Universidade do Minho
Escola de Engenharia
Departamento de Informática

Michael de Oliveira

On Quantum Bayesian Networks

December 2019

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Michael de Oliveira

On Quantum Bayesian Networks

Master dissertation
Master Degree in Computer Science

Dissertation supervised by
Luis Soares Barbosa(Univ. Minho)

December 2019

ACKNOWLEDGEMENTS

In the first place, I want to thank my parents and my brother for all the unconditional support, during this long journey. My big family for all the motivation, never forgetting sentences as “Quando acabas o curso rapaz?”, ”Ainda?!”,”Mas isso serve para quê?”.

A special thanks to my supervisor, Luis Barbosa, for all the help. Distinguishing the opportunity conceded to explore this work with freedom.

To my girl Margarida for being the best distraction that I could ever have, companion of many adventures and good moments.

A special thanks to my friends Filipe, Nelson and Vitor for never leaving me on the street or in the ditch.

A big thanks to my friends from “a nossa turma” that no one knows the number of elements. However, I distinguish with ease Areias, Igor, Frango, Leo, Rafarik and Tiago(DDS).

To all my colleagues from this course, which are now friends, that survived difficult moments with me. Without them, the course would be a very difficult journey.

Last but not least, the family ”Mecputos” I left behind in my first course. The course was not what I pretended but the friends I made are not exchangeable.

ABSTRACT

As a compact representation of joint probability distributions over a dependence graph of random variables, and a tool for modeling and reasoning in the presence of uncertainty, Bayesian networks are becoming increasingly relevant both for natural and social sciences, for example, to combine domain knowledge, capture causal relationships, or learn from incomplete datasets. Known as an NP- hard problem in a classical setting, Bayesian inference pops up as a class of algorithms worth to explore in a quantum framework.

The present dissertation explores this research field and extends the previous algorithm by embedding them in decision-making processes. In this regard, several attempts were made in order to find new and enhanced ways to deal with these processes. In a first attempt, the quantum device was considered to run a subprocess of the decision-making process, resulting in a quadratic speed-up for that subprocess. Afterward, "decision-networks" were taken into account and allowed a fully quantum implementation of a decision-making process, benefiting from a quadratic speed-up during the whole process. Lastly, a solution was found. It differs from the existing ones by the judicious use of the utility function in an entangled configuration. This algorithm explores the structure of input data to efficiently compute a solution. In addition, for each one of the algorithms developed, their computational complexity was determined in order to provide the information necessary to choose the most efficient one for a concrete decision problem.

A prototype implementation in Qiskit (a Python-based program development language for the IBM Q machines) was developed as a proof-of-concept. If Qiskit offered a simulation platform for the algorithm considered in this dissertation, string diagrams provided the verification framework for algorithmic proprieties. Further, string diagrams were studied with the intention to obtain formal proofs about the algorithms developed. This framework provided relevant examples and the proof that two different implementations for the same algorithm are equivalent.

RESUMO

As redes Bayesianas têm-se tornado cada vez mais importantes no domínio das ciências naturais e sociais, na medida em que permitem inferir relações de causalidade entre variáveis e aprender através de conjuntos incompletos de dados. Trata-se de uma representação compacta de distribuição de probabilidade conjunta feita sobre um grafo que representa dependências entre variáveis. Num contexto clássico, inferência sobre estas estruturas é visto como um problema de complexidade NP destacando-se como uma das classes de algoritmos a explorar num enquadramento quântico.

Esta dissertação explora este domínio de investigação e insere as redes Bayesianas num processo de tomada de decisão. Neste sentido, foram feitas várias tentativas para se encontrarem novas e melhores formas de lidar com estes processos. Numa primeira tentativa, considerou-se que o dispositivo quântico executava um subprocesso do processo de tomada de decisão, resultando numa aceleração quadrática do mesmo. Posteriormente, foram consideradas *decision networks* que permitiram uma implementação totalmente quântica de um processo de tomada de decisão. Através desta implementação foi possível obter uma aceleração quadrática durante todo o processo. Por fim, foi encontrada uma solução viável. Difere das já existentes pelo uso criterioso da função de utilidade num estado emaranhado. Este algoritmo explora a estrutura dos dados de entrada para calcular de forma eficaz uma solução. Além disso, para cada um dos algoritmos desenvolvidos, foi determinada a respetiva complexidade computacional de modo a que fossem conhecidas todas as informações necessárias para escolher o algoritmo mais eficiente para um determinado problema de decisão.

Foi desenvolvida uma implementação inicial no Qiskit (um software que permite o desenvolvimento de programas baseados em Python para as máquinas IBM Q) como prova de conceito. Se o Qiskit ofereceu uma plataforma de simulação para o algoritmo considerado nesta dissertação, os *string diagrams* forneceram a estrutura de verificação para propriedades algorítmicas. Além disso, estes diagramas foram estudados com a intenção de se obter provas formais sobre os algoritmos desenvolvidos. Esta estrutura forneceu exemplos relevantes e a prova de que duas implementações diferentes para o mesmo algoritmo são equivalentes.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 CONTEXT

Over the past years, methods and equipment used in scientific research have changed. The technological evolution has gifted researchers in different domains with new tools. Also, solutions to common problems are shared within the scientific community, stimulating a faster progress.

One main issue that affects a variety of domains is uncertainty. So, a tool to deal with uncertainty is needed. Graphical models, in particular Bayesian networks are commonly used for that purpose. The reason for choosing Bayesian networks is related to their causal description. As Bayesian networks have acyclic edges, they describe causal relations between the elements connected Pearl (2009); Nadkarni and Shenoy (2004). Every node has its parent nodes as *causes* and children as *consequences*. For models with cyclic dependencies it is not possible to define a relation of cause and consequence between the elements.

In Biology, Bayesian networks “are becoming the machine learning method of choice” Needham et al. (2007) given their capacity to learn from incomplete data and ability to avoid overfitting. For example, in Needham et al. (2007) they are used to describe a gene regulation network. Also, in the field of medical diagnosis, they are used since the last century Kahn et al. (1997); Arroyo-Figueroa and Sucar (1999).

In Weber et al. (2012), the authors make an overview of Bayesian networks for risk analysis and maintenance areas. Resorting to a large number of references they concluded that there is an “upward trend since 2000” for these structures. The arguments presented are related to the ones presented before, distinguishing Bayesian networks from other classical models by their polyvalency.

Bayesian networks present themselves as a powerful tool to tackle uncertainty, with a great diversity of applications. Some of them are unexpected, including classification of Chilean wines and even Terrorism risk management Pourret et al. (2008).

Additionally, these networks are not limited to static environments. Dynamic Bayesian networks (DBN) solve problems that change in time and are susceptible to uncertainty. For instance, DBN are used in ”fall prediction systems” of elderly people Nicholson (1997).

Also, in particle filtering, a very big problem in statistical physics, DBN are used as the main tool Doucet et al. (2000).

1.2 MOTIVATION

Despite the variety of uses and applications of Bayesian networks. They have a fundamental problem, related to the computational effort required to perform inference over them. This means that when the problem or the Bayesian network grows in dimension, the computational effort grows as well. Unfortunately, the rate of growth is exponential meaning that simply adding some nodes to the networks can make the problem unmanageable.

Although, typical algorithms are optimized to be used on a classical machine. In the last years, quantum computers have made their debut and the research around quantum computation has increased. The idea of a quantum computer was presented for the first time by Feynman Feynman (1999). He wanted to simulate a physical system with a computer that would behave as the system to be simulated. The intention was to reduce the complexity of the simulation by exploring the nature of the computer. Feynman's idea turned out to be right and quantum computers, with the potential to speedup simulations, start to emerge.

Figure 1.: Quantum processor developed by IBM (from Ralf Krauter).

It is known that quantum computers do not have the ability to solve more problems than classical ones. However, they distinguish themselves from their classical counterparts by providing speedups for certain problems. A variety of problems are solvable faster on a quantum computer. For instance, Grover's algorithm allows us to perform a search process with a quadratic speedup Grover (1996). As search processes are found in many classical problems, Grover's algorithm would be useful for a variety of applications.

Greater speedups are found in cryptography. Shor discovered that with the use of a quantum computer the cryptographic protocols that are used nowadays are breakable in a reasonable amount of time. This happens because these protocols are based on the fact that factorizing a prime number is a computationally hard problem. In fact, it is a difficult problem for a classical computer but not for a quantum computer. The algorithm that Shor presented uses the quantum Fourier transform which has an exponential speedup with respect to the Fourier transform performed on a classical computer.

Research in quantum computing is currently divided into different branches. A large number of groups are working to build a better quantum computer. The ones available, at the moment, are limited in resources and are highly affected by errors. Consequently, the best quantum computer would not outperform the best classical computer in any useful task. Other groups are studying how to use a perfectly working quantum computer to solve problems faster than traditional computers. The work of these groups is oriented to create new quantum algorithms and find problems that could be solved faster on such a device.

In summary, Bayesian inference is an important tool for a variety of domains, making it a candidate worth exploring in a quantum setting. The aim of this dissertation is the study of quantum variants of Bayesian networks, in which classical probabilities are replaced by quantum amplitudes, and of the corresponding quantum algorithms and heuristics.

1.3 CONTRIBUTIONS

The contributions of this dissertation can be summed up as:

1. A detailed study of the quantum Bayesian inference algorithm and its application.
2. A completely new algorithm for decision-making processes.
3. A precise definition of the computational complexity of the algorithms developed.
4. The use of a categorical formalism (string diagrams) to reason about the algorithms studied and introduced in the dissertation.

Part of the work covered in Chapter 4 was accepted for WOE (Worlds of Entanglement) international workshop and presented in Santiago de Chile, 12th February 2019. A full paper is currently being prepared as an invited submission resulting from the workshop. It should also be mentioned that this work was awarded a research grant by Fundação Calouste Gulbenkian related to the "Gulbenkian Programme New Talents in Quantum Technologies", 4th November 2018.

1.4 OUTLINE

This dissertation starts with an initial background in Chapter 2. It includes a review of the literature on classical Bayesian networks. It contains the mathematical description of Bayesian networks and the algorithms that handle the information represented by them.

Chapter 3, provides a review of quantum Bayesian networks, identifying and summarizing the main approaches to model and reason about Bayesian networks in a quantum setting. A quantum algorithm that has a computational advantage for Bayesian inference is distinguished.

The algorithm presented in the previous chapter is used in Chapter 4 to solve problems with computational advantage. In concrete, quantum algorithms for decision-making are described. Additionally, the implementation of a quantum simulator is documented at the end of the Chapter.

Chapter 5, offers an analysis of the computational effort required to perform each of the algorithms presented in the previous Chapter. The exact number of iterations is defined according to the characteristics of the given Bayesian network.

A formalization of the algorithms discussed in the previous chapter is presented in Chapter 6, using string diagrams Coecke and Kissinger (2017). Different algorithms are proved to equivalent in that context.

Finally, Chapter 7 summarizes the dissertation and guidelines for future work are proposed to extend the solutions presented throughout this dissertation.

2 BACKGROUND

2.1 CLASSICAL BAYESIAN NETWORKS

2.1.1 *Definition*

Bayesian Networks are probabilistic graphical models that support knowledge over a set **X** of variables from an uncertain domain. In this graphical model, the nodes represent the variables of a domain of study, which can be discrete or continuous. The edges represent probabilistic dependencies between the variables associated with the nodes that each edge joins. In concrete, a Bayesian Network is a *Directed Acyclic Graph* (DAG). This means that there are no cyclic dependencies between the variables. In addition, each node is associated with a *Conditional Probability Table* (CPT), which maps the conditional probability from the variable of the node to his parents in the graph, given as:

$$P(X_i|Parents(X_i)) \tag{1}$$

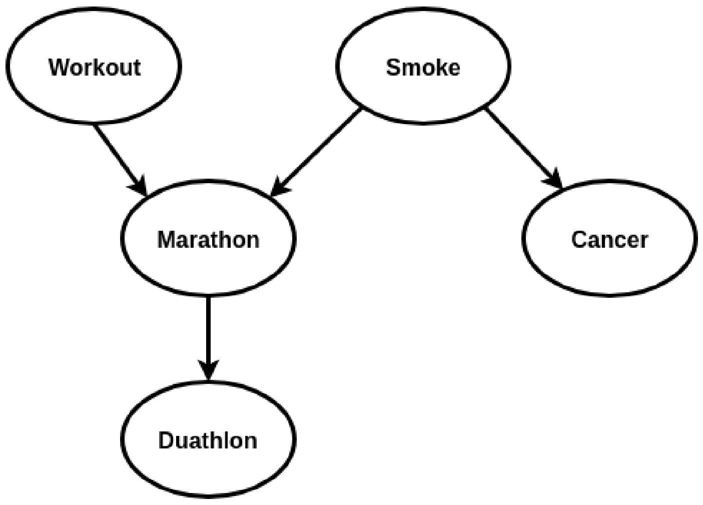
The content of CPTs are individual probabilities and probability densities functions for discrete and continuous variables, respectively Needham et al. (2007); Darwiche (2008); Networks et al. (2007).

As it can be seen in Figure 2, a Bayesian Network has to parts: the DAG which gives the observer qualitative information about the dependencies of the variables and the CPTs that provide quantitative information about these dependencies Networks et al. (2007).

This structure allows a compact representation of the *Joint Probability Distribution* (JPD) over all variables in **X**. The corresponding equation for the variables represented by the Bayesian Network, is:

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

A concrete value such as $P(X_1 = x1 \wedge ... \wedge X_n = x_n)$ is determined by the product of elements in the CPTs. These elements are selected by Equation (2) Russel and Norvig (2010).



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

Figure 2.: Bayesian Network over 5 variables. For every node there is a table that suports the conditional probalities related to its parents.

2.1.2 *Conditional independence*

The Bayesian Network structure makes it possible to extract independence statements. For example, it could be useful to ask if some variable Z is independent of Y given some other variable X. Formally, this independence would be written as $P(z, y|x) = P(z|x) * P(y|x)$, and denotes $I(Z, X, Y)$. Such statements can be derived from the DAG using a graphical method called *D-Separation* Darwiche (2008). To understand this method let us consider the 3 elementary networks.

Case 1: The chain $Z \rightarrow X \rightarrow Y$ (Figure 3). In this network, if X is known then Z is independent of Y. This independence is easy to recognize when we take an example. Let us suppose that the event Z is receiving a salary increase which influences the event X of playing in the lottery, while event Y represents winning the lottery. In this example, we can assume that the event of winning the lottery is independent of receiving a salary increase knowing that one played the lottery.

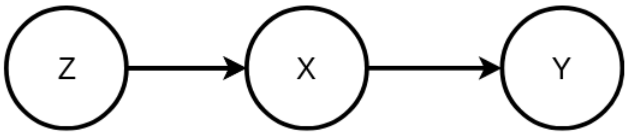


Figure 3.: Topology of a chain network.

Case 2: The second network is the fork $Z \leftarrow X \rightarrow Y$(Figure 4). In this topology we can assume that Z is independent of Y knowing X. Let us again take an example to explain that independence. Admit that X represents a raining event, Y represents wet streets and Z represents people carrying umbrellas. Therefore wet streets are independent of people caring umbrellas knowing that it rained.

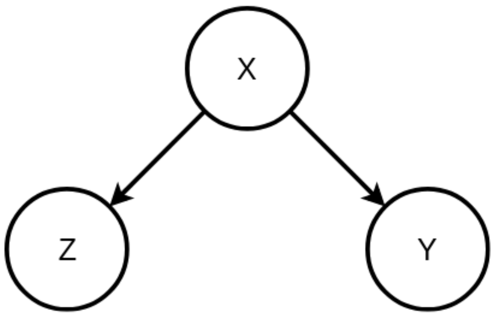


Figure 4.: Topology of a fork network.

Case 3: The last one is the collider $Z \rightarrow X \leftarrow Y$(Figure 5). This network has the opposite effect with respect to independence: Z is independent of Y only if X is unknown. There are two events that induce event X, so if we know that event X has happened, then by knowing

Z information about Y can be extracted. This relation can be determined in the CPT of node X. To answer any query of independence in the DAG, the d-separation method decomposes every path between two variables in a set of the elementary networks analyzed above.

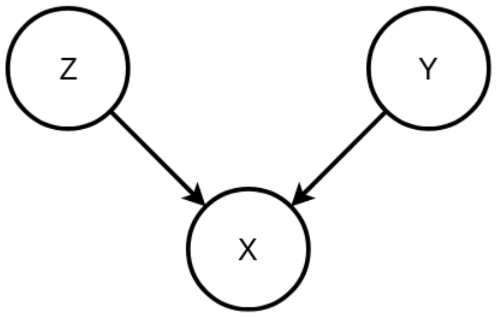


Figure 5.: Topology of a collider network.

So $I(Z, X, Y)$ is a true statement if every path between Z and Y has an elementary network, where X guarantees independence. For example, if between Z and Y there is only one path and if this path contains a chain network to which X belongs, then that statement is true. Note that if the path contains a collider network, the independence of that elementary network is proved if X is not the child node and none of his descendants. This method can be implemented in a polynomial time algorithm Henson et al. (2014); Darwiche (2008); Geiger et al. (1990).

Some of the algorithms related to Bayesian networks resort to the *Markov Blanket* (Figure 6), to ensure the independence of a node concerning all other nodes in the network. For the Markov blanket to provide that independence to a node, its parents, its children, and the children's parents must be given, as discussed in Russel and Norvig (2010).

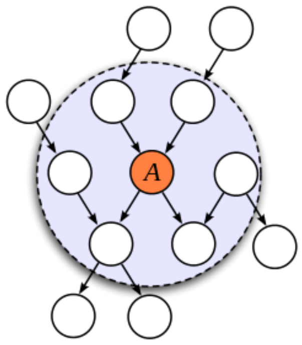


Figure 6.: Markov blanket for a node A (from Russel and Norvig (2010))

2.1.3 *Inference*

One powerful feature of a Bayesian network is to obtain the probability of some event knowing some other evidence. This process is known as inference. In a Bayesian Network, such a query is obtained with the use of Bayes theorem, which provides a way to write conditional probabilities as a function of the joint probability, as we can see in Equation (3). The joint probability is obtained by multiplication of values in the CPTs as mentioned before.

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{3}$$

The pieces of evidence given can change the possible worlds of an outcome in a way that some event that was almost certain, could unlikely happen. Imagine for example that we are going to make a bet on our favorite team next weekend. We know that the probability of winning is high, $P(Winning) = 2/3$. But the day before the game the best player of our team gets injured, forcing the probability of winning to go down, $P(Winning|CR7\ injured) = 1/6$. Most practical uses of Bayesian networks deal with this kind of inferences. A simple example is that of an *intelligent agent* that tries to find a way out of a certain room and uses his sensors to obtain the evidence and infer the probability of the outcomes of the actions he can make. Thus, this intelligent agent can maximize the "utility" of his actions depending on the causal relations mapped in the Bayesian Network. More about this topic will be discussed in more detail in section 4.1.

Inference in Bayesian Networks is a computationally difficult task, in the sense that it requires a high amount of computational resources, fundamentally (time and space). Therefore there are exact inference algorithms and approximate algorithms, the latter often leading to faster inferences, even if not in all cases.

Exact Inference

As mentioned before, inference is a computationally difficult task. In concrete, it is NP-hard, which means that at least it is as hard as the hardest problems in the NP class. Problems in the class NP (non-deterministic polynomial time) are problems that are not solvable in polynomial time but their solutions are verifiable in polynomial time. For example, 2-CNF (Conjunctive normal form) satisfiability problems are solvable in polynomial time and 3-CNF satisfiability problems are not. A CNF is a Boolean formula as in (4), where the binary variables are connected by conjunctions and closed by parentheses building a clause, and the clauses are connected by disjunctions. Moreover, every variable can be contradicted by the Not operator ($\neg$). The k-CNF satisfiability problem is essentially the problem of verifying if there is some assignment for the variables so that the whole CNF evaluated to true. The value of k determines how many variables exist in each clause Cormen et al. (2009).

$$(X_1 \land X_2) \lor (X_2 \land \neg X_3) \lor (\neg X_1 \land \neg X_4) \tag{4}$$

There are different algorithms for exact inference in Bayesian networks. The main difference between them is related to the way they use computational resources. Some of them use more time of computation, while others require more memory space on the device. For this reason, it is important to define the main parameters used to measure their complexity. For instance, the graph-theoretic notion of *Tree-Width* (w) is the most important parameter besides the number of nodes (n) Bacchus et al. (2009). The tree-width is determined after converting the Bayesian Network into a tree, for example, a jointree. There are different trees that can be constructed from the graph. So, the tree-width is defined as the smallest width from all trees that can be constructed by that graph. In concrete, the value of the tree-width is defined by the size of the largest cluster minus 1 Darwiche (2008). We shall now proceed to analyzing a number of inference algorithms.

VARIABLE ELIMINATION. This algorithm is one of the reference algorithms for inference. It is defined by computing the joint probability without repeating some calculations. For instance, the ones occurring in the *Enumeration* algorithm Russel and Norvig (2010).

To understand the *Variable Elimination* algorithm let us try to infer some probability from the Bayesian network in Figure 2. Suppose we want to obtain the probability of a friend performing a Duathlon, only knowing that he smokes, defined as $P(Duathlon|Smoke)$. It is necessary to obtain the joint probability $P(Duathlon, Smoke)$ from the original Bayesian network (Figure 2), resulting in the following expression:

$$P(Duathlon|Smokes) = \alpha * P(Duathlon, Smoke, Workout, Marathon) \tag{5}$$

$$\begin{aligned} P(Duathlon|Smokes) = \alpha * \sum_{Workout} \sum_{Marathon} P(Workout) * P(Smoke) \\ *P(Marathon|Workout, Smoke) * P(Duathlon|Marathon) \end{aligned} \tag{6}$$

The value of α is determined when both values $P(Duatlhon|Smokes)$ and $P(\neg Duathlon|Smokes)$ are known because it is just a normalization factor. Additionally, the variable Lung cancer will not be summed out because it is independent of the variable Duathlon. To confirm this the reader can apply the d-separation algorithm that was explained before. So, because there are some values that are constant, they can be put outside the sum to reduce the number of operations:

$$\begin{aligned} P(Duathlon|Smokes) = \alpha * P(Smoke) * \sum_{Workout} P(Workout) \sum_{Marathon} \\ *P(Marathon|Workout, Smoke) * P(Duathlon|Marathon) \end{aligned} \tag{7}$$

This optimization is considered by the enumeration algorithm but it still performs some repeated computations. For instance, in Figure 7 it can be seen that the terms $P(Duathlon|Marathon)$, $P(Duathlon|\neg Marathon)$ are determined twice.

That problem is solved by the variable elimination algorithm which evaluates the expression from right to left. With this procedure, the repeated values are maintained in the cache. In summary, this algorithm requires O($n \exp(w)$) time to obtain one query Russel and Norvig (2010); Darwiche (2008).

