NExt ApplicationS of Quantum Computing

D6.18: QPSA Quantum Walks and Markov Algorithms

Document Properties

Contract Number	951821
Contractual Deadline	09-06-2023
Dissemination Level	Public
Nature	Report
Editors	Mohamed Hibti, EDF R&D
Authors	Mohamed Hibti, EDF R&D Ahmed Zaiou, EDF R&D
Reviewers	Vicente Moret Bonillo, UDC Andres Gomez, CESGA
Date	09-06-2023
Keywords	Quantum walks, Dynamic Safety Assessment, Markov Graphs, Sequence identification
Status	Submitted
Release	1.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 951821

History of Changes

Release	Date	Author, Organisation	Description of Changes
1.0	23/06/2023	Mohamed Hibti, EDF R&D	Version after review
0.1	09/06/2023	Hibti M. (EDF R&D), Zaiou A. (EDF R&D Part of this work was done during M Zaiou PhD Thesis.)	Draft of the report descriving quantum walks and quantum algorithms for sequence identifications in dynamic safety assessment

Abstract

In this work, we present a quantum walks based approach for Probabilistic Safety Assessment problems in the dynamic Markovian framework.

After presenting well known quantum walk algorithms for detecting or finding marked elements in a graph or for finding paths in specific graphs, we propose two algorithms to address the more general problem of finding paths from some point to marked elements. The first algorithm is based on move and store strategy to built failure sequences from an initial state to the marked states. It uses a full register assembling qubits representing components of the system under study and other control and ancilla qubits (between n and 2n) to identify all the sequences. The second algorithm is based on a hybrid approach to consider different cascading circuits for solving relatively large instances.

We present some tests on a Qiskit simulation library that were performed to compare our algorithms against classical random walks. These tests showed that classical approach has advances in the first iteration but the hybrid approach scales better in the sense that it identifies a large number of paths and converges towards all the possible paths to these marked vertices.

Table of Contents

1 Executive Summary 6

List of Acronyms 7

List of Figures 8

List of Tables 9

2 Introduction 10

3 Markovian approach for safety assessment 12

4 Quantum walks 15
- 4.1 Discrete and continuous quantum walks . . . 15
- 4.2 Interpolated quantum walks . . . 15
- 4.3 Hitting and Mixing time . . . 16
- 4.4 Discriminant matrix of a Markov chain . . . 17

5 Quantum walk search algorithms 18

6 Dynamic Risk assessment 20
- 6.1 State-space method and some algorithms . . . 20
  - 6.1.1 Move and store . . . 21
  - 6.1.2 Quantum walk for sequences identification . . . 21
- 6.2 Quantum approach to find paths in an acyclic directed graph . . . 23
  - 6.2.1 How can we encode paths with quantum states? . . . 24
  - 6.2.2 How can we manage loops in the graph? . . . 24
  - 6.2.3 How we can use weights of the graph? . . . 25
  - 6.2.4 Quantum Oracle for quantum walks . . . 26
  - 6.2.5 Quantum algorithm to obtain all paths in an acyclic directed graph . . . 26
  - 6.2.6 Complexity analysis . . . 28
  - 6.2.7 Hybrid approach to find paths in an acyclic directed graph . . . 29
- 6.3 Results and tests . . . 31

7 Conclusion 34

8 Acknowledgements 35

Bibliography 36

1. Executive Summary

Probabilistic Safety Assessment (PSA) studies are used in different complex industrial systems (nuclear power plants, space and aviation, medical/pharmaceutical industries, petro-chemical industries, railways, etc.) to help prevent undesired events and analytically evaluate how they may occur in detail. They provide decision makers with a set of qualitative and quantitative insight regarding the risk in these systems. The method was introduced and adopted in the early seventies [35], and has since then been globally adopted, particularly in the nuclear industry. It was successfully used in different risk informed applications (see [29] for a detailed survey on the USA experience).

The discipline deals mainly with analysing the different scenarios that may lead to undesired outcomes in a systematic manner. The goal is to consider all combinations or sequences of elementary failures that create the conditions for these undesired outcomes, which are represented and evaluated using conditional probabilities.

This NEASQC report presents quantum walks algorithms and their use for PSA problems. After a brief overview of the techniques used in different quantum walk algorithms, we present many quantum search algorithms based on quantum walks and aiming to find or detect marked elements in a graph. We show how these algorithms were refined to consider different initial conditions regarding the Markov chains distributions and regarding their complexity. In general, a quadratic speedup was obtained in these algorithms, some improvements were shown along the different papers. Event if the computational complexity seems identical for some cases, it should be noted that in general restrictions and assumptions were removed or relaxed. Our objective was to understand how these algorithms work and how they mayu apply to PSA problems.

In this NEASQC report, we focus on the dynamic Markov approach for reliability which has many similarities with quantum walk frameworks. However, the same techniques can apply to the static approach. In the static approach, namely Fault Tree Analysis (FTA), the problem is to find minimal cutsets or prime implicants that may make true a Boolean formula representing the occurrence of an undesired event. This can be done either through the minimal content of the sequences or using similar techniques on s-t graphs representing systems or sequences [54].

While the problem of finding marked elements in graphs is of interest, PSA has an additional requirement: finding and storing the paths to these marked elements in the graph.

We present two algorithms (cf. [53]) in this report to search for the paths in question in graphs representing the evolution of the technical systems regarding the failure and successes of their elementary components in the Markovian process.

The first algorithm is based on move and store strategy to built failure sequences from an initial state to the marked states. It uses a full register assembling qubits representing components of the system under study and other control and working qubits (between n and $2n$) to help identify all the sequences. This approach showed a demanding number of qubits which is not compatible with a realistic implementation on current hardware for industrial systems of relevant size. The second algorithm is a hybrid approach to consider different cascading circuits for solving relatively large instances.

We present some tests that were performed to compare our algorithms against classical random walks. These tests showed that the classical approach has advances in the first iteration but our approach scales better in the sense that it identifies a large number of paths and converges towards all the possible paths to these marked vertices.

These algorithms however were tested only on a Simulator Qiskit library [21], and need to be tested on a real hardware to see in particular how the hybrid approach of section 6.2.7 behaves in a real hardware.

List of Acronyms

Term	Definition
CCF	Common Cause Failures 21
D&Q	Divide and quantum 10
FTA	Fault Tree Analysis 6
PSA	Probabilistic Safety Assessment 6
DTQW	Discrete-time quantum walks 15
HT	Hitting Time 16
MT	Mixing Time 16
CTQW	Continuous-time quantum walks 15

***Table 1**: Acronyms and Abbreviations*

List of Figures

Figure 1: Markov graph with 2 components that can fail or be repaired. . . . 12
Figure 2: Transitions in a general case [47]: We can see the different transitions from a state to another with a transition rate $\alpha_{i,j}(t)$ such as $\alpha_{i,j}(t)dt$ represents the probability of jumping from i to j within the interval $[t, t+dt]$. . . . 13

Figure 3: Interpolated quantum walk . . . 16

Figure 4: A one loop sequence . . . 20
Figure 5: Reliability Diagram . . . 21
Figure 6: Graph of the states and transitions . . . 21
Figure 7: Quantum walk Circuit: The circuit consists of state preparation (gate H) and a succession of move or transition oracles to the states corresponding to each potential transition . . . 22
Figure 8: The graph of transitions and their corresponding oracles (cf. colors) . . . 23
Figure 9: The path P_0 if $\alpha_{00} > \alpha_{01} > \alpha_{02} > \alpha_{03}$ . . . 23
Figure 10: A graph representing the failure paths of a small system without crediting any restoration 24
Figure 11: Example of the deletion of a loop . . . 25
Figure 12: Example of processing a path at time t . . . 25
Figure 13: Sub-circuit to generate the state $|\lambda'\rangle$ . . . 27
Figure 14: The architecture of our oracle . . . 27
Figure 15: (a) Quantum walks, (b) Random walks . . . 28
Figure 16: The general circuit to find the warping path between two sequences . . . 29
Figure 17: Subdivision of the big circuit for a big graph . . . 31
Figure 18: The running time spent for each approach . . . 32
Figure 19: Number of paths found according to time . . . 33

List of Tables

Table 1: Acronyms and Abbreviations 7
Table 2: Computational cost elements 18

2. Introduction

PSA studies are used in different complex industrial systems (nuclear power plants, space and aviation, medical/pharmaceutical industries, petro-chemical industries, railways, etc.) to help prevent undesired events and analytically evaluate how they may occur in detail. They provide decision makers with a set of qualitative and quantitative insight regarding the risk in these systems. The method was introduced and adopted in the early seventies [35], and has since then been globally adopted, particularly in the nuclear industry. It was successfully used in different risk informed applications (see [29] for a detailed survey on the USA experience).

The discipline deals mainly with analysing the different scenarios that may lead to undesired outcomes in a systematic manner. The goal is to consider all combinations or sequences of elementary failures that create the conditions for these undesired outcomes, which are represented and evaluated using conditional probabilities.

The problems listed below are central to the field of PSA, and this document discusses some of them in light of recent advances in quantum computing:

- The first step is modeling these scenarios into sequences of system (or human) failures that lead to an undesired outcome. These outcomes can then be evaluated to find the minimal combinations of elementary failures or paths that lead to them.
- The second problem is **how accurate this modeling** can be and to each extent the model can reflect **all the aspects** of the scenarios (their dynamic, their realism ...) and considering both **parametric and epistemic uncertainties**.
- The third is about how to **quantitatively evaluate the different metrics** regarding the undesired outcomes in such a way to be **not so conservative** and without at **all being optimistic**, but in a reasonable time that matches the requirement for operational decision making process. Recall that in the classical computing framework, many approximations are made and different strategies are followed to deal with the computational complexity.

Quantum computing is seen as a promising framework to tackle these problems. The problems appear to be good candidates for quantum computing because of their combinatorial nature and the capability of quantum superposition and quantum entanglement to help reach areas of the search space in faster ways. Furthermore, quantum walks have been successfully used for many similar graph problems and especially in the problems of searching marked vertices in a graph [32, 6, 52, 36, 13, 12]. The techniques used in this direction allowed to speedup backtracking algorithms [41] or [45], which can be also used to solve FTA problems.

In the NEASQC project, this report follows other works for solving PSA problems. An early report deals with solving tree search problems using Divide and quantum (D&Q) approach [45], to split a problem into portions that fit on small quantum computers. Then extended the approach to deal with quantum backtracking which allows better results than previous Grover-based methods by possibly pruning the search space. In [45], the FTA problem is reduced to Circuit-SAT which can also be reduced to k-SAT. Another work [27] presented a state of the art of quantum computing approaches dealing with SAT and FTA and presented some results of the implementation the D&Q in `ft-2-quantum` library[1], in addition to an extension of a cooperative approach [16].

In this work, we present a brief overview of the state of the art of quantum walk algorithms to first understand these techniques and then explore how they can be applied to PSA problems. In this report, we focus on the dynamic approach for reliability which has many similarities with quantum walk frameworks. However, the same techniques can apply to the static approach of finding minimal cutsets or prime implicants for a Boolean model through the minimal content of the sequences or using similar techniques although they are not fully based on the quantum walk framework (e.g. hybrid quantum techniques to find minimal cutsets in the Vertex Separator Problem [54] also been hypothesized to be possible and are in development [26]).

The quantum walk algorithms cited in this report aim to detect or find marked elements in a graph [32, 11, 6, 52, 36, 9, 10]. The techniques that were used in such algorithms are based on the transition matrix of the graphs in question and quantisation of the random walk in the graphs in question.

[1] `https://github.com/NEASQC/ft-2-quantum-sat`

The performance of these algorithms is mainly due to the use of the interpolated quantum walk (see subsection 4.2) introduced by Krovi et al. [32] and the quantum fast-forwarding technique introduced in [11], in addition to the discriminant matrix that fits the criteria of an ergodic[2] reversible[3] Markov chain.

While the problem of finding marked elements in graphs is of interest, PSA has an additional requirement: finding and storing the **paths** to these marked elements in the graph.

Two algorithms are presented in this report to search for the paths in question in acyclic graphs and some tests were performed for comparison against classical random walks ([53]).

