Instituto Superior Técnico

PIC1 - LEFT

Staggered Quantum Walk on Weighted Graphs

Project type: Scientific

Student: João Martins

Supervisors: Yasser Omar

Academic Year 2024/2025

Staggered Quantum Walk on Weighted Graphs

João Martins*
Instituto Superior Técnico

(Dated: June 28, 2025)

Quantum walks are among the main tools capable of harnessing quantum advantage in graph problems. The staggered quantum walk has shown promise in addressing problems on spatial graphs, having achieved a quadratic speedup for spatial search. In this work, we present an extension of this model to weighted graphs. As a result, we study the application of the staggered quantum walk to a welded tree, and of our extended model to a simple random weighted graph, verifying the expected scaling for the number of steps $O(\sqrt{N})$. Furthermore, we test our model on a weighted graph containing welded trees, taking the first steps towards upgrading the *marked vertex* problem into a pathfinding algorithm for our new Weighted Staggered Quantum Walk.

I. INTRODUCTION

Quantum mechanics has fundamentally altered our understanding of the physical world, introducing concepts that, while not necessarily complex, challenge our everyday intuition. One such concept is superposition. The idea that a system can exist in multiple, even contradictory, states simultaneously. This shift in perspective gave rise to quantum computation.

Unlike classical computers, which process information using bits that are either 0 or 1, quantum computers use *qubits*, which can be in a superposition of both. Classical algorithms follow Boolean logic, but quantum algorithms must adhere to a different set of rules, those defined by quantum mechanics.

This new paradigm has led to the development of quantum algorithms that surpass classical approaches in specific tasks. A prominent example is Shor's prime factoring algorithm, which achieves an exponential speedup over the best known classical methods, which poses serious implications for cryptography, as many widely used schemes depend on the hardness of factoring large integers [1]. Quantum computers also show clear advantages in simulating quantum systems, with important applications across Condensed Matter Physics, Molecular Chemistry, and Materials Science [2].

Many computational problems can also be framed using graph theory. Among the most notable quantum algorithms is Grover's search algorithm, which provides a quadratic speedup for unstructured search tasks [3]. Furthermore, the study of Quantum Walks (QW) (the quantum analogue of classical random walks) has proven to be a powerful framework for tackling graph-based problems.

Graph-based problems appear in a wide range of real-world applications, from social networking analysis and search on the Internet to logistics, biological systems, and quantum chemistry. These problems often involve identifying optimal paths, detecting clusters, or searching for specific nodes in large, complex structures. Quantum walks, which naturally capture the dynamics of information propagation on graphs, offer a powerful tool for addressing these challenges.

Naturally, one of the most important and widely applicable classes of graphs is that of weighted graphs, as the inclusion of edge weights allows for the modelling of more complex relationships and dynamics. This added structure enables a more faithful representation of real-world systems, such as transportation networks or communication systems. Using the Staggered Quantum Walk model, which has already been proven to be promising in problems such as spatial search [4], we propose its extension to weighted graphs. Our proposal aims to significantly broaden its scope and practical utility. It opens the door to exploring a wider range of computational problems within the quantum computing framework, particularly those where edge weights play a crucial role in determining optimal paths.

A. Quantum Walks

The quantum walk is a powerful technique for building quantum algorithms and for simulating complex quantum systems. A quantum walk takes place on a graph whose vertices are the places the walker may step and whose edges tell the possible directions the walker can choose to move. Space is discrete, but time can be discrete or continuous [5].

In the case of a Discrete-Time Quantum Walk (DTQW), the motion consists of a succession of steps from one vertex to the next. The walker starts in some initial state $|\psi_0\rangle$ and the dynamic in its simplest form is described by a unitary operator U, which will be our *evolution operator*. Naturally, we can describe our state as follows:

$$|\psi(t)\rangle = U^t |\psi_0\rangle \tag{1}$$

where $t \in \mathbb{Z}_0^+$.