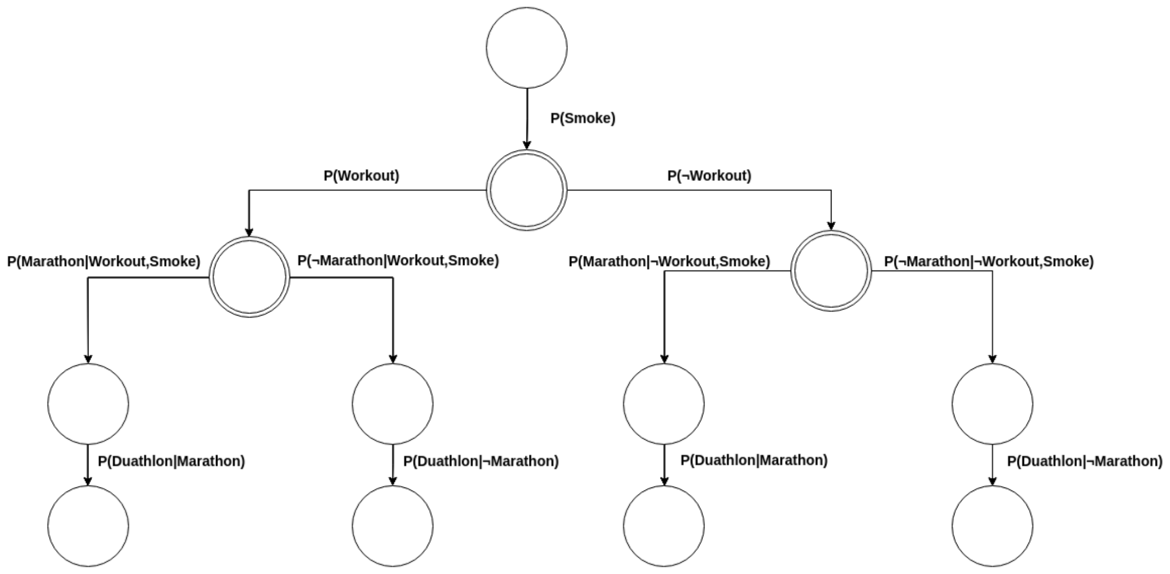


Figure 7.: Data structure for the Enumeration algorithm

CLUSTERING ALGORITHMS A typical example of a clustering algorithm is the *Jointree* algorithm. This algorithm constructs a jointree from a given Bayesian Network, which reduces the complexity of the inference because those nodes became singly connected, as it also happens in a polytree. For that, the nodes in the initial network need to be joined into clusters, as in Figure 8. To obtain this arrangement various steps need to be performed as described in C.Huang and A.Darwiche (1996).

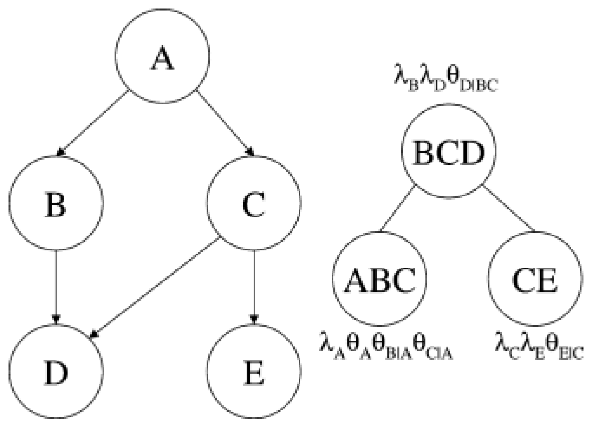

A
B
C
D
E
$\lambda_B\lambda_D\theta_{D|BC}$
BCD
ABC
CE
$\lambda_A\theta_A\theta_{B|A}\theta_{C|A}$
$\lambda_C\lambda_E\theta_{E|C}$

Figure 8.: Transformation of a belief network into the a jointree (from Park and Darwiche (2004)).

The main advantage to do inference in a jointree is the reduced number of steps required. Even in a polytree, which takes O(n) steps to perform one query, it would take O(n^2) to perform a query for every n. In contrast, a jointree would only need O(n) steps for the same task. For this reason, clustering algorithms are the most used in industrial applications. These algorithms are a little more complex than the variable elimination ones because some clusters have a number of variables in common, which means they need to deal with repeated variables to ensure that the information given by the clusters agree. Finally, it is important to refer that the complexity of the inference is still NP-hard. This means that if the inference takes exponential resources with a variable elimination algorithm, the CPTs in the jointree will occupy also exponential memory space Russel and Norvig (2010).

Aproximate Inference

These algorithms try to reduce the complexity of the inference by doing it in an approximate way. There are several approaches to that. Some of them are based on stochastic sampling and others reduce the complexity of the given network to a more simpler one.

REJECTION SAMPLING. This algorithm is very simple: it takes the Bayesian network and directly samples values from it. The procedure is to go through the network in topological order and sample each variable. In this way, we obtain samples with values for all variables. For example, in Figure 9 the first two steps of the algorithm are applied to the network from Figure 2.

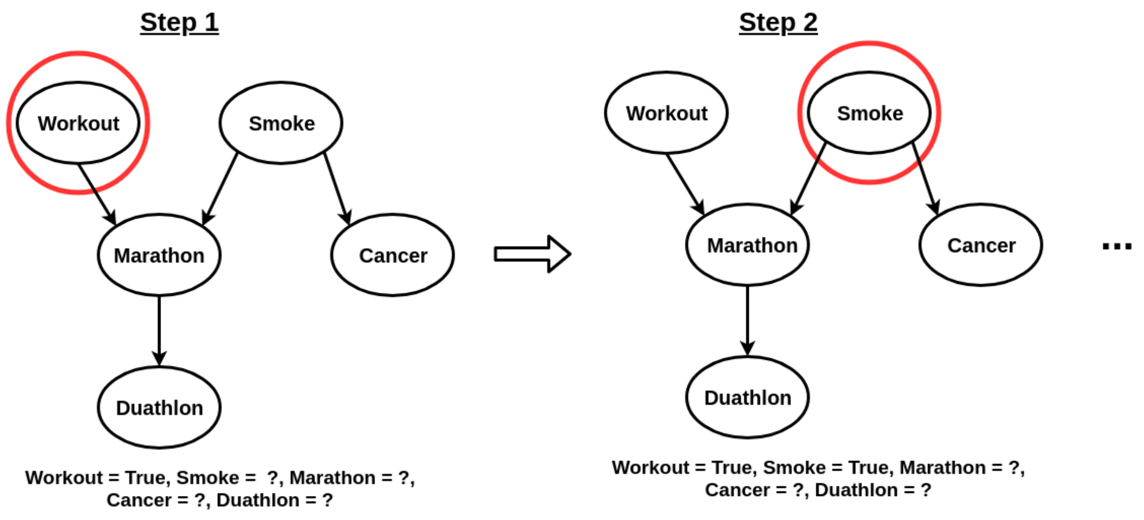

Step 1
Workout
Smoke
Marathon
Cancer
Duathlon
Workout = True, Smoke = ?, Marathon = ?, Cancer = ?, Duathlon = ?
Step 2
Workout
Smoke
Marathon
Cancer
Duathlon
...
Workout = True, Smoke = True, Marathon = ?, Cancer = ?, Duathlon = ?

Figure 9.: Representation of the first steps of the rejections sampling algorithm, performed on the network from Figure 2.

For the network considered, the first values sampled are Workout and Smoking, after that, the variable Marathon can be obtained, which is used for the value of Duathlon. In the end, a sample as the following is produced:

$$(Workout = True) \wedge (Smoke = True) \wedge (Marathon = False) \wedge (Duathlon = False) \quad (8)$$

The algorithm creates various samples and calculates the probabilities from the corresponding frequencies. Recall the Bayesian Theorem (3), a query $P(Workout|Smoke)$ can be written as:

$$P(Workout|Smoke) = \frac{P(Workout, Smoke)}{P(Smoke)} \quad (9)$$

So for the example in Equation (8), this yields:

$$P(Workout|Smoke) \approx \frac{\#Samples(Smoke = True, Workout = True)}{\#Samples(Smoke = True)} \quad (10)$$

This value becomes more accurate as the number of samples grows, so there is a balance between time and accuracy. In concrete, the value for the standard deviation of the error is related to the number of samples. This algorithm, however, fails to take advantage of every sample, considering only those where the evidence appears. So, if the evidence probability is small, then a larger number of samples will be discarded. For this reason, it is inefficient and not used in real applications Russel and Norvig (2010).

LIKELIHOOD WEIGHTING. In order to avoid wasting samples, there is another approach that can be adopted by this algorithm. It forces the evidence variables to the values given by the query, whereas nonevidence variables are normally sampled. By doing that the samples are no more equally weighted. Every time an evidence variable is forced to a value it does not respect the real probability with which this value would happen. So, to solve this problem every sample is associated with a weight related to the real probability of the value imposed. In this way, every sample has a weight that represents its real probability out of a forced sampling mechanism. Therefore, the weight of each sample with E evidence variables, Z nonevidence variables and m number of evidence variables, is obtained by,

$$w(z,e) = \prod_{i=1}^{m} P(e_i|parents(E_i)) \tag{11}$$

Consequently, the conditional probability for the query $P(x|e)$ is determined as,

$$P(x|e) \approx \alpha * \sum_{y} N_{ws}(x,y,e) * w(x,e) \tag{12}$$

When N_{ws} becomes large,

$$P(x|e) = \alpha' * \sum_{y} S_{ws}(x,y,e) * w(x,e) \tag{13}$$

where $S_{ws}(z,e)$ is the probability of the nonevidence variables defined as,

$$S_{ws}(z,e) = \prod_{i=1}^{l} P(z_i|parents(Z_i)) \tag{14}$$

Due to the fact that every sample is used, the algorithm is more efficient than rejection sampling. But it has a problem: when one of the evidence variables occurs late on the topological order, then the samples are guided by nonevidence variables, which is not, of course, the best scenario. It would be of great interest to allow the evidence variables to guide the sampling distributions so that we would get samples in correspondence to the probability induced by them Russel and Norvig (2010).

OTHER APPROXIMATE ALGORITHMS There are other approaches to do approximate inference. One is the *Gibbs Sampling* algorithm based on *Markov Chain Monte Carlo* algorithms Walsh (2004). Here the evidence variables are fixed and then one of the nonevidence variables is randomly chosen. Subsequently, the probabilities of the states of the variable are determined through its Markov blanket, which is a simple process. After that, the state is sampled. The result is used as one sample for the whole process and the next step takes this new value into account. In the end, the result of $P(x|e)$ is given by:

$$P(x|e) \approx \frac{\#Samples(X = x)}{\#Samples} \tag{15}$$

It is important to refer that the Gibbs sampling on a Bayesian Network has to fulfill some properties. For instance, the transition probability distribution must be ergodic, i.e. from every state in the state space there is a possible transition to every other state. Therefore, there are no periodic cycles. It is important to refer that this algorithm is a better approach than the ones mentioned before but it still has some problems, for example, if some probability in the CPTs is 0 or 1 the state space can be disconnected Kulaga (2006); Russel and Norvig (2010); Darwiche (2008).

Another technique is based on a variational method, which tries to obtain an approximate value by finding a simpler network. The task is to find a simpler and better-behaved network for performing inference. A computationally good behaved network should have, for instance, a small tree-width. The algorithm turns out to be an optimization problem because it tries to maximize the *Kullback-Leibler Divergence* (KL-divergence), of the two networks. The KL-divergence, given by Equation (16) measures how much one probabilistic distribution diverges from another. The quality of the values obtained afterward is directly related to this divergence measure. In Figure 10, the KL-divergence between two Gaussian distributions is presented.

$$KL(q(x)||p(x)) = -\sum_{x} q(x) * \log \frac{q(x)}{p(x)} \tag{16}$$

The algorithm works like other variational methods because it searches for the best parameters for the new network. This algorithm is able to obtain faster results but it does not guarantee the same convergence to the real value as the Gibbs sampling Jordan et al. (1999); Darwiche (2008).

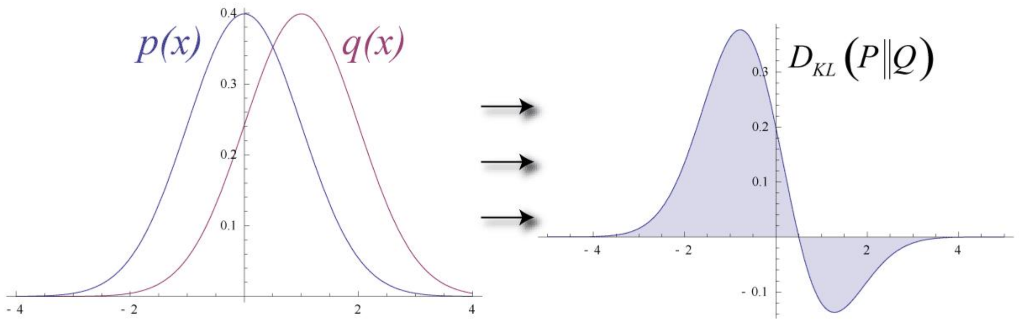


Figure 10.: Kullback-Leiber divergence. On the left are represented two Gaussian probability distribution and on the right the corresponding KL-divergence area, which has to be integrated, for the measure (from Suzen (2017)).

2.2 DYNAMIC BAYESIAN NETWORKS

Bayesian Networks can also represent time models. In this case, the network is divided into slices, each slice corresponding to a different time frame. To keep the model simple the local network is invariant along different time slices, containing the same variables and topology. As a result, there is an occurrence of each variable for each time frame $X_0, ..., X_t, X_{t+1}$. Additionally, there are temporal dependencies between the variables, for instance, the previous position of an object has a direct influence on the next one. These dependencies define the transition model $P(X_{t+1}|X_t)$. Normally in each time slice, there are also evidence variables which means that the network in each time slice maps the sensor model $P(E_t|X_t)$. A *Dynamic Bayesian Network* is totally defined when the initial values of the variables are given $P(X_0)$ to the transition and sensor model. In Figure 11, a simple dynamic Bayesian network is presented. It models the behaviour of an automatic brake system. For that, it considers the velocity of the car, the distance to the front car and the state of the brakes. Additionally, in this system, the transition model describes that the velocity in the next time frame is dependent on the previous, while the distance in the next time frame depends on the previous distance and velocity.

Finally, it is important to mention that a dynamic Bayesian network represents a first-order Markov process. This means that a state is only dependent on the previous one, in other words, the present state is independent of past states except for the previous one Li et al. (2018).

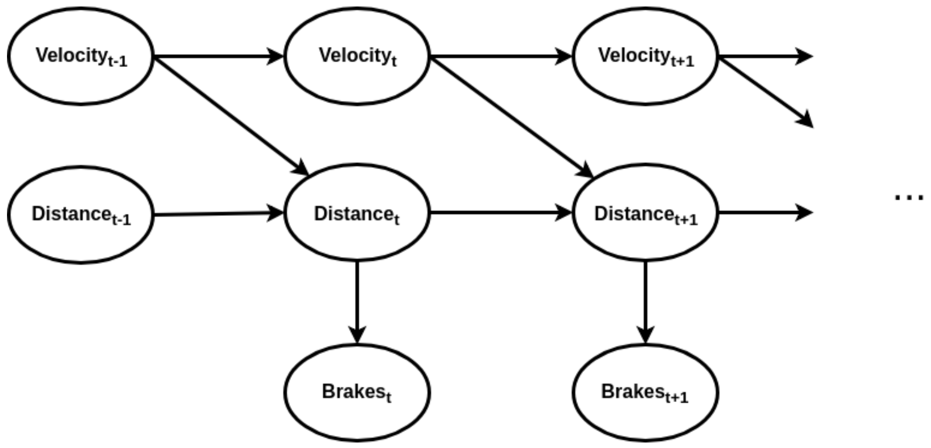


Figure 11.: A representation of an automatic brake model.

3 QUANTUM BAYESIAN NETWORKS

3.1 BAYESIAN NETWORKS AS A TOOL FOR QUANTUM MODELING

Quantum mechanics is the most successful theory to explain how elementary particles, such as electrons, behave. The theory has in its nature a probabilistic interpretation. The observables, which are the physical quantities measurable in a laboratory, not only are quantified for some concrete values "quanta", but also have a non-deterministic character House (2017). Moreover, these particles exhibit "social" behaviour, i.e. two particles can behave as if they are only one, somehow sharing information Oliveira et al. (2007). This effect is called entanglement, and it has become a highly researched topic. A new era of communications is emerging to explore it, for the safety of their users Kimble (2008).

As the number of experiments confirming this behaviour increased, the more accepted the theory became. And remember, even Einstein was not convinced that the universe would work that way Einstein et al. (1935). Nevertheless, no experiment has disproved quantum mechanics Aspect et al. (1981), which came as a disappointment for several Physicists.

In quantum mechanics, a particle is described by its wave-function (Figure 12). This function holds the information that can be extracted from the particle or system, and its unit is the probability density House (2017).

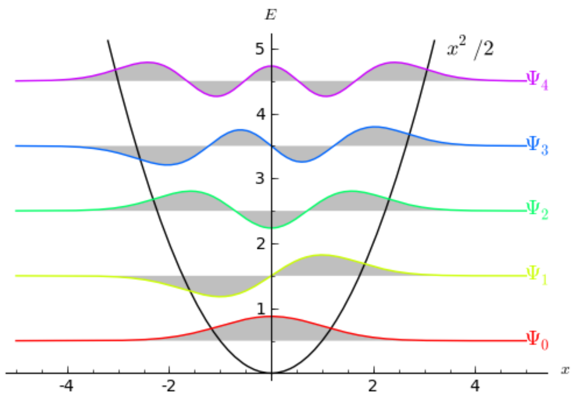


Figure 12.: Wave-functions for the Harmonic Oscillator (from Kapoor (2012)).

As a result of this probabilistic nature, Bayesian networks and similar models were considered to specify physical systems. They allow performing *belief propagation*, which is the process of passing information between the nodes, necessary to perform inference. The information obtained after the inference process provides a better understanding of a system in a posterior state. For instance, in Leifer and Poulin (2007), Bayesian networks are used to create and analyze error correction codes. These codes are intended to reduce or even eliminate the presence of noise in quantum based computations. For that, they measure part of the quantum state and predict using precisely these graphical models, which would be the best operation to correct, as much as possible, the effect caused by the noise at that time step.

Another example is the usage of the same structures for quantum many-body simulations. They allow Physicists to simulate particle systems and predict their behaviour. For instance, spin-systems (Figure 13) are of great interest to simulate superconducting qubits. The use of graphical structures for simulation is also found in Tucci (1995), where only Bayesian networks are considered and their systems are written down by the second quantization formalism.

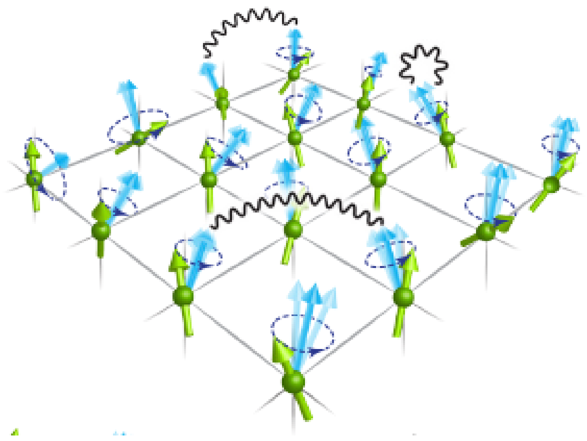


Figure 13.: Spin system as an instance of a many-body quantum system (from Knap).

There are other approaches where a special kind of Bayesian networks called "Qudot Nets" is used as a representation of real qubits Sakkaris (2016). The vertices of the networks are the states of the qubits and the edges correspond to the quantum operations. Belief propagation is applied to obtain the outcomes of the computations performed on the qubits. In other words, these inference algorithms work as a classical simulation of quantum algorithms.

In this sequence, Leifer has defined some of the most basic elements of probability theory Leifer and Spekkens (2011). A very interesting one for this text is "Quantum Bayes Theorem":

$$\rho_{A|B} = \frac{\rho_{B|A} * \rho_A}{\rho_B} \tag{17}$$

Resorting to this theorem, quantum states (ρ_A) and quantum conditional states ($\rho_{B|A}$), the author defines belief propagation for quantum systems. However, despite the previous works presented, this framework intends to demonstrate the fundamental proprieties of the quantum systems. Representing or simulating a physical system is not the main intent. For instance, in Leifer and Spekkens (2011) the authors demonstrate that there is an isomorphism between the state that characterizes two physical systems in the same time step and the time evolution operator that defines the transition of a system between two different time steps. So, there is no reason to distinguish these two elements, as it happens in the traditional quantum mechanical formalism.

Explaining and predicting quantum mechanical experiences with the use of probabilistic theories is a research topic by itself. But there is an interpretation the other way around where quantum mechanical systems are used to simulate probabilistic computations. A more primitive idea came with Feynman's concept of a quantum computer, to simulate quantum mechanical systems Feynman (1999). The concept of a quantum computer turns out to be extremely useful for the purpose that Feynam presented. Several computational problems were proven to be solved efficiently in the concept of such a machine. For instance, the discrete Fourier transform has an exponential speedup on a quantum computer Nielsen and Chuang (2010). That has a great impact in many engineering applications since solutions for problems that were computationally intractable before are now reachable. In cryptography, Shor's algorithm could compromise the current cryptographical protocols, with the use of the quantum Fourier transform to factorize prime numbers in a reasonable amount of time Shor (1995). This happens, because most of the current cryptographical protocols are based on the assertion that factorizing a prime number is a computationally difficult problem Katz and Lindell (2014).

3.2 QUANTUM ANALOGS FOR BAYESIAN NETWORKS

Bayesian networks are a model used to describe a probabilistic system and every time they were mentioned in this dissertation they had a classical nature. But they are not limited to a classical framework. There are some interesting examples where these models are written in a quantum framework.

In Moreira and Wichert (2018) the authors present "quantum Bayesian networks" that are not a classical object to describe a quantum mechanical one, as before, but a quantum mechanical system used to describe human behaviour. A classical Bayesian network can be used to describe human behaviour but when paradoxical results appear Pothos and Busemeyer (2009); Shafir and Tversky (1992), their complexity grows exponentially. This happens because more latent variables have to be considered. Latent variables are extra nodes in Bayesian Networks that cannot be observed but are inferred from the data. However, when "quantum Bayesian networks" are considered the paradoxical results do not generate an exponential increase. The network maintains its dimension and the results occur due to interference along the inference process.

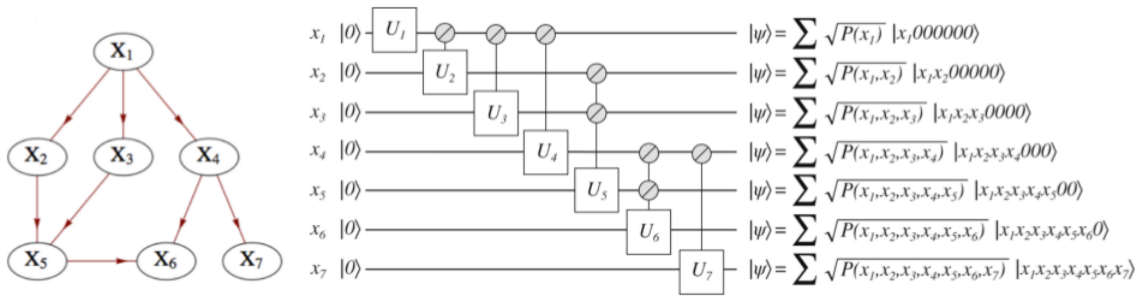
Following the idea that a quantum computer could solve probabilistic problems faster, a lot of research has been done to speedup these processes. Inference over a Bayesian network was not an exception, as it is found in a lot of real applications Jensen and Nielsen (2007); Neapolitan (2003).

A quantum algorithm for inference over Bayesian networks was introduced by Low et al. (2014), 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 is represented by a single qubit and probabilities are mapped into the coefficients of the quantum state:

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \Leftrightarrow |\Psi\rangle = \alpha\,|Var_1 = true\rangle + \beta\,|Var_1 = false\rangle \tag{18}$$

Whenever two variables share an edge in the network 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 through the application of a specific sort of gates — controlled rotations — to the state qubits (Figure 14). The fact that a rotation is controlled by another qubit creates entanglement between them (mutual information). The amplitude of the rotation defines the value of the coefficients.