This report is organized as follows: in section 3 we present the Markovian approach for safety assessment, while in section 4, we present quantum walks and some techniques and notions used in the different algorithms we refer to. Section 5 is dedicated to present the short review of quantum walks based search algorithms. In section 6, we present our algorithms for safety assessment in the dynamic case, with some primary benchmarks. In section 7, we give some concluding remarks and perspectives.

[2] A Markov chain is called an ergodic chain if it is possible to go from every state to every state not necessarily in one move [23].

[3] A Markov chain is reversible if, in steady state, the backward running sequence of states is statistically indistinguishable from the forward running sequence. More formally, for every pair of states i and j, the transition probability from state i to state j is the same as the transition probability from state j to state i.

3. Markovian approach for safety assessment

A Markov chain or Markov process is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event (memorylessness). In other word, the future evolution of the system depends only on its current state and not on the trajectories (set of states) it went through [51].

Formally, a discrete-time Markov chain is a sequence of random variables $X_1, X_2, X_3, \ldots$ with the Markov property, namely that the probability of moving to the next state depends only on the present state and not on the previous states:

$$\Pr(X_{n+1} = x \mid X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \Pr(X_{n+1} = x \mid X_n = x_n),$$

if both conditional probabilities are well defined, that is, $\Pr(X_1 = x_1, \ldots, X_n = x_n) > 0$..

A stationary distribution of a Markov chain is a probability distribution that remains unchanged in the Markov chain as time progresses. Typically, it is represented as a row vector π whose entries are probabilities summing to 1, and given transition matrix P, it satisfies

$$\pi = \pi \mathbf{P}$$

In other words, π is invariant by the matrix P.

Markov chains are used in PSA to represent different states of the system, where each node is a state and each transition means either the failure or repair of a component in the system. No other evolution aspects are considered in this framework, we are mainly interested by the failure and repair events.

If we consider the following system with two redundant components we have the following states: AB (A and B are safe), $A\bar{B}$ (B is failing), $\bar{A}B$ (A is failing) and $\bar{A}\bar{B}$ (both A^{and} B are failing). The Markov graph corresponding to this small system is shown in figure 1.

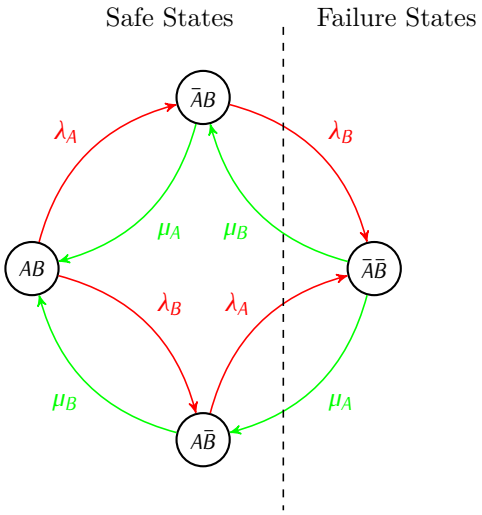


***Figure 1**: Markov graph with 2 components that can fail or be repaired.*

Let's consider a more general case for a given state i in a Markov chain. We want to represent the probabilities for the three following cases:

- Transitioning from any state to i
- The state staying as i (i.e. i transitioning to itself)
- Transitioning from i to another state

The following presentation is a summary of [47][1].

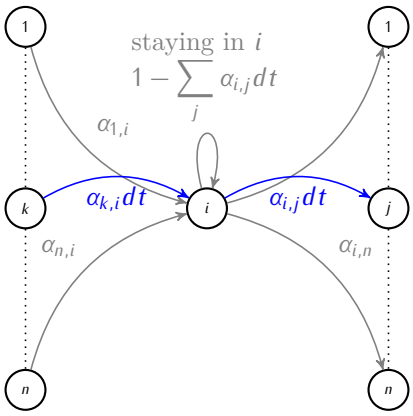


***Figure 2**: Transitions in a general case [47]: We can see the different transitions from a state to another with a transition rate $\alpha_{i,j}(t)$ such as $\alpha_{i,j}(t)dt$ represents the probability of jumping from i to j within the interval $[t, t + dt]$.*

The case in which the transition probability varies according to t is a semi-Markov process[2], which generally requires more complicated analysis. If the transition probabilities are constant in time, it is a homogeneous Markov process. We consider the latter in this document.

If we consider a system S with n states, the probability to move to state i from every state k is the sum of the possible transitions to state i

$$\sum_{k \neq i} \alpha_{k,i} P_k(t).dt$$

where $P_k(t)$ is the probability of the system to be in state k at time t.

Similarly, the probability to leave state i is the sum of all its outgoing transitions:

$$(\sum_{k \neq i} \alpha_{i,k}) P_i(t).dt$$

while the probability that S remains in state i is

$$(1 - (\sum_{k \neq i} \alpha_{i,k}).dt) P_i(t) = 1 - \alpha_{i,i}$$

Therefore, the probability for the system to be in state i at $t + dt$ is the sum of the probability to jump to i from another state and the probability to stay in i. This can be expressed in equation 3.1.

$$Pi(t + dt) = \sum_{k \neq i} \alpha_{k,i} P_k(t) dt + (1 - \sum_{i \neq k} \alpha_{i,k} dt) P_i(t) \tag{3.1}$$

Equation 3.2 gives the derivative $\frac{d(P_i(t))}{dt}$ of the probability of the system to be in state i at time t.

$$\frac{dP_i(t)}{dt} = -\alpha_i P_i(t) + \sum_{k \neq i} \alpha_{k,j} P_k(t) \tag{3.2}$$

where $\alpha_i = \sum_{k \neq i} \alpha_{i,k}$ is the transition rate out of state i.

The reliability problem in PSA is to find the probability P_m of reaching the failure states which can be considered as the set of marked nodes M in the Markov graph. In addition to these probabilities, we are

[1] Chapter 31, pp 475-477.

[2] Semi-Markov processes are generalizations of Markov processes in which the time intervals between transitions have an arbitrary distribution rather than an exponential distribution (which is the case for Markov processes).

interested by the trajectories of the Markov graph that start from some initial state and end in the marked states.

For the probability computation, there are many classical approaches that were identified; considering the reliability assumptions in [42, 1, 25] (a more recent and complete presentation can be found in [47]) for solving this problem; Laplace transform, matrix inversion or numerical methods.

Two main cases are of interest regarding safety studies: if we are interested by the reliability $R(t) = P(T > t)$ then the graph marked states are failure states and thus are considered as absorbing (i.e. a state that has no outgoing transitions other than to itself). If the target is availability, $A(t) = P$(System is up at time t), the marked elements has transitions out.

4. Quantum walks

In this section, we present some definitions and concepts that may help a casual reader to understand some methods used in the different algorithms based on quantum walk.

Quantum walks are quantum mechanical analogs of classical random walks. With random walks, the path of the walk is described by stochastic transitions, while in quantum walks these transitions happen in superposition; that is, many positions of the "walker" exist in a probability amplitude. They are fundamental to quantum algorithms, quantum computing , and quantum information processing [8, 14, 4, 5, 24].

4.1 Discrete and continuous quantum walks

Quantum walks can be classified into two main types: Continuous-time quantum walks (CTQW) [22] which are characterized by a continuous evolution of the system and Discrete-time quantum walks (DTQW) [3] that can be separated in a repeated application of two operations called "coin" (acting on the state) and "shift" (acting on the position of the walker) reminding of gate operations in quantum information processing. The behaviour of the quantum walks may then be influenced by a proper choice of the coin and shift operation.

DTQW are analogous to classical random walks, where the walker takes discrete steps at regular intervals of time. In DTQWs, the walker is a quantum system (e.g., a qubit) that moves through a graph. At each step, the coin is flipped, and the quantum system evolves under the influence of the coin and the graph topology. The evolution is governed by the unitary operator, which describes the time evolution of the quantum system.

CTQWs are analogous to classical diffusion processes, where the walker moves continuously in time. In CTQWs, the walker is also a quantum system, but it evolves under the influence of the Hamiltonian operator, which describes the energy of the system. The Hamiltonian is defined based on the graph topology and the quantum coin, and it determines the rate at which the walker moves from one vertex to another.

The main difference between DTQWs and CTQWs is the nature of the evolution of the quantum system. In DTQWs, the system evolves in discrete steps, while in CTQWs, the system evolves continuously in time. This difference has implications for the behavior of the quantum walk, such as the speed of spreading and the degree of localization of the walker.

4.2 Interpolated quantum walks

An interpolated quantum walk [32] is a type of quantum walk that allows for a gradual transition between two different quantum walks. In a quantum walk, a quantum particle (such as an atom or photon) moves through a series of locations or "nodes" that are connected by paths, and the particle's state changes as it interacts with the nodes and paths.

In an interpolated quantum walk, the particle's state is gradually transformed from one quantum walk to another over time, with the transition controlled by a parameter that varies smoothly from one value to another. This allows for more flexible and controllable quantum walks that can be tailored to specific applications.

Formally an interpolated walk of some initial one, is the superposition of an initial walk with an absorbing walk with a parameter ϵ (between 0 and 1, close to 0").

$$(Absorbing\ walk).(1-\epsilon) + (Initial\ walk).\epsilon$$

For instance, if we consider the walk in the graph of figure 3 (a), we can interpolate it by modifying the transition probabilities of the outgoing transition of the node $\bar{A}\bar{B}$ to which a loop is added (see figure 3 (b)).

Note that for PSA problems, the failure states are absorbing in the case of reliability, and that this is a natural property of such graphs.

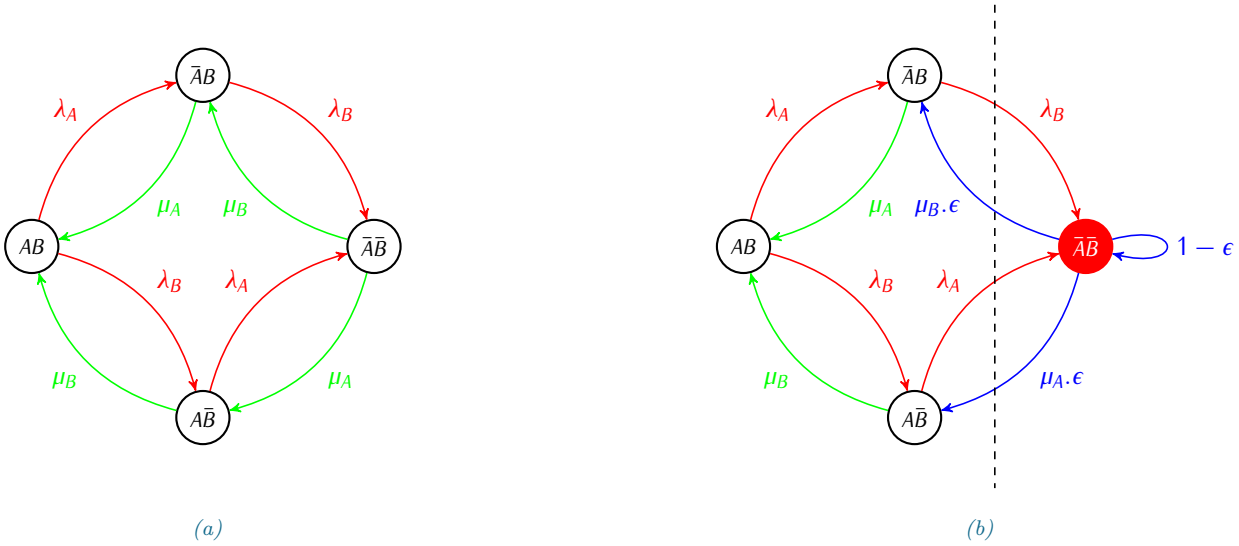


***Figure 3**: Interpolated quantum walk*

4.3 Hitting and Mixing time

In the classical case, the Hitting Time (HT) is the average time (number of steps) that the walker needs to reach a given position j, when it starts from a particular vertex (c stands for classical) [44]:

$$h^c(j) = \sum_{m=0}^{\infty} mp(j,m)$$

Where $p(j,m)$ is the transition probability from j to m.