In the case of a Continuous-Time Quantum Walk (CTQW), there is a transition rate that controls the jump probability, which starts with a small value and increases continuously so that the walker eventually steps to the next vertex. The dynamic is described by the unitary operator $U(t) = e^{itH}$, where t is time and H is an Hermitian matrix, whose entries are non-zero only if they correspond to neighbouring vertices. This matrix is usually the adjacency matrix (A) or the Laplacian matrix (L). There are other instances that are closely related to the previous ones [6]. Naturally, similar to equation 1, assuming that $|\psi_0\rangle$ is our initial state, the state of the system will be:

$$|\psi(t)\rangle = U(t) |\psi_0\rangle = e^{itH} |\psi_0\rangle \tag{2}$$

In this work, our focus will be DTQW, therefore, we will start to introduce some of the main models for this case. Additionally, it is necessary to mention that in Appendix A can be found some important definitions for Graph Theory.

* joao.ferreira.martins@tecnico.ulisboa.pt

B. Discrete-Time Quantum Walk models

1. Coined Quantum Walk

One of the simplest models of classical random walks is the *coined walk*. In this case, we present its analogue, the Coined Quantum Walk (CQW). When applied to the motion of a particle on a line, this model uses two registers: one to store the particle's position and the other to determine its next move. In the classical case, the second register would be updated through a random process, effectively guiding the direction of the walk. We can translate this into the quantum case ([7], [8]). The walker (particle) is described as a vector $|n\rangle$ in a Hilbert space $\mathcal{H}_P$ of infinite dimension, the computational basis of which is $\{|n'\rangle : n' \in \mathbb{Z}\}$. The evolution of the walk depends on the quantum "coin". If we obtain "heads" ($|0\rangle$, for example), we move to $|n+1\rangle$. Similarly, if we obtain "tails"($|1\rangle$), we move to $|n-1\rangle$. Therefore, the Hilbert space $\mathcal{H}_C$ of this register will be $\{|0\rangle, |1\rangle\}$. As a result, the state will be a vector on the Hilbert space $\mathcal{H} = \mathcal{H}_C \otimes \mathcal{H}_P$

Additionally, we need to define the evolution operator. We first define the *shift operator* S, which will change the position of the walker. Naturally, S acts as follows:

$$S|0\rangle|n\rangle = |0\rangle|n+1\rangle \tag{3a}$$
$$S|1\rangle|n\rangle = |1\rangle|n-1\rangle \tag{3b}$$

If we give a complete description of the S operator, we obtain

$$S = |0\rangle\langle 0| \otimes \sum_{n=-\infty}^{\infty} |n+1\rangle\langle n| + |1\rangle\langle 1| \otimes \sum_{n=-\infty}^{\infty} |n-1\rangle\langle n| \tag{4}$$

In addition to S, we define the coin operator, which is defined as $(C \otimes I_P)$ due to only acting on the first register. The most used operator is known as the *Hadamard Coin*:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{5}$$

However, there are also other coins, such as the *Fourier Coin* and the *Grover Coin*.

In summary, the evolution operator for one time step will be:

$$U = S(H \otimes I_P) \tag{6}$$

The evolution is illustrated in figure 1 and the state after t steps is given by equation 1.

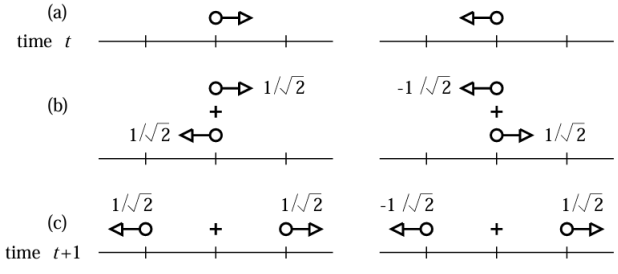


FIG. 1: Coined Quantum Walk on a Line. Taken from Ref. [7]. (a) The particle is in a position n with the first register as $|0\rangle$ (left) and $|1\rangle$ (right); (b) state of the walker after applying the coin operator $(H \otimes I_P)$; (c) state of the walker after applying the shift operator S.