X1
X2
X3
X4
X5
X6
X7
x_1 $|0\rangle$ U_1
x_2 $|0\rangle$ U_2
x_3 $|0\rangle$ U_3
x_4 $|0\rangle$ U_4
x_5 $|0\rangle$ U_5
x_6 $|0\rangle$ U_6
x_7 $|0\rangle$ U_7
$|\psi\rangle = \sum \sqrt{P(x_1)}\, |x_1000000\rangle$
$|\psi\rangle = \sum \sqrt{P(x_1,x_2)}\, |x_1x_200000\rangle$
$|\psi\rangle = \sum \sqrt{P(x_1,x_2,x_3)}\, |x_1x_2x_30000\rangle$
$|\psi\rangle = \sum \sqrt{P(x_1,x_2,x_3,x_4)}\, |x_1x_2x_3x_4000\rangle$
$|\psi\rangle = \sum \sqrt{P(x_1,x_2,x_3,x_4,x_5)}\, |x_1x_2x_3x_4x_500\rangle$
$|\psi\rangle = \sum \sqrt{P(x_1,x_2,x_3,x_4,x_5,x_6)}\, |x_1x_2x_3x_4x_5x_60\rangle$
$|\psi\rangle = \sum \sqrt{P(x_1,x_2,x_3,x_4,x_5,x_6,x_7)}\, |x_1x_2x_3x_4x_5x_6x_7\rangle$

Figure 14.: Quantum circuit build to encode a Bayesian network in to a quantum state (from Low et al. (2014)).

Measuring this state leads to its collapse into a classical value:

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \xrightarrow{\text{measurement}} |\Psi\rangle = |0\rangle \text{ or } |\Psi\rangle = |1\rangle \tag{19}$$

Additionally, the probability of some state becoming the outcome of the measurement is the square norm of the corresponding amplitude:

$$P(|\Psi\rangle = |0\rangle) = |\alpha|^2 \text{ and } P(|\Psi\rangle = |1\rangle) = |\beta|^2, |\alpha|^2 + |\beta|^2 = 1 \tag{20}$$

The quantum state is the superposition of all entries in the original joint probability distribution table:

$$\begin{aligned}|\Psi\rangle = |\gamma_1|^2\,|Var_1 = true, Var_2 = true\rangle + |\gamma_2|^2\,|Var_1 = true, Var_2 = false\rangle +\\ |\gamma_3|^2\,|Var_1 = false, Var_2 = true\rangle + |\gamma_4|^2\,|Var_1 = false, Var_2 = false\rangle\end{aligned} \tag{21}$$

This means that a measurement produces a sample, as in the rejection sampling algorithm. Consequently, this process has similarities with rejection sampling. However, it is not an efficient way to do inference because every time a measurement is performed over the state, it collapses and needs to be reconstructed.

In a second stage, Grover's algorithm Grover (1996) is applied to amplify the states that have the right values for the evidence variable. Grover's algorithm allows for a quadratic speedup in search problems, a fact that explains its ubiquity and relevance to many quantum programs. In this 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 where they lack it:

$$\begin{aligned} |\Psi_{init}\rangle = &\sqrt{P(e)}\,|Var_1, Var_2, ..., evidences\rangle \\ &+\sqrt{1-P(e)}\,|Var_1, Var_2, ..., \neg evidences\rangle \end{aligned} \tag{22}$$

Then Grover's algorithm is applied to search for the state that has the right values for the evidence variables.

$$|\Psi_{init}\rangle \xrightarrow{\text{Grover}} |\Psi_{final}\rangle = |Var_1, Var_2, ..., evidences\rangle \tag{23}$$

The last stage amounts simply to observe this state and use the result as a sample. Table 1 shows the comparison between the classical and the quantum versions. The latter exhibits a quadratic speedup but only if the Bayesian network is not too densely connected. Otherwise the price of encoding into a quantum state will be too high, as the corresponding term grows exponentially.

Table 1.: Classical *vs* quantum complexity.

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

As before, e represents the evidence variables, while n quantifies the total number of variables represented in the Bayesian network. Finally, m represents the density of the graph Darwiche (2008).

It is important to mention that the inference process remains in the NP-class. The quantum inference process presented before does not change the exponential characteristic of the problem. The authors of Moreira and Wichert (2018) showed that inference process performed with an quantum Bayesian network does not change the difficulty of the problem.

3.3 OTHER MODELS AND APPLICATIONS

Bayesian networks are a very generic structure. For this reason, some special cases of these networks are known by a specific name. For example, Markov processes (Figure 15) and

Kalman filters are a special case of dynamic Bayesian networks Russel and Norvig (2010), and, have also been explored for quantum analogs.

Some very interesting results appeared for quantum Markov decision processes. In Ying and Ying (2014), they were used to describe the semantics of quantum programs. In concrete, the reachability of invariant states was analyzed and the results showed that for the quantum case this problem is undecidable. This means that a generic algorithm to compute reachability does not exist.

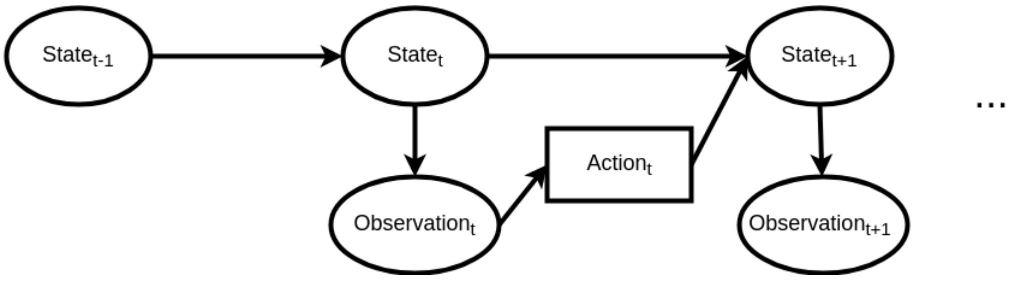


Figure 15.: First order Markov decision process.

A similar result was found in Barry et al. (2014), which studies a slightly different case focused on partially observable Markov processes (POMP) and their quantum analog. POMP are named that way because some nodes are not directly observable. However, also in this case, the reachability problem is undecidable. Both structures in a quantum framework have undecidable reachability.

These two examples proved the existence of fundamental differences between classical models and their quantum analogs. Furthermore, these differences revealed important implications for their applications and formal verification.

In other research fields, these differences are studied for better or faster results. For instance, in Paparo et al. (2014) a quantum analog for a projective simulation model Briegel and De las Cuevas (2012) is presented. This kind of models are used for active learning agents because of the structure changes with every iteration. They are commonly used in robotics because it is impossible, in most cases, to have a perfect model of the environment. So, the agents learn with every interaction how to behave in such an environment. Their internal processes are very prone to use quantum walks, which is the reason for the speedup. Also, they do not require a universal quantum computer. A physical system that supports quantum walks such as optical systems and trapped ions are enough for a complete implementation. A few years later, a proof-of-principle experiment based on trapped ions was executed Sriarunothai et al. (2018), demonstrating a real quantum enhancing speedup for these models.

3.4 SUMMARY

Bayesian networks and other graphical models come in hand to describe probabilistic systems. Quantum mechanics and quantum computing are good candidates to be described by these models. Such descriptions are useful in a variety of research areas related to physics. An application of great importance are simulations of quantum mechanical systems.

Quantum computing reveals to be a powerful playground for such models, allowing them to represent and implement related processes with an advantage in performance. These differences in performance could have a huge impact on real industrial applications. For this reason, topics related to quantum probabilistic programming, quantum machine learning, and quantum artificial intelligence remain very active research topics within the scientific community.

4 QUANTUM BAYESIAN DECISION-MAKING

4.1 DECISION MAKING

Decisions are made by humans in their everyday life. The choice between the food we eat and the clothes we wear are simple examples. Decision making can be defined as the process of deliberation between the options available. It is also related to the notion of intelligence, which is one of the features that distinguishes us the most from other animals. Another relevant feature of the human being is laziness, so we look to optimize our everyday tasks and decision making is not an exception.

In the field of artificial intelligence, the idea of equipping a machine with such a decision-making process already exists. It is the result of combining probability theory with utility theory Russel and Norvig (2010). From probability theory, the intelligent agent receives a framework to model the world that surrounds it, and from utility theory, the notion of usefulness that each *result* (R) can have. The utility of each outcome can be quantified by a *utility function* (U). A well known object that behaves as a utility function for humans is money because it quantifies the value of objects and services (Figure 16).

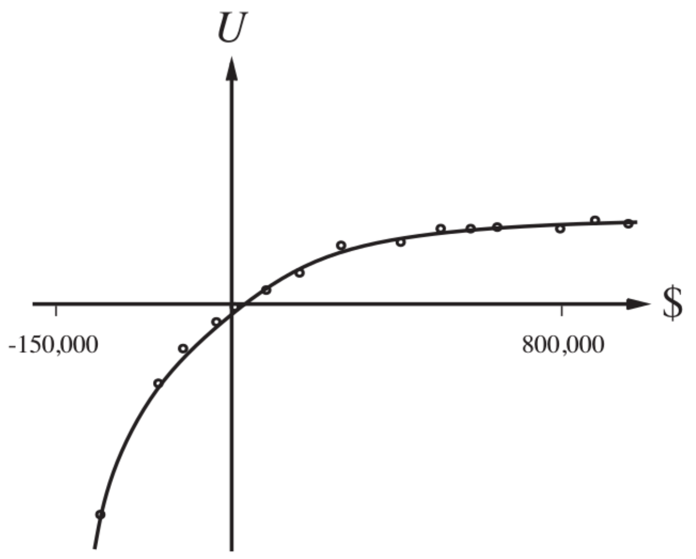


Figure 16.: Empirical utility of money (from Russel and Norvig (2010))

However, both elements have to be combined in a way that respects the notion of rational decision making. If rationality was not a requirement, the decision process could be performed with any heuristic or even in a completely random way. Given that, the model of the environment does not impose any restriction of rationality, all axioms fall into the utility function Russel and Norvig (2010). For instance, the utility function has to respect **Transitivity**, which means that the following has to be true:

$$(U(r_1) > U(r_2)) \wedge (U(r_2) > U(r_3)) \implies U(r_1) > U(r_3) \tag{24}$$

If the utility function does not respect this axiom then the agent ruled by this function could be fooled. Imagine that $r_1 = 10€$, $r_2 = 5€$, $r_3 = 2€$ and the following is true:

$$U(r_1) > U(r_2) \wedge U(r_2) > U(r_3) \wedge U(r_1) < U(r_3) \tag{25}$$

then, someone could exchange 2€ by 10€ with the agent, until the agent has no money. This happens, because the agent thinks that the utility of 2€ is bigger than the utility of 10€ $(U(r_3) > U(r_1))$. With that, the agent would run out of money and still believe that it had made the best choices.

The same function and the probability distributions have to respect:

- **Orderability**: Imposes that some preference has to exist between any two outcomes.

$$\begin{gathered} \forall_{r_1,r_2 \in Res} \\ (\mathrm{U(r_1)} > U(r_2) = True) \vee (U(r_1) < U(r_2) = True) \vee (U(r_1) = U(r_2) = True) \end{gathered} \tag{26}$$

- **Continuity**: If some outcome (r_2) is the middle of some other two (r_1, r_3), in relation to preference. Then, there is a probability distribution for r_1 and r_3 that has the same expected utility as having outcome r_2 with certainty.

$$U(r_1) > U(r_2) > U(r_3) \implies \exists p\ [p, r_1; 1-p, r_3] = EU(r_2) \tag{27}$$

- **Substitutability**: The relation between two outcomes r_1 and r_2 is preserved if they are inserted in more complex distribution.

$$\forall_{\phi \in \{<,>,=\}}\ U(r_1)\ \phi\ U(r_2) \implies [p, r_1; 1-p, r_3]\ \phi\ [p, r_2; 1-p, r_3] \tag{28}$$

- **Monotonicity**: If outcome r_1 is preferred over outcome r_2, then a distribution composed by these two outcomes with higher probability for r_1 is also preferred.

$$U(r_1) > U(r_2) \implies (p > q \Leftrightarrow [p, r_1; 1-p, r_2] > [q, r_1; 1-q, r_2]) \tag{29}$$

- **Decomposability**: Any compound distribution can be translated in a simple distribution by the rules of probability:

$$[p, r_1; 1-p, [q, r_2; 1-q, r_3]] = [p, r_1; (1-p)q, r_2; (1-p)(1-q), r_3] \tag{30}$$

Additionally to the axioms imposed, the decision process has to follow the *maximum expected utility* (MEU) principle C. Fishburn (1982). This principle, claims that a rational entity is expected to choose the *action* (A) with the greatest *expected utility* (EU) with respect to its own set of beliefs. The expected utility of some action is the value that quantifies the reward expected, on average, by picking exactly that one. The Equation used to compute the expected utility for a certain action, with knowledge about the state of some variables called *evidences* (E) is:

$$EU(a|e) = \sum_R P(r|a,e) * U(r) \tag{31}$$

Summarizing, the maximum expected utility principle becomes :

$$action = argmax_a EU(a|e) \tag{32}$$

In general, the MEU principle is found all over the domain of artificial intelligence. But, this does not mean that all follow exactly equations (31) and (32). For some tasks, it is not enough to consider only the next state, as it happens in (31). In such problems the intelligent agent seeks for the ideal *policy* ($\pi*$) within all possible policies (π), with a policy being simply a mapping from states to actions.

The utility of each policy ($U^{\pi}(s)$) is determined by computing all possible combination of interactions with the environment until the goal state is reached. Afterward, with respect to the highest expected utility principle, the optimal policy is selected:

$$\pi_s * = argmax_{\pi} U^{\pi}(s) \tag{33}$$

The computational work increases, but the underlying principle is the same in both cases.

It is important to mention that dynamic Bayesian networks discussed in Section 2.2 are often used to model the environment, which means that, at this point, all elements of a decision process were presented. In Figure 17, the main processes of an intelligent agent are demonstrated. Solutions of this kind are commonly used and have excellent results Bongard (2008).

Although, more complex processes exist. For instance, in reinforcement learning models, the agents adapts its behaviour depending on its interactions with the environment. If some action generates a positive feedback it is rewarded and the opposite happens for negative feedback. So the utility of each actions changes with the history of the interactions. These types of model are frequently used in robotics because the robot can not have access to a model of its environment or, if it can, it is too complex to perform inference over it.

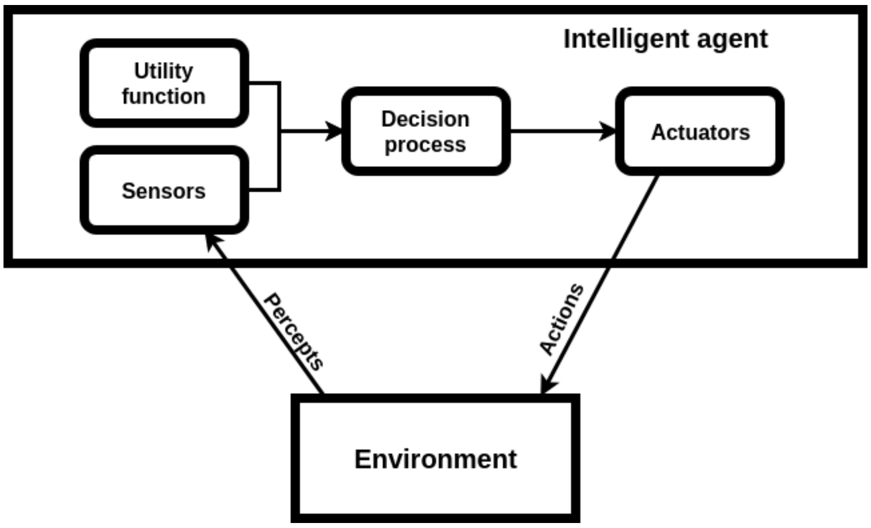


Figure 17.: Model of an intelligent agent. The interaction with the environment and the utility function are given to the decision process. This process outputs to the actuators the next action.

4.2 QUANTUM ASSISTED DECISION MAKING

Decision-making Processes are very useful. Unfortunately, the quality of their result is not only bounded by their theoretical limitations but by the computational power of the machines running them. As inference is an NP-hard problem, decision making is at least of the same complexity, as it has an inference process as a subprocess.

A quantum computer running the inference algorithm presented in Chapter 3 , could be used to speedup decision-making processes. The conditional probabilities could be computed with a quadratic speedup, according to Equation (34).

$$EU(a|e) = \underbrace{\sum_R \underbrace{P(Result = r|a,e)}_{Quantum} * U(r)}_{Classical} \tag{34}$$

At this point, there is no doubt that a decision-making process performed on a quantum computer could, in some cases, reduce the time of computation. It is important to mention that the complexity of inference is not reduced, a quadratic speedup applied to an NP-problem has almost no effect when the dimension of the problem increases. Still, such a speedup will be observable in real applications and have a great impact on them.

The algorithm presented before is a hybrid one, it uses a quantum and a classical Turing machine. The subprocess that suffers the speedup is the one performed on the quantum machine. So, if a larger part of the computational work was performed on the quantum machine then the entire process, in principle, would be faster.

By using *Decision Networks* (Figure 18) it will be possible to do exactly that. Decision networks are a Bayesian network with extra nodes for actions and utilities Russel and Norvig (2010). This means that the network will have a utility node that is connected to the outcome node, or the outcome nodes if in a multi-attribute utility function. To compute the expected utility of some action it is only necessary to perform the inference algorithm until the utility node is reached.

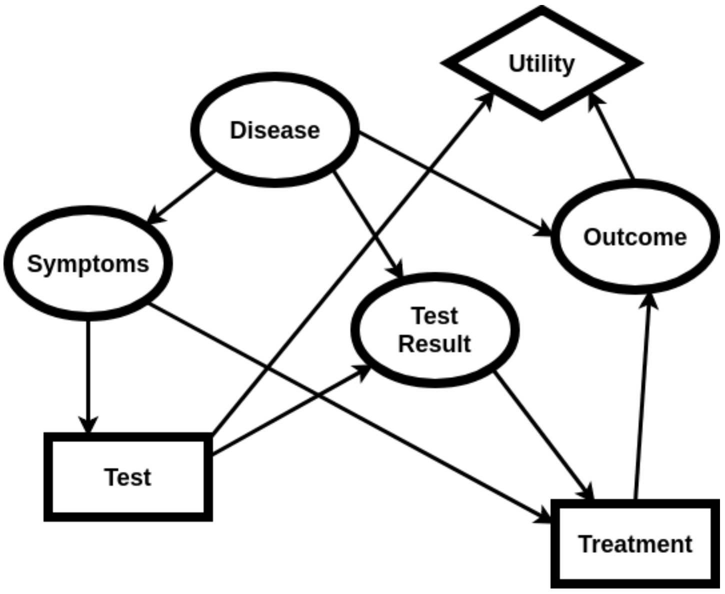


Figure 18.: Diagnostic assistant modeled with a decision network (from David Poole (2010).

Classically, the computational work needed to determine the expected utility is the same in both cases because the exact inference process applied to a decision network does the same as Equation (31). In the quantum case, the use of decision networks allows us to perform a larger part of the process with the quantum inference algorithm :

$$EU(a|e) = \sum_R \underbrace{P(Result = r|a,e) * U(r)}_{Quantum} \tag{35}$$

This has, as consequence, a quadratic speedup not only for the subprocess that computes the conditional probabilities ($P(Result = r|a,e)$) as before, but a quadratic speedup for the utility term of each outcome ($P(Result = r|a,e) * U(r)$).

A quantum decision-making algorithm based on the quantum inference algorithm, presented in Chapter 3, applied to a decision network, can be defined by the following iterations:

- Observe the current state of the evidence variables.
- For each possible *action*:
  - Until the expected utility of $action_i$ is precise enough:
    - Encode the decision network as a quantum state ($|\Psi_{init}\rangle$).
    - Perform a Grover search for the correct state of the *action* and *evidence* variables.

$$|\Psi_{init}\rangle \xrightarrow{\text{Grover}} |\Psi_{final}\rangle = |Var_1, Var_2, ..., action_i, evidences\rangle \tag{36}$$

    * Observe the final state($|\Psi_{final}\rangle$).

- Return the action with the highest utility.

There are important implications to mention. First of all, the utility function has to be normalized to be encoded as a quantum state. Secondly, the expected utility which is sampled is also a normalized value. Additionally, if the utility function is a multi-attribute function the number of variables on which it depends is important because it can change the tree-width of the network. The tree-width is one of the parameters that define the complexity of the quantum inference algorithm as seen in Table 1. So, for some decision networks a quadratic speedup can be obtained by using a quantum computer, while for others the quantum process is slower.

4.3 A NEW QUANTUM ALGORITHM FOR DECISION MAKING

In this section, a different approach for decision making which, in principle, will take an increased advantage of the quantum resources will be presented. The idea is quite simple: instead of sampling the conditional probabilities ($P(Result = r|a, e)$) or utility terms ($P(Result = r|a, e) * U(r)$), the quantum state remains unobserved until the utility function is applied. The utility is not part of the state, as in the decision networks, but a transformation applied to the quantum state:

$$|\Psi\rangle \xrightarrow{\text{Utilty function}} |\Psi'\rangle \tag{37}$$

The intention is to apply this 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. By maintaining the action variable in a superposition, the utility function will be applied in parallel to all conditional probability terms used in the decision-making process. Also, by observing later on the action variable all the utility terms will automatically be summed out for that action as it happens in Equation (31). From this state the expected utility of some action ($EU(a)$) will be directly sampled.

