of useful samples (#*Samples*). It is important to notice that not all samples are useful since samples with different values for the evidence variables are not used.

2.1.2 Bayesian Networks for Decision Making

Bayesian networks are equipped with an *utility function* in order to support decision-making processes. Its purpose is to quantify the utility of possible outcomes. The *expected utility* (*EU*) of an outcome is the product of its probability and the associated utility value. Formally, to find the expected utility of some action a, one computes

$$EU(a|e) = \sum_r P(Result = r|a, e) * U(r) \tag{6}$$

If the *EU* values of all feasible actions is known, it is possible to choose the 'best', or more profitable, one:

$$action = argmax_a EU(a|e) \tag{7}$$

This is indeed the maximum expected utility principle; a rational entity is expected to choose the action with the greatest expected utility with respect to her set of beliefs (Russel and Norvig 2010).

The previous principle describes many algorithms and solutions used in artificial intelligence. For example, in reinforcement learning a great number of *agents* and *robots* are built on a process that attempts to find the optimal policy. This works with an instance of a Bayesian network (Markov decision process) and a more complex utility function (the so-called discounted reward function), where the agent also accounts for future rewards ($R(S_t)$), which are reachable from the starting state. Further, it values present rewards over future rewards with the use of a discount factor γ^t.

$$U_\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right] \tag{8}$$

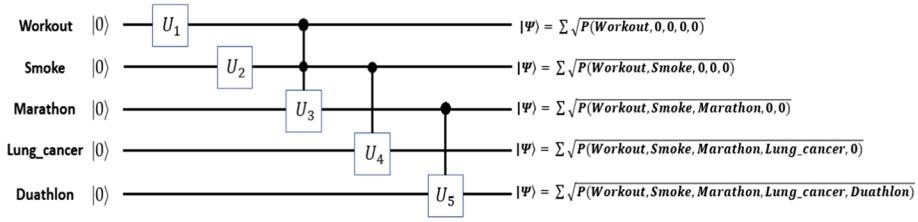


Fig. 2 Encoding circuit

Therefore, any algorithm that computes the best decision by Eqs. (6) and (7) has the potential to be applied to all other instances derived from the principal one, as a consequence of having the same computational pattern.

3 Quantum Bayesian Inference

In brief, a quantum algorithm can be regarded as a targeted manipulation of a quantum state (realised as an assembly of *qubits*) with a subsequent measurement to retrieve relevant information. States are represented by column vectors of complex numbers whose sum of moduli squared is 1, often represented in the so-called Dirac notation (Nielsen and Chuang 2010) as $|\Psi\rangle$. Typically, they correspond to linear combinations of basis states affected by complex coeficients, as in, for example, Eq. (9). The dynamics of a quantum system is represented by (the multiplication of the quantum state by) unitary matrices and is therefore reversible in time, as long as no measurement is involved. The reversal corresponds simply to a composition with the adjoint of the unitary matrix that represents forward evolution. Such an evolution can be expressed as a sequence of only a few elementary transformations represented as *quantum gates*, which only act on one or two qubits at a time. Therefore, quantum algorithms are widely formulated as *circuits* built of these elementary gates, as depicted, for example, in Fig 2.

A quantum algorithm for inference on Bayesian networks was introduced by Low et al. (2014), which, as mentioned above, is based on an improved quantum version of the Rejection Sampling algorithm. This algorithm is able to generate samples quadratically faster than the classical version, provided that the network is not too densely connected. The algorithm is divided into 3 stages, detailed in the sequel.

In the first stage, the Bayesian network is encoded into a quantum state. For this, a binary variable can be represented by a single qubit and the probabilities are mapped to the coefficients of the quantum state:

$$|\Psi\rangle = \alpha|Var_1 = true\rangle + \beta|Var_1 = false\rangle \Leftrightarrow |\Psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{9}$$

with the corresponding density matrix (Barnett 2009),

$$\rho_\Psi = |\Psi\rangle\langle\Psi| = \begin{pmatrix} \alpha^2 & \alpha * \beta \\ \alpha * \beta & \beta^2 \end{pmatrix} \tag{10}$$

Whenever two variables share an edge in the network they are related, and therefore not independent from each other, such a relationship is expressed through state entanglement. Entanglement represents a strong correlation between quantum states, therefore expressing shared information between different elements. The envisaged state is achieved though the application of a specific sort of gates—controlled rotations—to the state qubits. The fact that a rotation is controlled by another qubit permits the creation of entanglement between them. The amplitude of the rotation defines the value of the coefficients. For instance, a circuit that encodes the Bayesian network in Fig. 1 is represented in Fig 2.

The whole quantum state is equivalent to a superposition of all on entries of the original joint probability distribution table:

$$\begin{aligned}|\Psi'\rangle = &\gamma_1^2|Var_1 = true, Var_2 = true, \ldots\rangle + \gamma_2^2|Var_1 = true, Var_2 = false, \ldots\rangle \\ &+ \gamma_3^2|Var_1 = false, Var_2 = true, ..\rangle + \gamma_4^2|Var_1 = false, Var_2 = false, \ldots\rangle \\ &+ \cdots\end{aligned} \tag{11}$$

Afterward, a measurement[1] to this state produces a sample, as in the Rejection Sampling algorithm, as the probability of each sample is the same as the one in the distribution:

$$\begin{aligned}P(Var_1 =&true, Var_2 = true, \ldots) = Tr(P_0 * \rho_{\Psi'}) \\ =&\begin{pmatrix}1 & 0 & \cdots & 0\\ 0 & 0 & \cdots & 0\\ \vdots & \vdots & \ddots & \cdots\\ 0 & 0 & \cdots & 0\end{pmatrix} * \begin{pmatrix}\gamma_1^2 & \cdots & \cdots & \cdots\\ \cdots & \gamma_2^2 & \cdots & \cdots\\ \cdots & \cdots & \ddots & \cdots\\ \cdots & \cdots & \cdots & \gamma_n^2\end{pmatrix} = \gamma_1^2 * 1 + \gamma_2^2 * 0 + \cdots + \gamma_n^2 * 0 = \gamma_1^2\end{aligned}$$

At this point, a quantum analog to Rejection Sampling is created. However, it is not an efficient way to do inference because every time we measure the state it collapses, and it is necessary to reconstruct the state entailing the need for a subsequent reconstruction.