Moreover, the previous example was for a simple infinite line, but the CQW can be applied to more general graphs (Appendix B).

2. Szegedy Quantum Walk

Another discrete-time QW model that has contributed greatly to the development of the field is the Szegedy Quantum Walk (SzQW) [9]. In order to understand the SzQW, we will briefly introduce some notions related to Markov Chains.

A classical discrete-time *stochastic process* is a sequence of random variables $\{X_t : t \in \mathbb{N}\}$, where X_t represents the state of the process at time t and X_0 is the initial state. The *state space* $\mathcal{S}$ is assumed to be discrete, for example, $\mathcal{S} = \mathbb{N}$.

A *Markov chain* is a stochastic process where the future depends only on the present state:

$$\begin{aligned}\text{Prob}(X_{t+1} = j \mid X_t = i, X_{t-1} = i_{n-1}, \ldots, X_0 = i_0) = \\ = \text{Prob}(X_{t+1} = j \mid X_t = i).\end{aligned} \tag{7}$$

Let $p_{ij} = \text{Prob}(X_{t+1} = j \mid X_t = i)$ and assume that p_{ij} is independent of t (*time-homogeneous Markov chain*). The matrix $P = [p_{ij}]$ is called the *transition matrix*, satisfying:

$$p_{ij} \geq 0 \quad \text{and} \quad \sum_{j \in \mathcal{S}} p_{ij} = 1 \quad \text{for all } i \in \mathcal{S}. \tag{8}$$

Any time-homogeneous Markov chain can be represented as a digraph $\Gamma(V, A)$, where $V = \mathcal{S}$ is the set of vertices and $(i, j) \in A$ if and only if $p_{ij} > 0$. If P is symmetric, then Γ becomes an undirected graph, and the Markov chain becomes a simple random walk.

Szegedy's quantum walk is defined on a bipartite graph obtained by duplicating an underlying graph associated with a discrete-time Markov chain, just like it is showed in figure 2.

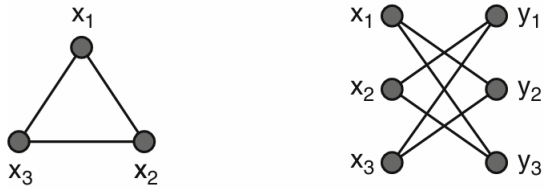


FIG. 2: Extension of the original graph into a bipartite graph. Taken from Ref. [5]

The transition probabilities, satisfy $\sum_{y \in Y} p_{xy} = 1, \quad \forall x \in X$ and $\sum_{x \in X} q_{yx} = 1, \quad \forall y \in Y$.

We define a basis spanned by $\{|\alpha_x\rangle\}$ and $\{|\beta_y\rangle\}$ where, using the computational basis of $\mathcal{H}^{n^2} = \mathcal{H}^n \otimes \mathcal{H}^n$, its elements are defined as:

$$|\alpha_x\rangle = |x\rangle \otimes \left(\sum_{y \in Y} \sqrt{p_{xy}}\,|y\rangle\right) \tag{9a}$$
$$|\beta_y\rangle = \left(\sum_{x \in X} \sqrt{q_{yx}}\,|x\rangle\right) \otimes |y\rangle \tag{9b}$$

and its orthogonal projections

$$\Pi_A = \sum_{x \in X} |\alpha_x\rangle\langle\alpha_x| \tag{10a}$$

$$\Pi_B = \sum_{y\in Y} |\beta_y\rangle \langle\beta_y| \tag{10b}$$

From these, we define the reflection operators:

$$\mathcal{R}_A = 2\Pi_A - I_{n^2} \tag{11a}$$

$$\mathcal{R}_B = 2\Pi_B - I_{n^2} \tag{11b}$$

The evolution operator for the quantum walk is then:

$$W_P = \mathcal{R}_B\mathcal{R}_A \tag{12}$$

At time t, the state of the quantum walk is $(W_P)^t$ applied to the initial state. This operator-based construction enables analysis via spectral decomposition, leading to insights into properties such as hitting time, which can outperform classical analogues on the underlying graph.