For a better understanding of that algorithm, let us follow a generic quantum state that encodes a Bayesian network:

$$|\Psi\rangle = |Result, Action, Evidences, Var_1, ...\rangle \tag{38}$$

In the first step, Grover's search algorithm is applied to obtain the correct values of the evidence variables, without considering the action variable as one:

$$|\Psi\rangle = |Result, Action, Evidences, Var_1, ...\rangle \xrightarrow{\text{Grover}} |\Psi'\rangle = |Result, Action, evidences, Var_1, ...\rangle \tag{39}$$

The same state can be written as a superposition of all concrete bases of the result variable, which, being Boolean, yields:

$$|\Psi'\rangle = \alpha * |r, Action, evidences, Var_1, ...\rangle + \beta * |\neg r, Action, evidences, Var_1, ...\rangle \tag{40}$$

where

$$|\alpha|^2 + |\beta|^2 = 1 \tag{41}$$

Considering that the action variable is also a Boolean, the same state can be written as:

$$\begin{aligned} |\Psi'\rangle = \gamma_1 |r, a_1, evidences, Var_1, ...\rangle + \gamma_2 |r, a_2, evidences, Var_1, ...\rangle \\ + \gamma_3 |\neg r, a_1, evidences, Var_1, ...\rangle + \gamma_4 |\neg r, a_2, evidences, Var_1, ...\rangle \end{aligned} \tag{42}$$

with

$$|\gamma_1|^2 + |\gamma_2|^2 + |\gamma_3|^2 + |\gamma_4|^2 = 1 \tag{43}$$

The coefficients that describe the state and determine the probability of observing each base are related by the following expressions:

$$|\alpha|^2 = |\gamma_1|^2 + |\gamma_2|^2 \tag{44}$$

$$|\beta|^2 = |\gamma_3|^2 + |\gamma_4|^2 \tag{45}$$

Afterward, the utility function is applied as an operator U to the state $|\psi\rangle$. This operator changes the amplitudes of the state, in correspondence to the utility of each value of the result variable:

$$\begin{aligned} U|\Psi'\rangle = \frac{\sqrt{U(r)} * \alpha}{k} |r, Actions, evidences, Var_1, ...\rangle \\ + \frac{\sqrt{U(\neg r)} * \beta}{k} |\neg r, Actions, evidences, Var_1, ...\rangle \end{aligned} \tag{46}$$

k is a normalization term, so that,

$$|\frac{\sqrt{U(r)} * \alpha}{k}|^2 + |\frac{\sqrt{U(\neg r)} * \beta}{k}|^2 = 1 \tag{47}$$

yielding,

$$\begin{aligned} U \left|\Psi'\right\rangle = \gamma_1' \left|r, a_1, evidences, Var_1, ...\right\rangle + \gamma_2' \left|r, a_2, evidences, Var_1, ...\right\rangle \\ + \gamma_3' \left|\neg r, a_1, evidences, Var_1, ...\right\rangle + \gamma_4' \left|\neg r, a_2, evidences, Var_1, ...\right\rangle \end{aligned} \tag{48}$$

Thus,

$$\frac{U(r) * |\alpha|^2}{k^2} = |\gamma_1'|^2 + |\gamma_2'|^2 \tag{49}$$

which yields, by equation (44),

$$\frac{U(r)}{k^2} = \frac{|\gamma_1'|^2 + |\gamma_2'|^2}{|\gamma_1|^2 + |\gamma_2|^2} \tag{50}$$

$$|\gamma_1'|^2 + |\gamma_2'|^2 = \frac{U(r)}{k^2} * (|\gamma_1|^2 + |\gamma_2|^2) \tag{51}$$

$$|\gamma_1'|^2 + |\gamma_2'|^2 = \frac{U(r)}{k^2} * |\gamma_1|^2 + \frac{U(r)}{k^2} * |\gamma_2|^2 \tag{52}$$

concluding that

$$|\gamma_1'|^2 = \frac{U(r)}{k^2} * |\gamma_1|^2 \tag{53}$$

$$|\gamma_2'|^2 = \frac{U(r)}{k^2} * |\gamma_2|^2 \tag{54}$$

Rewriting state $U\left|\Psi'\right\rangle$ in the bases where the action variable is defined and the result variable is not, leads to

$$U \left|\Psi'\right\rangle = \omega_1 \left|R, a_1, evidences\right\rangle + \omega_2 \left|R, a_2, evidences\right\rangle \tag{55}$$

knowing that,

$$\begin{aligned} |\omega_1|^2 = |\gamma_1'|^2 + |\gamma_3'|^2 \\ |\omega_2|^2 = |\gamma_2'|^2 + |\gamma_4'|^2 \end{aligned} \tag{56}$$

where the probabilities of measuring each state of the action variable are:

$$\begin{aligned} P(a_1) = |\omega_1|^2 = \frac{U(r)}{k^2} * |\gamma_1|^2 + \frac{U(\neg r)}{k^2} * |\gamma_3|^2 \\ P(a_2) = |\omega_2|^2 = \frac{U(r)}{k^2} * |\gamma_2|^2 + \frac{U(\neg r)}{k^2} * |\gamma_4|^2 \end{aligned} \tag{57}$$

Moreover, given that

$$\begin{aligned} |\gamma_1|^2 = P(a_1, r|evidences)\ ,\ |\gamma_2|^2 = P(a_2, r|evidences) \\ |\gamma_3|^2 = P(a_1, \neg r|evidences)\ ,\ |\gamma_4|^2 = P(a_2, \neg r|evidences) \end{aligned} \tag{58}$$

and

$$P(r|a_1, evidences) = \frac{P(a_1, r|evidences)}{P(a_1|evidences)} \tag{59}$$

we conclude that

$$|\gamma_1|^2 \propto P(r|a_1, evidences) \tag{60}$$

and that the transformation yields a state where

$$\begin{aligned} P(a_1|evidences) = \frac{U(r)}{k^2} * P(a_1, r|evidences) \\ + \frac{U(\neg r)}{k^2} * P(a_1, \neg r|evidences) \end{aligned} \tag{61}$$

As after the transformation the probability of some action is proportional to its expected utility

$$P(a_1|evidences) \propto EU(a_1|e) \tag{62}$$

$$P(a_1|evidences) = \varrho_1 * EU(a_1|e) \tag{63}$$

Finally, if initially

$$P(a_1|evidences) = P(a_1) \tag{64}$$

then

$$P(r|a_1, evidences) = \frac{P(r, a_1|evidences)}{P(a_1)} \tag{65}$$

and

$$P(a_{n'}) = P(a_n), n \neq n' \tag{66}$$

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

$$\varrho_{n'} = \varrho_n = \frac{1}{k^2 * P(a)} \, , n \neq n' \tag{67}$$

Thus, the values of the proportionality constants ϱ_n between all the Expected Utilities remain the same. This means that an action with a greater probability has a greater expected utility,

$$\begin{aligned} U\left|\Psi'\right\rangle = \; & \omega_1 \left|Result, action_1, evidences, Var_1, ...\right\rangle + \omega_2 \left|Result, action_2, evidences, Var_1, ...\right\rangle \\ & + \omega_3 \left|Result, action_3, evidences, Var_1, ...\right\rangle + \omega_4 \left|Result, action_4, evidences, Var_1, ...\right\rangle + ... \end{aligned}$$

$$\begin{aligned} |\omega_1|^2 = EU(action_1|e) * \varrho \, , |\omega_2|^2 = EU(action_2|e) * \varrho \, , \\ |\omega_3|^2 = EU(action_3|e) * \varrho \, , |\omega_4|^2 = EU(action_4|e) * \varrho \, , ... \end{aligned} \tag{68}$$

Consequently, to choose the action with the greatest probability/utility (Figure 19), 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.

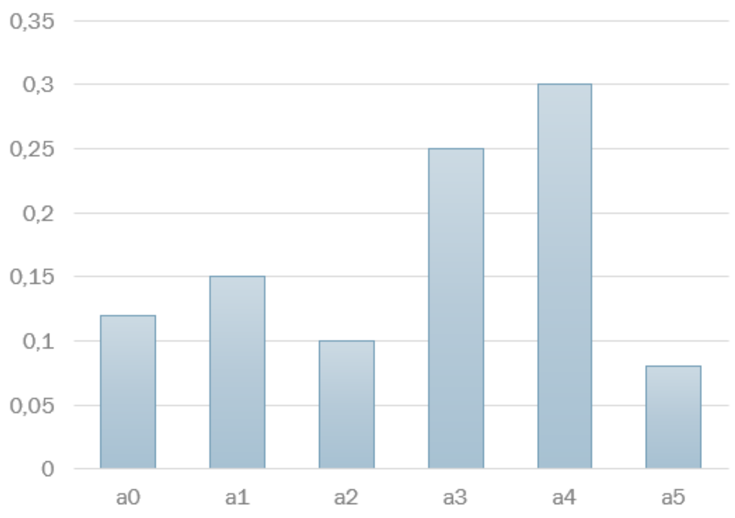


Figure 19.: Probability distribution of an action variable.

Remember, that this decision process requires that Equation (66) 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 in beforehand. Additionally, the action variable should be independent of the evidence variables as in Equation (64), which means that the topology of the network has to be as in Figure 20. 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.

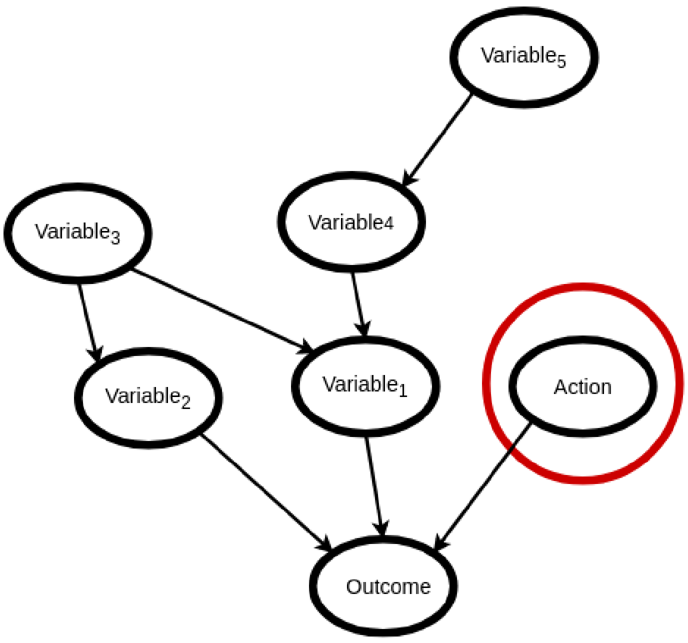


Figure 20.: Bayesian network with an independent action node.

The new quantum decision-making algorithm presented can be completely described by the following iterations:

- Observe the current state of the *evidence* variables.
- Until $(\forall_{n\setminus\{max\}} EU(action_{max}) - EU(action_n) > \delta_{action_{max}} + \delta_{action_n})$:
  - Encode the Bayesian network as a quantum state ($|\Psi_{init}\rangle$).
  - Perform a Grover search for the correct state of the *evidence* variables.

$$|\Psi_{init}\rangle \xrightarrow{\text{Grover}} |\Psi'\rangle = |Result, Action, evidences, Var_1, ...\rangle \tag{69}$$

  - Apply the Utility function (U) to state $|\Psi'\rangle$.

$$|\Psi'\rangle \xrightarrow{\text{Utility function}} |\Psi_{final}\rangle \tag{70}$$

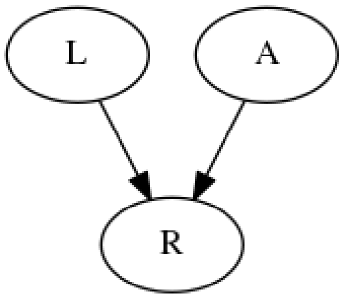
  - Observe the *action* variable of the final state ($|\Psi_{final}\rangle$).
- Return the action with the highest utility.

4.4 PROOF-OF-CONCEPT IMPLEMENTATION

The algorithm presented in the previous Section 4.3 was implemented on the IBM Q quantum simulator as a proof-of-concept. The purpose of the following example is to present a valid instance of the algorithm, which invalidated the use of the real quantum device. At our disposal was IBMs 20-qubit machine, which is based on superconducting circuits Steffen et al. (2011). This machine has 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 too. The output of a circuit with a considerable number of gates would be majorly noise. Due to that, 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. Although, that does not mean that the current devices are completely useless. There are problems where a *Noisy Intermediate-Scale Quantum* (NISQ) devices could have an impact, in the near future Preskill (2018). The applications of these NISQ devices are related to simulations in chemistry and many-body quantum physics. Also, 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).

Over the last years, quantum devices have had a lot of progress. For example, the number of qubits are increasing in each technology, 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. That one day could 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 (Figure 21) was selected for the decision process.



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

Figure 21.: Bayesian network over 3 variables. Node L represents the evidence variable, A the action variable and R the outcome variable.

The network was encoded to a quantum state with use of the tecnique presented in Low et al. (2014), producing the circuit shown in Figure 22.

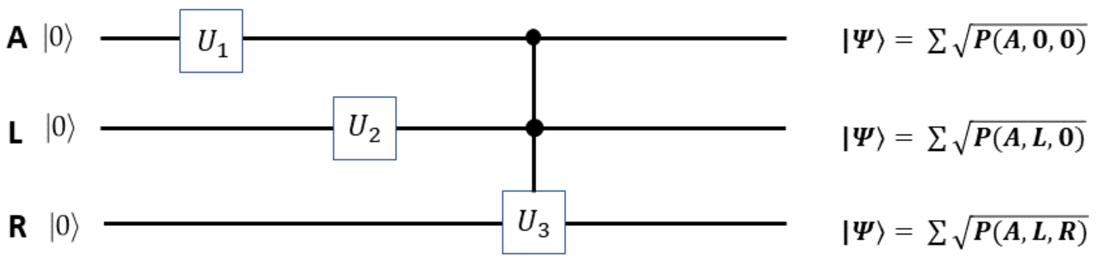


Figure 22.: Quantum circuit composed by rotations and controlled-rotations.

The subsequent application of Grover's algorithm yields the quantum state corresponding to equation:

$$|\Psi_{network}\rangle = \alpha\,|r, A, l\rangle + \beta\,|\neg r, A, l\rangle \tag{71}$$

The incorporation of an utility function over the quantum state requires its normalization, i.e.

$$U(R) = \begin{cases} 7 & , R = r \\ 3 & , R = \neg r \end{cases} \xrightarrow{\text{normalization}} U(R) = \begin{cases} \frac{7}{10} & , R = r \\ \frac{3}{10} & , R = \neg r \end{cases} \tag{72}$$

The normalized utility function generates an isomorphic state whose bases are the domain of the function and the amplitudes its images:

$$\left|\Psi_{utility}\right\rangle = \sqrt{\frac{7}{10}} * |r\rangle + \sqrt{\frac{3}{10}} * |\neg r\rangle \tag{73}$$

At this point the two states in the quantum registers, being mutually independent, can be combined as follows,

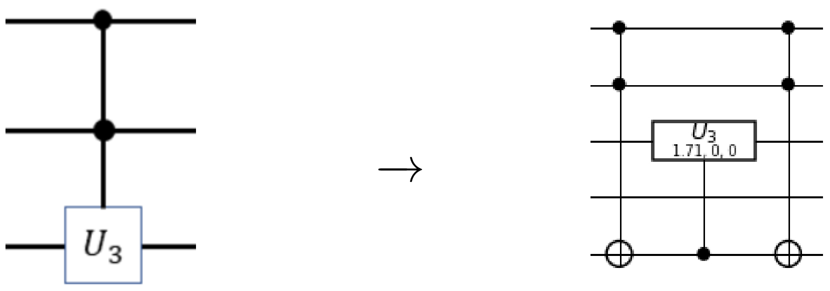
$$\begin{aligned}|\Psi_{total}\rangle = |\Psi_{network}\rangle \otimes \left|\Psi_{utility}\right\rangle = \alpha * \sqrt{\frac{7}{10}}\left|r, A, l, r'\right\rangle + \beta * \sqrt{\frac{7}{10}}\left|\neg r, A, l, r'\right\rangle \\ + \alpha * \sqrt{\frac{3}{10}}\left|r, A, l, \neg r'\right\rangle + \beta * \sqrt{\frac{3}{10}}\left|\neg r, A, l, \neg r'\right\rangle\end{aligned} \tag{74}$$

State $|\Psi_{total}\rangle$ already contains all the relevant terms — all one has to do is to amplify them resorting again to Grover's algorithm. For the example at hands, such is the case when $r \wedge r'$ and $\neg r \wedge \neg r'$ hold, yielding

$$|\Psi_{total}\rangle = \frac{\alpha * \sqrt{\frac{7}{10}}}{k}\left|r, A, l, r'\right\rangle + \frac{\beta * \sqrt{\frac{3}{10}}}{k}\left|\neg r, A, l, \neg r'\right\rangle \tag{75}$$

The resulting state is exactly the one deduced in Equation (68). This state represents the final state of the new algorithm developed during the work of this dissertation. Meaning that the previously presented operations correspond to an possible implementation.

To compute the same algorithm on IBM's quantum simulator, each part has be to converted in a concrete quantum circuit. Every state has to be encoded, which in concept is simple with the use of rotations and controlled-rotations. The major difficulty exists when the rotation is controlled by more than one qubit. In such cases, this operation has to be decomposed in 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 any possible computation is performed on that framework. In concrete, some tools decompose complex operations into simpler Vartiainen et al. (2004); Möttönen et al. (2004). For the Bayesian network chosen only one rotation (Figure 23a) requires a decomposition (Figure 23b).



(a) Multiqubit controlled-rotation. (b) Multiqubit controlled-rotation on QISKIT.

Figure 23.: Decomposition of a two qubit controlled-rotation into available gates on QISKIT.

The next task is to create a circuit version of Grover's search algorithm (Figure 24). To do so, a phase inversion of the correct states has to be performed. In order to obtain the correct circuit for this process the Programmer should make use of the tools referenced before. After the phase inversion, the inversion about the mean is applied to all states. This process can be obtained again by the same tools. Interestingly, the inversion about the mean processes is equal for all Grover searches with the same dimension.

For the decision-making processes presented none of these tools were necessary. An example of a 4-qubit Gover implementation was sufficient Strömberg and Blomkvist Karlsson (2018).

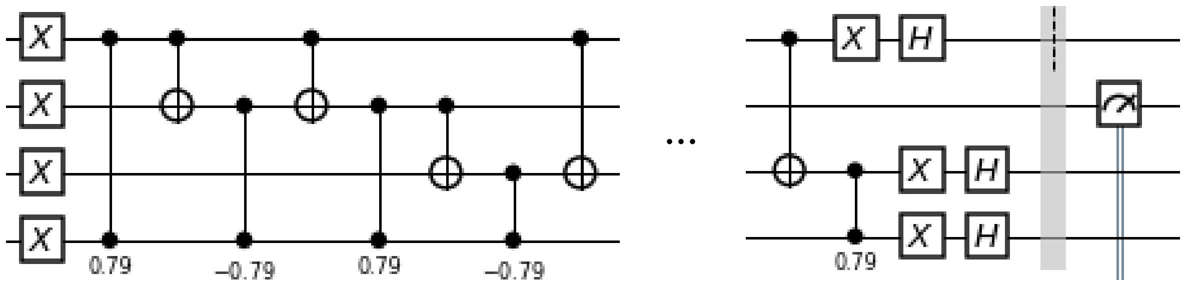


Figure 24.: Circuit representation of Grover's search algorithm for the decision-making process.

Finally, these circuits have to be constructed on "Qiskit" which runs on a "jupyter notebook" (Appendix A). The circuits created are sent as a job to IBM's servers and the results are sent back to the client. The number of times that the circuits are computed is determined by the user. It is important to mention that IBM imposes an upper limit on the number of times that a submitted job is computed. This limit had no interference with our example because only a small amount of samples was necessary.

Actually, a significant number of samples is generated. The number of samples for each state of the action variable should be similar to the theoretical probability. Table 2 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 2 can be explained by deficiencies of the implementation. First, note that Grover's algorithm is probabilistic meaning that the result is never entirely precise. On the other hand, the number of iterations in Grover's algorithm was an integer number; thus, if $n.m$ non-integer iterations are required the usage of the one bellow n or the one above $n+1$, generates a small variation. Actually, it is possible to perform a Grover search with a non-integer number of iterations Zekrifa et al. (2000), but it would not be relevant for this example.

Table 2.: Comparison between theoretical and experimental results after sampling from state (75).

States	Theoretically expected probability	Percentage of Samples
$Action_0$	0,58	0.544
$Action_1$	0,42	0.456

4.5 SUMMARY

Decision-making processes are part of the algorithms that benefit from quantum machines. Especially, the decision-making processes supported by Bayesian networks take advantage of inference algorithms created for such structures. Additionally, the decision-making process can be optimized as seen in this Chapter by exploring the quantum resources deeper or in a different way.

Unfortunately, no quantum computer or quantum device in the world is able to run the presented algorithms. The output of such machines would be majorly noise and the information processed impossible to recover. Decoherence is nowadays the biggest problem to solve in order to construct a faithful quantum computer.

5

COMPLEXITY ANALYSIS

In Chapter 4 different quantum enhanced algorithms were presented. The purpose of this Chapter is to characterize those algorithms in terms of complexity. Complexity is the area of computer science that divides problems in categories according to the computational work involved. The computational work is defined not by absolute values but by the inherent difficulty, which is quantified by the growth rates of the computational work with the dimension of the problem. For instance, NP-problems are known to grow exponentially with the dimension of the problem.

For the analyses, all the decision-making processes were considered. To simplify let us denote by *Process A* the new quantum algorithm (section 4.3), the quantum inference algorithm over the decision network by *Process B* and the decision-making process that computes the conditional probabilities with the same quantum inference algorithm by *Process C*.

All of these algorithms generate samples to determine which is the best action. The number of operations (I_t) in each algorithm will be defined by the number of iterations per sample (I_s) and the number of samples (S) necessary, as in Equation (76).

$$I_t = S * I_s \tag{76}$$

5.1 NUMBER OF ITERATIONS

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

$$I_s = \sqrt{\frac{1}{P(state)}} \tag{77}$$

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

$$P(state) = \sum_r U(r) * P(r,e) \tag{78}$$

knowing that,

$$1 = \sum_r U(r) \tag{79}$$

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

$$\frac{1}{N_r} \tag{80}$$

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

$$\frac{P(e)}{N_r} \tag{81}$$

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

$$\frac{P(e)}{N_r} * \frac{1}{N_r} = \frac{P(e)}{N_r^2} \tag{82}$$

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

$$Mean(P(state)) = \sum_r \frac{P(e)}{N_r^2} = \frac{P(e) * N_r}{N_r^2} = \frac{P(e)}{N_r} \tag{83}$$

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

$$I_s = \sqrt{\frac{N_r}{P(e)}} \tag{84}$$

defining 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

$$P(state) = P(e,a) \tag{85}$$

In this case, we have to apply the requirements determined by Process A described in (64) and (66) to later make a correct comparison

$$P(e,a) = P(e) * P(a) = \frac{P(e)}{N_a} \tag{86}$$

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

$$I_s = \sqrt{\frac{N_a}{P(e)}} \tag{87}$$

As Process C has a goal state with the same probability of Process B, it shares the same number of iterations.

Finally, the mean number of iterations by sample and by process is presented in Table 3.

Table 3.: Number of samples per iterations per sample and per process.

Process A	Process B	Process C
$\sqrt{\frac{N_r}{P(e)}}$	$\sqrt{\frac{N_a}{P(e)}}$	$\sqrt{\frac{N_a}{P(e)}}$

5.2 NUMBER OF SAMPLES

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

$$(pi - \pi_i)^2 = \frac{A * \pi_i(1-\pi_i)}{N}, (i = 1, 2, ..., k) \tag{88}$$

The value A represents the upper $\alpha * 100 - th$ percentile of a *Chi-Square Distribution* (figure 25) 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).

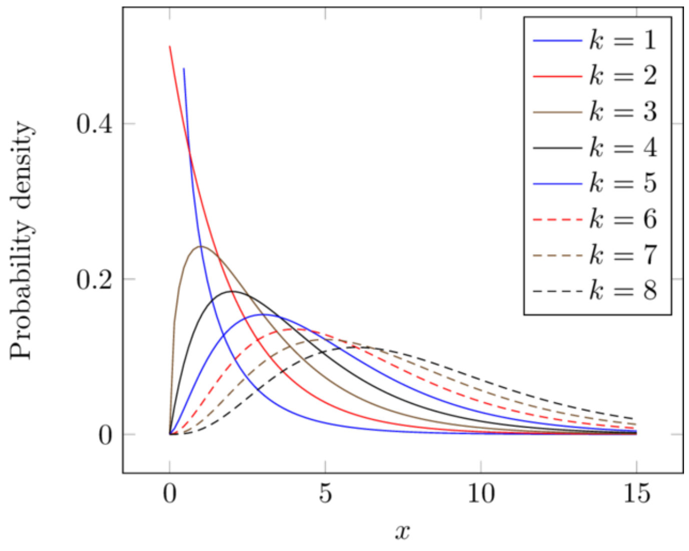


Figure 25.: Chi-square distributions with different degrees of freedom (k).