In the quantum case (i.e. quantum walk), the definition of the hitting time is not the same. The measurement destroys the quantum characteristics of the walk. and, in general, the $m \to \infty$ limits of $U^m |\psi_0\rangle$ and $p^m(x)$ do not exist. Nevertheless, if we average the distribution over time, in the limit of infinite upper bound on time it does converge to a probability distribution which can be evaluated.

In quantum mechanics, the Mixing Time (MT) and HT are measures of how long it takes for a quantum system to reach equilibrium or to transition between different states.

The hitting time, also known as the collision time, is the time it takes for a quantum system to transition from one state to another. Specifically, it is the time it takes for the system to reach a particular state with a specified probability. Given a quantum system with an initial state $|\psi_0\rangle$ and a target state $|\psi_T\rangle$, the hitting time T_H is the minimum number of discrete time steps required for the system to evolve from $|\psi_0\rangle$ to $|\psi_T\rangle$, with the evolution governed by a unitary operator U that describes the dynamics of the system. Mathematically, the hitting time T_H is defined as:

$$T_H = mint|U^t |\psi_0\rangle = |\psi_T\rangle$$

where U^t represents the application of the unitary operator U for t time steps. In this document, it represents the time to reach some marked element in a graph.

Mixing Time: Let $\mathcal{G}$ be a graph and $|\pi_0\rangle$ be the initial state of a quantum walker on $\mathcal{G}$. Let $\mathcal{U}$ be the unitary evolution operator that governs the dynamics of the quantum walk. The mixing time of the quantum walk on $\mathcal{G}$ is defined as the smallest time t_{mix} such that

$$\left\| \mathcal{U}^t(|\pi_0\rangle \otimes |\pi_{\text{eq}}\rangle) - |\pi_{\text{eq}}\rangle \otimes |\pi_{\text{eq}}\rangle \right\|_{\text{tr}} \leq \epsilon,$$

where π_{eq} is the stationary distribution[1] of the quantum walk, ϵ is a small constant, and $\|\cdot\|_{\text{tr}}$ denotes the trace norm[2]. In other words, the mixing time is the smallest time at which the distribution of the quantum walker becomes close to the stationary distribution, up to an error of ϵ.

Hitting Time: Let v be a vertex in $\mathcal{G}$ and let P_v be the projector onto the subspace spanned by the vertex v. The hitting time of the quantum walk on $\mathcal{G}$ starting from vertex u is defined as the smallest time t_{hit} such that

$$\left\|\mathcal{U}^t(|\pi_0\rangle \otimes |P_v\rangle) - |P_v\rangle \otimes |P_v\rangle\right\|_{\text{tr}} \leq \epsilon,$$

where $|\pi_0\rangle$ is the initial state of the quantum walker on $\mathcal{G}$, ϵ is a small constant, and $\|\cdot\|_{\text{tr}}$ denotes the trace norm. In other words, the hitting time is the smallest time at which the probability of the quantum walker being at vertex v is close to the desired probability, up to an error of ϵ.

4.4 Discriminant matrix of a Markov chain

The discriminant matrix is a key component of the Szegedy quantum walk [50]. It enhances the efficiency of the algorithm for searching marked vertices in a graph, by reducing the number of iterations required to locate the marked vertices.

The discriminant matrix quantifies the difference between the quantum and classical random walks by measuring the deviation of the eigenvalues of the unitary operator from the real line. The larger the deviation, the more effective the Szegedy walk is in locating the marked vertices. The discriminant matrix is a useful tool in quantum walks because it allows us to analyze the behavior of the quantum walk on the graph and to design efficient algorithms for various graph problems. There are many different formal definitions of the discriminant matrix that seem coherent with each other (e.g. [50] for a general case, [32] or [9] for a Markov chain). In this report we don't necessarily use it per se, but is is used in the quantum walk based algorithms for finding marked elements in a graph and allow symmetry when the initial transition matrix is not symmetric. Many results that stand for ergodic reversible Markov chains can be recovered in general non-reversible one due to the singular values of the "discriminant" matrix associated with it ([39]).

Given a Markov chain with state space S and transition matrix P, the discriminant matrix D can be defined as:

$$D = (I - P)^{(-1)} * (I - P^T)$$

where I is the identity matrix and P^T is the transpose of the transition matrix P.

[1] Note that under certain conditions (ergodicity) a Markov Chain has a stationary distribution π with entries $(\pi_j : j \in S)$ such that:

- $\pi_j \geq 0, \forall j$ and $\sum_j \pi_j = 1$
- $\pi = \pi P$ where P is the transition matrix of the chain $(\pi_j = \sum_i \pi_i p_{(i,j)}, \forall j)$

[2] See [30] for a potential intuition for the trace norm. It is considered as a way of turning the rank of a matrix (which is very discontinuous) into a norm (which is continuous). Specifically, the trace norm is the unique norm with the property that $P_{tr} = rank(P)$ for every orthogonal projection $P \in M_n(C)$ (M_n being the space of $n \times n$ complex matrices).

5. Quantum walk search algorithms

There are different types of continuous and discrete time algorithms that were used to search marked elements in different graphs (hypercubes [46], lattices [20], [7], triangular graphs [37], Johnson graphs ...).

DTQW are mainly based on a coined quantum walk [2] [34] on the vertices or on Szegedy quantum walks on the edges.

All these algorithms were developed to speed up the hitting times of reaching marked elements in specific graphs. Indeed, given some initial distribution σ, which may be an arbitrary or a stationary distribution (in this case noted π) and the nature of the underlying graphs where the walks occur, the algorithms aimed to evaluate and reduce as much as possible the time to detect or find one or all the marked elements of the graph. This time is given as a function of the following parameters of table 2:

***Table 2**: Computational cost elements*

Costs	Abbreviation	Classical	Quantum
Setup cost	S	Sampling $X_0 \sim \sigma$	Create $\|\sigma\rangle$
Update cost	U	Sampling $y \sim N_x$	map $\|x\rangle \to \frac{1}{\sqrt{d_x}} \sum_{y \in N_x} \|y\rangle$
		N_x being the neighbors set of x	d being the degree of x
Checking cost	C	Checking whether an element is marked	$\|z\rangle \to \begin{cases} \|z\rangle & if \quad z \in M \\ -\|z\rangle & if \quad z \notin M \end{cases}$

After the introduction of Szegedy quantum walk [50], where given some stationary distribution π, the algorithm can **detect** if a marked element exists with a computational cost of $S + \sqrt{HT}(U + C)$ (HT being the classical hitting time), many algorithms were proposed for the search of marked elements. In [40] [38], Magniez et al. showed that from a stationary distribution, one can **find** a marked element $m \in M$ in a time $S + \frac{1}{\sqrt{\epsilon\delta}}U + \frac{1}{\sqrt{\epsilon}}C$ where δ is the spectral gap and ϵ a small fraction such that

$$\epsilon = P(X \in M | X \sim \pi) = \frac{|M|}{|V|}.$$

In [13], A. Belovs introduced electrical networks that were successfully[1] used [19] for classical random walks to leverage quantum walks. He showed that for any initial distribution σ, one can detect if a marked element exists in $S + \sqrt{RK}(U + C)$ (R the effective resistance[2] of the corresponding graph and K the number of nodes).

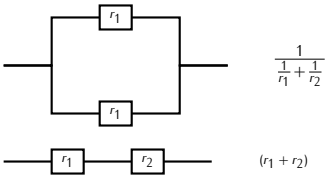
- $RK = HT$ if $\sigma = \pi$.
- $RK = CT$ if $\sigma = \langle\tau\rangle$.

In [32], Krovi et al. showed that if $\sigma = \pi$, one can **find** a marked element $m \in M$ in $S + \sqrt{HT}(U + C)$ where HT is the classical hitting time and when there are multiple marked elements (any element of M), the cost is defined as an extended hitting time HT^+ defined as follows:

$$HT^+(P, M) := \lim_{s \to 0} HT(s)$$

[1] If we consider a random walk as an electrical current flowing through the graph, where the probabilities of transitioning to neighboring vertices are proportional to the conductance of the corresponding edges. The electrical potential at each vertex can be determined based on the flow of current and the resistance values. This potential distribution can then be used to analyze various properties of the random walk, such as hitting times, cover times, and mixing times.

[2] R can be obtained by the same rules that serve to determine an electrical circuit resistance as follows :



where $HT(s)$ is the interpolated hitting time. $HT(s)$ is defined as follows:

$$HT(s) := \sum_{k=1}^{n-1} \frac{|\langle v_k(s)|U\rangle|^2}{1-\lambda_k(s)}$$

where the λ_k are the eigenvalues of the discriminant matrix $D = D(P)$ in nondecreasing order and $|v_k\rangle$ their corresponding eigenvectors, and $|U\rangle$ is the unit vector [32].

In [18], Dohotaru and Hloyer showed that if $\sigma = \pi$ one can **find** a marked element $m \in M$ in $S+\sqrt{HT}U+\frac{1}{\sqrt{\epsilon}}C)$ note the following inequalities ([10]).

$$S + \sqrt{HT}(U + \frac{1}{\sqrt{\epsilon}}C) \leq S + \sqrt{HT}(U + C)$$
$$S + \sqrt{HT}(U + \frac{1}{\sqrt{\epsilon}}C) \leq S + \frac{1}{\sqrt{\epsilon\delta}}U + \frac{1}{\sqrt{\epsilon}}C)$$

In [6], Ambainis et al. showed that if $\sigma = \pi$ one can **find** a marked element $m \in M$ in $S + \sqrt{HT}(U + C)$ and Apers et al. in [10] proposed another approach that have a quadratic speedup even starting with any distribution σ and finding the marked elements (generalizing the results of Belovs and Ambainis et al.). Indeed, for any σ, one can find $m \in M$ in $S + \sqrt{R(\tau)m}(\sqrt{\tau}U + C)$. The performance of these algorithms is mainly due to the use of interpolated quantum walk (see subsection 4.2) introduced by Krovi et al. [32] and the quantum fast-forwarding technique introduced in [11]. Theorem 1 in [11] states that for any quantum state $|v\rangle$ and $\epsilon > 0$, there exists a time t_0 such that for all $t \geq t0$ the quantum fast forwarding algorithm outputs (within $O(\sqrt{\frac{t}{\epsilon}}log(\frac{1}{\epsilon})$ steps) a quantum state (up to normalization) ϵ-close to $D^t|v\rangle$, with probability $D^t|v\rangle_2^2$, and otherwise outputs "Fail" (Where D is the discriminant matrix of the transition matrix).

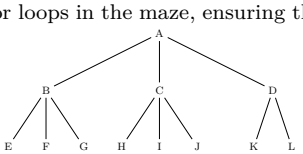
In the continuous framework, Apers et al. [9] proposed CTQW to get a quadratic speedup over classical random walks generalizing a previous result by Chakrabarti et al. [15].

All the aforementioned algorithms are dealing with the problem of finding a (any) marked element in a graph. The PSA problem, however, requires finding not only marked elements but all paths to them. This raises the question: can these algorithms serve as a basis for quantum path finding algorithms that provide an advantage over their classical counterparts?

In [43], it is shown that quantum walks can be used to find a path between two marked elements. In some specific graphs (N-Tree Mazes[3]) where the classical search has a cost of $O(MN)$ (M being the deft of the N-tree), the quantum walk can have a $O(\sqrt{M}N)$. In [31] quantum walks were used to find paths between marked elements in a tree maze. Two approaches were proposed, one based on applying on the computation of the eigenvalues and eigenvectors of the walk operator, the other is based on measurements.

In our case we are looking for all the paths that start from some initial element to all the marked elements except those paths with very low probabilities (the probability here is the resulting probability of the path knowing each edge has a probability of occurrence). The main objective is to evaluate the whole cumulative probability of reaching some undesired event (marked element).

[3] *A maze can be represented by a tree data structure, since mazes branch off into different paths, which in turn branch off into more paths. When you reach a dead end in a maze, you must backtrack to an earlier branching point* [49]. An N-tree maze is a type of maze that is represented by a rooted tree structure consisting of nodes connected by edges. The root node, serves as the starting point of the maze, and each non-leaf node can have up to N child nodes ($N \geq 2$). Note that there are no cycles or loops in the maze, ensuring that there is a unique path from the root node to each leaf node. As in the example hereafter.