In Ref. [10], it was proven the equivalence between CQW, SzQW and the Staggered QW model that we will study for simple (unweighted) graphs. This model is also important to our goal as it enables the application to weighted graphs.

3. Electrical Networks Quantum Walk

The two last examples are some of the most foundational models in the discrete-time QW framework. More recently, there has been some development in another interesting possibility that gave useful insights for our proposal for the weighted staggered quantum walk, as we will see. The first connection was first established in Ref. [11]. Subsequently, other authors studied and developed this model, applying it to different problems [12–14] or suggesting new concepts, such as a multidimensional approach [15, 16].

Let $G(V, E, w)$ be a simple weighted graph. We can consider G as an electrical network with each edge assigned a positive weight $w_{u,v}$, that is, its conductance (Appendix A 2). This weight assignment gives rise to a weighted superposition of neighbours in the neighbourhood $\Gamma(u)$ of any vertex u, normalized by the quantity $w_u = \sum_{v\in\Gamma(u)} w_{u,v}$, which is known as the *star state* of u:

$$\frac{1}{\sqrt{w_u}} \sum_{v\in\Gamma(u)} \sqrt{w_{u,v}}\ |u, v\rangle$$

This state can be thought of as a quantum encoding of the probability of moving from a vertex u to each neighbour v. Instead of using E, we can assign arbitrary directions to each edge, creating a new edge set $\vec{E}$. As a consequence, our star state will depend on the Boolean variable $\Delta_{u,v}$, which is equal to 0 if $(u, v) \in \vec{E}$ and 1 if $(v, u) \in \vec{E}$:

$$|\psi_u\rangle = \frac{1}{\sqrt{w_u}} \sum_{v\in\Gamma(u)} (-1)^{\Delta_{u,v}} \sqrt{w_{u,v}}\ |u, v\rangle \tag{13}$$

We should also note that the Hilbert space is $\mathcal{H}^{|E|}$, as the computational basis is $\{|u, v\rangle : (u, v) \in E\}$. In this model, we study $s-t$ flows on our graph G between any two of its vertices s and t, which can be thought of as the initial state and the marked vertex, respectively.

Due to considering the graph as an electrical network, we define the flow θ, which will be analogous to an electrical current. The flow θ can be defined as the quantity that minimises the energy of the electrical flow:

$$\mathcal{E}(\theta) = \sum_{(u,v)\in\vec{E}} \frac{\theta_{u,v}^2}{w_{u,v}}. \tag{14}$$

The minimal energy of the $s-t$ electrical flow θ is called the *effective resistance* $\mathcal{R}_{s,t}$ and this flow gives rise to the normalized electrical flow state $|\theta\rangle$:

$$|\theta\rangle = \frac{1}{\sqrt{2\mathcal{R}_{s,t}}} \sum_{(u,v)\in\vec{E}} \frac{\theta_{u,v}}{\sqrt{w_{u,v}}}(|u, v\rangle + |v, u\rangle) \tag{15}$$

Kirchhoff's Law states that this $s-t$ electrical flow is conserved at every vertex $u \in V\backslash s, t$, which means that the amount of flow θ entering u is equal to the amount of flow exiting u. This law can be equivalently read in terms of $|\psi_u\rangle$ and $|\theta\rangle$, in which case it states that for every vertex $u \in V\backslash\{s, t\}$ we require

$$\langle\psi_u|\theta\rangle = 0. \tag{16}$$

In the case where θ is the $s-t$ electrical flow, we define the (unnormalized) state associated with the induced potential vector p (with the convention that $p_t = 0$) as

$$|p\rangle = \sqrt{\frac{2}{\mathcal{R}_{s,t}}} \sum_{u\in V\backslash\{s\}} p_u\sqrt{w_u}\ |\psi_u\rangle \tag{17}$$

where p_u is according to equation A4.