In a second stage, the *Amplitude Amplification* algorithm (Brassard et al. 2000) is applied to amplify the states that have the right values for the evidence variable. It allows for a square root speed up in search problems, a fact that explains its relevance and ubiquity to many quantum programs. In our case, the quantum state that encodes the Bayesian network is divided into two orthogonal states, one where the evidence variables have the right value and another state where they lack it:

$$\begin{aligned}|\Psi_{init}\rangle = &\sqrt{P(e)}|Var_1, Var_2, \ldots, evidences\rangle \\ &+ \sqrt{1 - P(e)}|Var_1, Var_2, \ldots, \neg evidences\rangle\end{aligned} \tag{13}$$

Next, the amplitude amplification algorithm is applied to search for the state that has the right values for the evidence variables (Brassard et al. 2000).

[1] Notice, that the use of the density matrix notation with projectors is equivalent to the description of the measurement in the Dirac notation:

$$\langle Var_1 = true, Var_2 = true, \ldots|\Psi'\rangle \equiv Tr(P_0 * \rho_{\Psi'}) \tag{12}$$

The matrix density notation was not required, as we are not dealing with mixed quantum states. However, it helped, later on, to expose the main ideas more clearly.

Table 1 Classical versus quantum complexity

Process type	Complexity
Classical	$O(n * m * P(e)^{-}1)$
Quantum	$O(n * 2^m * P(e)^{\frac{-1}{2}})$

$$Q^k * |\Psi_{init}\rangle = \cos\left(\frac{2k+1}{2} * \theta\right)|Var_1, Var_2, \ldots, evidences\rangle + \sin\left(\frac{2k+1}{2} * \theta\right)|Var_1, Var_2, \ldots, \neg evidences\rangle$$

where Q represent the operator that amplifies the selected elements, k the number of iterations, and θ the initial probabilities,

$$\theta = 2sin^{-1}(\sqrt{P(e)}) \quad \wedge \quad \theta = 2cos^{-1}(\sqrt{1 - P(e)}) \tag{14}$$

then for the right number of iteration (k'), the final quantum state approximates with great probability to the pretended state,

$$Q^{k'} * |\Psi_{init}\rangle = |\Psi_{final}\rangle \approx |Var_1, Var_2, \ldots, evidences\rangle \tag{15}$$

The last stage amounts simply to observe this state and use the result as a sample. In Table 1 we can see the comparison between the classical and the quantum versions. The latter exhibits a quadratic speed up but only if the Bayesian network is not too densely connected, meaning that m the number of edges between the nodes (n) is not to large. Otherwise the price of encoding it to a quantum state will be too high, as the corresponding term grows exponentially.

4 Quantum Decision-Making

Clearly, a quantum computer could be used to work out the conditional probabilities with a quadratic speed up for decision problems, according to Eq. (16).

$$EU(a|e) = \sum_r \underbrace{P(Result = r|a, e)}_{Quantum} * \underbrace{U(r)}_{Classical} \tag{16}$$

In this section, however, we would like to propose a different approach which, in principle, will increase the advantage of having the quantum resources. The idea is quite simple: Instead of sampling the conditional probabilities, the quantum state remains unobserved until the utility function is applied. The intention is to apply a transformation to the outcome variable and look to what happens to the action variable. As both the outcome and the action variables are entangled, a transformation applied to the former will produce an effect on the latter.

The new algorithm modifies the process described in the previous section to infer a conditional probability by preventing the action variable to be used as an evidence variable. Thus, after an application of the amplitude amplification algorithm and tracing out the non-evidence variables (NE), as nothing happens to them during the proposed process, we have,

$$tr_{NE}(Q_{Search_1}|\Psi_{init}\rangle) = |\Psi_{A,R}\rangle = \gamma_{a,r}|a, r, evidences\rangle + \gamma_{a,\neg r}|a, \neg r, evidences\rangle + \gamma_{\neg a,r}|\neg a, r, evidences\rangle + \gamma_{\neg a,\neg r}|\neg a, \neg r, evidences\rangle \tag{17}$$

or, equivalently,

$$|\Psi_{A,R}\rangle = \begin{pmatrix} \gamma_{a,r} \\ \gamma_{a,\neg r} \\ \gamma_{\neg a,r} \\ \gamma_{\neg a,\neg r} \end{pmatrix} \tag{18}$$

describing only the states of the featured variables. The utility function $U(r)$ is then applied to this state ($|\Psi_{A,R}\rangle$). Therefore, a quantum state $|\Psi_U\rangle$ isomorphic to the utility function will be created:

$$U(R) = \begin{cases} U(r) \\ U(\neg r) \end{cases} \rightarrow |\Psi_U\rangle = \begin{pmatrix} \frac{\sqrt{U(r)}}{n} \\ \frac{\sqrt{U(\neg r)}}{n} \end{pmatrix}$$

where n is a normalization term such that $\langle\Psi_{A,R}|\Psi_{A,R}\rangle$ sums up to 1. Also, by creating both states in memory, the whole product state $|\Psi_{A,R}\rangle \otimes |\Psi_U\rangle$ becomes,

$$|\Psi_{Syt}\rangle = |\Psi_{A,R}\rangle \otimes |\Psi_U\rangle = \begin{pmatrix} \gamma_{a,r} \\ \gamma_{a,\neg r} \\ \gamma_{\neg a,r} \\ \gamma_{\neg a,\neg r} \end{pmatrix} \otimes \begin{pmatrix} \frac{\sqrt{U(r)}}{n} \\ \frac{\sqrt{U(\neg r)}}{n} \end{pmatrix} = \begin{pmatrix} \gamma_{a,r} * \frac{\sqrt{U(r)}}{n} \\ \gamma_{a,r} * \frac{\sqrt{U(\neg r)}}{n} \\ \gamma_{a,\neg r} * \frac{\sqrt{U(r)}}{n} \\ \gamma_{a,\neg r} * \frac{\sqrt{U(\neg r)}}{n} \\ \gamma_{\neg a,r} * \frac{\sqrt{U(r)}}{n} \\ \gamma_{\neg a,r} * \frac{\sqrt{U(\neg r)}}{n} \\ \gamma_{\neg a,\neg r} * \frac{\sqrt{U(r)}}{n} \\ \gamma_{\neg a,\neg r} * \frac{\sqrt{U(\neg r)}}{n} \end{pmatrix} \tag{19}$$