6. Dynamic Risk assessment

The probabilistic approach to risk assessment is an approach to identify and evaluate in terms of probabilities or frequencies all the scenarios that may be triggered by some initiator. In the static case, we only look for combinations of events that may make some undesired outcome realisable, given some initiator. Accident scenarios are modeled using Boolean algebra (fault trees) and their quantification give an overview of the frequency/probability of the undesired outcomes. For each outcome the probability is the sum (in the probabilistic sense) of the combinations probabilities (cutsets contributions).

In the dynamic case, however, we are not looking for events combinations but for sequences of failure and success events that ends in some undesired outcome. Different aspects are then considered and in the literature, different approaches are considered as dynamic ([48, 28, 33, 17]). However, these approaches may have different definitions of what they consider as a dynamic framework. Some of the aspects they may have in common are consideration of events order (to better consider passive redundancy and standby systems), the possibility to consider recoveries and time. Therefor considering Markov models as in this report can be common to all these methods.

When dealing with dynamic models, the physical behavior of a complex system and its components is taken into account, in particular how their reliability would evolve due to degradation of a component performance, changes in its operation modes, accidents and other phenomena that would increase the probability of a certain component to fail at, or after, a given time.

In these approaches, the system has different possible *state configurations*, expressed in terms of combinations of failed/operating states of each single component of the system, and the *transition probabilities* to move from a state to another. This method is called *State Space Model* which produces huge models even for medium size real life systems.

Currently, classical algorithms that are used to solve this kind of problems suffer from a number of limitations mainly due to computational complexity issues (the combinatorial explosion and the difficulty to explicitly model the different states the system can go through during its evolution). Another obstacle, in the industry, is the way these problems are formulated and solved. This requires generally familiarisation with advanced mathematical models and a very good knowledge of the underlying assumptions and approximations that may be made in order to get confident results [48].

This method has many characteristics that make it a good candidate quantum computing application.

- The state space can be encapsulated by the superposition of qubits.
- Many quantum algorithms or quantum classes of algorithms could be of interest to explore the state space and have a significant speedup against classical computation.

6.1 State-space method and some algorithms

In this section we present an algorithm to deal with the state space method. The main objective here is to consider all the sequences from some initial state to some degraded or failure state of interest. For practical reasons, we consider only sequences with a maximum of one loop (cf. example in figure 4) assuming that during a scenario only one recovery can be considered due to time and personnel resources.

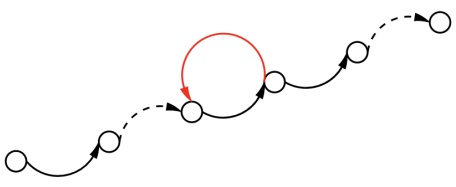


***Figure 4**: A one loop sequence*

Our aim here is to use a minimal number of qubits; for a system of N components, we need a quantum register of $3N$ qubits and N moves.

In the example of figure 5, we have a system of three components $\{x_1, x_2, x_3\}$ and we look for finding all the scenarios that go from the initial state (all the components are safe) to a failing state (for instance $\{\bar{x_1}, \bar{x_3}\}$ where both x_1 and x_3 are failing).

We can represent a state of the system by a quantum state (for instance $|abc\rangle$, with a, b and c in $\{0, 1\}$), with the j^{th} qubit representing the j^{th} component, and 0 for safe component and 1 for failing component. A failure state of the system could be $|011\rangle$

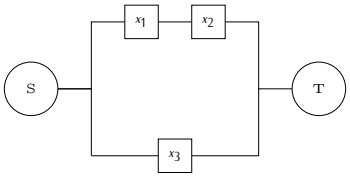


***Figure 5**: Reliability Diagram*

In this case, the following graph represents the set of all transitions between all the states of the system. The aim of the algorithm is to determine all the sequences that go from $|000\rangle$ to some failure state (let it be $|011\rangle$), allowing only one loop. A loop here corresponds to the reparation/recovery of a component during a sequence.

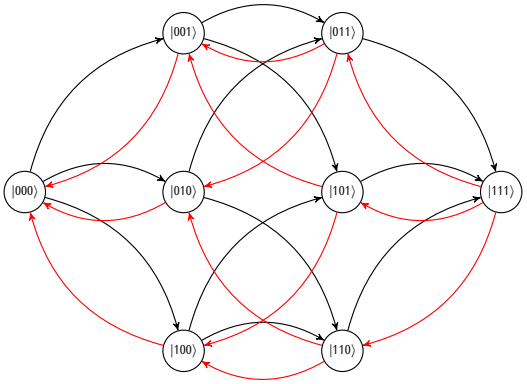


***Figure 6**: Graph of the states and transitions*

6.1.1 Move and store

To identify all the possible sequences from an initial state[1] which may be $|000\rangle$ or any other state in the graph, we will use quantum walks with a storage strategy. Starting from the state $|000\rangle$ the idea is to collect all its successors, which means all states that correspond to a failure of some components (i.e. $|100\rangle, |010\rangle, |001\rangle$). We assume for now that Common Cause Failures (CCF) are not considered (two or more components failing at the same time or within a short duration). Therefore, we get 3 paths, and if we consider any of these states we can also look for its successors. The advantage quantum computing provides us here is the "parallel way" this process is performed.

If we consider $|100\rangle$, we can move to $|000\rangle, |110\rangle, |101\rangle$. Now, the paths are connected to form all the sequences following algorithm 1:

6.1.2 Quantum walk for sequences identification

In this approach, we consider using only between $2n$ and $3n$ qubits with n being the number of components of the system. With the first n qubits, we can represent all the states of the system and the rest of the qubits (between n and $2n$) can be used to identify the paths.

[1] We assume initial state is a state where all the components work as expected or are waiting in a standby position without assuming any failure.

Algorithm 1: Paths to the failure state

Input : The number of elements to handle n, the number of iterations k and a failure state $|failure\rangle$
Output: The list of paths CH between the start state and the failure state $|failure\rangle$.
Begin:
Initialize the list of paths to the initial state $CH = \{\{|0\rangle^{\otimes n}\}\}$.
Initialize the set of safe states to the state $\psi = \{|0\rangle^{\otimes n}\}$
And the current state to $|\phi\rangle = |0\rangle^{\otimes n}$.
for k *times* **do**
  for n *times* **do**
    Apply a quantum walk to each $|\psi_i\rangle \in \psi$ with $|\psi_i\rangle \neq |failure\rangle$ and add the new paths to the list.

$$O(\psi_i)\,|\phi\rangle = \sum_i \alpha_i\,|x_i\rangle$$

    Add the $|x_i\rangle$ after each list ending at $|\psi_i\rangle$ to the list CH.
    Update state $|\phi\rangle$ and the state of all safe states ψ.

$$|\phi\rangle = O(\psi)\,|\phi\rangle = \sum_i^p \beta_i\,|\phi_i\rangle$$

$$\psi = \{|\phi_i\rangle\,, i = 0, ..., p\}$$

  $\psi = \{|0\rangle^{\otimes n}\}$
Remove the list of paths that do not end with the $|failure\rangle$ state of the list CH.
return CH

Consider the graph in figure 6, to represent the nodes of the graph we use 3 qubits (n = 3). Let $|res\rangle$ be the resulting state, then we are looking to get at the end of the execution of our algorithm a superposition of the form :

$$|res\rangle = \sum_i |i\rangle \sum_j \alpha_{ij}\,|w_j\rangle$$

where the $|i\rangle$ represent path indexes and the $\sum_j \alpha_{ij}\,|w_j\rangle$ represent the path. The order of the states in the path will be specified by the α_{ij} in a decreasing manner. More precisely, the first state will be the state having the greater value α_{ij} the last one is the state with the smallest value. Recall that when dealing with sequences the probabilities are multiplicative and the α_{ij} are decreasing since we multiply by numbers less than or equal to one.

As an example, if we consider the graph in figure 6, we need 3 move oracles and 9 qubits. 3 to represent the states of the system and 6 to identify the paths.

Initially we start by applying the Hadamard gates on the qubits used to identify the paths (they are used to index the results in the n-th qubits). After that we apply the oracles of movements: the first to move from the initial state to $|001\rangle$, $|010\rangle$ and $|100\rangle$, the second to move to $|011\rangle$, $|110\rangle$ and $|101\rangle$ and finally the third to the state $|111\rangle$ (See Figure 7).

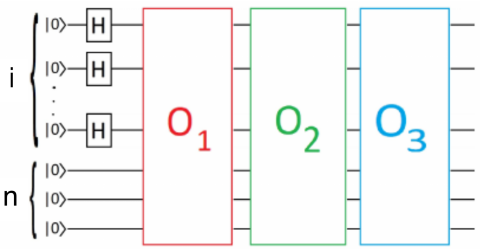


***Figure 7**: Quantum walk Circuit: The circuit consists of state preparation (gate H) and a succession of move or transition oracles to the states corresponding to each potential transition*

In the graph of figure 8, we can see in red, blue or green the transitions performed in each oracle.

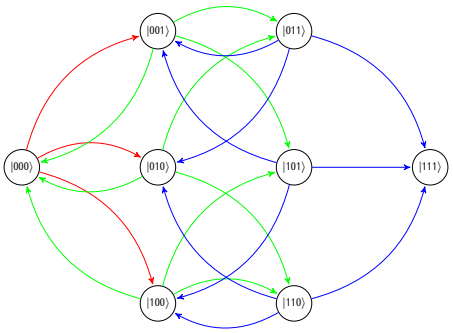


***Figure 8**: The graph of transitions and their corresponding oracles (cf. colors)*

In general, we use between $2n$ and $3n$ qubits for a system of n components to represent all the direct sequences that start at the initial state and end in a failure state. These qubits are used to find at the end the following superposition:

$$|res\rangle = \sum_i |i\rangle \left(\sum_j \alpha_{ij} |w_{ij}\rangle\right)$$

where we use n qubits for the states of the system $|w_{ij}\rangle$ and between n and $2n$ qubits for the paths identifiers $|i\rangle$.

To read the paths we take for example the path $P_0 = \sum_j \alpha_{0j} |w_{0j}\rangle$ identified by $|i\rangle = |000000\rangle$. The order of the states $|w_{0j}\rangle$ is specified in the path by the probabilities α_{0j}. The path P_0 for the graph 6 is:

$$P_0 = \alpha_{00} |000\rangle + \alpha_{01} |100\rangle + \alpha_{02} |110\rangle + \alpha_{03} |111\rangle$$

If $\alpha_{00} > \alpha_{01} > \alpha_{02} > \alpha_{03}$ we present the path P_0 in figure 9:

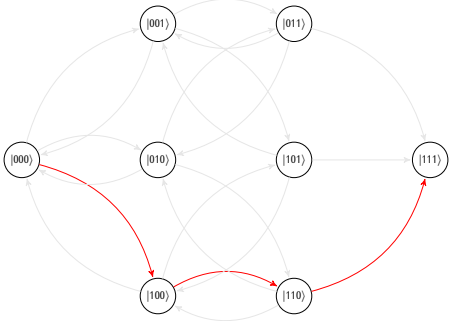


***Figure 9**: The path P_0 if $\alpha_{00} > \alpha_{01} > \alpha_{02} > \alpha_{03}$*

This approach is quite demanding in terms of qubits and should be difficult to implement for real life instances in current hardware. In the next section, we present another hybrid approach [53] to find paths in an acyclic graph.

6.2 Quantum approach to find paths in an acyclic directed graph

Let $G = (\mathsf{V}, \mathbb{E}, \mathsf{C})$ be a graph of n vertices $\mathsf{V} = \{v_1, \ldots, v_n\}$ connected by edges defined in the set $\mathbb{E}$, where each $e_{i,j} \in \mathbb{E}$ represents the connection from the vertices v_i to v_j and has a weight $P_{i,j}$. C is the set of marked vertices on the graph. A path λ in the graph is a succession of steps in the graph that reach a marked vertex $c \in \mathsf{C}$, where the probability of λ is calculated as follows:

$$P(\lambda) = \prod_{e_{i,j} \in \lambda} P_{i,j}$$

The objective of this subsection is to find all paths λ from a given vertex to certain marked vertices using quantum walks. For that, in order to create the space in which these quantum walks pass, we represent the vertices of the graph by qubits, and we build with the help of these qubits a superposition containing all paths λ from the source of the graph v_0 to all marked vertices $c \in C$. In order to apply quantum walks and to store the steps of the walks, we propose our approach based on a path saving strategy using for each vertex of the graph a qubit and each state of these qubits is a path in the graph.