This model seems to have a higher affinity for the study of weighted graphs. Therefore, it gives great insight into how we can explore and extend the next model to new problems. In Appendix C can be found the definition of the evolution operator and how it is applied. We decided to not include that part in the introduction due to its fundamental difference compared to our model.

II. STAGGERED QUANTUM WALK AND ITS EXTENSION

A. Staggered Quantum Walk

1. Model

The Staggered Quantum Walk (SQW) was presented as a broader model compared to the SzQW [5, 17, 18]. The fundamental difference from the other models is the construction of a tessellation cover for a graph and the use of it to construct the evolution operator.

A *graph tessellation* $\mathcal{T}$ is a partition of the vertex set into cliques. An edge *belongs* to the tessellation $\mathcal{T}$ if and only if its endpoints belong to the same clique in $\mathcal{T}$. The set of edges belonging to $\mathcal{T}$ is denoted by $E(\mathcal{T})$. An element of a tessellation is called a *polygon*. The *size* of a tessellation $\mathcal{T}$ is the number of polygons in $\mathcal{T}$.

Given a graph G with edge set $E(G)$, a *graph tessellation cover* of size k of G is a set of k tessellations $\mathcal{T}_1$,...,$\mathcal{T}_k$, whose union covers the edges, that is $\bigcup_{i=1}^{k} E(\mathcal{T}_i) = E(G)$.

A graph G is called *k-tessellable* if there is a tessellation cover of size at most k. The size of a smallest tessellation cover of G is called *tessellation cover number* and is denoted by $T(G)$.

Firstly, let $G(V, E)$ be a simple connected graph such that $|V| = N$. Let $\mathcal{H}^N$ be the N-dimensional Hilbert space spanned by the computational basis $\{|v\rangle\,,\ v \in V\}$, that is, each vertex v is associated with a vector $|v\rangle$ of the computational basis. In the

staggered model, there is a one-to-one correspondence between the set of vertex labels and the states of the computational basis, just like in the CQW and SzQW. Furthermore, we need to compute a tessellation cover for our graph. Let us use figure 3 as example.

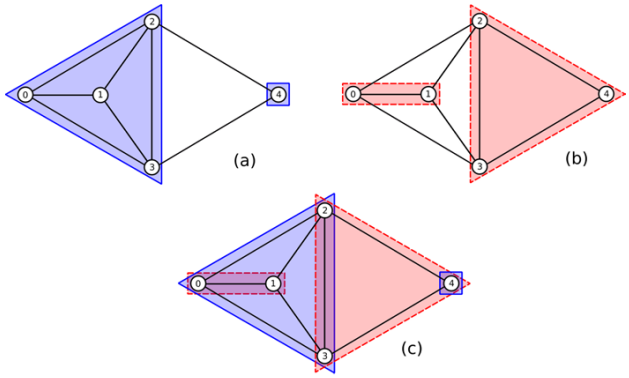


FIG. 3: Tesselation Cover for a graph. Taken from ref. [18]. (a) blue tessellation; (b) red tessellation; (c) tessellation cover, which covers all edges of the graph

Step 1: Compute a tessellation cover for the graph. Note that a polygon is always a clique, but not necessarily a maximal clique. Therefore, the tessellation cover is not unique. However, this will not be a problem for the extension that will be further proposed. From now on, we suppose that a tessellation cover $\{\mathcal{T}_1, ..., \mathcal{T}_k\}$ of size k is known.