Then, this state $|\Psi_{Syt}\rangle$ already contains the relevant terms where the Utility function is applied to the correct bases. All one has to do is to amplify them resorting again to amplitude amplification algorithm. For the example at hands, such is the case when $r \wedge r$ and $\neg r \wedge \neg r$ hold, yielding,

$$Q_{Search_2}|\Psi_{Syt}\rangle \approx \begin{pmatrix} \frac{\gamma_{a,r} * \sqrt{U(r)}}{n'} \\ 0 \\ 0 \\ \frac{\gamma_{a,\neg r} * \sqrt{U(\neg r)}}{n'} \\ \frac{\gamma_{\neg a,r} * \sqrt{U(r)}}{n'} \\ 0 \\ 0 \\ \frac{\gamma_{\neg a,\neg r} * \sqrt{U(\neg r)}}{n'} \end{pmatrix} \tag{20}$$

At this moment the amplitudes of the action variable hold the solution to the decision problem. To understand how the Results variable and the Utility function are traced out as follows,

$$\begin{aligned} \rho_A &= tr_{R,U}(\rho_{Syt}) \\ &= \begin{pmatrix} \left(\frac{\gamma_{a,r} * \sqrt{U(r)} + \gamma_{a,\neg r} * \sqrt{U(\neg r)}}{n'}\right)^2 & \cdots \\ \cdots & \left(\frac{\gamma_{\neg a,r} * \sqrt{U(r)} + \gamma_{\neg a,\neg r} * \sqrt{U(\neg r)}}{n'}\right)^2 \end{pmatrix} \end{aligned} \tag{21}$$

A measurement yields,

$$P(a') = tr(P_0 * \rho_{Syt}) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} * \rho_{Syt} = \left(\frac{\gamma_{a,r} * \sqrt{U(r)} + \gamma_{a,\neg r} * \sqrt{U(\neg r)}}{n'}\right)^2 \tag{22}$$

and,

$$P(\neg a') = tr(P_1 * \rho_{Syt}) = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} * \rho_{Syt} = \left(\frac{\gamma_{\neg a,r} * \sqrt{U(r)} + \gamma_{\neg a,\neg r} * \sqrt{U(\neg r)}}{n'}\right)^2 \tag{23}$$

Combining the deduced probability of the action variable with,

$$\gamma_{a,r}^2 = P(a, r, evidences) \tag{24}$$

and

$$P(r|a, evidences) = \frac{P(a, r, evidences)}{P(a, evidences)} \tag{25}$$

we conclude that

$$\gamma_{a,r}^2 \propto P(r|a, evidences) \tag{26}$$

and the transformation yields a state where

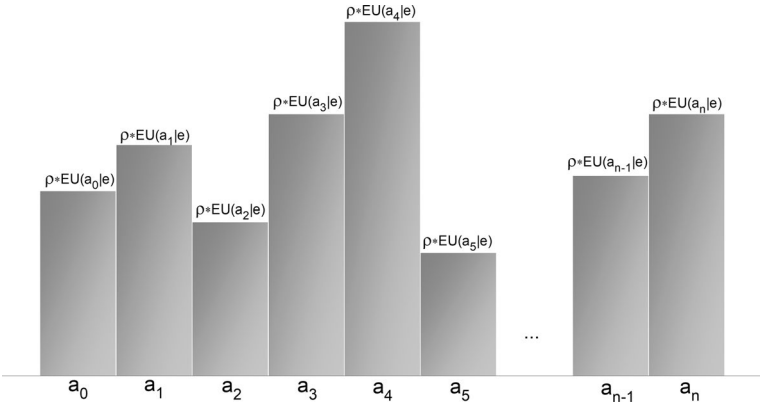


Fig. 3 Probability distribution of an action variable

$$P(a_i') = tr(P_i * \rho_{Syt}) = \frac{\gamma^2_{a_i,r_1} * U(r_1) + \gamma^2_{a_i,r_2} * U(r_2) + \cdots + \gamma^2_{a_i,r_n} * U(r_n)}{n'^2} \tag{27}$$

So, after applying there transformations the probability of some action (a_i) is proportional to its expected utility [Eq. 6],

$$P(a_i') \propto EU(a_i|e) \tag{28}$$

$$P(a_i') = \varrho_{a_i} * EU(a_i|e) \tag{29}$$

Finally, if initially the Bayesian networks respects,

$$P(a|evidences) = P(a) \tag{30}$$

then

$$P(r|a, evidences) = \frac{P(r, a|evidences)}{P(a)} \tag{31}$$

and

$$P(a_{i'}) = P(a_i), i \neq i' \tag{32}$$

The constant of proportionality takes the following value for all actions:

$$\varrho_{i'} = \varrho_i = \frac{1}{n'^2 * P(a)}, \quad i \neq i' \tag{33}$$

Thus, the values of the proportionality constants ϱ_i between all the Expected Utilities remain the same. This means that an action with a greater probability has a greater expected utility. Consequently, to choose the action with the greatest probability/utility (Fig. 3), it is enough to resort to a limited collection of samples, rather than obtaining first all the conditional probabilities. Moreover, this provides a more precise way of choosing an action because the sampling method always yields an approximation and the error affecting the computed values grows for every conditional probability determined.