6.2.1 How can we encode paths with quantum states?

In section 5, the algorithms presented can walk through the graph to detect or finf marked elements, but it is not clear whether these walks can be done all over the complete set of paths. Indeed finding a marked element using a quantum walk can be done following only one path. Moreover, even if many paths were followed we have no clue about storing the steps of the paths that lead to a marked element. We propose a strategy to encode paths in order to extract them at the end of all steps. In what remains in this document, the path is encoded with a quantum state as follows:
Suppose that we have a state $|\lambda\rangle = |v_0 v_1 \dots v_n\rangle$ which represents the path λ.

- we say that the vertex v_i is in the path λ if and only if the qubit v_i of the state $|\lambda\rangle$ is in state $|1\rangle$.
- we say that the edge $e_{i,j} = (v_i, v_j)$ is in the path λ if and only if the two qubits v_i and v_j are in states $|1\rangle$ and $|1\rangle$ respectively and the qubits v_k for $i < k < j$ are all in states $|0\rangle$.

With this encoding of vertices using qubits, we can keep the history of each path.

Take the example of figure 10, in this example we represent 3 different paths, each path is represented by a color and the quantum state that represent it is also represented with the same color.

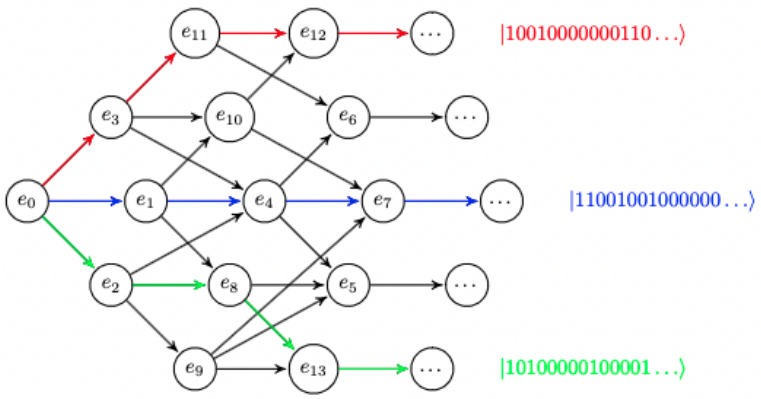


***Figure 10**: A graph representing the failure paths of a small system without crediting any restoration*

The two principal differences between our approach and that of quantum walks represented in the section 5 are:

- with our approach we can find the order of the vertices in paths and with the approach present in 5 we can't;
- the second difference is in the number of used qubits, with our approach for a graph with N vertices we use N qubits versus with the approach present in section 5 only $log_2 N$ qubits are needed. Note that we can possibly have output of 2^N states, and beyond $N = 40$ it is difficult to store classically and measure. This why we thought of making a hybrid version, i.e. processing the circuit part by part and then processing the results classically (and of course this represents a cost for processing large graphs).

6.2.2 How can we manage loops in the graph?

The configuration that we propose to encode paths doesn't allow us to handle the loops in graphs, which can cause some problems if we have graphs with loops. The solution that we can use, is to remove these

loops and change the structure of the graph to keep the possibility of having a single loop configured with more vertices. In the PSA field, assuming that an element can be repaired only once in a scenario. So, to remove these loops and improve the structure of the graph, we propose this small transformation: Consider a graph $G = (\mathsf{V}, \mathbb{E}, \mathsf{C})$, if $(v_i, v_j) \in \mathbb{E}$ and $(v_j, v_i) \in \mathbb{E}$ (there is a loop between v_i and v_j).

1. We remove the edge (v_j, v_i) from the set $\mathbb{E}$;
2. we add a new vertex v_i^* in the set of the vertices V;
3. we add the edge (v_j, v_i^*) in E and also we add the edges $(v_i^*, v_l), \forall v_l \in Succ(v_i) \setminus v_j$. Where $Succ(v_i)$ is the set of successors of v_i.

With the removal of the edge (v_j, v_i) from the set E, we have ensured the removal of the loop, and adding the vertex v_i^* and the edges (v_j, v_i^*) and $(v_i^*, v_l), \forall v_l \in Succ(v_i) \setminus v_j$ makes it possible to keep the loop only one time on paths of the graph, where the loop $v_i \longrightarrow v_j \longrightarrow v_i$ can be represented in the new graph structure by $v_i \longrightarrow v_j \longrightarrow v_i^*$.

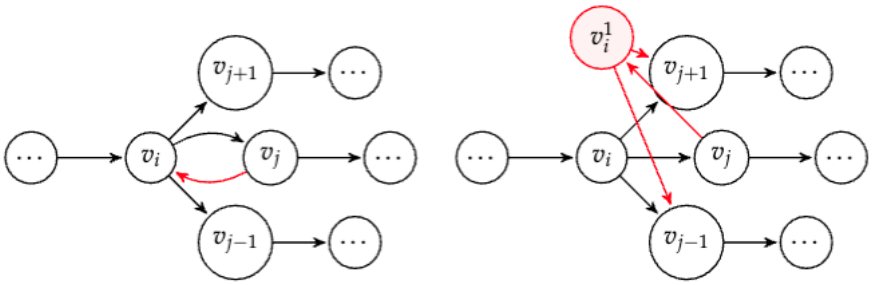


***Figure 11**: Example of the deletion of a loop*

In the figure 11, in order to remove the loop between v_i and v_j we add the vertex v_i^* and the edges $(v_j, v_i^*)(v_i^*, v_{j+1})$ and (v_i^*, v_{j-1}).

6.2.3 How we can use weights of the graph?

The question that remains now is how we can apply quantum walks in a weighted graph and with our configuration? To answer this question, we begin by defining exactly the challenge that we are addressing. Let us suppose that we have done t steps (we have already done t walks on the graph), and that we want to go to the step $t + 1$, so we have a number of paths, we suppose k paths, $\lambda_i, i = 0, \ldots, k$. The goal is then to go forward in the paths at the time t, so we have to go forward to the successors of each path ending (the last vertex of the path). Take the example of the figure 12 where we show an example of a path at time t.

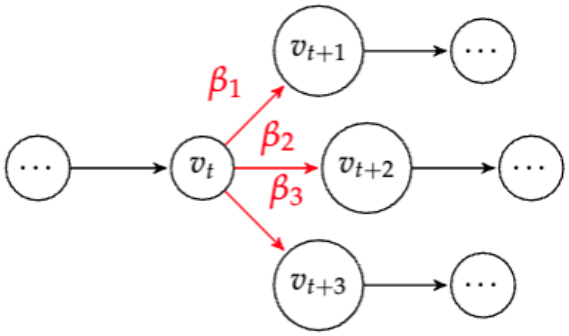


***Figure 12**: Example of processing a path at time t*

Suppose that this path is represented by the state $|\lambda\rangle = |v_0 \ldots v_t\rangle \otimes |v_{t+1} v_{t+2} v_{t+3}\rangle \otimes |v_{t+k+1} \ldots v_n\rangle$. At time t we have already successfully built the path to v_t and we have nothing after v_t. Therefore the quantum state

that represents the path at time t is $|\lambda\rangle = |v_0 \dots v_t\rangle \otimes |000\rangle \otimes |0 \dots 0\rangle$. the vertex v_t has three successors, so if we walk through the three successors we get 3 different paths, where each path contains in addition one of the 3 successors. So, from the state $|\lambda\rangle$ we will extract 3 different states, where each state has a probability β_i. Formally, we are looking for an oracle O that allows us to perform the following transformations:

$$\begin{aligned}|\lambda'\rangle &= O\,|\lambda\rangle \\ &= \frac{\beta_1}{\beta_1+\beta_2+\beta_3}\,|\lambda_1\rangle + \frac{\beta_2}{\beta_1+\beta_2+\beta_3}\,|\lambda_2\rangle \\ &+ \frac{\beta_3}{\beta_1+\beta_2+\beta_3}\,|\lambda_3\rangle\end{aligned}$$

with the three qubits $|v_{t+1}\rangle$,$|v_{t+2}\rangle$ and $|v_{t+3}\rangle$ are in the state $|1\rangle$ respectively in the states $|\lambda_1\rangle$,$|\lambda_2\rangle$ and $|\lambda_3\rangle$ as follows:

$$\begin{aligned}|\lambda_1\rangle &= |v_0 \dots v_t\rangle \otimes |100\rangle \otimes |0 \dots 0\rangle \\ |\lambda_2\rangle &= |v_0 \dots v_t\rangle \otimes |010\rangle \otimes |0 \dots 0\rangle \\ |\lambda_3\rangle &= |v_0 \dots v_t\rangle \otimes |001\rangle \otimes |0 \dots 0\rangle\end{aligned}$$

In quantum walks, we use the Hadamard gate to advance through the graph, the problem with using this gate is that we can't specify the probability for each step of progress and also we can't deal with the case where vertices have 3 or more successors.
Then, to specify the probabilities and also to handle the case of more than two successors for each vertex of the graph, we define the general quantum gate $U(\theta, \phi, \lambda)$. This gate is represented by the following matrix:

$$U(\theta,\phi,\lambda) = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i(\phi+\lambda)}\cos(\theta/2) \end{pmatrix}$$

with the three parameters θ, ϕ and λ, we can make any rotation of $|\psi\rangle$ with respect to x, y and z axis. If we set phi to 0 and lambda to 0 we find the matrix:

$$U(\theta) = U(\theta,0,0) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$$

With this matrix, if we are in the state $|0\rangle$, we can exactly specify the probability of shifting to the state $|1\rangle$ with the θ rotation angle as follows:

$$U(\theta)\,|0\rangle = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos(\theta/2) \\ \sin(\theta/2) \end{pmatrix}$$

We use the $U(\theta)$ gate and we apply the following sub-circuit 13 to find the state $|\lambda'\rangle$. In this circuit, we take the qubit $|v_t\rangle$ as a control to apply the walk to all the paths that have reached the vertex v_t, the first gate $U(\beta_1)$ applies the walk to the vertex v_{t+1} according to the weight β_1 of the edge between v_t and v_{t+1}. After that, we apply the gate X on the qubits v_{t+1} and we use it also as a control to apply the walk to the second successor v_{t+2} and similarly for the third successor v_{t+3}. At the end we apply the two X gates on the two qubits v_{t+1} and v_{t+2} to come back to the state after the walk.

6.2.4 Quantum Oracle for quantum walks

In the general case, suppose that we have k vertices $\{v_t, \dots v_{t+k}\}$ and each $v_i, i \in \{t, \dots, k\}$ has the successors $v_i^s, s \in N$ and each edge $e = (v_i, v_i^s)$ has a probability $\beta_{(v_i, v_i^s)}$, the oracle is defined as follows (Figure 14):

This quantum oracle takes as input a number k of inputs, and it allows to apply the walks to all the successors of each input while keeping the memory of each path.

6.2.5 Quantum algorithm to obtain all paths in an acyclic directed graph

We use the general purpose oracle in figure 14 to introduce the algorithm 2 that allows us to find all the paths between a source and all the marked vertices in an acyclic directed graph.