Step 2: Compute the vectors associated with each polygon in each tessellation so that each polygon belongs to the subspace spanned by its vertices. Suppose that tessellation $\mathcal{T}_i$ has p polygons, each denoted by α_j, that is, $\mathcal{T}_i = \{\alpha_j : 1 \leq j \leq p\}$. We associate a unit vector with each polygon as follows:

$$|\alpha_j\rangle = \frac{1}{\sqrt{|\alpha_j|}} \sum_{v \in \alpha_j} |v\rangle \tag{18}$$

where $|\alpha_j|$ is the number of vertices of the polygon α_j.

For example, for the blue tessellation in figure 3(a), its vectors will be $|\alpha_0\rangle = \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle)$, which is the 4-clique on the left, and $|\alpha_1\rangle = |4\rangle$, which is the 1-clique (vertex) on the right.

Naturally, the states associated with the polygon in the red tessellation will be analogous.

Step 3: Define the operator associated with each tessellation. The operator associated with a tessellation $\mathcal{T}_i$ will be:

$$W_i = 2 \sum_{j=1}^{p} |\alpha_j\rangle \langle\alpha_j| - I \tag{19}$$

There are a few observations to make. Firstly, by construction, W is unitary and hermitian ($W^2 = I$) because $\langle\alpha_j|\alpha_{j'}\rangle = \delta_{jj'}$ for $1 \leq j, j' \leq p$. Secondly, W is an orthogonal reflection on the subspace spanned by $\{|\alpha_j\rangle : 1 \leq j \leq p\}$.

Moreover, we define a linear operator H as a *local operator* with respect to a graph G when $\langle v_1| H |v_2\rangle = 0$ if v_1 and v_2 ($v_1 \neq v_2$) are non-adjacent.

We can easily show that W is a local operator. Suppose that we have v_1 and v_2 that are non-adjacent. Each vertex belongs to one polygon. Therefore, if v_1 belongs to a polygon α_1, v_2 does not. Let us assume, without loss of generality, that v_2 belongs to α_2. Then $\langle v_1| W |v_2\rangle = \langle v_1|v_2\rangle - 2(\langle v_1|\alpha_1\rangle \langle\alpha_1|v_2\rangle + \langle v_1|\alpha_2\rangle \langle\alpha_2|v_2\rangle) = 0$. We conclude, as required when defining a quantum walk, that W is a local unitary operator.

Step 4: Define the evolution operator. For a tessellation cover $\{\mathcal{T}_1, ..., \mathcal{T}_k\}$, the evolution operator will be:

$$U = \prod_{j=1}^{k} W_j \tag{20}$$

We can further extend this model ([19]) by defining:

$$U = e^{i\theta_1 W_1} e^{i\theta_2 W_2} ... e^{i\theta_k W_k} \tag{21}$$

where we can easily verify that, due to $W^2 = I$, we can expand each term as

$$e^{i\theta_j W_j} = \cos\theta_j \; I + i \sin\theta_j \; W_j \tag{22}$$

Naturally, for the case $\theta_j = \frac{\pi}{2}$ for every j, we get the initial evolution operator defined in equation 20. For further reference, we will always assume $\theta_1 = \theta_2 = ... = \theta$.

2. *Quantum Search*

Let us formalize the usual problem for quantum search. Firstly, we suppose that there exists a set M of *marked vertices* that we want to find. Secondly, let us suppose that there exists a function $f(x)$ which outputs $f(x) = 1$ if $x \in M$ or $f(x) = 0$ if $x \notin M$. This function can be called an *Oracle*, as it essentially confirms whether we have, in fact, found a marked vertex or not. As a result, we can write the problem for searching a marked vertex in the following way:

Problem: Given the oracle of the graph $G(V, E)$, the goal is to output the marked vertex.