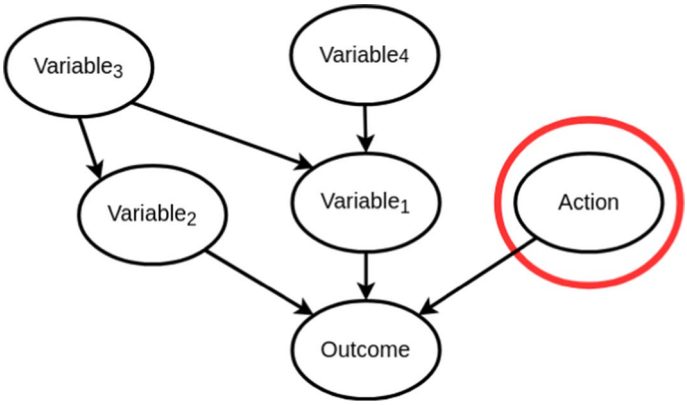


Fig. 4 Bayesian network with an independent action node

Remember, that this decision process requires that Eq. (32) has to be initially true. This expresses the rational choice which considers all actions as equal at the beginning. In other words, the intelligent agent is not biased beforehand. Additionally, the action variable should be independent of the evidence variables as in Eq. (30), which means that the topology of the network has to be as in Fig. 4. This requirement ensures that the intelligent agent is not biased by the current state of his environment and performs his decisions in order to achieve the best outcome in the future state.

5 Complexity

The purpose of this Section is to characterize computational complexity of the proposed algorithm and compare it to the solution that computes the conditional probabilities with use of the quantum inference algorithm (Eq. (16)). So, to simplify let us denote by Process A the new quantum algorithm and by Process B the second one.

Both algorithms generate samples to determine which is the best action. The number of operations (I_t) in each algorithm is defined by the number of iterations per sample (I_s) and the number of samples (S) necessary, as in Eq. (34).

$$I_t = S * I_s \tag{34}$$

5.1 Number of Iterations

The number of iterations per sample of the two processes are defined by the number of *Search* iterations that are necessary to apply in each case. Also, the number of iterations necessary to find the goal state in a quantum search is defined by the probability of this state:

$$I_s = \sqrt{\frac{1}{P(state)}} \tag{35}$$

For Process A, this is the probability of the state which has already the utility function applied to it,

$$P(state) = \sum_r U(r) * P(r, e) \tag{36}$$

knowing that,

$$1 = \sum_r U(r) \tag{37}$$

Assuming that any distribution is possible for U(r) and P(r,e), we conclude that the probability can take any value between 0 and P(e). We also know that the mean value for $U(r)$ is:

$$\frac{1}{N_r} \tag{38}$$

where N_r represents the dimension of the outcome variable. Thus, P(e,r) can be described as:

$$\frac{P(e)}{N_r} \tag{39}$$

So the mean value for the product of the two values $U(r) * P(e, r)$ will be,

$$\frac{P(e)}{N_r} * \frac{1}{N_r} = \frac{P(e)}{N_r^2} \tag{40}$$

if they are independent, which is the case because the utility function is independent of the information present in the Bayesian Network. The mean value for the sum can be computed by the sum over the mean terms

$$Mean(P(state)) = \sum_r \frac{P(e)}{N_r^2} = \frac{P(e) * N_r}{N_r^2} = \frac{P(e)}{N_r} \tag{41}$$

This mean value for the probability will be used to define the number of steps:

$$I_s = \sqrt{\frac{N_r}{P(e)}} \tag{42}$$

defining in this way the number of iterations necessary to obtain a sample with Process A.

The same has to be done for Process B, where the probabilty of the goal state is

$$P(state) = P(e, a) \tag{43}$$

In this case, we have to apply the requirements determined by Process A described in (30) and (32) in order to make a correct comparison at a later stage,

$$P(e, a) = P(e) * P(a) = \frac{P(e)}{N_a} \tag{44}$$

where N_a is the dimension of the action variable. Finally, we estimate the number of iterations as

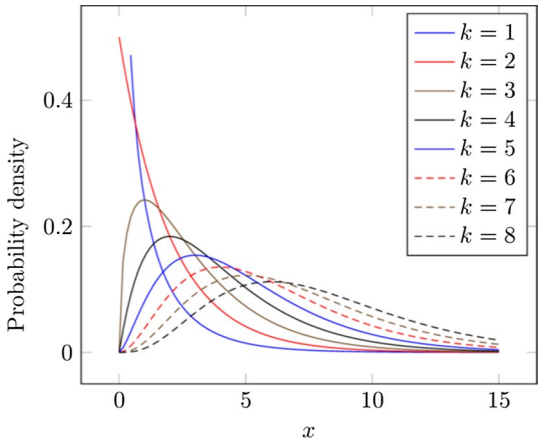


Fig. 5 Chi-square distributions with different degrees of freedom (k)

$$I_s = \sqrt{\frac{N_a}{P(e)}} \tag{45}$$

5.1.1 Number of Samples

The next step is to obtain the number of samples necessary for each process. Recall that the simultaneous error terms for a *Multinomial Distribution* are:

$$(pi - \pi_i)^2 = \frac{A * \pi_i(1 - \pi_i)}{N}, \quad (i = 1, 2, \ldots, k) \tag{46}$$

The value A represents the upper $\alpha * 100$th percentile of a *Chi-Square Distribution* (Fig. 5) with k-1 degrees of freedom, π_i represent the probability of category i, N is the number of samples, and the difference on the left side of the equation represents the error term (Goodman 1965).

Writing the same equation as a function of N, yields

$$N = \frac{A * \pi_i(1 - \pi_i)}{\delta^2}, (i = 1, 2, \ldots, k) \tag{47}$$

Equation (47), defines the number of samples necessary for Process A and Process B, since, both processes are sampling from a quantum state with multiple bases.