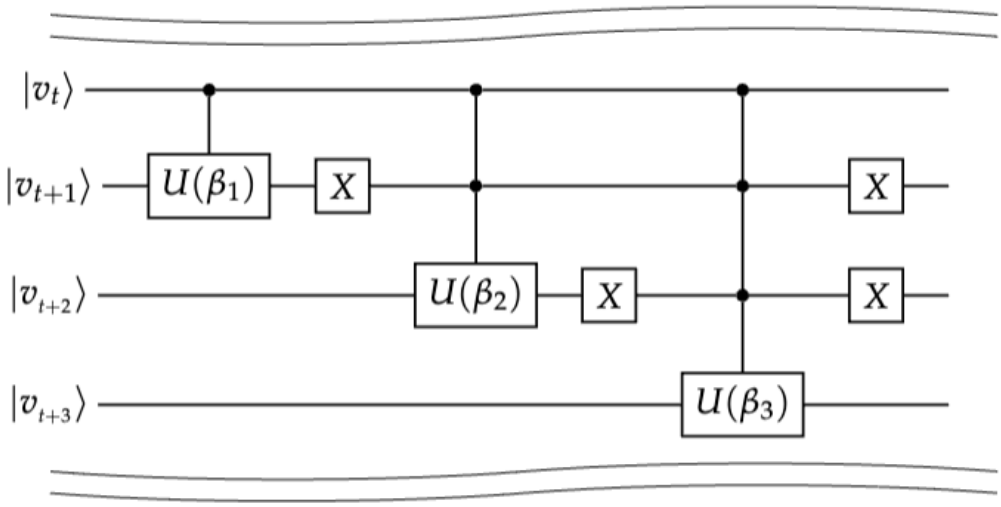


***Figure 13**: Sub-circuit to generate the state $|\lambda'\rangle$*

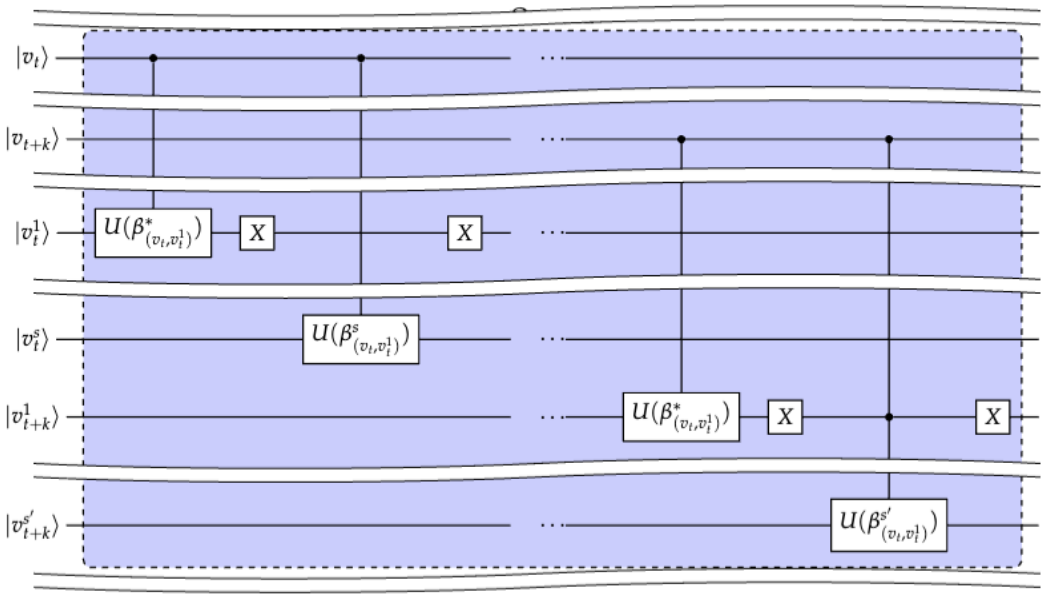


***Figure 14**: The architecture of our oracle*

The algorithm 2 takes as input the graph $G = (\mathsf{V}, \mathbb{E}, \mathsf{C})$, and the source v_0, with V is the set of vertices, $\mathbb{E}$ the set of edges, and C the set of marked vertices in the graph. We start by initializing n qubits to the state $|0\rangle^{\otimes n}$, where $n = |\mathsf{V}|$ is the number of vertices. We activate qubit q_0, and we initialize the list M by the source v_0.

After this initialization step, we start to apply the quantum walks with our configuration and the oracle that we have presented previously. In each iteration, we go to the successors of each element of M to add a step in each path to all successors, in this step for each path λ_i and v_t the end of the path λ_i at time t, we will add k (k is the number of successors of v_t) similar paths up to step t, and in each path among these k paths we will add a successor.

With the quantum oracle, just calling the gate that allows the target qubits to rotate by a specified angle around the base axes allows us to create and add these steps. After that we remove all the elements of M and we add the new vertices of the instant $t+1$ if they are not part of the set of the marked vertices C. We repeat these three iterations as long as the set M is different to the empty set.

Algorithm 2: Quantum algorithm to obtain all paths in a DAG (QAPAG)

Input : Graph $G = (\mathbf{V}, \mathbb{E}, \mathbf{C})$, source v_0.
Initialization
Initialize $n = |V|$ qubits in the state $|0\rangle^{\otimes n}$.
Apply the gate X into the qubit $|v_0\rangle$.
Initialize the set of steps to be handled M with the initial state $M = \{v_0\}$
while $M \neq \emptyset$ **do**
  Apply the oracle $O(M, \{Succ(v_t), \forall v_t \in M\})$.
  Empty M, $M = \{\}$.
  For each $v_i \in Succ(v_t)$ add v_i to M if $Succ(v_i) \neq \emptyset$ and $v_i \notin C$.
Measure the circuit and extract the set of paths P_s.
Return: P_s

At the end we measure the circuit to find the superposition that contains all the states that represents the paths. For each state of this superposition, we extract the corresponding path.

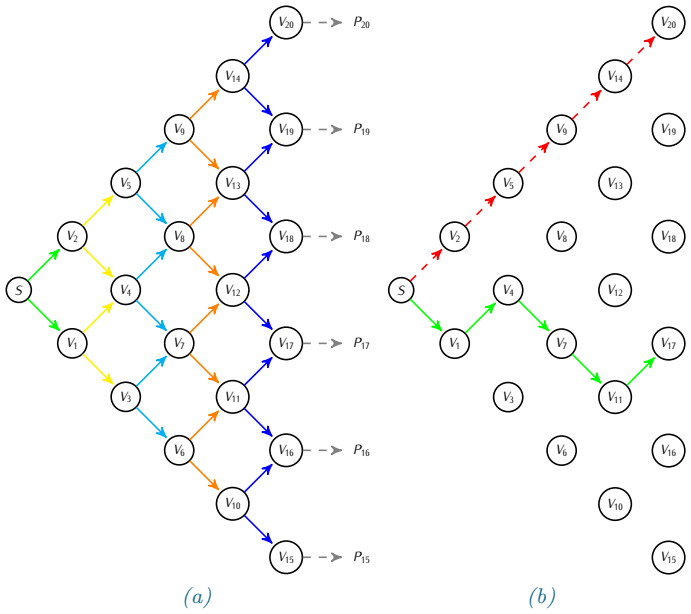


***Figure 15**: (a) Quantum walks, (b) Random walks*

We take for example the graph (b) of figure 15, with the circuit of figure 16 we can apply all the possible steps on the graph to attract the marked vertices. In the graph we show with the colors of each oracle the step that applies.

Each oracle M_i represents a specific case of the general circuit that we have shown in figure 14. In the circuit of figure 16, we see that the oracle M_1 takes only the source S and allows to go to the two qubits v_1 and v_2 in order to apply the rotation that we have shown with the gate $U(\theta)$. The second oracle takes as inputs the two output qubits of the first oracle and the third takes the outputs of the second oracle and so on, until the end. By measuring the circuit we find the superposition that includes all the paths between the source S and all the red vertices of the graph.

6.2.6 Complexity analysis

For the memory complexity, for the graph $G = (\mathbf{V}, \mathbb{E}, \mathbf{C})$, we use $N = |\mathbf{V}|$ qubits (for each vertex we use a single qubit). For the computational complexity, we use at most m $Multi - control - U(\beta)$ gates where $m = |\mathbb{E}|$ is the number of edges. These gates are used to build oracles, which are a combination of gates with a single control qubit and M_s control qubits, where M_s is the maximum number of successors of the vertices of the graph. In addition to that we use at most $2m$ X gates.

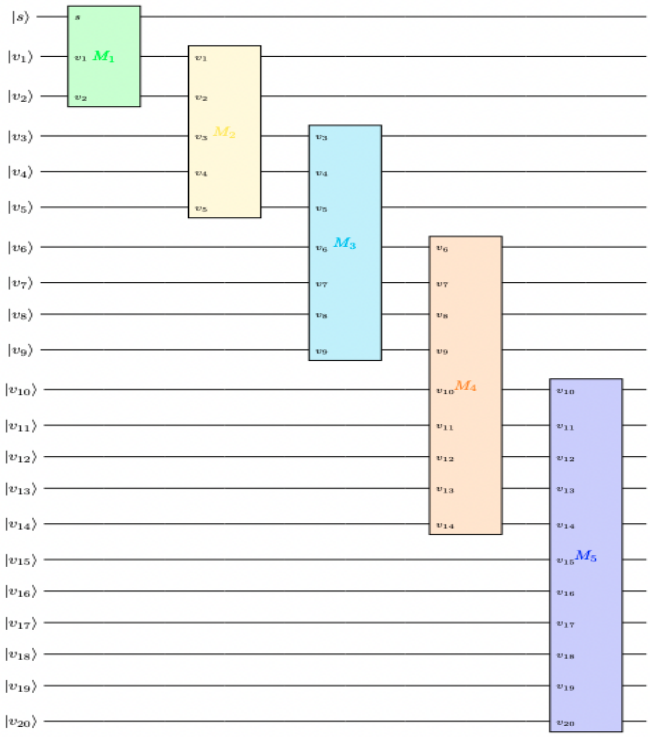


***Figure 16**: The general circuit to find the warping path between two sequences*

6.2.7 Hybrid approach to find paths in an acyclic directed graph

The biggest limitation of the above approach is that it requires a lot of qubits compared to a standard quantum walk, as in our approach we use $|V|$ qubits to represent each path. The approach that we have presented in the section 5. The approach presented in section 4 requires only $log_2 n$ qubits, while ours requires N qubits. In addition to that, the number of qubits currently available in quantum computers or quantum simulators is very small (around 130 or 340 in the best projections for a universal machine). With this, a hybrid approach that can explore even a small number of qubits becomes necessary. The hybridization process means that we should find an approach that allows us to use our algorithm in a way that benefits from the advantage of quantum computers according to the number of qubits available today for large problems. To do this, we need to divide the large circuit generated by the algorithm 2 into several smaller circuits and process them separately, then combine the results at the end. The questions now are:

- How can we cut these circuits?
- And how can we aggregate their results?

To answer these two questions, we propose our approach in this section.

Let's suppose that we have a weighted graph $\mathbb{G} = (\mathsf{V}, \mathbb{E}, \mathsf{C})$, and the number of vertices $N = |\mathsf{V}|$ is very large compared to the number of qubits N_q available in the quantum computer that we are using ($N >> N_q$). Then finding the results with algorithm 2 only becomes impossible.

We start by defining a filter for the search, we define P_{min} as the minimal probability for the paths, i.e. we search for all the paths that have a probability (calculated with the formula 6.2) greater than P_{min}. Of course here, if we want to search all the paths without taking into account the probability P_{min}, we search with $P_{min} = 0$.
Now we start processing the graph $\mathbb{G}$, beginning with the source of the graph s. Assuming we have N_q qubits available in the quantum computer, we extract a subgraph of N_q vertices from the source s. This extraction is done as follows: we take the N_q vertices which are the closest successors to the source s. We start by adding the source and the direct successors of s to the set of vertices, then for each successor we do the same thing as long as we have the number of selected vertices less than N_q. The algorithm 3 allows us to do this extraction.

Algorithm 3: Extraction of a sub-graph

Input : Graph $\mathbb{G} = (\mathbf{V}, \mathbb{E}, \mathbf{C})$, $n = |\mathbf{V}|$, N_q, source s
Initialisation
Initialize a new graph $\mathbb{G}' = (\mathbf{V}' = \emptyset, \mathbb{E}' = \emptyset, \mathbf{C}' = \emptyset)$
Initialize a list $Todo = [S]$
while $|\mathbf{V}'| < N_q$ *and* $|Todo| > 0$ **do**
  Take an element v from the list $Todo$.
  if $|Succ(v)| + |\mathbf{V}'| < N_q$ **then**
    for *all element* v_i *in* $Succ(v)$ **do**
      Add v_i to $\mathbf{V}'$
      Add (v, v_i) to $\mathbb{E}'$ **if** $v_i \in \mathbf{C}$ **then**
        Add v_i to $\mathbf{C}'$
      Add v_i to the $Todo$ list if it doesn't already exist
  Delete v from $Todo$ list
Return: $\mathbb{G}' = (\mathbf{V}', \mathbb{E}', \mathbf{C}')$

Then we use the algorithm 2 of the previous approach to find all the paths of this sub-graph. From the results obtained after this first step, we eliminate the paths that have a probability lower than P_{min}, and we add to the result list the paths that have a probability higher than P_{min} and that arrived at a marked vertex $c_i \in \mathbf{C}$. For the rest of the paths, i.e., the paths that have a probability higher than P_{min} and they haven't yet arrived to a marked vertex (the current paths), we take the end of each path among these current paths and we extract a sub-graph from each item of these vertices, and we use the algorithm 2 with the first oracle take all these output vertices as inputs.