Writing the same equation in function of N we obtain

$$N = \frac{A * \pi_i(1 - \pi_i)}{\delta^2}, (i = 1, 2, ..., k) \tag{89}$$

Equation (89), defines the number of samples necessary for Process A and Process C. Since, both processes are sampling from a quantum state with multiple bases. Process B requires sampling from binomial states, for which the *Wald Interval* Schuld and Petruccione (2018) will be used:

$$\pi_i = pi \pm z\sqrt{\frac{\pi_i\,(1 - \pi_i)}{N}} \tag{90}$$

in function of N,

$$N = z^2 \cdot \frac{\pi_i\,(1 - \pi_i)}{\delta^2} \tag{91}$$

5.3 TOTAL NUMBER OF OPERATIONS

The total number of operations will be caracterized 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. The total number of operations for each process is shown in Table 4.

Table 4.: Mean number of operations for each process.

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

The computational work of all processes is caracterized as a function of the tree-width of the network (m, m'), the number of nodes (n), the dimension of the result variable (N_r), the dimension of the action variable (N_a), the probability of the evidence variables $(P(e))$, the error (δ) and the probability(π_i) of the state. Additionally, the decision-making process requires inequality (92) to be satisfied. It assures that the decision maker chooses with certainty the best action.

$$\forall_{n \backslash \{max\}} EU(action_{max}) - EU(action_n) > \delta_{action_{max}} + \delta_{action_n} \tag{92}$$

At this point, with use of Table 4 and the restriction defined by expression (92), it is possible to perform a fair comparison between the processes. Initially, it is important to refer that the topology of the network determines if any of the quantum algorithm presented performs the decision-making process faster then a classical one. This happens, because the number of operations needed to encode the network grows exponentially with the the value of tree-width. If the encoding term 2^m is bigger than the quadratic speedup generated by the quantum inference algorithm, than the decision-making process performs faster on a classical computer. Moreover, the same factor destinguishes the between Process C and the remaning ones because the tree-width can change when the utility function is considered. If the utility node, in the decision network is too densely connected (Figure 26) then Process A and Process B could be slower then the classical algorithms, while Process C remains faster. Remark that Process A encodes the initial Bayesian network and the utility function independently. However, when Grover's algorithm is applied to select the correct states (Equation 75) to amplify.It requires approximately the same number of operations than the ones required to encode the decision network.

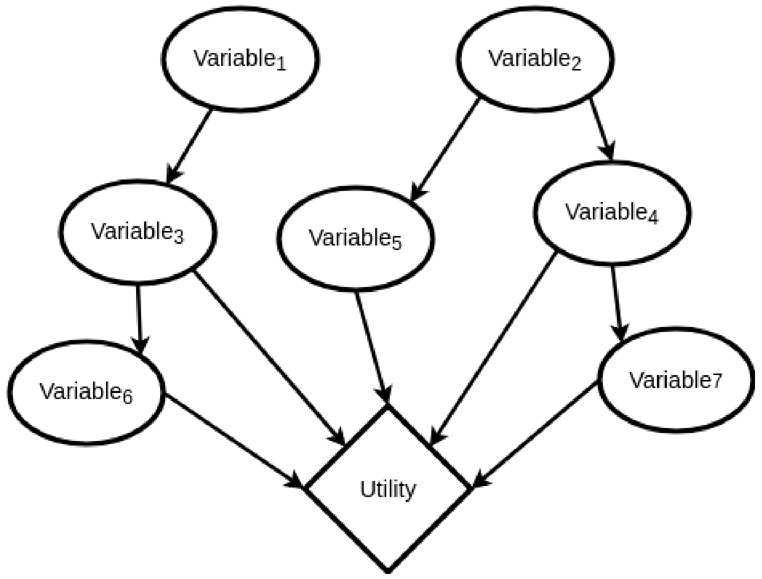


Figure 26.: Decision network with a densly connected utility node.

As seen before, the exponential term 2^m has the greatest impact on the number of iterations of each processes. However, when this number is equal for all process, then other factors are used to distinguish the presented processes. Apart from this term, it is easy to understand that Process B is faster then Process C, because Process B implements the whole decision process one step further with the quantum inference algorithm. Taking advantage during that step from a quadratic speedup.

To compare Process A and Process C it is necessary to consider all terms that are different. So, the error term δ_a for Process A is related to directly sampling values for the Expected utilities, while in Process C the Expected Utility is determined indirectly. For this reason, in Process C it is necessary to apply error propagation rules:

$$EU(a|e) + \delta_{EU(a|e)} = \sum_R (P(Result = r|a,e) + \delta_c) * U(r) \tag{93}$$

before we apply error propagation to this equation we need to normalize the equation in a way that $EU(a|e)/k$ is equal to $P(a)$.

$$P(a) + \delta_a = \sum_R (P(Result = r|a,e) + \delta_c) * F(r) \tag{94}$$

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

$$F(r) = \frac{U(r)}{\sum_a \sum_r U(r) * P(r|a,e)} \tag{95}$$

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

$$F(r) = \frac{U(r)}{\sum_a \sum_r P(r|a,e) * U(r)} = \frac{U(r)}{\sum_a \frac{1}{N_r}} = \frac{U(r)}{\frac{N_a}{N_r}} = \frac{N_r * U(r)}{N_a} \tag{96}$$

$$F(r) = \frac{1}{N_a} \tag{97}$$

Now, writing down the equation that determines the error term δ_a in function of the error term δ_c:

$$\delta_a = \sqrt{\sum_R \delta_c^2 * F(r)^2} \tag{98}$$

using equation 97 we obtain that:

$$\delta_a = \sqrt{\sum_R \delta_c^2 * (\frac{1}{N_a})^2} \tag{99}$$

After that if we assume that δ_c is similar, which is in favor of Process C because it minimizes the δ_a term:

$$\delta_a = \sqrt{N_r * \delta_c^2 * (\frac{1}{N_a})^2} \tag{100}$$

obtaining,

$$\delta_a = (\frac{\sqrt{N_r}}{N_a}) * \delta_c \tag{101}$$

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

$$\sqrt{\frac{N_a}{N_r}} * \frac{A_{r,\alpha} * \frac{1}{N_r} * (1 - \frac{1}{N_r}) * \delta_a^2 * N_a}{A_{a,\alpha} * \frac{1}{N_a} * (1 - \frac{1}{N_a}) * \delta_c^2} \tag{102}$$

Using 101,

$$\sqrt{\frac{N_a}{N_r}} * \frac{N_r}{N_a} * \frac{A_{r,\alpha} * \frac{1}{N_r} * (1 - \frac{1}{N_r})}{A_{a,\alpha} * \frac{1}{N_a} * (1 - \frac{1}{N_a})} \tag{103}$$

also,

$$\sqrt{\frac{N_r}{N_a}} * \frac{A_{r,\alpha} * \frac{1}{N_r} * (1 - \frac{1}{N_r})}{A_{a,\alpha} * \frac{1}{N_a} * (1 - \frac{1}{N_a})} \tag{104}$$

and,

$$\sqrt{\frac{N_r}{N_a}} * \frac{A_{r,\alpha} * (\frac{1}{N_r} - \frac{1}{N_r^2})}{A_{a,\alpha} * (\frac{1}{N_a} - \frac{1}{N_a^2})} \tag{105}$$

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

$$A_{\alpha,k} \geq k + 2 * \log \frac{1}{\alpha} - \frac{5}{2} \tag{106}$$

With this equation it is possible to define a better value for the difference of the computational effort:

$$\sqrt{\frac{N_r}{N_a}} * \frac{(N_r + 2 * \log \frac{1}{\alpha} - \frac{7}{2}) * (\frac{1}{N_r} - \frac{1}{N_r^2})}{(N_a + 2 * \log \frac{1}{\alpha} - \frac{7}{2}) * (\frac{1}{N_a} - \frac{1}{N_a^2})} \tag{107}$$

As

$$\lim_{N_r \to \infty} (N_r + 2 * \log \frac{1}{\alpha} - \frac{7}{2}) * (\frac{1}{N_r} - \frac{1}{N_r^2}) = 1 \tag{108}$$

and,

$$\lim_{N_a \to \infty} (N_a + 2 * \log \frac{1}{\alpha} - \frac{7}{2}) * (\frac{1}{N_a} - \frac{1}{N_a^2}) = 1 \tag{109}$$

It is possible to approximate the expression to:

$$\sqrt{\frac{N_r}{N_a}} * \frac{(N_r + 2 * \log \frac{1}{\alpha} - \frac{7}{2}) * (\frac{1}{N_r} - \frac{1}{N_r^2})}{(N_a + 2 * \log \frac{1}{\alpha} - \frac{7}{2}) * (\frac{1}{N_a} - \frac{1}{N_a^2})} \approx \sqrt{\frac{N_r}{N_a}} \tag{110}$$

The result shows that A is faster when the *result* variable has a greater dimension than the action variable. When the opposite happens then Process C is faster. Theoretically, if the decision-making process is applied to a real world application. The dimension of the environment should be exponentially greater then the number of actions that the agents has at his disposal. Also, the number of possible results increases if the decision-making process computes outcomes further in time. It is expected that Process A is faster then Process C most of the times.

Finally, only one comparison is left. Process A has to be compared to Process B. Again the error propagation rules are required. Both processes are sampling Expected utility but their magnitudes are different:

$$P(a) + \delta_a = (P(u|a,e) + \delta_b) * F(r) \tag{111}$$

The mean value for $P(a)$ is $\frac{1}{N_a}$, while the mean value for $P(u|a,e)$ is $\frac{1}{2}$. So the mean value for $F(r)$ has to be:

$$F(r) = \frac{2}{N_a} \tag{112}$$

therefore,

$$\delta_a = \sqrt{(\frac{2}{N_a})^2 * \delta_b^2} \tag{113}$$

and,

$$\delta_a = (\frac{2}{N_a}) * \delta_b \tag{114}$$

Repeating the processes as before, the relation between both process is described by:

$$\sqrt{\frac{N_a}{N_r}} * \frac{z * \frac{1}{2} * (1 - \frac{1}{2}) * \delta_a^2 * N_a}{A_{a,\alpha} * \frac{1}{N_a} * (1 - \frac{1}{N_a}) * \delta_b^2} \tag{115}$$

with 114,

$$\sqrt{\frac{N_a}{N_r}} * \frac{z * \frac{1}{2} * (1 - \frac{1}{2}) * N_a * 4}{A_{a,\alpha} * \frac{1}{N_a} * (1 - \frac{1}{N_a}) * N_a^2} \tag{116}$$

yielding,

$$\sqrt{\frac{4}{N_r * N_a}} * \frac{z * \frac{1}{2} * (1 - \frac{1}{2})}{A_{a,\alpha} * \frac{1}{N_a} * (1 - \frac{1}{N_a})} \tag{117}$$

using Equation 109 again, the total term tends to:

$$\sqrt{\frac{c}{N_r * N_a}}, c = constant \tag{118}$$

Meaning that as the dimension of the problem increases, Process B gets faster then Process A. There will be values for N_r and N_a for which Process A is faster, but in the long run Process B solves most of the cases faster.

It is of utmost importance to refer that the comparison made was for decision problems where the action with the highest expected utility is pretended, which is, in general, the goal. However, some problems are slightly different and this can change which one of the algorithms performs better.

For instance, in reinforcement learning the action with the highest expected utility is not always desired. In these models, the agent selects one action from all possible actions, considering their expected utility. Thus, actions with higher expected utility are selected with more frequency and vice versa. In order to do so, the agent has to compute the expected utility for every action and then randomly choose one of the actions respecting the probability distribution. All classical solutions require this procedure and two of the quantum algorithm developed too (Process B and Process C). However, Process A has a great advantage for this problem. Their samples are obtained exactly by the distribution pretended, which means that only one sample from Process A solves this decision problem. Therefore Process A reveals itself to be both efficient and useful for applications in reinforcement learning.

5.4 SUMMARY

The complexity of a problem determines if a solution is found in a reasonable amount of time. There are problems that not even the best supercomputers in the world would find a solution in our lifetime. So, it is important to know the complexity of the problems we want to solve in the first place. Additionally, all extra information can help to reduce the time of computation.

For the decision-making process, a precise definition of the complexity would enable us to choose the fastest process to reach the optimal action. For all processes presented in the previous chapter, there is now a precise definition (Table 4). Also, a complete comparison between the processes was presented. Distinguishing when each of the processes has some advantage over the remaining, in a typical decision problem.

6

PICTURING A QUANTUM BAYESIAN ALGORITHM

6.1 STRING DIAGRAMS

In computer science, it is very important to verify the correctness of programs. Programs should be considered correct not by repeated experimental verification but by mathematical proof. In some applications, a software ”bug” could stay for months unseen and then cost millions or even lifes. Due to that, a great amount of work was done in this direction Harton et al. (2008); Jacobs (1999); Burstall (1972). Frameworks that help software developers to verify their programs are now available.

The theory of software verification is strongly connected to category theory. Category theory is a mathematical subject that formalizes mathematical structures. This kind of abstraction applied to programs allows to understand their behaviour without computing concrete values. Also, more generic proprieties are proved in this framework.

Formal program analysis is not limited to classical programs running on classical Turing machines. The same idea can be applied to quantum computers. To verify proprieties about the quantum algorithms, a category that contains all basic proprieties of quantum mechanics has to be considered. In Coecke (2010), Coecke considered the dagger monoidal categories as a semantic framework for quantum computation. Additionally, he describes a graphical language, known as string diagrams, which represents reasoning in such a category. Every proof that can be performed in the categorical notation, can also be performed in the corresponding graphical language, if the axioms of the category are respected by the graphical language Coecke and Spekkens (2012). To understand how algorithms are represented in this frameworks, a better understanding of categories is required.

A category $\mathbf{C}$ is constituted by objects $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$ and morphisms $\mathbf{f} : \mathbf{A} \to \mathbf{B}$, $\mathbf{g} : \mathbf{B} \to \mathbf{C}, \ldots$. Every object has an unique morphism to itself $\mathbf{1_A} : \mathbf{A} \to \mathbf{A}$ and morphisms are composable:

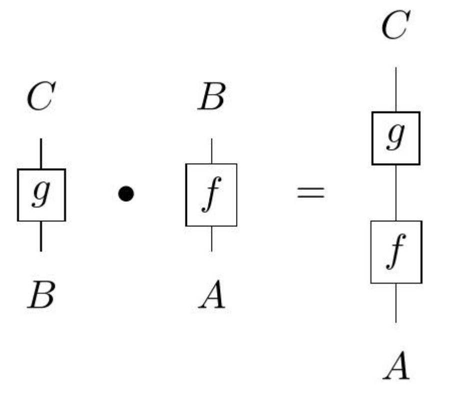
$$\begin{tikzcd}
A \arrow[loop left, "1_A"] \arrow[r, "f"] \arrow[dr, "g\circ f"'] & B \arrow[loop right, "1_B"] \arrow[d, "g"] \\
& C \arrow[loop right, "1_C"]
\end{tikzcd}$$

Figure 27.: Composition of morphisms.

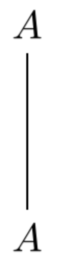
The composition of morphisms is associative:

$$l \bullet (g \bullet h) = (l \bullet g) \bullet h \tag{119}$$

These are the most basic elements of a category. The structure can be richer when different kinds of categories are considered. In the corresponding graphical language, the morphisms are associated with boxes and the objects to strings. Thereby the composition of morphisms and the identity morphism are represented as follows:



(a) Composition of morphism represented in graphical language.



(b) Representation of the identity morphism in graphical language.

From Figure 28a, it easy to see morphisms could represent processes and objects information or data types. Also, the composition of morphism represents the sequential composition of processes. To include the notion of parallel composition a monoidal structure is required. This category is equipped with a bifunctor (morphism between categories) $\mathbf{F} : \mathbf{C} \otimes \mathbf{C} \to \mathbf{C}$ and a unit object $\mathbf{I}$. The functor respects associativity:

$$l \otimes (g \otimes h) = (l \otimes g) \otimes h \tag{120}$$

and the unit object behaves as an identity element to the functor:

$$I \otimes A = A = A \otimes I \tag{121}$$

Additionally, a pair morphisms that are parallel composed are composable to:

$$(g \otimes f) \bullet (h \otimes l) = (g \bullet h) \otimes (f \bullet l) \tag{122}$$

in the graphical language the such a composition is represented as,

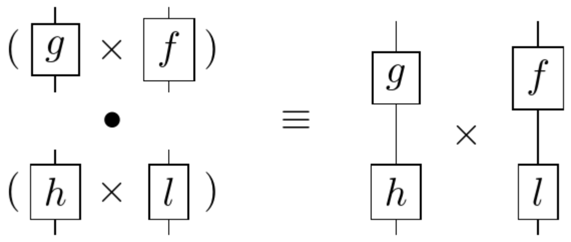


Figure 29.: Parallel composition of a pair of morphisms.

Monoidal categories are symmetric when the isomorphisms $\sigma_{A,B} : A \otimes B \to B \otimes A$ exist for all objects A and B. Those isomorphisms are represented graphically as:

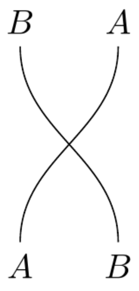


Figure 30.: The $\sigma_{A,B} : A \otimes B \to B \otimes A$ isomorphism represents the swap operator within a process a theory

In the quantum case, the use of parallel composition enables the representation of multipartite systems and is known in the quantum formalism as the tensor product.

The representation of quantum operations is related to morphisms between states. States are described by Coecke as a morphisms $\mathbf{e} : \mathbf{I} \to \mathbf{A}$ from the zero object ($\mathbf{I}$) to another object ($\mathbf{A}$) Coecke and Kissinger (2017). In string diagrams, the representation of states is the following:

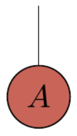


Figure 31.: Representation of a state in a graphical language.

States and processes are described by a monoidal category without any fault. The notion of a dagger category is required to describe the duality present in quantum mechanics. Such a category has an involution functor $\mathbf{C}^{\mathrm{op}} \to \mathbf{C}$ that associates each morphism $\mathbf{f} : \mathbf{A} \to \mathbf{B}$ to his "adjoint" morphism $\mathbf{f}\dagger : \mathbf{B} \to \mathbf{A}$. The same occurs for every state $\mathbf{e} : \mathbf{I} \to \mathbf{A}$ where $\mathbf{e}\dagger : \mathbf{A} \to \mathbf{I}$. The "adjoint" object of a state is called an effect in Coecke and Kissinger (2017). For quantum algorithms, an effect behaves as a measurement and is represented in string diagrams as:

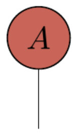


Figure 32.: Representation of an effect (dagger state) in a graphical language.

A dagger category is structurally rich enough to represent complete quantum algorithms and protocols, while string diagrams are used as a useful syntax. In Coecke and Kissinger (2017), the most general quantum algorithms and protocols are represented and studied with the corresponding graphical language. For instance, Grover's search algorithm is represented in this framework as in Figure 33. This abstract representation shows that Grover's algorithm is composed of equal unitary iterations. Also, every iteration can be decomposed in a phase inversion, applied to the selected elements, and an inversion about the mean.

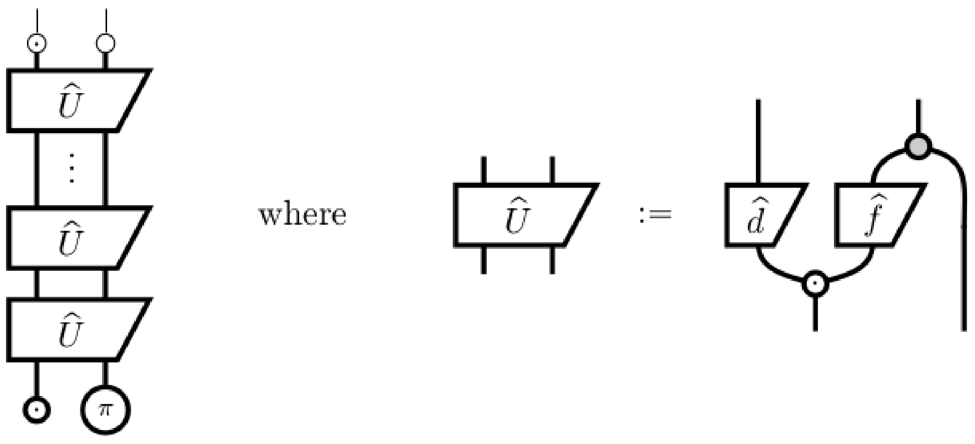


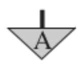
Figure 33.: Grover's algorithm represented in string diagrams (from Coecke and Kissinger (2017)).

Afterward, the authors used this representation to prove that any search algorithm composed of unitary operations can not be faster than Grover's search algorithm. Such a result is very important to understand how faster a quantum computer can solve problems that are search driven.

6.2 OTHER USES OF STRING DIAGRAMS

String diagrams are normally used to describe quantum processes or protocols. However, their use extends to different applications. Some of these applications are of interest to this dissertation. This happens, because they use string diagrams to describe topics discussed in previous chapters.

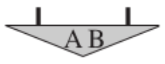
In Coecke and Spekkens (2012), the authors present string diagrams for classical and quantum Bayesian inference. They use symmetric monoidal categories with compact structures and Frobenius structures. With the considered category and structures they obtain a very general framework to describe classical inference. For instance, the morphisms from the zero object (**I**) to another object (**A**), **e** : **I** → **A**, in this framework, are representations of normalized states,



$$: \mathrm{I} \to A = (p_1, p_2, \ldots, p_n)$$

Figure 34.: Graphical representation of a normalizes state and its semantics in probability theory (from Coecke and Spekkens (2012)).

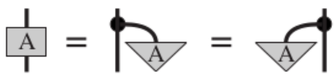
and as before the monoidal structure equips the framework with the parallel composition, used to represent multi-variable states,



$$: \mathrm{I} \rightarrow A \otimes B = (p_{1,1}, p_{1,2}, \ldots, p_{n,m})$$

Figure 35.: Graphical representation of a normalizes multi-variable state (from Coecke and Spekkens (2012)).

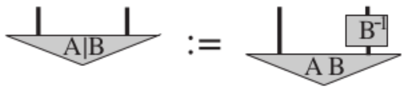
another important piece is the existence of modifiers, which are the representation of transformations,



$$: A \rightarrow A = M[(p_1, p_2, \ldots, p_n) \otimes I] = \begin{pmatrix} p_1 & 0 & \ldots & 0 \\ 0 & p_2 & \ldots & 0 \\ \vdots & & & \\ 0 & 0 & \ldots & p_n \end{pmatrix}$$

Figure 36.: Graphical representation of a state transformer (from Coecke and Spekkens (2012)).

With the use of these elements, a conditional state can be constructed simply by,



$$: \mathrm{I} \rightarrow A \otimes B = (p_{i|j} | i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\})$$

Figure 37.: Conditional states in the graphical language (from Coecke and Spekkens (2012)).

and considering that it is a symmetric monoidal category, the morphism from Figure 30 exist for every pair of objects, allowing the construction of Bayes theorem as,

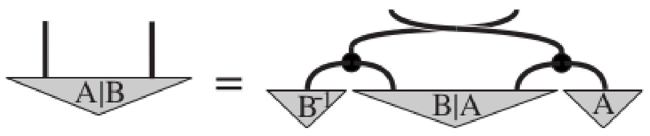


Figure 38.: Bayes Theorem written in a graphical language (from Coecke and Spekkens (2012)).

Also, the notion of conditional independence is supported by this framework. The general independence rules between the two elements are represented in this framework by four graphical Equations (39). These equations are very different from the conditional independence rules that were presented in Chapter 2 for Bayesian networks. In a Bayesian network, testing conditional independence boils down to simple tests on the topology of the network.