5.1.2 Total Number of Operations

The total number of operations is characterized by the product of the terms deduced in the previous sections and the number of operations necessary to encode the network as a quantum state. Additionally, Process B requires at least $2(N_a + N_r)$ operations to apply the Utility function and sum the respective terms for the expected utilities. Finally, the total number of operations for each process to solve the decision problem is shown in Table 2.

When the decision problem is totally defined all that it requires is to plug the number in the equation and look which process performs better. However, a comparison was

Table 2 Mean number of operations for each process

Process A	Process B
$n * 2^m * \sqrt{\frac{N_r}{P(e)}} * \frac{A*\pi_i(1-\pi_i)}{\delta_a^2}$	$n * 2^m * \sqrt{\frac{N_a}{P(e)}} * \frac{A*\pi_i(1-\pi_i)}{\delta_c^2} * N_a + 2(N_a + N_r)$

performed (as described in Appendix) and the relation between the computational effort is asymptotically over the term,

$$\frac{ProcessB}{ProcessA} \geq \sqrt{\frac{N_r}{N_a}} \tag{48}$$

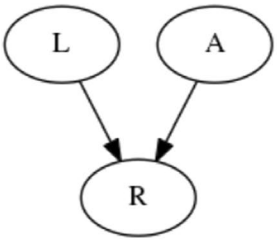
This result shows that Process A is faster when the outcome variable has a greater dimension than the action variable. This is a quite normal scenario in real applications because the number of states in which an agent can transit is tremendously smaller than the possible states that his environment can evolve. Also, for a fixed number of action this process allows the agent to explore quadratically more outcomes with the same computational effort, making him a *wiser* decision maker.

Finally, this process can also be compared with a quantum version of *decision networks* (Russel and Norvig 2010), as discussed in Oliveira (2019). The results show that again the best process depends on the characteristics of the problem. Although, it is important to mention that process A solves a decision process that wants to sample an action from a distribution based on the expected utilities in a extremely efficient way with only *one sample*, this kind of decision processes could be used and studied for applications in reinforcement learning.

6 Proof-of-Concept Implementation

The algorithm presented in Sect. 4 was implemented on the IBM Q quantum simulator as a proof-of-concept. At our disposal was the IBM 20-qubit machine, which is based on superconducting circuits (Steffen 2011). This machine specifies an error term associated with each gate used in a quantum circuit and a life-time for each qubit. So, as the number of gates grows the error of the outcome grows as well. The output of a circuit with a considerable number of gates would be majorly noise. Thus, the decision processes presented before, which is based on a search problem, would be impossible to compute with a manageable error term.

IBM's best quantum computer is not the only that fails to solve such problems. The best quantum devices, in the world, are not even near to solve problems related to search problems with a higher dimension. However, that does not mean that the current devices are completely useless. There are problems where a *Noisy Intermediate-Scale Quantum* (NISQ) devices may have an impact, in the near future (Preskill 2018). The applications



L	P(L)
T	2/3
F	1/3

A	P(A)
T	1/2
F	1/2

L	A	R	P(R| L,A)
F	T	T	1/5
F	T	F	4/5
F	F	T	3/5
F	F	F	2/5
T	T	T	4/7
T	T	F	3/7
T	F	T	1/8
T	F	F	7/8

Fig. 6 Bayesian network over 3 variables. Node L represents the evidence variable, A the action variable and R the outcome variable

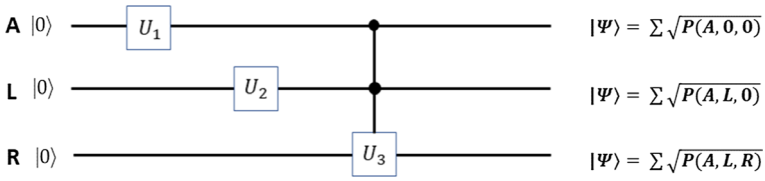


Fig. 7 Quantum circuit composed by rotations and controlled-rotations

of these NISQ devices are related to simulations in chemistry and many-body quantum physics.[2]

Over the last years, quantum devices have had a lot of progress. For example, the number of qubits are smoothly increasing, the gate errors are reducing (Schäfer et al. 2018) and entanglement between them is becoming stronger (Kues et al. 2017; Pirandola et al. 2006). This progress has been giving hope to construct a powerful universal quantum computer, which one day may have a great impact on our everyday life. But to validate results as pretended, in this section a classical simulator has to be used. However, the same simulator struggles to compute the outcomes, if the number of qubits used increases. As mentioned before the complexity to simulate a quantum computer on a classical computer is too high. Given that, a very simple Bayesian network (Fig. 6) was selected for the decision process.

The network was encoded to a quantum state with use of the technique presented in Low et al. (2014), producing the circuit shown in Fig. 7.

A value for the evidence variable L was selected ($L = False$) and the following utility function,

$$U(R) = \begin{cases} 7, & R = r \\ 3, & R = \neg r \end{cases} \tag{49}$$

was applied with use of the proposed algorithm. However, to compute the algorithm on IBM's quantum simulator, each part has be to converted in a concrete quantum circuit (Fig. 8). Every state has to be encoded, which in theory is simple with the use of rotations and controlled-rotations. The major difficulty exists when the rotation is controlled

[2] It is interesting to mention that the major companies investing in quantum computing are constructing devices based on different technologies. Microsoft devices are based on topological quantum computing (Nayak et al. 2008), while Intel is exploring spin qubits (Vandersypen et al. 2017).