When analysing the oracle, it can be expressed by creating an operator $\mathcal{O}$, which will act on a state as follows:

$$|x\rangle |q\rangle \xrightarrow{\mathcal{O}} |x\rangle |q \oplus f(x)\rangle \tag{23}$$

where $|x\rangle$ is the index register, $|q\rangle$ is the oracle qubit and $\oplus$ denotes the addition modulo 2. In case we define $|q\rangle = \frac{(|0\rangle - |1\rangle)}{\sqrt{2}}$, then equation 23 becomes:

$$|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \xrightarrow{\mathcal{O}} (-1)^{f(x)} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \tag{24}$$

Due to neither the index register nor the oracle qubit changing after applying $\mathcal{O}$, we can omit the oracle qubit and write equation 24 as

$$|x\rangle \xrightarrow{\mathcal{O}} (-1)^{f(x)} |x\rangle \tag{25}$$

This means that the oracle $\mathcal{O}$ identifies the marked states by changing the phase of the solution. As a result, we can define the operator R, which has the function of the oracle and can be defined as

$$R = I - 2 \sum_{v \in M} |v\rangle \langle v| \tag{26}$$

As a result, we can create a *modified evolution operator* [2]. This new operator U' can be defined by equation 27.

$$U' = UR \tag{27}$$

B. Extension to Weighted graphs

We now present our original work, which is an extension of the Staggered Quantum Walk to weighted graphs. As we can verify in equation 18, uniform weights were assumed for each polygon, which means that there is no bias in the evolution of the system, which is to be expected when analysing unweighted graphs. However, we can use this freedom to define our polygon states to encode edge weights, as we will now explain.

Firstly, let $G(V, E, w)$ be an arbitrary weighted graph. We will now present the process on how to construct this model, using figure 4.

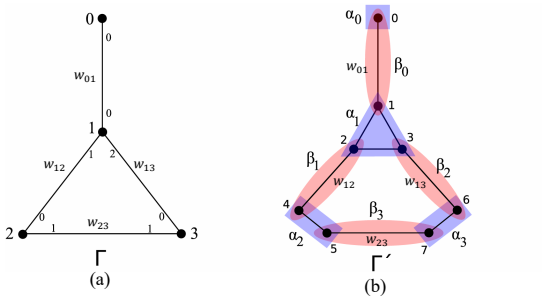


FIG. 4: Example of an arbitrary graph G (a) and a new graph G' after applying clique–insertion operator (b)

Step 1: Apply the clique-insertion operator on G, creating G'. Notice that the only weights that will matter are the ones that were previously defined in the original graph and will therefore remain at the edges that connect the cliques.

Step 2: Define a tessellation cover. Naturally, this new graph G' will have two tessellations. One for the new $d(v)$-cliques that have replaced each vertex v in G and another for the edges that connect each clique on G'. Without loss of generality and following the same logic as in figure 4(b), we will name "blue tessellation" when we refer to the tessellation that includes the cliques and "red tessellation" when we refer to the tessellation from the edges.

Step 3: Define the new states for each polygon. For the red tessellation, we will use the same definition as we used before according to equation 18. For the blue tessellation, we will define it as

$$|\alpha_i\rangle = \frac{1}{\sqrt{\sum_j w_{ij}}} \sum_j \sqrt{w_{ij}} \ |v_{ij}\rangle \tag{28}$$

where we define $|v_{ij}\rangle$ as the vector that represents a vertex that belongs to the i-th clique and is adjacent to another vertex in the j-th clique. Naturally, the edge that connects both cliques will be the edge that had the weight w_{ij}.