Algorithm 4: Hybrid Quantum algorithm to obtain all paths in an acyclic graph (HQAPAG)

Input : A weighted acyclic graph $\mathbb{G} = (\mathbf{V}, \mathbb{E}, \mathbf{C})$, source v_t, number of qubit available in the quantum computer N_q, and the minimal probability of the paths P_{min}.
Initialisation
Initialize the set of walks to be processed M with the source $M = \{v_t\}$
Initializes the set of paths that arrived at a given vertex marked at $P_g = \emptyset$ and the current paths at $P_t = \emptyset$.
while $M \neq \emptyset$ **do**
  v_t is the first item of M
  Extract a sub-graph $\mathbb{G}_i$ of $\mathbb{G}$ from v_t such that the number of vertices of $\mathbb{G}_i$ is less than N_q.
  Extract the paths $paths_i$ from $\mathbb{G}_i$ using the algorithm 2
  Calculate the exact probability of each path
  Remove the item v_t from the list M
  Update the two path sets P_g,P_t and the set M with the new list of found paths $paths_i$ according to P_{min}.
Return: P_g

Of course if the number of qubits allows to do these tasks at the same time, if not, we can process all the outputs in a separate way. And we do this again until all the paths arrive to the marked vertices or the end of the graph. This whole process is summarized in the algorithm 4.

In the algorithm 4, we proposed to use for each iteration a single source since the number of qubits today in quantum computers is very small. If the number of qubits is a little bit larger, we can simply process all the elements of the list M at the same time in each step. In order to make it more clear, we show the circuit in the general case of a large graph in the figure 17.

Each box marked in blue in the circuit 17 is an iteration of our algorithm 4, where each box takes no more than N_q qubits as input, in the circuit $|\psi\rangle$ is the superposition found after the first step and $|\psi_f\rangle$ is the superposition of the execution at the last step. The results are extracted in the classical way after each part of the circuit, then we take each output as input to the next circuit, this is a better way because we don't need to initialize the superposition and then avoid its computational cost, but we have to call the small circuit which comes several times and this takes time but of course gives good results while waiting for large quantum computers.

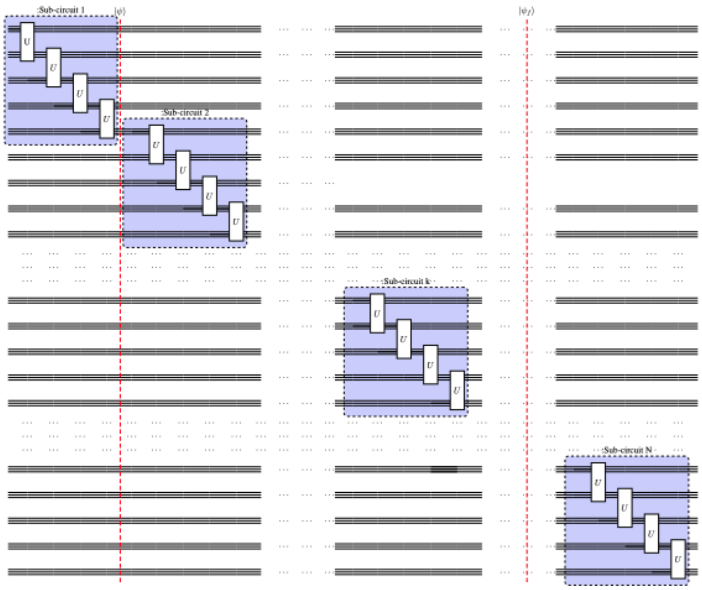


***Figure 17**: Subdivision of the big circuit for a big graph*

With this hybrid approach, we can handle large graphs with our algorithm 2. That means that we can get all paths from the source (or any vertex) of the graph to the marked vertices in an acyclic weighted graph. The weights here play the role in the probability of each state in the superposition at the end, where we find the order of importance of each path. The exact probability of each path with our approach is calculated in a classical way along the running of our hybrid algorithm 4.

6.3 Results and tests

In this subsection, we start by comparing our proposed approach in 3 with the classical Random walk algorithm. To do, we set the maximum running time to 20s, and the minimal probability of sequences to $P_{min} = 10^{-8}$. We have randomly generated 6 graphs of different size and we have chosen the marked vertices in these graphs in a random way, after that we have applied our algorithm and the classical algorithm to find the paths to these marked vertices and we have represented the results in the table 6.3.

Graph	1	2	3	4	5	6
Number of vertices	50	60	70	80	100	120
Number of edges	235	285	335	385	485	585
Number of Paths found by RW	2060	5182	6682	7726	8762	9405
Number of Paths foundby our approach	2463	17134	50989	317324	389869	1475909

In Table 6.3, we can see that in all 6 cases, our approach finds more paths than the classical approach. Also, when we increase the size of the graph, we find a very large number of additional paths compared to the random walk approach.

To find these results we have used the IBM quantum simulator with 32 qubits, then, we have used our algorithm 4 with the setting $N_q = 30$. In order to compare the time spent for each approach, we present the comparative results in the figure 18.

The results presented here are based on tests over an IBM Qiskit library [21], both tests (classical and quantum simulation) are made on the same computer. In figure 18, we can see that the classical time reaches the maximum time given for each approach (20s), and that nevertheless it doesn't succeed to find the number of paths found by our approach, as we have already shown in the table 6.3. For our approach we can see that the time spent to run the quantum circuits is very small compared to the classical time, which shows that even if we use a simulator with a very small number of qubits (32 qubits). We remark in our hybrid approach's results that the classical processing time increases drastically with the graph size, as classical processing is necessary to join the paths found in each partition of the graph. However, it still

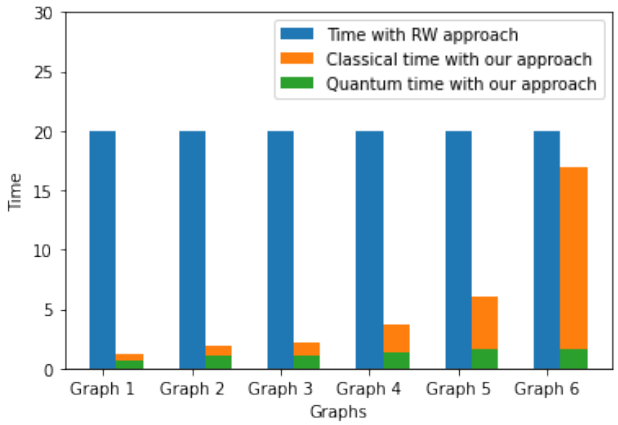
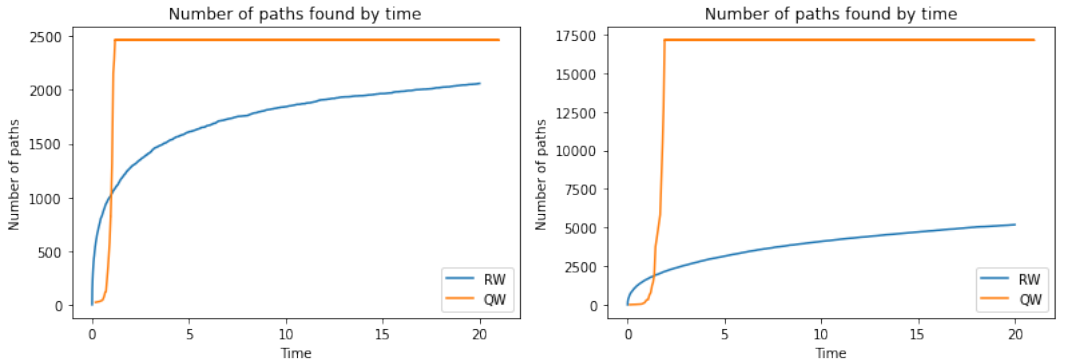


***Figure 18**: The running time spent for each approach*

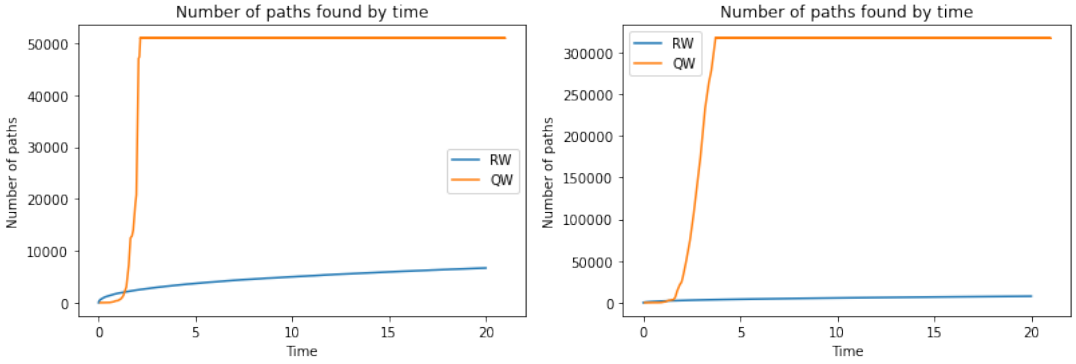
beats the random walk by a large margin, both in terms of the number of paths found and computation time. This classical time spent in our approach will decrease with the increase of the number of qubits in the simulator that we use. In order to compare the functionality of our approach, we show the results of the number of paths found according to time in the figure 19.

In the figure 19, for all 6 graphs, we can see that the search for paths starts with a small advance from the classical random walk, then our approach goes up very quickly and remains stable, contrary to the classical approach which goes up very slowly all the time.

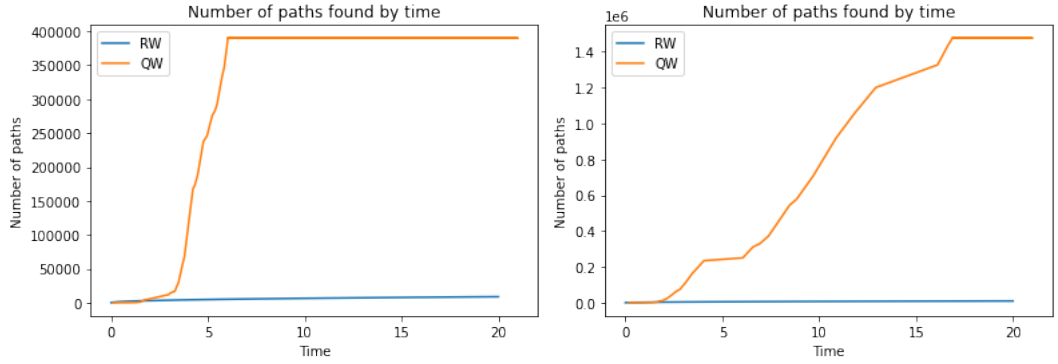
The classical approach has advances in the first iteration because it searches the paths one by one and our approach propagates the graph in a parallel way, to search all the paths at the same time, which requires some time to reach the marked vertices. But when our approach arrives, it arrives with a large number of paths, which implies the increase of the number of paths in this manner. In the case of our approach the results after some time remain unchanged, which means that our approach converges towards all the possible paths to these marked vertices.



(a) Graph 1 *(b)* Graph 2



(c) Graph 3 *(d)* Graph 4



(e) Graph 5 *(f)* Graph 6

***Figure 19**: Number of paths found according to time*

7. Conclusion

In this work, we presented a brief overview of the state of the art of quantum walk algorithms applied to find marked elements in graphs. The main objective is to understand these techniques and then explore how they can be applied to PSA problems in the dynamic Markovian framework.