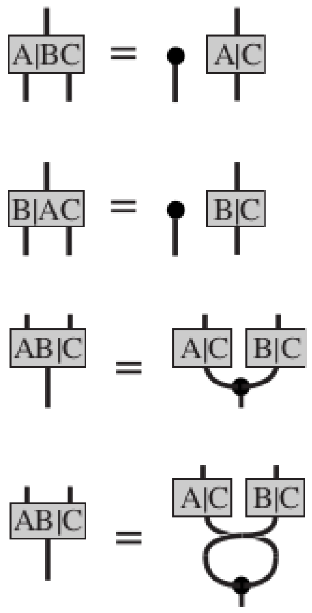


Figure 39.: Conditional independence rules written in Coecke and Spekkens (2012) graphical language.

For the quantum case, the authors presented a graphical calculi for dagger compact categories. The description is oriented for the operators and traces. Allowing, therefore, the representation of mixed states (mixed states are quantum states that are not correctly defined by one wave function). Altogether, the authors depicted a very abstract version of inference. This kind of abstraction can be helpful to find solutions and provide proofs about those subjects.

Another interesting application was presented in Hedges et al. (2016). The authors applied string diagrams to the theoretical notion of game theory. String diagrams are a very flexible language, they express completely different subjects by simply exchanging the underlying category, changing with that their semantics. That work intends to study the composability of games. Game theory studies the interaction of intelligent players in a game. These games can be simple or complex. In some cases, a game can be formed by the composition of smaller games. That work starts by defining simple games, which are called open games. They are the building blocks and their algebraic expression is the following:

$$G : X \otimes S^* \to Y \otimes R^* \tag{123}$$

which in string diagrams has the following representation:

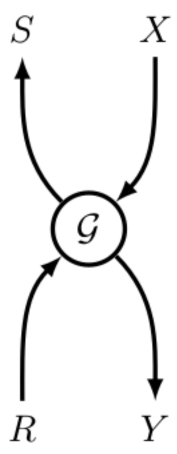


Figure 40.: Open game depicted in a graphical language (from Hedges et al. (2016)).

The variables X, S, Y, R are representative of sets, having each of them their one role in the game:

- X represents histories and observations possibles in G. The strategies of the players in G may be defined by X. Also, X can be partial observable by a player in G.
- Y is the set of possible choices in G, meaning that Y works as an output value of G.
- R is the set of outcomes that each player in G is trying to optimize.
- S represents the utility generated by G and is called *coutility*. Players in G do not generate coutilities but players outside G do. In this way, S serves as a mechanism to communicate back to the earlier stages of the game.

These games are composable in both ways, sequential:

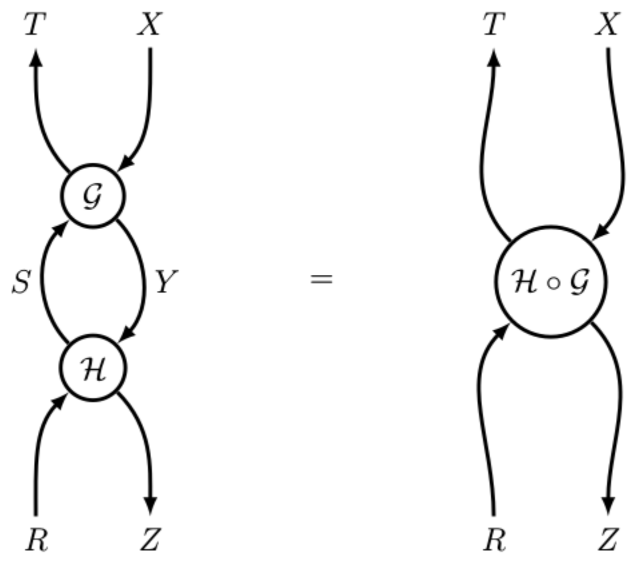


Figure 41.: Sequential composition of open games (from Hedges et al. (2016)).

and in parallel, with a tensor product:

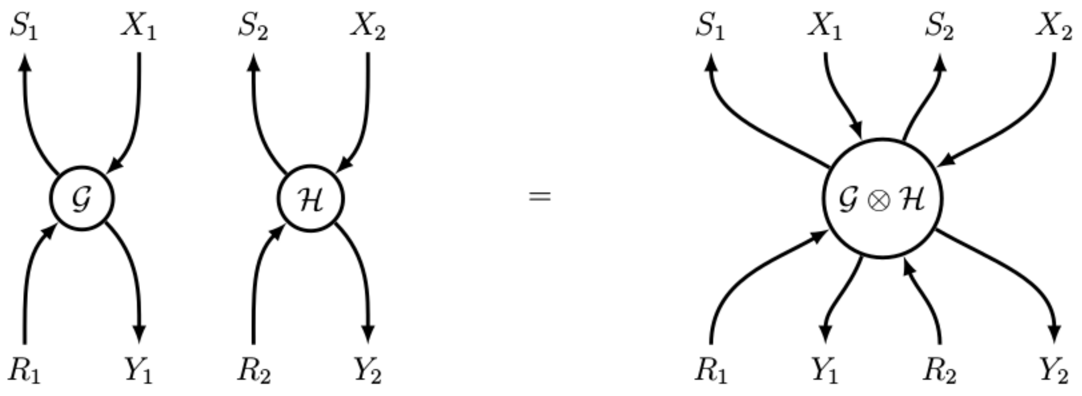


Figure 42.: Parallel composition of open games (from Hedges et al. (2016)).

These games are able to represent decision problems like the one discussed in Chapter 4. For that, a game will be simplified to one player making only one move. In Figure 43, a single-player decision problem is presented. The player receives observations X, then he makes a decision and outputs and action Y. Afterward, the player receives a utility value for that action. It is important to notice that the player needs a relational order for utility values. Without a relational order, the player would not distinguish good choices from bad ones.

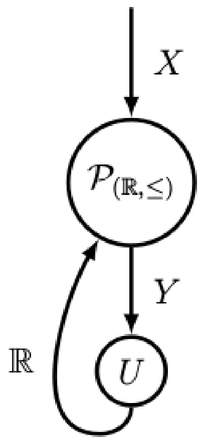


Figure 43.: Graphical representation of a decision problem (from Hedges et al. (2016)).

Furthermore, the same paper uses the definitions presented before to study the composition and coordination of games composed of a higher number of elements. These cases are difficult to study, but with this graphical framework, some operations are made simpler. For instance, the composition of games becomes something very simple to perform and analyze.

6.3 ALGORITHMIC ANALYSIS

A new algorithm for decision-making processes and its implementation was presented in section 4.3 and section 4.4, respectively. It is based on a quantum inference algorithm composed with the application of a utility function. Both processes involve the use of Grover's search algorithm. Due to that, it would be of interest to prove that **two composed quantum search algorithms are equivalent to one search with the union of the restrictions.** If the previous affirmation holds then the new decision-making process can be done with only one Grover search.

First, the composition of two quantum-maps will be analyzed. Quantum-maps are processes used in quantum algorithms to invert the phase of certain elements. Grover's search algorithms use these processes to select the elements intended to be found. Afterward, the same elements are amplified by a process called inversion about the mean. So, is **the composition of two quantum-maps equivalent to a quantum-map with the union of the restrictions?** In Figure 44, a quantum map is represented in string diagrams with two different abstractions. This abstraction will help us to find an answer to the previous question.

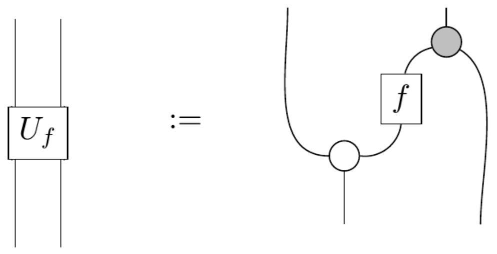


Figure 44.: Unitary transformation represented in string diagrams.

So, the composition of two quantum maps is:

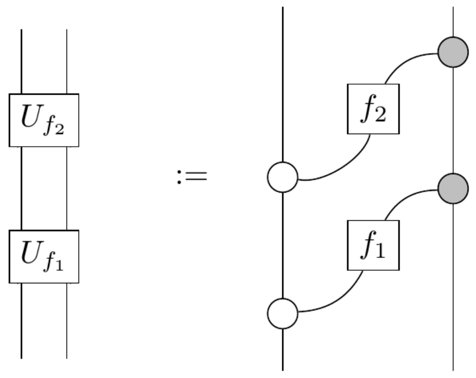


Figure 45.: Composition of unitary transformations in a string diagrams.

The string diagrams on the right hand side can be modeled in a way to respect the axioms of the underlying category:

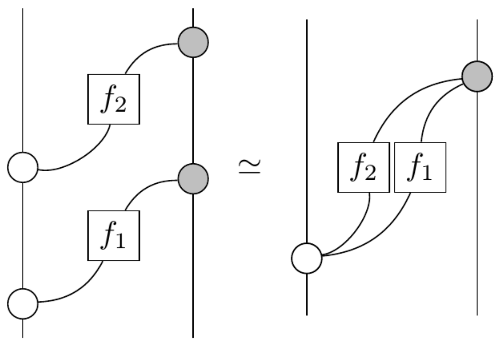


Figure 46.: Equivalent diagrams obtained by one graphical transformation.

While the quantum map with the union of the restrictions is represented by:

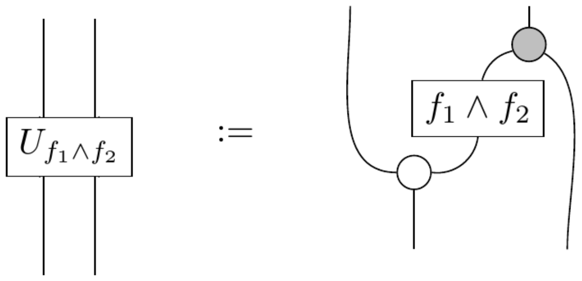


Figure 47.: Equivalent representation of the an unitary tansformation.

and after decomposing the same in simpler operations,

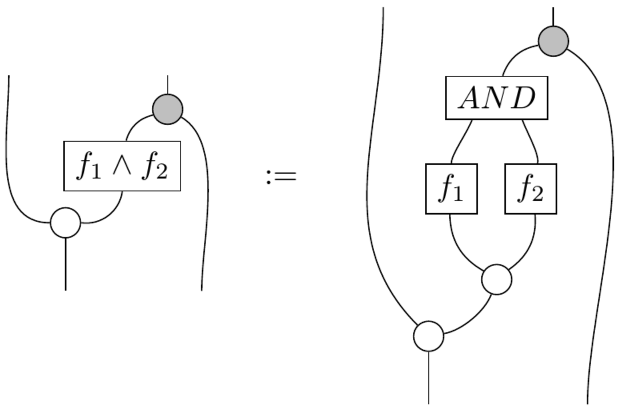


Figure 48.: Equivalent representation after a decomposition in simpler operations.

and,

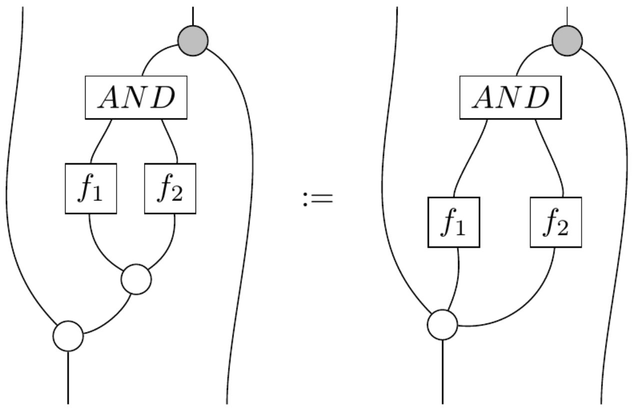


Figure 49.: Equivalent diagrams with use of one graphical transformation.

Since the two branches are different with respect to the AND operator, the processes are not equivalent. Another way to obtain the same result is to use the definition of the elements in the string diagram and show that they are not equivalent. From Coecke and Kissinger

(2017), the white dot operates as copy process, while the grey dot works as XOR process, meaning that the branch generated by the composition of maps (Figure 46) translates to:

$$\begin{aligned} XOR \bullet (f_1 \otimes f_2) \bullet COPY(x) &= \\ = XOR \bullet (f_1 \otimes f_2) \bullet (x \otimes x) &= \\ = XOR \bullet (f_1(x) \otimes f_2(x)) &= \\ = XOR(f_1(x), f_2(x)) & \end{aligned} \tag{124}$$

While the same branch for the quantum map with the union of the restriction translates to:

$$\begin{aligned} AND \bullet (f_1 \otimes f_2) \bullet COPY(x) &= \\ = AND \bullet (f_1 \otimes f_2) \bullet (x \otimes x) &= \\ = AND \bullet (f_1(x) \otimes f_2(x)) &= \\ = AND(f_1(x), f_2(x)) & \end{aligned} \tag{125}$$

Demonstrating that the branches are different:

$$XOR(f_1(x), f_2(x)) \neq AND(f_1(x), f_2(x)) \tag{126}$$

With that, a second proof was given. **The composition of two quantum maps does not correspond to the quantum map with the union of the restrictions.** There is an easy intuition to understand why these processes are not equivalent. Imagine that some element is selected by the first quantum map and is again selected by the second quantum map. It obtains a phase shift twice which is equivalent to none. Consequently, the elements that we are searching with the union of the restrictions are not selected.

Now we pursue to verify if something similar is true when we compose Grover's algorithm. Classically, there is no doubt that if a search process is composed with another one, then the outcome is the same as a search with the union of the restrictions. The intent is to prove the same for a quantum search. In string diagrams, we want to prove that the equivalence shown in Figure 50 is true.

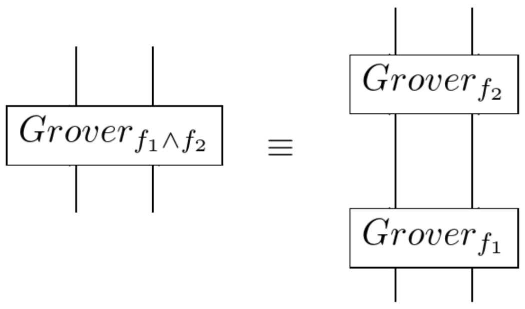


Figure 50.: Illustration 1.

The left hand side of Figure 50 generates as result the state represented in Figure 51.

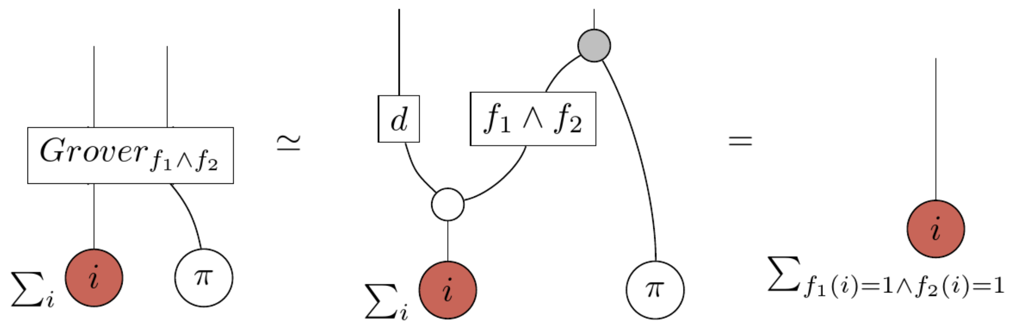


Figure 51.: Illustration 2.

To prove the equivalence in Figure 50 holds, then the right hand side has to reduce to the same final state. For that, the string diagram can be transformed as shown in Figure 52.

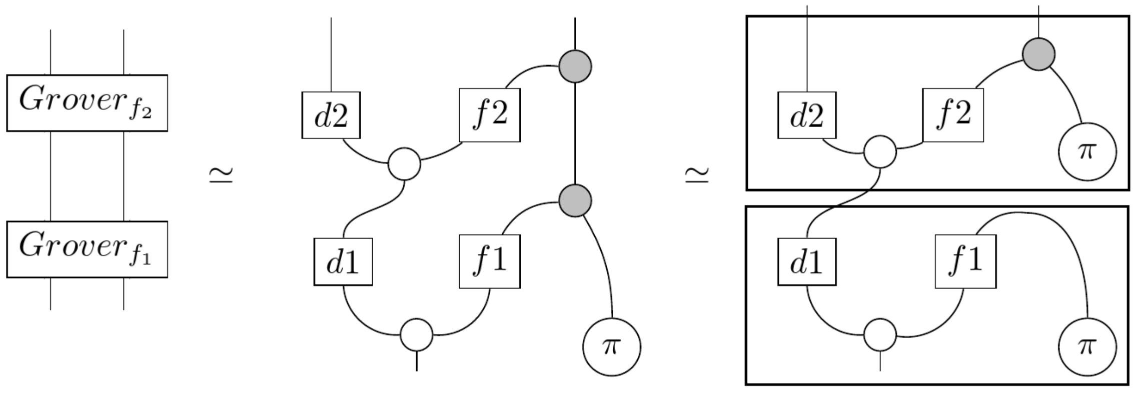


Figure 52.: Illustration 3.

Proceeding with the transformations, the first block of operations in the last diagram of Figure 52 can be reduces to the state in Figure 53.

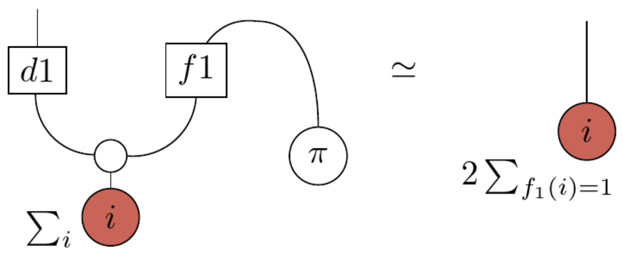


Figure 53.: Illustration 4.

Which can be composed to the second block of operations, as shown in Figure 54.

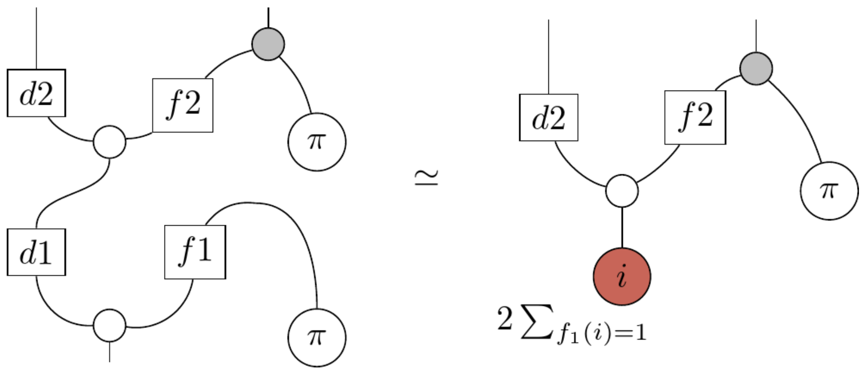


Figure 54.: Illustration 5.

Using a decomposition method presented in Coecke and Kissinger (2017), the same can be represented as in Figure 55.

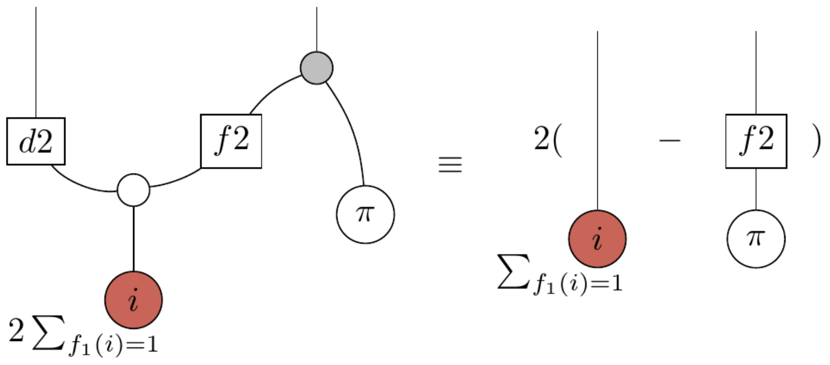


Figure 55.: Illustration 6.

This decomposition can be taken one step further as shown in Figure 56.

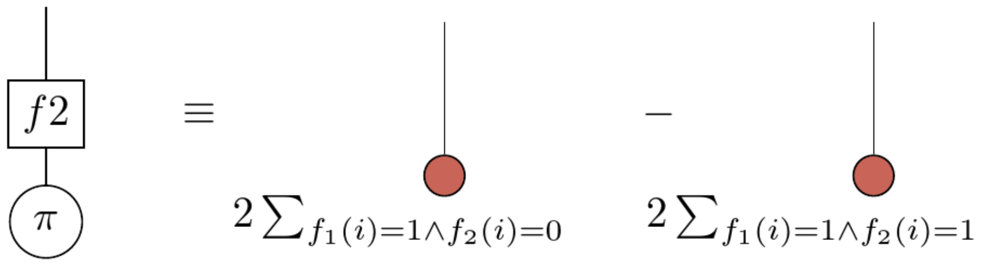

f2
π
≡
$2\sum_{f_1(i)=1\wedge f_2(i)=0}$
−
$2\sum_{f_1(i)=1\wedge f_2(i)=1}$

Figure 56.: Illustration 7.

Resulting in the final state, which is represented in Figure 57.

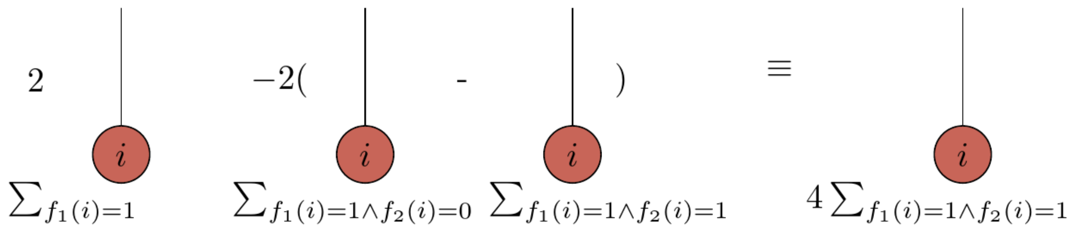

2
$\sum_{f_1(i)=1}$
$-2($
$\sum_{f_1(i)=1 \wedge f_2(i)=0}$
-
$\sum_{f_1(i)=1 \wedge f_2(i)=1}$
$)$
$\equiv$
$4\sum_{f_1(i)=1 \wedge f_2(i)=1}$
i

Figure 57.: Illustration 8.

The final state contains exactly the base elements pretended. The only difference is the scalar value 4 which has no impact on the result. This proves that **two composed quantum search algorithms are equivalent to one search with the union of the restrictions**.

For the new algorithm presented in section 4.3, this means that the two different implementations generate the same result. Therefore, the utility function can be applied after the Grover search for the correct values of the evidence variables or within. The computational complexity of both implementations is exactly the same. So the Programmer should choose the one he prefers.

6.4 SUMMARY

In computer science, the use of formal methods to verify the correctness of algorithms is well known. However, their use does not only extend to academic research, but also a variety of industrial applications rely on them to provide the safety of their workers or clients. For instance, the software running on airplanes should never fail, for the safety of the occupants.

With the increasing investment in quantum computers and quantum software, the research of formal methods applied to quantum algorithms has grown. This chapter focused on string diagrams which are a graphical framework used to verify and prove proprieties about algorithms. They revealed themselves a very powerful tool, since they have the capacity to describe an algorithm at different levels of abstraction in a graphical way.

7 CONCLUSION

7.1 CONCLUSIONS

The dissertation aimed to study quantum variants of Bayesian networks and its corresponding algorithms. The review of the classical literature, demonstrated that Bayesian networks are a powerful tool to describe probabilistic systems. Therefore, they are found in a variety of domains and applications. Unfortunately, inference over those structures revealed to be an NP-complete problem. Meaning that the amount of computational work grows exponentially with the dimension of the problem. As a result, none of the classical algorithms developed to perform inference over Bayesian networks represents an efficient solution.