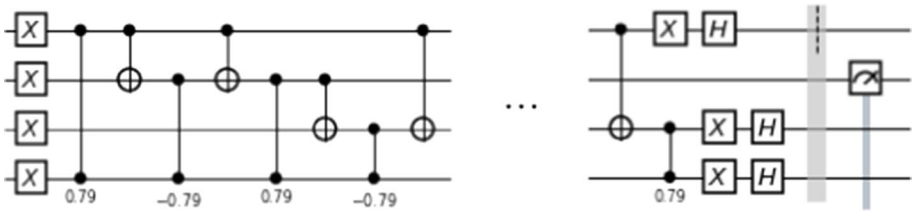


Fig. 8 Circuit representation of the amplitude amplification algorithm for the decision-making process

Table 3 Comparison between theoretical and experimental results after sampling from the constructed state

States	Theoretically expected probability	Percentage of Samples
$Action_0$	0.58	0.544
$Action_1$	0.42	0.456

by more than one qubit. In such cases, this operation has to be decomposed to simpler and available operations in the working framework. The existence of an equivalent circuit is guaranteed by the fact that the simulator is a universal quantum computer, meaning that it may perform any possible computation. In practice, there are tools to decompose complex operations into simpler ones (Vartiainen et al. 2004; Möttönen et al. 2004).

Finally, these circuits have to be built on "Qiskit" which runs on a "jupyter notebook" (Github link "Quantum Bayesian Decisions"). The circuits created are sent as a job to IBM's servers and the results are sent back to the client. For this example, a significant number of samples was generated. The number of samples for each state of the action variable must be similar to the theoretical probability. Table 3 shows that this is indeed the case: the experimental result is quite similar to the one foreseen by theory.

The small discrepancies pointed out in Table 3 can be explained by deficiencies of the implementation. First, note that the amplitude amplification algorithm is probabilistic, i.e. the result is never entirely precise. On the other hand, the number of iterations in the amplitude amplification algorithm was an integer number; thus, if $n.m$ non-integer iterations are required the usage of the value bellow n or the one above $n + 1$, generates a small variation. Actually, it is possible to perform a a quantum search with a non-integer number of iterations (Zekrifa et al. 2000), but this would not add relevance to this results.

7 Conclusions and Future Work

A quantum algorithm which solves the generic decision-making problem was presented. It is a very curious solution for couple of reasons. First, it has a proven computational advantage over the classical and the semi-classical solutions when the parameters are in the correct relation. Secondly, it samples from a very particular probability distribution, which classically would require an tremendous amount of computational work to recreate. Moreover, it benefits from the structure of the data, which no classical algorithm could benefit from, making it a very interesting example to illustrating the differences between classical and quantum computations.

To support the theoretical work described the algorithm was implemented in IBM's quantum simulator as a proof-of-concept. The results were as in correspondence with what theory anticipated for the example chosen, further confirming the ideas presented.

As an extension to this work we propose a search for decision problems that take advantage by sampling from the probability distribution created by this solution. Also, the decision-making process discussed here was related to a static model, in which, neither the utility function nor the Bayesian network change in time. It would be of interest to verify if the decision-making process could benefit from an additional learning process (Jonsson and Barto 2007; Robinson and Hartemink 2010; Tong and Koller 2001). Enabling an agent to adapt its behaviour to a changing environment, would probably result in better outcomes, raising the number of possible applications.

Appendix

A Complexity Comparison

The decision-making processes we aim at comparing require inequality (50) to be satisfied. It assures that the decision maker chooses with certainty the best action.

$$\forall_{n\setminus\{max\}} EU(action_{max}) - EU(action_n) > \delta_{action_{max}} + \delta_{action_n} \tag{50}$$

Thus, to compare Process A and Process B it is necessary to consider all terms that are different. Therefore, the error term δ_a for Process A is related to directly sampling values for the expected utilities, while in Process B the expected utility is determined indirectly. For this reason, in Process B it is necessary to apply error propagation rules:

$$EU(a|e) + \delta_{EU(a|e)} = \sum_R (P(Result = r|a, e) + \delta_b) * U(r) \tag{51}$$

Before applying error propagation to this equation, we need to normalize it so that $EU(a|e)/k$ is equal to $P(a)$.

$$P(a) + \delta_a = \sum_R (P(Result = r|a, e) + \delta_b) * F(r) \tag{52}$$

where the normalization function ($F(r)$) is expressed as,

$$F(r) = \frac{U(r)}{\sum_a \sum_r U(r) * P(r|a, e)} \tag{53}$$

Here, again, the mean value of $U(r)$ is used:

$$F(r) = \frac{U(r)}{\sum_a \sum_r P(r|a, e) * U(r)} = \frac{U(r)}{\sum_a \frac{1}{N_r}} = \frac{U(r)}{\frac{N_a}{N_r}} = \frac{N_r * U(r)}{N_a} \tag{54}$$

$$F(r) = \frac{1}{N_a} \tag{55}$$

Expressing the equation that determines the error term δ_a as a function of the error term δ_b yields

$$\delta_a = \sqrt{\sum_R \delta_b^2 * F(r)^2} \tag{56}$$

Using Eq. (55) we obtain:

$$\delta_a = \sqrt{\sum_R \delta_b^2 * (\frac{1}{N_a})^2} \tag{57}$$

Then, assuming that δ_b is similar, which is in favor of Process B because it minimizes the δ_a term:

$$\delta_a = \sqrt{N_r * \delta_b^2 * (\frac{1}{N_a})^2} \tag{58}$$

yielding,

$$\delta_a = (\frac{\sqrt{N_r}}{N_a}) * \delta_b \tag{59}$$

With the relation between the error terms determined, it is possible to compare the difference of the computational effort involved in both processes, assuming again the mean terms for the probabilities:

$$\sqrt{\frac{N_a}{N_r}} * \frac{A_{r,\alpha} * \frac{1}{N_r} * (1 - \frac{1}{N_r}) * \delta_a^2 * N_a}{A_{a,\alpha} * \frac{1}{N_a} * (1 - \frac{1}{N_a}) * \delta_b^2} + \frac{2 * N_a * N_r}{n * 2^m * \sqrt{\frac{N_r}{P(e)}} * \frac{A_{a,\alpha} * \frac{1}{N_a} * (1 - \frac{1}{N_a})}{\delta_a^2}} \tag{60}$$