While in the quantum walks algorithms there is an interest in detecting and finding marked elements in a graph, we proposed two algorithms to address the more general problem of finding sequences to marked elements. The first algorithm is based on move and store strategy to built failure sequences from an initial state to the marked states. It uses a full register assembling qubits representing components of the system under study and other control and working qubits (between n and 2n) to help identify all the sequences.

This approach showed a demanding number of qubits which is not compatible with a realistic implementation on current hardware for industrial systems of relevant size. Therefore another approach was proposed to consider a hybrid approach to consider different cascading circuits for solving relatively large instances.

We presented some tests that were performed to compare our algorithms against classical random walks. These tests showed that classical approach has advances in the first iteration but our approach scales better in the sense that it identifies a large number of paths and converges towards all the possible paths to these marked vertices.

These algorithms however were tested only on a Simulator Qiskit library [21], and need to be tested on a real hardware to see in particular how the hybrid approach of section 6.2.7 behaves in a real hardware.

8. Acknowledgements

We would like to thank the reviewers Mr Vicente Moret Bonillo and Mr Andres Gomez for accepting to review this report and for their valuable suggestions and corrections. Many thanks to Mr. Pedro Henrique Pons Fiorentin for providing a careful review of first version of this document and many helpful suggestions.

Bibliography

[1] V. A., *Sûreté de fonctionnement des systèmes industriels.* Collection de la Direction des Etudes et Recherches d'Electricité de France, Eyrolles, 1988.

[2] D. Aharonov, A. Ambainis, J. Kempe, and U. V. Vazirani, "Quantum walks on graphs," in *STOC*, 2000.

[3] Y. Aharonov, L. Davidovich, and N. Zagury, "Quantum random walks," *Phys. Rev. A*, vol. 48, pp. 1687–1690, Aug 1993. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.48.1687

[4] A. Ambainis, "Quantum Search Algorithms," *SIGACT News*, vol. 35, no. 2, pp. 22–35, 2004.

[5] ——, "New Developments in Quantum Algorithms," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010, pp. 1–11.

[6] A. Ambainis, A. Gilyén, S. Jeffery, and M. Kokainis, "Quadratic speedup for finding marked vertices by quantum walks," 2019.

[7] A. Ambainis, J. Kempe, and A. Rivosh. (2004) Coins Make Quantum Walks Faster.

[8] A. Andris, "Quantum Walks and Their Algorithmic Applications," *International Journal of Quantum Information*, vol. 01, no. 04, pp. 507–518, 2003.

[9] S. Apers, S. Chakraborty, L. Novo, and J. Roland, "Quadratic speedup for spatial search by continuous-time quantum walk," *Physical Review Letters*, vol. 129, no. 16, oct 2022. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.129.160502

[10] S. Apers, A. Gilyén, and S. Jeffery, "A unified framework of quantum walk search," 03 2021.

[11] S. Apers and A. Sarlette, "Quantum fast-forwarding: Markov chains and graph property testing," *Quantum Information & Computation*, vol. 19, pp. 181–213, 2018.

[12] Y. Atia and S. Chakraborty, "Improved upper bounds for the hitting times of quantum walks," *Physical Review A*, vol. 104, no. 3, Sep 2021. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.104.032215

[13] A. Belovs, "Quantum walks and electric networks," 2013.

[14] E. Campos, S. E. Venegas-Andraca, and M. Lanzagorta, "Quantum tunneling and quantum walks as algorithmic resources to solve hard K-SAT instances," *Scientific Reports*, 2021.

[15] S. Chakraborty, L. Novo, and J. Roland, "Finding a marked node on any graph via continuous-time quantum walks," *Physical Review A*, vol. 102, no. 2, aug 2020. [Online]. Available: https://doi.org/10.1103%2Fphysreva.102.022227

[16] S. T. Cheng and M. H. Tao, "Quantum cooperative search algorithm for 3-SAT," *Journal of Computer and System Sciences*, 2007.

[17] F. Di Maio, A. Rai, and E. Zio, "A dynamic probabilistic safety margin characterization approach in support of Integrated Deterministic and Probabilistic Safety Analysis," *Reliability Engineering and System Safety*, 2016.

[18] C. Dohotaru and P. Hloyer, "Controlled Quantum Amplification," in *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), I. Chatzigiannakis, P. Indyk, F. Kuhn, and A. Muscholl, Eds., vol. 80. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, pp. 18:1–18:13. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2017/7489

[19] P. G. Doyle and J. L. Snell. (2000) Random Walks and Electric Networks.

[20] W. Dür, R. Raussendorf, V. M. Kendon, and H.-J. Briegel, "Quantum walks in optical lattices," *Physical Review A*, vol. 66, no. 5, Nov 2002. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.66.052319

[21] M. D. S. et al. ANIS, "Qiskit: An Open-source Framework for Quantum Computing," 2021.

[22] E. Farhi and S. Gutmann, "Quantum computation and decision trees," *Phys. Rev. A*, vol. 58, pp. 915–928, Aug 1998. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.58.915

[23] C. M. Grinstead and J. L. Snell, *Introduction to probability*. American Mathematical Soc., 1997.

[24] J. Guan, Q. Wang, and M. Ying, "An HHL-based algorithm for computing hitting probabilities of quantum walks," *Quantum Information and Computation*, 2021.

[25] A. Hartmanns, M. Klauck, D. Parker, T. Quatmann, and E. Ruijters, "The quantitative verification benchmark set," in *Tools and Algorithms for the Construction and Analysis of Systems*, T. Vojnar and L. Zhang, Eds. Cham: Springer International Publishing, 2019, pp. 344–350.

[26] P. F. P. Henrique, "Quantum-hybrid approaches for probabilistic safety assessment," Master's thesis, ENSIMAG - EDF R&D, 2023 (ongoing).

[27] M. Hibti, A. Zaiou, B. Younes, and B. Matei, "State of the art of sat and psa solvers in the light of quantum computing." [Online]. Available: https://www.neasqc.eu/wp-content/uploads/2022/10/NEASQC_D6.8_Review_State_of_the_art_of_SAT_and_PSA_solvers_in_the_light_of_quantum_computing_v1-2.pdf

[28] Hu, Y.-S., Modarres, M., "Evaluating system behavior through Dynamic Master Logic Diagram (DMLD) modeling," *Reliability Engineering and System Safety*, vol. 64, no. 2, pp. 241–269, 1999.

[29] P. Jackson, Christopher and Apostolakis, G. and Mosleh, Ali and Garrick, B. and Zama, Toshiyuki and Cunningham, Mark and Grantom, C. and Balali, Samaneh and Fleming, Karl and Johnson, David and Kiper, Kenneth and Nelson, *Risk-Informed Decision Making: A Survey of United States Experience*. Unpublished, 2017.

[30] N. Johnston, "What is the intuition for the trace norm (nuclear norm)?" MathOverflow. [Online]. Available: https://mathoverflow.net/q/278055

[31] D. Koch and M. Hillery, "Finding paths in tree graphs with a quantum walk," *Physical Review A*, vol. 97, no. 1, jan 2018. [Online]. Available: https://doi.org/10.1103%2Fphysreva.97.012308

[32] H. Krovi, F. Magniez, M. Ozols, and J. Roland, "Quantum Walks Can Find a Marked Element on Any Graph," *Algorithmica*, vol. 74, pp. 851–907, 2015.

[33] Labeau, P.E., Smidts, C., Swaminathan, S.,, "Dynamic reliability: towards an integrated platform for probabilistic risk assessment." *Reliability Engineering and Systems Safety*, vol. 68, pp. 219–254, 2000.

[34] G. Leung, P. Knott, J. Bailey, and V. Kendon, "Coined quantum walks on percolation graphs," *New Journal of Physics*, vol. 12, no. 12, p. 123018, Dec 2010. [Online]. Available: http://dx.doi.org/10.1088/1367-2630/12/12/123018

[35] H. W. Lewis, R. J. Budnitz, W. D. Rowe, H. J. Kouts, F. Von Hippel, W. B. Loewenstein, and F. Zachariasen, "RISK ASSESSMENT REVIEW GROUP REPORT TO THE U. S. NUCLEAR REGULATORY COMMISSION," *IEEE Transactions on Nuclear Science*, 1979.

[36] P. H. G. Lugão, R. Portugal, M. Sabri, and H. Tanaka. (2022) Multimarked Spatial Search by Continuous-Time Quantum Walk.

[37] F. Magniez, M. Santha, and M. Szegedy, "Quantum algorithms for the triangle problem," *SIAM Journal on Computing*, 2007.

[38] F. Magniez, A. Nayak, P. C. Richter, and M. Santha, "On the hitting times of quantum versus random walks," *Algorithmica*, vol. 63, no. 1–2, p. 91–116, May 2011. [Online]. Available: http://dx.doi.org/10.1007/s00453-011-9521-6

[39] F. Magniez, A. Nayak, J. Roland, and M. Santha, "Search via quantum walk," *SIAM Journal on Computing*, vol. 40, no. 1, p. 142–164, Jan 2011. [Online]. Available: http://dx.doi.org/10.1137/090745854

[40] ——, "Search via quantum walk," *SIAM Journal on Computing*, vol. 40, no. 1, p. 142–164, Jan 2011. [Online]. Available: http://dx.doi.org/10.1137/090745854

[41] A. Montanaro, "Quantum-walk speedup of backtracking algorithms," *Theory of Computing*, vol. 14, no. 1, pp. 1–24, 2018.

[42] A. Pagès and M. Gondran, *Fiabilité des systèmes*, ser. Collection de la Direction des études et recherches d'Électricité de France. Eyrolles, 1980. [Online]. Available: https://books.google.fr/books?id=WJ24ngEACAAJ

[43] D. Reitzner, M. Hillery, and D. Koch, "Finding paths with quantum walks or quantum walking through a maze," *Physical Review A*, vol. 96, no. 3, sep 2017. [Online]. Available: https://doi.org/10.1103%2Fphysreva.96.032323

[44] D. Reitzner, D. Nagaj, and V. Buzek, "Quantum walks," *arXiv preprint arXiv:1207.7283*, 2012.

[45] M. Rennela, S. Brand, A. Laarman, and V. Dunjko, "Divide & Quantum Mathematical Foundation," NEASQC Project, Tech. Rep., 2021.

[46] N. Shenvi, J. Kempe, and K. B. Whaley, "Quantum random-walk search algorithm," *Physical Review A*, vol. 67, no. 5, May 2003. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.67.052307

[47] J. Signoret and A. Leroy, *Reliability Assessment of Safety and Production Systems: Analysis, Modelling, Calculations and Case Studies*, ser. Springer Series in Reliability Engineering. Springer International Publishing, 2021. [Online]. Available: https://books.google.fr/books?id=vIMlEAAAQBAJ

[48] N. Siu, "Risk assessment for dynamic systems: An overview," *Reliability Engineering and System Safety*, 1994.

[49] A. Sweigart, *The Recursive Book of Recursion: Ace the Coding Interview with Python and JavaScript*. No Starch Press, 2022. [Online]. Available: https://books.google.fr/books?id=1lWPzgEACAAJ

[50] M. Szegedy, "Quantum speed-up of markov chain based algorithms," in *45th Annual IEEE Symposium on Foundations of Computer Science*, 2004, pp. 32–41.

[51] A. Villemeur, *Sureté de Fonctionement des Systèmes Insdustriels*. Eyrolles, 1997.

[52] Y. Xu, D. Zhang, and L. Li, "Robust quantum walk search without knowing the number of marked vertices," *Physical Review A*, vol. 106, no. 5, Nov 2022. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.106.052207

[53] A. Zaiou, "Quantum machine learning approaches for graphs and sequences : application to nuclear safety assessment," Ph.D. dissertation, 2022, thèse de doctorat dirigée par Mateï, Basarab Informatique Paris 13 2022. [Online]. Available: http://www.theses.fr/2022PA131048

[54] A. Zaiou, Y. Bennani, M. Hibti, and B. Matei, "Quantum approach for vertex separator problem in directed graphs," in *Artificial Intelligence Applications and Innovations*, I. Maglogiannis, L. Iliadis, J. Macintyre, and P. Cortez, Eds. Cham: Springer International Publishing, 2022, pp. 495–506.