In the quantum domain, a quantum version of a Bayesian network was presented, with the capacity to model human behaviour. The advantage of such a quantum Bayesian network was an exponential reduction in the dimension of the model. For the inference problem, no quantum algorithm has an exponential speedup. However, a quantum algorithm for Bayesian inference was analyzed, which has a quadratic speedup for some networks.

On a second contribution, quantum algorithms for decision-making processes were developed. These algorithms are based on the quantum algorithm for inference previously presented. They take advantage of a quadratic speedup during the inference process, which is a subprocess of the whole process. Additionally, the possibility to further explore the quantum resources was considered and resulted in a completely ”quantum” decision-making process. In general, quantum computing has a great impact on artificial intelligence Schuld and Petruccione (2018), since it provides speedups for a variety of techniques. Some of the algorithms employ the quantum machine for subprocesses, while others are completely quantum driven.

It was under the scope of this dissertation to figure out the computational complexity of the algorithms developed. Therefore, the computational work was quantified for each one of the algorithms, allowing with it an optimal choice between them, when the decision problem is well defined. The results showed that not all cases benefit from the use of a quantum device. However, the ones who showed themselves to be candidates for computational speed-up are promising.

Later on, a formal analysis was performed on one of the algorithms. In order to do so, string diagrams were studied. These are a mathematical framework to study quantum algorithms and protocols. The results demonstrated that two different implementations generated the same result, thus offering the "quantum Programmer" a choice between both options.

In conclusion, quantum computing is an active research topic and new discoveries are expected in the next years. The work performed during this dissertation expanded the domain knowledge, with an algorithm that benefits from the structure present in the data, which no classical algorithm could benefit from. Therefore, it should serve as an example that new and completely different solutions are expected, as a result of the intrinsic differences that exist amongst classical and quantum computations.

7.2 PROSPECTS FOR FUTURE WORK

Although the work presented in this dissertation led to a real application, some ideas could be further extended. For instance, the decision-making process discussed here was related to a static model, in which, neither the utility function nor the Bayesian network changed 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 intelligent agent to adapt its behaviour to a changing environment, would result in better outcomes, raising with that the number of possible applications. Additionally, it should be considered to be performed on the quantum device for potential speedups or even better results.

Another idea that could be further explored is the use of a quantum device to build a Bayesian network from data. This task has great applications in data science and is known to be a computationally difficult problem. The review of the literature revealed that there are some approaches to solve this problem more efficiently using a quantum device O'Gorman et al. (2014); Tucci (2014). However, the number of solutions is small compared to what is already known in a classical setting. It would be worth checking for more efficient solutions and its implications for quantum complexity theory.

BIBLIOGRAPHY

Gustavo Arroyo-Figueroa and Luis Enrique Sucar. A Temporal Bayesian Network for Diagnosis and Prediction. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, pages 13–20, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-614-9. URL http://dl.acm.org/citation.cfm?id=2073796.2073798.

Alain Aspect, Philippe Grangier, and Gérard Roger. Experimental Tests of Realistic Local Theories via Bell's Theorem. *Phys. Rev. Lett.*, 47(7):460–463, 1981. doi: 10.1103/PhysRevLett.47.460. URL https://link.aps.org/doi/10.1103/PhysRevLett.47.460.

Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi. Solving #SAT and Bayesian Inference with Backtracking Search. *Journal of Artificial Intelligence Research*, page 52, 2009.

Jennifer Barry, Daniel T. Barry, and Scott Aaronson. Quantum partially observable Markov decision processes. *Physical Review A*, 90, 2014. doi: 10.1103/PhysRevA.90.032311.

J Bongard. Probabilistic Robotics. Sebastian Thrun, Wolfram Burgard, and Dieter Fox. (2005, MIT Press.) 647 pages. *Artificial Life*, 14(2):227–229, 2008. ISSN 1064-5462. doi: 10.1162/artl.2008.14.2.227.

Hans J Briegel and Gemma De las Cuevas. Projective simulation for artificial intelligence. *Scientific Reports*, 2:400, may 2012. URL https://doi.org/10.1038/srep00400http://10.0.4.14/srep00400.

Rod M Burstall. An Algebraic Description of Programs with Assertions, Verification and Simulation. *SIGPLAN Not.*, 7(1):7–14, 1972. ISSN 0362-1340. doi: 10.1145/942578.807068. URL http://doi.acm.org/10.1145/942578.807068.

Peter C. Fishburn. *The Foundations of Expected Utility Theory*. 1982. doi: 10.1007/978-94-017-3329-8.

C.Huang and A.Darwiche. Inference in belief networks: A procedual guide. *International Journal of Approximate Reasoning*, pages 225–263, 1996.

B Coecke and A Kissinger. *Picturing Quantum Processes: {A} First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

Bob Coecke. Quantum picturalism. *Contemporary Physics*, 51(1):59–83, jan 2010. doi: 10.1080/00107510903257624.

Bob Coecke and Robert W Spekkens. Picturing classical and quantum Bayesian inference. *Synthese*, 186(3):651–696, jun 2012. ISSN 1573-0964. doi: 10.1007/s11229-011-9917-5. URL `https://doi.org/10.1007/s11229-011-9917-5`.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN 0262033844, 9780262033848.

A. Darwiche. Chapter 11 Bayesian Networks. *Foundations of Artificial Intelligence*, 3(07): 467–509, 2008. ISSN 15746526. doi: 10.1016/S1574-6526(07)03011-8.

Alan Mackworth David Poole. Artificial Intelligence - foundations of computational agents – 9.3.1 Decision Networks, 2010. URL `https://artint.info/html/ArtInt{_}219.html`.

Arnaud Doucet, Nando de Freitas, Kevin P Murphy, and Stuart J Russell. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI '00, pages 176–183, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-709-9. URL `http://dl.acm.org/citation.cfm?id=647234.720075`.

A Einstein, B Podolsky, and N Rosen. Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? *Phys. Rev.*, 47(10):777–780, 1935. doi: 10.1103/PhysRev.47.777. URL `https://link.aps.org/doi/10.1103/PhysRev.47.777`.

Richard P Feynman. Feynman and Computation. chapter Simulating, pages 133–153. Perseus Books, Cambridge, MA, USA, 1999. ISBN 0-7382-0057-3. URL `http://dl.acm.org/citation.cfm?id=304763.305688`.

Dan Geiger, Thomas Venna, and Judea Pearl. -separation: from theorems to algorithms. In *Procedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, number 1, pages 138–148, 1990. URL `http://arxiv.org/abs/1405.2572http://dx.doi.org/10.1088/1367-2630/16/11/113043`.

Leo A Goodman. On Simultaneous Confidence Intervals for Multinomial Proportions. *Technometrics*, 7(2):247–254, 1965. doi: 10.1080/00401706.1965.10490252. URL `https://amstat.tandfonline.com/doi/abs/10.1080/00401706.1965.10490252`.

Lov K Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM. ISBN 0-89791-785-5. doi: 10.1145/237814.237866. URL `http://doi.acm.org/10.1145/237814.237866`.

Heather K Harton, Murali Sitaraman, and Joan Krone. Formal Program Verification. In *Wiley Encyclopedia of Computer Science and Engineering*, 2008.

Jules Hedges, Evguenia Shprits, Viktor Winschel, and Philipp Zahn. Compositionality and String Diagrams for Game Theory. 2016.

Joe Henson, Raymond Lal, and Matthew F Pusey. Theory-independent limits on correlations from generalised Bayesian networks. *New J.Phys.*, 2014. doi: 10.1088/1367-2630/16/11/113043. URL http://arxiv.org/abs/1405.2572http://dx.doi.org/10.1088/1367-2630/16/11/113043.

J E House. *Fundamentals of Quantum Mechanics*. Elsevier Science, 2017. ISBN 9780128092552. URL https://books.google.pt/books?id=YLkxDQAAQBAJ.

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

Bart Jacobs. Coalgebras in Specification and Verification for Object-Oriented Languages. 1999.

Finn V Jensen and Thomas D Nielsen. *Bayesian Networks and Decision Graphs*. Springer Publishing Company, Incorporated, 2nd edition, 2007. ISBN 9780387682815.

Anders Jonsson and Andrew Barto. Active Learning of Dynamic Bayesian Networks in Markov Decision Processes. In *Proceedings of the 7th International Conference on Abstraction, Reformulation, and Approximation*, SARA'07, pages 273–284, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-73579-3. URL http://dl.acm.org/citation.cfm?id=1770681.1770705.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2): 183–233, nov 1999. ISSN 1573-0565. doi: 10.1023/A:1007665907178. URL https://doi.org/10.1023/A:1007665907178.

Charles E Kahn, Lisa M Roberts, Katherine A Shaffer, and Peter Haddawy. Construction of a Bayesian network for mammographic diagnosis of breast cancer. *Computers in biology and medicine*, 27 1:19–29, 1997.

Priyanka Kapoor. Wave functions for the quantum harmonic oscillator, 2012. URL https://priyankacool10.wordpress.com/2012/05/28/wave-functions-for-the-quantum-harmonic-oscillator/.

Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2014. ISBN 1466570261, 9781466570269.

H J Kimble. The quantum internet. *Nature*, 453:1023, jun 2008. URL https://doi.org/10.1038/nature07127http://10.0.4.14/nature07127.

Michael Knap. Collective Quantum Dynamics. URL http://users.ph.tum.de/ga32pex/.

Michael Kues, Christian Reimer, Piotr Roztocki, Luis Romero Cortés, Stefania Sciara, Benjamin Wetzel, Yanbing Zhang, Alfonso Cino, Sai T Chu, Brent E Little, David J Moss, Lucia Caspani, José Azaña, and Roberto Morandotti. On-chip generation of high-dimensional entangled quantum states and their coherent control. *Nature*, 546:622, jun 2017. URL https://doi.org/10.1038/nature22986http://10.0.4.14/nature22986.

Tomasz Kulaga. The Markov Blanket Concept in Bayesian Networks and Dynamic Bayesian Networks and Convergence Assessment in Graphical Model Selection Problems. (October), 2006.

Matthew Leifer and David Poulin. Quantum Graphical Models and Belief Propagation. *Annals of Physics*, 323:1899–1946, 2007. doi: 10.1016/j.aop.2007.10.001.

M.~S. Leifer and R.~W. Spekkens. Towards a Formulation of Quantum Theory as a Causally Neutral Theory of Bayesian Inference. *arXiv e-prints*, page arXiv:1107.5849, jul 2011.

Chongxuan Li, Max Welling, Jun Zhu, and Bo Zhang. Graphical Generative Adversarial Networks. *CoRR*, abs/1804.0, 2018. URL http://arxiv.org/abs/1804.03429.

Guang Hao Low, Theodore J Yoder, and Isaac L Chuang. Quantum inference on Bayesian networks. *Phys. Rev. A*, 89(6):62315, jun 2014. doi: 10.1103/PhysRevA.89.062315. URL https://link.aps.org/doi/10.1103/PhysRevA.89.062315.

Catarina Moreira and Andreas Wichert. Are quantum-like Bayesian networks more powerful than classical Bayesian networks? *Journal of Mathematical Psychology*, 82:73–83, 2018. ISSN 10960880. doi: 10.1016/j.jmp.2017.11.003. URL https://doi.org/10.1016/j.jmp.2017.11.003.

Mikko Möttönen, Juha J Vartiainen, Ville Bergholm, and Martti M Salomaa. Quantum Circuits for General Multiqubit Gates. *Phys. Rev. Lett.*, 93(13):130502, 2004. doi: 10.1103/PhysRevLett.93.130502. URL https://link.aps.org/doi/10.1103/PhysRevLett.93.130502.

Sucheta Nadkarni and Prakash P Shenoy. A Causal Mapping Approach to Constructing Bayesian Networks. *Decis. Support Syst.*, 38(2):259–281, 2004. ISSN 0167-9236. doi: 10.1016/S0167-9236(03)00095-2. URL http://dx.doi.org/10.1016/S0167-9236(03)00095-2.

Chetan Nayak, Steven H Simon, Ady Stern, Michael Freedman, and Sankar Das Sarma. Non-Abelian anyons and topological quantum computation. *Rev. Mod. Phys.*, 80(3):1083–1159, 2008. doi: 10.1103/RevModPhys.80.1083. URL https://link.aps.org/doi/10.1103/RevModPhys.80.1083.

Richard Neapolitan. *Learning Bayesian Networks*. 2003. doi: 10.1145/1327942.1327961.

Chris J Needham, James R Bradford, Andrew J Bulpitt, and David R Westhead. A Primer on Learning in Bayesian Networks for Computational Biology. *PLOS Computational Biology*, 3(8):1–8, 2007. doi: 10.1371/journal.pcbi.0030129. URL https://doi.org/10.1371/journal.pcbi.0030129.

Bayesian Networks, F Faltin, and R Kenett. 35 Bayesian Networks. *Encyclopedia of Statistics in Quality & Reliability*, 1(1):4, 2007. ISSN 18780326. doi: 10.1002/wics.48.

Ann Nicholson. A Case Study in Dynamic Belief Networks: Monitoring Walking, Fall Prediction and Detection. *Lect. Notes Comput. Sci.*, 1114, 1997.

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

Bryan O'Gorman, Alejandro Perdomo-Ortiz, Ryan Babbush, Alán Aspuru-Guzik, and V Smelyanskiy. Bayesian Network Structure Learning Using Quantum Annealing. *The European Physical Journal Special Topics*, 224, 2014. doi: 10.1140/epjst/e2015-02349-9.

Ivan Oliveira, Roberto Sarthour, Tito Bonagamba, Eduardo Azevedo, and Jair C C Freitas. *NMR Quantum Information Processing*. Elsevier Science, San Diego, USA, 1st edition, 2007. ISBN 0444527826, 9780444527820.

Giuseppe Davide Paparo, Vedran Dunjko, Adi Makmal, Miguel Angel Martin-Delgado, and Hans J Briegel. Quantum Speedup for Active Learning Agents. *Phys. Rev. X*, 4(3):31002, jul 2014. doi: 10.1103/PhysRevX.4.031002. URL https://link.aps.org/doi/10.1103/PhysRevX.4.031002.

James Park and Adnan Darwiche. A differential semantics for jointree algorithms. *Artificial Intelligence*, 156:197–216, 2004. doi: 10.1016/j.artint.2003.04.004.

Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009. ISBN 052189560X, 9780521895606.

Stefano Pirandola, David Vitali, Paolo Tombesi, and Seth Lloyd. Macroscopic Entanglement by Entanglement Swapping. *Phys. Rev. Lett.*, 97(15):150403, 2006. doi: 10.1103/PhysRevLett.97.150403. URL https://link.aps.org/doi/10.1103/PhysRevLett.97.150403.

Emmanuel Pothos and Jerome Busemeyer. Quantum probability explanation for violations of 'rational' decision theory. *Proceedings. Biological sciences / The Royal Society*, 276:2171–2178, 2009. doi: 10.1098/rspb.2009.0121.

Oliver Pourret, Patrick Naim, and Bruce Marcot. *Bayesian Networks. A Practical Guide to Applications*. 2008. doi: 10.1002/9780470994559.

John Preskill. Quantum {C}omputing in the {NISQ} era and beyond. *Quantum*, 2:79, 2018. ISSN 2521-327X. doi: 10.22331/q-2018-08-06-79. URL https://doi.org/10.22331/q-2018-08-06-79.

Frank Grotelüschen Ralf Krauter. Quanteninternet - Das Web Q.0 nimmt Gestalt an. URL https://www.deutschlandfunk.de/quanteninternet-das-web-q-0-nimmt-gestalt-an.740.de.html?dram:article{_}id=425283.

Joshua W Robinson and Alexander J Hartemink. Learning Non-Stationary Dynamic Bayesian Networks. *J. Mach. Learn. Res.*, 11:3647–3680, 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1953047.

Stuart Russel and Peter Norvig. *Artficial Intelligence : a modern approach*. Upper Saddle River, NJ : Prentice Hall, ed, 3rd edition, 2010.

Perry Sakkaris. QuDot Nets: Quantum Computers and Bayesian Networks. 2016.

V M Schäfer, C J Ballance, K Thirumalai, L J Stephenson, T G Ballance, A M Steane, and D M Lucas. Fast quantum logic gates with trapped-ion qubits. *Nature*, 555:75, feb 2018. URL https://doi.org/10.1038/nature25737http://10.0.4.14/nature25737.

M Schuld and F Petruccione. *Supervised Learning with Quantum Computers*. Springer, 2018.

Eldar Shafir and Amos Tversky. Thinking through uncertainty: Nonconsequential reasoning and choice. *Cognitive Psychology*, 24(4):449–474, 1992. ISSN 0010-0285. doi: 10.1016/0010-0285(92)90015-T.

Peter W Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *arXiv e-prints*, pages quant–ph/9508027, 1995.

Th Sriarunothai, S Wölk, G S Giri, N Friis, V Dunjko, H J Briegel, and Ch Wunderlich. Speeding-up the decision making of a learning agent using an ion trap quantum processor. *Quantum Science and Technology*, 4(1):15014, 2018. doi: 10.1088/2058-9565/aaef5e. URL https://doi.org/10.1088{%}2F2058-9565{%}2Faaef5e.

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

Philip Strömberg and Vera Blomkvist Karlsson. 4-qubit Grover's algorithm implemented for the ibmqx5 architecture, 2018.

Mehmet Suzen. Pratical Kullback-Leiber Divergence: Discrete Case, 2017. URL `http://memosisland.blogspot.com/2015/08/practical-kullback-leibler-kl.html`.

Simon Tong and Daphne Koller. Active Learning for Parameter Estimation in Bayesian Networks. *Proc 13th Conf Neural Information Processing*, 2001.

Robert Tucci. Quantum Circuit For Discovering from Data the Structure of Classical Bayesian Networks. 2014.

Robert R Tucci. Quantum Bayesian Nets. *International Journal of Modern Physics B*, 9(3): 295–337, jan 1995. doi: 10.1142/S0217979295000148.

Lieven M K Vandersypen, Hansjoachim Bluhm, Jennifer S Clarke, Andrew Dzurak, R Ishihara, Andrea Morello, David J Reilly, Lars R Schreiber, and Menno Veldhorst. Interfacing spin qubits in quantum dots and donors—hot, dense, and coherent. *npj Quantum Information*, 3:1–10, 2017.

Juha J Vartiainen, Mikko Möttönen, and Martti M Salomaa. Efficient Decomposition of Quantum Gates. *Phys. Rev. Lett.*, 92(17):177902, 2004. doi: 10.1103/PhysRevLett.92.177902. URL `https://link.aps.org/doi/10.1103/PhysRevLett.92.177902`.

B Walsh. Markov Chain Monte Carlo and Gibbs Sampling. *Lecture Notes for EEB 581, version 26, April*, 2004.

Philippe Weber, Gabriela Medina-Oliva, Christophe Simon, and Benoît Iung. Overview on Bayesian networks applications for dependability, risk analysis and maintenance areas. *Engineering Applications of Artificial Intelligence*, 25:671–682, 2012. doi: 10.1016/j.engappai.2010.06.002.

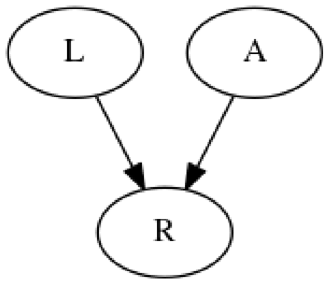
Shenggang Ying and Mingsheng Ying. Reachability analysis of quantum Markov decision processes. *ArXiv*, abs/1406.6146, 2014.

Djabeur Mohamed Seifeddine Zekrifa, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum Amplitude Amplification and Estimation. *AMS Contemporary Mathematics Series*, 305, 2000. doi: 10.1090/conm/305/05215.

A

QISKIT IMPLEMENTATION

To illustrate the Qiskit implementation of the algorithm proposed in this dissertation a Bayesian network will be used in the context of a decision problem. The Network will be as simple as possible, containing only 3 binary variables. The first is the evidence variable (L), another one the action variable (A) and still another which depends on the first two is the result variable (R). Every variable/node has a conditional probability table that maps the dependencies between them. The agent will use the network to decide which are the best actions to take knowing the evidence that he obtains.