Let us call the term on the right,

$$t_1 = \frac{2 * N_a * N_r}{n * 2^m * \sqrt{\frac{N_r}{P(e)}} * \frac{A_{a,\alpha} * \frac{1}{N_a} * (1 - \frac{1}{N_a})}{\delta_a^2}} \tag{61}$$

Using 59,

$$\sqrt{\frac{N_a}{N_r}} * \frac{N_r}{N_a} * \frac{A_{r,\alpha} * \frac{1}{N_r} * (1 - \frac{1}{N_r})}{A_{a,\alpha} * \frac{1}{N_a} * (1 - \frac{1}{N_a})} + t_1 \tag{62}$$

also,

$$\sqrt{\frac{N_r}{N_a}} * \frac{A_{r,\alpha} * \frac{1}{N_r} * (1 - \frac{1}{N_r})}{A_{a,\alpha} * \frac{1}{N_a} * (1 - \frac{1}{N_a})} + t_1 \tag{63}$$

and,

$$\sqrt{\frac{N_r}{N_a}} * \frac{A_{r,\alpha} * (\frac{1}{N_r} - \frac{1}{N_r^2})}{A_{a,\alpha} * (\frac{1}{N_a} - \frac{1}{N_a^2})} + t_1 \tag{64}$$

From Inglot (2010) we obtain a lower bound for $A_{\alpha,k}$. Although these terms are different for distinct values of α, we consider the one where α is not leaning to zero too fast. Thus,

$$A_{\alpha,k} \geq k + 2 * \log \frac{1}{\alpha} - \frac{5}{2} \tag{65}$$

With this equation it is possible to define a better value for the difference between the computational efforts,

$$\sqrt{\frac{N_r}{N_a}} * \frac{(N_r + 2 * \log \frac{1}{\alpha} - \frac{7}{2}) * (\frac{1}{N_r} - \frac{1}{N_r^2})}{(N_a + 2 * \log \frac{1}{\alpha} - \frac{7}{2}) * (\frac{1}{N_a} - \frac{1}{N_a^2})} + t_1 \tag{66}$$

As

$$\lim_{N_r \to \infty} (N_r + 2 * \log \frac{1}{\alpha} - \frac{7}{2}) * (\frac{1}{N_r} - \frac{1}{N_r^2}) = 1 \tag{67}$$

and,

$$\lim_{N_a \to \infty} (N_a + 2 * \log \frac{1}{\alpha} - \frac{7}{2}) * (\frac{1}{N_a} - \frac{1}{N_a^2}) = 1 \tag{68}$$

it is possible to approximate the expression to

$$\sqrt{\frac{N_r}{N_a}} * \frac{(N_r + 2 * \log \frac{1}{\alpha} - \frac{7}{2}) * (\frac{1}{N_r} - \frac{1}{N_r^2})}{(N_a + 2 * \log \frac{1}{\alpha} - \frac{7}{2}) * (\frac{1}{N_a} - \frac{1}{N_a^2})} + t_1 \approx \sqrt{\frac{N_r}{N_a}} + \frac{2 * N_a * \sqrt{N_r * P(e)} * \delta_a^2}{n * 2^m} \tag{69}$$

Writing the term of δ_a as a function of its dimension and a factor that adjusts the precision,

$$\delta_a = \frac{1}{c * N_a} \tag{70}$$

we obtain,

$$\sqrt{\frac{N_r}{N_a}} + \frac{2 * \sqrt{N_r * P(e)}}{N_a * c^2 * n * 2^m} \tag{71}$$

Rewriting this the expression as,

$$\sqrt{\frac{N_r}{N_a}} * (1 + \frac{2 * \sqrt{P(e)}}{\sqrt{N_a} * c^2 * n * 2^m}) \tag{72}$$

Because,

$$\frac{2 * \sqrt{P(e)}}{\sqrt{N_a} * c^2 * n * 2^m} \geq 0 \tag{73}$$

for any value of the composing variables, then,

$$\sqrt{\frac{N_r}{N_a}} * (1 + \frac{2 * \sqrt{P(e)}}{\sqrt{N_a} * c^2 * n * 2^m}) \geq \sqrt{\frac{N_r}{N_a}}. \tag{74}$$

we prove that the relation between Process A and B is under bounded by,

$$\sqrt{\frac{N_r}{N_a}} \tag{75}$$

References

Al-Jarrah, O.Y., Yoo, P.D., Muhaidat, S., Karagiannidis, G.K., & Taha, K. (2015). Efficient machine learning for big data: review. CoRR abs/1503.0, arXiv:1503.05296

Barnett, S. (2009). *Quantum Information*. USA: Oxford University Press Inc.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2016). Quantum machine learning. *Nature,549,*. https://doi.org/10.1038/nature23474.

Brassard, G., Hoyer, P., Mosca, M., & Tapp, A. (2000). Quantum amplitude amplification and estimation. arXiv: quant–ph/0005055

Darwiche, A. (2008). Chapter 11 Bayesian networks. *Foundations of Artificial Intelligence, 3*(07), 467–509. https://doi.org/10.1016/S1574-6526(07)03011-8.

Giovannetti, V., Lloyd, S., & Maccone, L. (2008). Quantum random access memory. *Physical Review Letters, 100,* 160501. https://doi.org/10.1103/PhysRevLett.100.160501.

Goodman, L. A. (1965). On simultaneous confidence intervals for multinomial proportions. *Technometrics, 7*(2), 247–254. https://doi.org/10.1080/00401706.1965.10490252.

Harrow, A. W., Hassidim, A., & Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical Review Letters, 103*(15), 150502. https://doi.org/10.1103/PhysRevLett.103.150502.

Inglot, T. (2010). Inequalities for quantiles of the chi-square distribution. *Probability and Mathematical Statistics,30*.

Jonsson, A., Barto, A. (2007). Active learning of dynamic bayesian networks in markov decision processes. In: Proceedings of the 7th international conference on abstraction, reformulation, and approximation. pp. 273–284. SARA'07, Springer-Verlag, Berlin, Heidelberg, http://dl.acm.org/citation.cfm?id=1770681.1770705.