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

The agent only needs to perform the following calculations to find out which is the best action to take:

$$EU(a|e) = \sum_{S} P(Result = s|a, e) * U(s)$$

$$action = argmax_a EU(a|e)$$

To confirm that this process is doing what is expected, the results for this problem are determined :

-First, we assume action a_0 when variable A is false and a_1 when true;

-The evidence variable L takes value False;

The utility function is given by:

$$U(R) \begin{cases} 7 & R = True \\ 3 & R = False \end{cases}$$

Thus the expected utilities of the actions are:

$$\begin{aligned} EU(a_0) = P(R = True|L = False, A = False) * U(R = True) \\ +P(R = False|L = False, A = False) * U(R = False) \end{aligned}$$

$$EU(a_0) = \frac{3}{5} * 7 + \frac{2}{5} * 3 = 5,4$$

In percentage:

$$EU(a_0) \approx 58\%$$

$$\begin{aligned} EU(a_1) = P(R = True|L = False, A = True) * U(R = True) \\ +P(R = False|L = False, A = True) * U(R = False) \end{aligned}$$

$$EU(a_1) = \frac{1}{5} * 7 + \frac{4}{5} * 3 = 3,8$$

In percentage of the total utility of the actions:

$$EU(a_1) \approx 42\%$$

These percentages are the relationships from where one should sample the actions with the used technique.

```
In [1]: import matplotlib.pyplot as plt
        %matplotlib inline
        import numpy as np
        from math import pi
        from qiskit import QuantumCircuit,
        ClassicalRegister, QuantumRegister, execute
        from qiskit.tools.visualization import circuit_drawer
        from qiskit.quantum_info import state_fidelity
        from qiskit.tools.visualization import plot_histogram
        from qiskit.tools.monitor import job_monitor, backend_monitor,
        backend_overview
```

```
In [2]: from qiskit import IBMQ
        IBMQ.enable_account('1657029970d40f6aca619ea0b546e547c280ce69242dd',
                            url='https://q-console-api.mybluemix.net/api',
                            hub='ibm-q-academic',
                            group='univ-minho',
                            project='quantalab')
```

A.0.1 *Preparation of the Quantum Registers*

```
In [3]: q_v = QuantumRegister(4, name='qv')
        q_ext = QuantumRegister(14, name='qext')
        c_v = ClassicalRegister(4)
        c_ext = ClassicalRegister(14)

        circuit = QuantumCircuit(q_v,q_ext,c_v, c_ext)
```

A.0.2 *Preparation of the state of the Bayesian Network*

```
In [4]: #Variable L
        circuit.u3(1.23,0,0,q_v[0])

        #Variable A
        circuit.h(q_v[1])

        #Variable R
        circuit.ccx(q_v[0],q_v[1],q_ext[0])
        circuit.cu3(1.71,0,0,q_ext[0],q_v[2])
        circuit.ccx(q_v[0],q_v[1],q_ext[0])
        circuit.barrier(q_v,q_ext)

        circuit.x(q_ext[1])
        circuit.cx(q_v[0],q_ext[1])
        circuit.ccx(q_ext[1],q_v[1],q_ext[2])
        circuit.cu3(0.927,0,0,q_ext[2],q_v[2])
        circuit.ccx(q_ext[1],q_v[1],q_ext[2])
        circuit.cx(q_v[0],q_ext[1])
        circuit.x(q_ext[1])
        circuit.barrier(q_v,q_ext)

        circuit.x(q_ext[3])
        circuit.cx(q_v[1],q_ext[3])
        circuit.ccx(q_v[0],q_ext[3],q_ext[4])
        circuit.cu3(0.722,0,0,q_ext[4],q_v[2])
        circuit.ccx(q_v[0],q_ext[3],q_ext[4])
        circuit.cx(q_v[1],q_ext[3])
        circuit.x(q_ext[3])

        circuit.barrier(q_v,q_ext)

        circuit.x(q_ext[5])
        circuit.x(q_ext[6])
        circuit.cx(q_v[0],q_ext[5])
        circuit.cx(q_v[1],q_ext[6])
        circuit.ccx(q_ext[5],q_ext[6],q_ext[7])
        circuit.cu3(1.77,0,0,q_ext[7],q_v[2])
        circuit.ccx(q_ext[5],q_ext[6],q_ext[7])
        circuit.cx(q_v[1],q_ext[6])
        circuit.cx(q_v[0],q_ext[5])
        circuit.x(q_ext[6])
        circuit.x(q_ext[5])

        circuit.barrier(q_v,q_ext)
```

A.0.3 *Preparation of the Utility function*

```
In [5]: circuit.u3(2,0,0,q_v[3])
        circuit.barrier(q_v,q_ext)
```

A.0.4 *Application of the Utility function*

Grover's algorithm will be performed. The probability related to the correct state after k iterations is obtained using the expression:

$$\sin^2\left(\left(k+\frac{1}{2}\right)\theta\right)$$

Knowing that the value of $\theta = 1,17rad$, after 1 iteration we obtain the envisaged state with 96, 6% of certainty. A better result would require k to be a rational number, although an iteration can be applied an integer number of times.

```
In [6]: ######################################
        ### Oracle for 0000,0011,0100,1110 ###
        ######################################


        #0000
        circuit.x(q_v[0])
        circuit.x(q_v[1])
        circuit.x(q_v[2])
        circuit.x(q_v[3])

        circuit.cu1(pi/4, q_v[0], q_v[3])
        circuit.cx(q_v[0], q_v[1])
        circuit.cu1(-pi/4, q_v[1], q_v[3])
        circuit.cx(q_v[0], q_v[1])
        circuit.cu1(pi/4, q_v[1], q_v[3])
        circuit.cx(q_v[1], q_v[2])
        circuit.cu1(-pi/4, q_v[2], q_v[3])
        circuit.cx(q_v[0], q_v[2])
        circuit.cu1(pi/4, q_v[2], q_v[3])
        circuit.cx(q_v[1], q_v[2])
        circuit.cu1(-pi/4, q_v[2], q_v[3])
        circuit.cx(q_v[0], q_v[2])
        circuit.cu1(pi/4, q_v[2], q_v[3])

        circuit.x(q_v[0])
        circuit.x(q_v[1])
        circuit.x(q_v[2])
        circuit.x(q_v[3])




        ##1100
        circuit.x(q_v[0])
        circuit.x(q_v[1])
circuit.cu1(pi/4, q_v[0], q_v[3])
circuit.cx(q_v[0], q_v[1])
circuit.cu1(-pi/4, q_v[1], q_v[3])
circuit.cx(q_v[0], q_v[1])
circuit.cu1(pi/4, q_v[1], q_v[3])
circuit.cx(q_v[1], q_v[2])
circuit.cu1(-pi/4, q_v[2], q_v[3])
circuit.cx(q_v[0], q_v[2])
circuit.cu1(pi/4, q_v[2], q_v[3])
circuit.cx(q_v[1], q_v[2])
circuit.cu1(-pi/4, q_v[2], q_v[3])
circuit.cx(q_v[0], q_v[2])
circuit.cu1(pi/4, q_v[2], q_v[3])

circuit.x(q_v[0])
circuit.x(q_v[1])

#0100

circuit.x(q_v[0])
circuit.x(q_v[2])
circuit.x(q_v[3])

circuit.cu1(pi/4, q_v[0], q_v[3])
circuit.cx(q_v[0], q_v[1])
circuit.cu1(-pi/4, q_v[1], q_v[3])
circuit.cx(q_v[0], q_v[1])
circuit.cu1(pi/4, q_v[1], q_v[3])
circuit.cx(q_v[1], q_v[2])
circuit.cu1(-pi/4, q_v[2], q_v[3])
circuit.cx(q_v[0], q_v[2])
circuit.cu1(pi/4, q_v[2], q_v[3])
circuit.cx(q_v[1], q_v[2])
circuit.cu1(-pi/4, q_v[2], q_v[3])
circuit.cx(q_v[0], q_v[2])
circuit.cu1(pi/4, q_v[2], q_v[3])

circuit.x(q_v[0])
```

```
        circuit.x(q_v[2])
        circuit.x(q_v[3])


        #0111

        circuit.x(q_v[0])

        circuit.cu1(pi/4, q_v[0], q_v[3])
        circuit.cx(q_v[0], q_v[1])
        circuit.cu1(-pi/4, q_v[1], q_v[3])
        circuit.cx(q_v[0], q_v[1])
        circuit.cu1(pi/4, q_v[1], q_v[3])
        circuit.cx(q_v[1], q_v[2])
        circuit.cu1(-pi/4, q_v[2], q_v[3])
        circuit.cx(q_v[0], q_v[2])
        circuit.cu1(pi/4, q_v[2], q_v[3])
        circuit.cx(q_v[1], q_v[2])
        circuit.cu1(-pi/4, q_v[2], q_v[3])
        circuit.cx(q_v[0], q_v[2])
        circuit.cu1(pi/4, q_v[2], q_v[3])

        circuit.x(q_v[0])

In [7]: ######################
        #### Amplification ####
        ######################
        circuit.h(q_v[0])
        circuit.h(q_v[1])
        circuit.h(q_v[2])
        circuit.h(q_v[3])
        circuit.x(q_v[0])
        circuit.x(q_v[1])
        circuit.x(q_v[2])
        circuit.x(q_v[3])
        ######## cccZ #########
        circuit.cu1(pi/4, q_v[0], q_v[3])
        circuit.cx(q_v[0], q_v[1])
        circuit.cu1(-pi/4, q_v[1], q_v[3])
        circuit.cx(q_v[0], q_v[1])
        circuit.cu1(pi/4, q_v[1], q_v[3])
        circuit.cx(q_v[1], q_v[2])
        circuit.cu1(-pi/4, q_v[2], q_v[3])
        circuit.cx(q_v[0], q_v[2])
        circuit.cu1(pi/4, q_v[2], q_v[3])
        circuit.cx(q_v[1], q_v[2])
        circuit.cu1(-pi/4, q_v[2], q_v[3])
        circuit.cx(q_v[0], q_v[2])
        circuit.cu1(pi/4, q_v[2], q_v[3])
        ####### end cccZ #######
        circuit.x(q_v[0])
        circuit.x(q_v[1])
        circuit.x(q_v[2])
        circuit.x(q_v[3])
        circuit.h(q_v[0])
        circuit.h(q_v[1])
        circuit.h(q_v[2])
        circuit.h(q_v[3])

In [8]: circuit.barrier(q_v,q_ext)
        circuit.measure(q_v[1],c_v[1])

In [9]: circuit.draw(output='mpl',scale=0.5)

Out[9]:
```

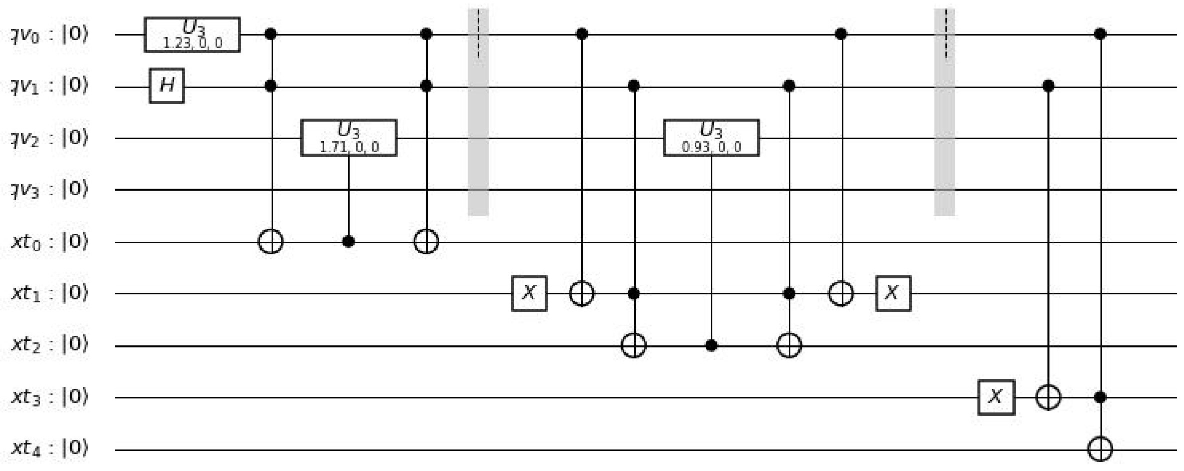


Figure 58.: State preparation circuit (part 1).

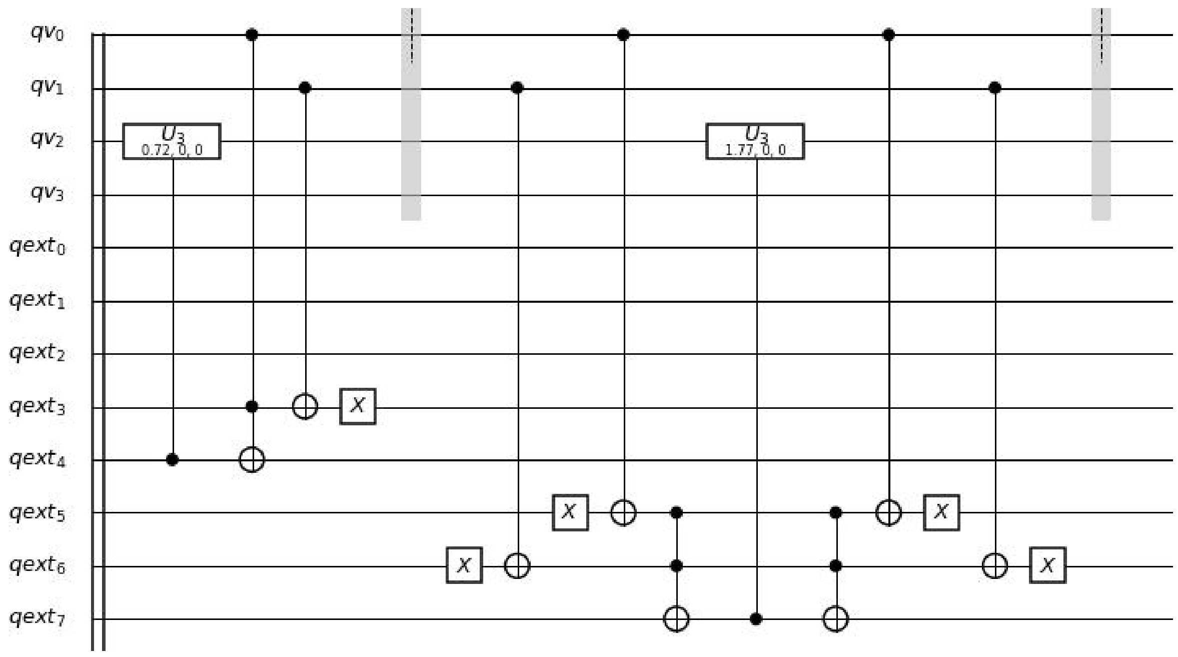


Figure 59.: State preparation circuit (part 2).

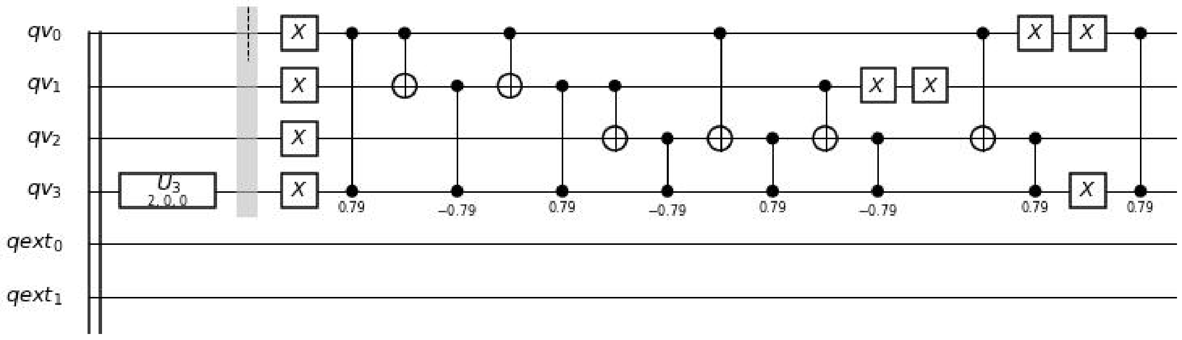


Figure 60.: Grover search circuit (part 1)

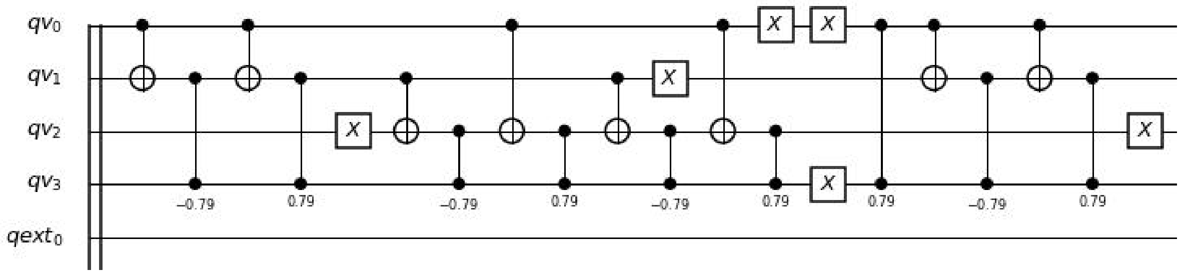


Figure 61.: Grover search circuit (part 2)

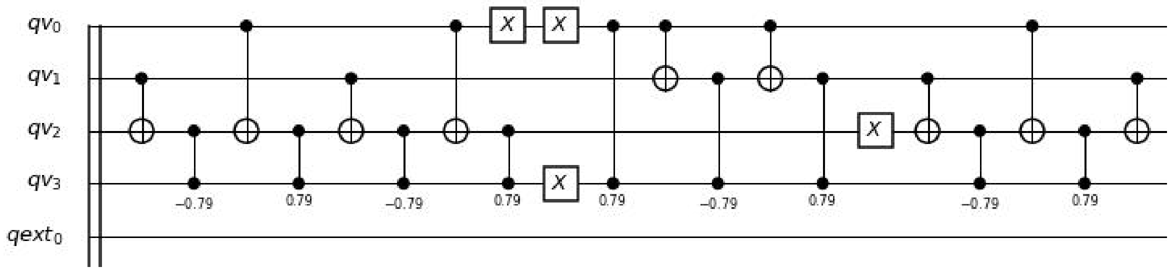


Figure 62.: Grover search circuit (part 3)

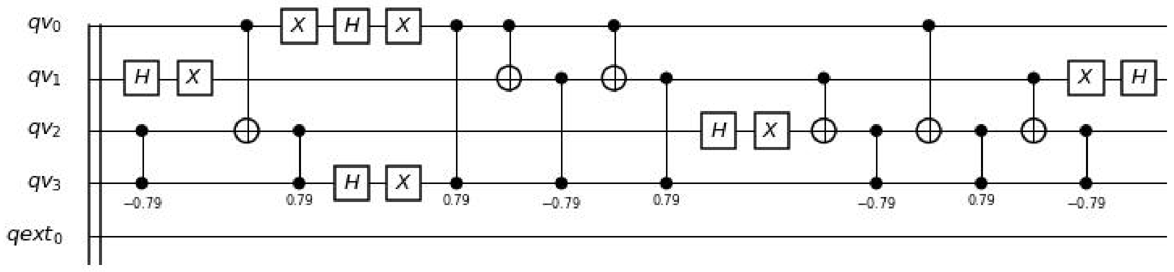


Figure 63.: Grover search circuit (part 4)

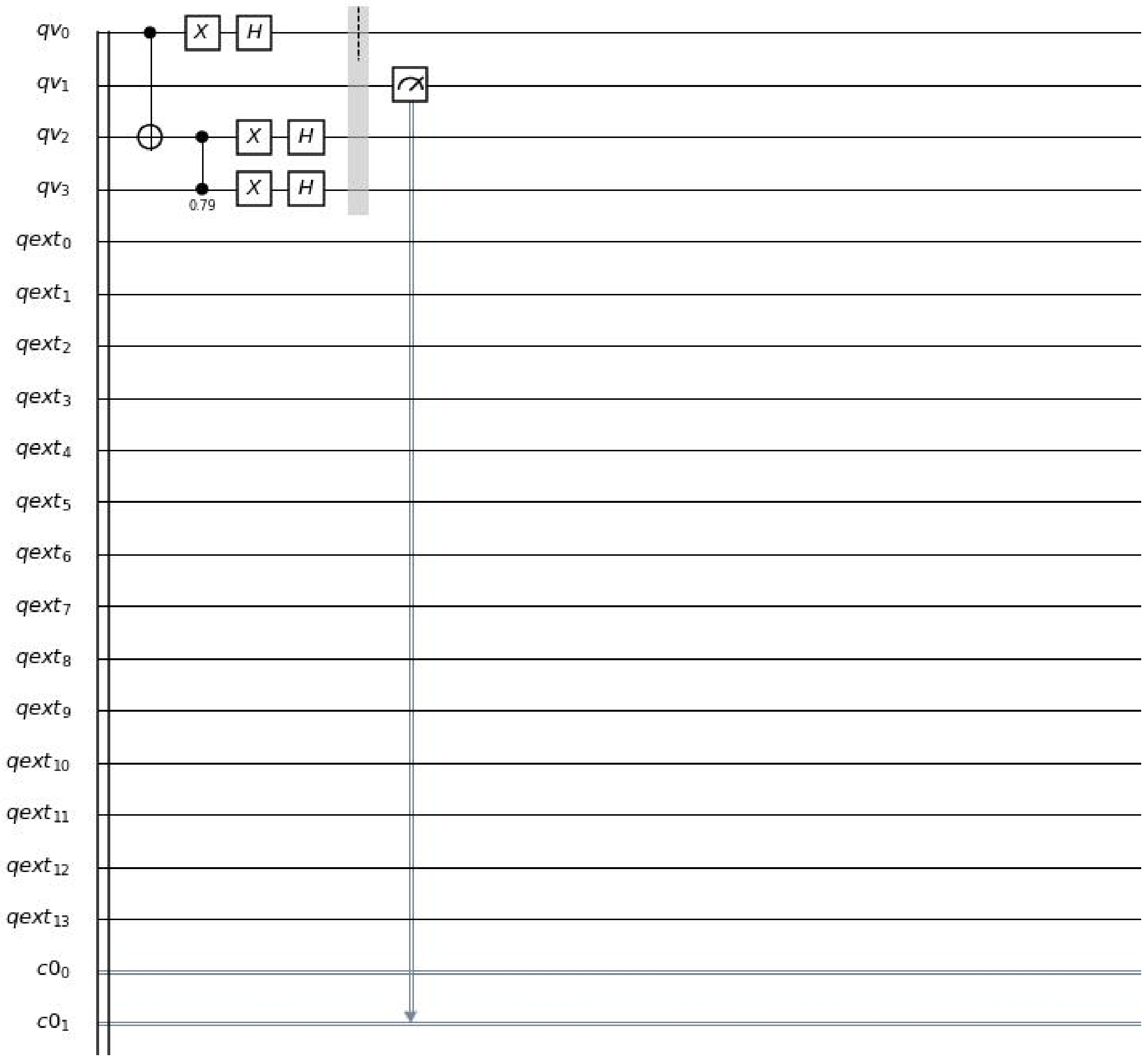


Figure 64.: Grover search circuit (part 5)

```
In [10]: backend = IBMQ.get_backend('ibmq_qasm_simulator')

         backend = IBMQ.get_backend('ibmq_qasm_simulator')
         backend.name()
         job = execute(circuit, backend,shots=5000)
         job_monitor(job, interval=5)
```

```
HTML(value="<p style='font-size:16px;'>Job Status: job is being initialized </p>")
```

```
In [11]: counts=job.result().get_counts(circuit)
         job.result().get_counts(circuit)
```

```
Out[11]: {'00000000000000 0010': 2320, '00000000000000 0000': 2680}
```

From the measurements, it is possible to conclude that the action a_0 has the biggest coefficient. So the action chosen by the agent will be the first. The values are not exactly those theoretically expected but the difference can be justified by the fact that the Grover's algorithm is probabilistic and gives the result only up to a certain level. The initial state of Bayesian Network after applying the utility function can be found in "Quantum Bayesian Decisions (Initial_State)", and the hole state after applying the Grover algorithm in "Quantum Bayesian Decisions Verification". The second file measures states that are not measured by the algorithm, but their outcomes are used to validate the results.

```
In [12]: plot_histogram(counts)
```

```
Out[12]:
```

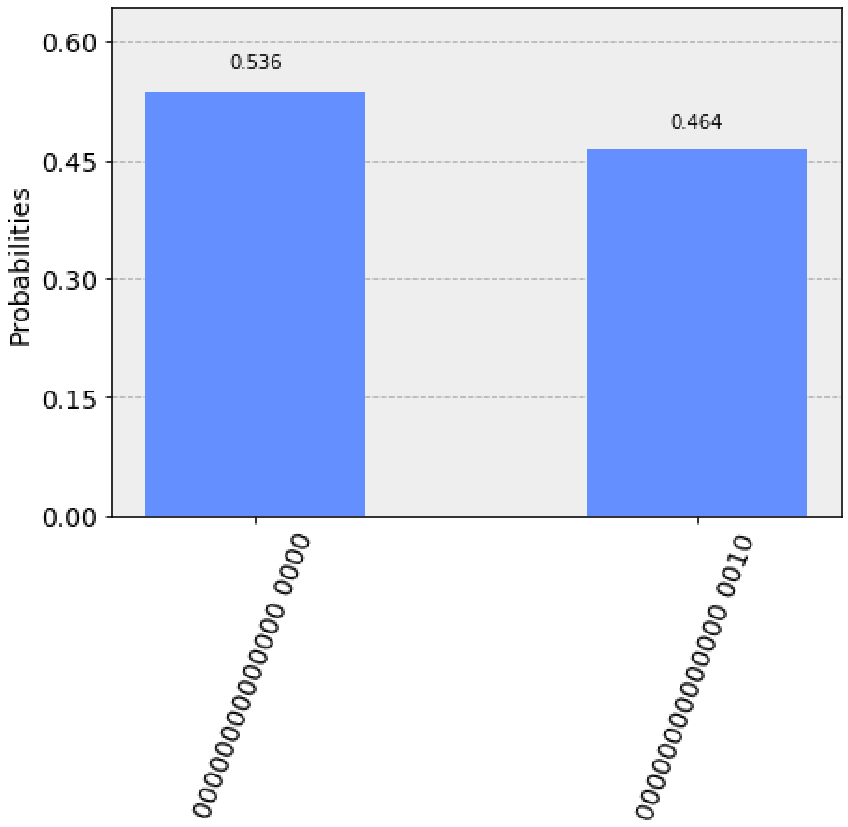


Figure 65.: Results obtained after 5000 runs.