Kues, M., Reimer, C., Roztocki, P., Cortés, L. R., Sciara, S., Wetzel, B., et al. (2017). On-chip generation of high-dimensional entangled quantum states and their coherent control. *Nature, 546,* 622. https://doi.org/10.1038/nature22986.

Li, C., Welling, M., Zhu, J., Zhang, B. (2018). Graphical generative adversarial networks. CoRR abs/1804.0, arXiv:1804.03429.

Low, G. .H., Yoder, T. J., & Chuang, I. L. (2014). Quantum inference on Bayesian networks. *Physical Review A, 89*(6), 62315. https://doi.org/10.1103/PhysRevA.89.062315.

Mansinghka, V.K. (2009). Natively probabilistic computation. Ph.D. thesis, Massachusetts Institute of Technology.

Möttönen, M., Vartiainen, J. J., Bergholm, V., & Salomaa, M. M. (2004). Quantum circuits for general multiqubit gates. *Physical Review Letters, 93*(13), 130502. https://doi.org/10.1103/PhysRevLett.93.130502.

Nayak, C., Simon, S. H., Stern, A., Freedman, M., & Das Sarma, S. (2008). Non-Abelian anyons and topological quantum computation. *Reviews of Modern Physics, 80*(3), 1083–1159. https://doi.org/10.1103/RevModPhys.80.1083.

Neal, R. M. (1996). *Bayesian learning for neural networks*. Berlin, Heidelberg: Springer-Verlag.

Nielsen, M. A., & Chuang, I. L. (2010). *Quantum Computation and Quantum Information (10th Anniversary Edition)*. Cambridge: Cambridge University Press.

Oliveira, M. (2019). *On Quantum Bayesian Networks*. Physical engineering: University of Minho.

Ozols, M., Roetteler, M., & Roland, J. (2013). Quantum rejection sampling. In Innovations in theoretical computer science 2012, Cambridge, MA, USA, January 8–10, 2012. vol. 5, p. 33. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2493252.2493256.

Pirandola, S., Vitali, D., Tombesi, P., & Lloyd, S. (2006). Macroscopic Entanglement by Entanglement Swapping. *Physical Review Letters, 97*(15), 150403. https://doi.org/10.1103/PhysRevLett.97.150403.

Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. *Quantum, 2,* 79. https://doi.org/10.22331/q-2018-08-06-79.

Robinson, J.W., & Hartemink, A.J. (2010). Learning Non-Stationary Dynamic Bayesian Networks. *Journal of Machine Learning Research11*, 3647–3680, http://dl.acm.org/citation.cfm?id=1756006.1953047.

Russel, S., & Norvig, P. (2010). Artificial intelligence: a modern approach (3 rd ed.). Upper Saddle River, NJ : Prentice Hall. https://www.bibsonomy.org/bibtex/20533b732950d1c5ab4ac12d4f32fe637/mialhoma.

Sakkaris, P. (2016). QuDot Nets: quantum computers and bayesian networks. arXiv e-prints. arXiv:1607.07887.

Schäfer, V. M., Ballance, C. J., Thirumalai, K., Stephenson, L. J., Ballance, T. G., Steane, A. M., & Lucas, D. M. (2018). Fast quantum logic gates with trapped-ion qubits. *Nature, 555,* 75. https://doi.org/10.1038/nature25737.

Steffen, M., DiVincenzo, D. .P., Chow, J. .M., Theis, T. .N., & Ketchen, M. .B. (2011). Quantum computing: An IBM perspective. *IBM Journal of Research and Development, 55*(5), 13:1-13:11. https://doi.org/10.1147/JRD.2011.2165678.

Tong, S., & Koller, D. (2001). Active learning for parameter estimation in bayesian networks. In *Proceedings of 13th conference neural information processing*.

Vandersypen, L. .M. .K., Bluhm, H., Clarke, J. .S., Dzurak, A., Ishihara, R., Morello, A., et al. (2017). Interfacing spin qubits in quantum dots and donors—hot, dense, and coherent. *NPJ Quantum Information, 3,* 1–10.

Vartiainen, J. J., Möttönen, M., & Salomaa, M. M. (2004). Efficient Decomposition of Quantum Gates. *Physical Review Letters, 92*(17), 177902. https://doi.org/10.1103/PhysRevLett.92.177902.

von Neumann, J. (1951). Various techniques used in connection with random digits. In: A. S. Householder, G. E. Forsythe, & H. H. Germond (Eds.), *Monte Carlo method*, National Bureau of Standards Applied Mathematics Series (vol. 12, chap. 13, pp. 36–38). Washington, DC: US Government Printing Office.

Zekrifa, D. M. S., Hoyer, P., Mosca, M., & Tapp, A. (2000). Quantum Amplitude Amplification and Estimation. *AMS Contemporary Mathematics Series,305*. https://doi.org/10.1090/conm/305/05215.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Michael de Oliveira is a Physical Engineer (graduated from the University of Minho). He obtained his master's from the Computer Science department of the University of Minho. Currently, he is working with the Quantum and Linear-Optical Computation group at the International Iberian Nanotechnology Laboratory, as a PhD student. His mains research interests are Quantum Computation and Information, Computer Science, and Mathematical Physics. During his academic degree, he has been awarded with the distinguishing grant for "New Talents in Quantum Technologies" from the Calouste Gulbenkian Foundation.

Luis Soares Barbosa is a full professor at the Computer ScienceDepartment, Universidade do Minho, and a senior researcher at theHigh Assurance Software laboratory, HASLab INESC TEC. Recently hejoined INL, the International Iberian Nanotechnology Laboratory, tolead the new Quantum Software Engineering Group. His researchfocuses on program semantics, logics and calculi applied to rigoroussoftware analysis, design, and construction. Most of his work isframed on Coalgebra and Modal Logic. Luis has second academicaffiliation to the United Nations University, currently serving asDeputy Head of its Operational Unit on Policy-driven ElectronicGovernance, UNU-EGOV.