For example, the red tessellation will be:

$$|\beta_0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad |\beta_1\rangle = \frac{|2\rangle + |4\rangle}{\sqrt{2}},$$
$$|\beta_2\rangle = \frac{|3\rangle + |6\rangle}{\sqrt{2}}, \quad |\beta_3\rangle = \frac{|5\rangle + |7\rangle}{\sqrt{2}}$$

and the blue tessellation will be:

$$|\alpha_0\rangle = |0\rangle$$
$$|\alpha_1\rangle = \frac{1}{\sqrt{w_{01} + w_{12} + w_{13}}} \left(\sqrt{w_{01}}\,|1\rangle + \sqrt{w_{12}}\,|2\rangle + \sqrt{w_{13}}\,|3\rangle\right)$$
$$|\alpha_2\rangle = \frac{1}{\sqrt{w_{12} + w_{23}}} \left(\sqrt{w_{12}}\,|4\rangle + \sqrt{w_{23}}\,|5\rangle\right)$$
$$|\alpha_3\rangle = \frac{1}{\sqrt{w_{13} + w_{23}}} \left(\sqrt{w_{13}}\,|6\rangle + \sqrt{w_{23}}\,|7\rangle\right)$$

We can now note why, according to our construction, there is no need to define weights inside each clique from the blue tessellation.

The following steps are completely analogous to what was previously defined for the Staggered Quantum Walk, in terms of creating the evolution operator.

III. APPLICATIONS

A. Welded tree Analysis

1. *Welded Tree Problem*

One of the most well-known problems in graph theory is the welded tree, probably because of the advantage that the quantum approach presents compared to the classical one. Consequently, it has been an object of discussion in the context of quantum walks [6, 12, 16, 20]. In addition to this, welded trees are part of the last problem that we will study. Therefore, we consider that an analysis on it would be pertinent. This problem can be formulated as follows.

Problem: Given a welded tree, an oracle for the graph, and the name of the *entrance*, the goal is to output the name of the *exit*.

A welded tree graph W consists of two balanced binary trees of depth n with roots s (entrance) and e (exit), and a random cycle that alternates between the leaves of the two binary trees. It is also important to note that the number of nodes is $N = 2^{n+2} - 2$, where n is the depth of the tree. An example of this is shown in figure 5.

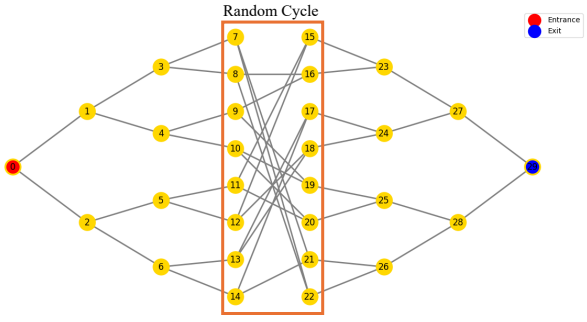


FIG. 5: Welded Tree Graph where each tree has depth 3

Note that the name *random cycle* might be considered misleading as in graph theory *cycle* has a different definition. However, the literature uses this expression, so we will keep it that way. Also note that all vertices in the random cycle also have degree 3. The red and blue nodes will be the entrance (initial state) and the exit (marked vertex).

2. *Staggered Quantum Walk on a welded tree*

In order to analyse the SQW model in a welded tree, we need to compute its tessellation cover. In figure 6, we have an example.

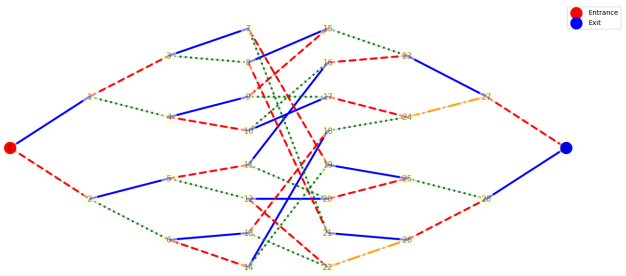


FIG. 6: Tesselation cover of a welded tree of depth 3 with a random cycle.

In a colouring of the edges of a graph G, the edges incident with a vertex get distinct colours, so we define the *edge-chromatic number*, $\chi'(G)$, as the smallest number of colour necessary to colour all edges. Notice that in a welded tree, due to its structure, computing a tessellation cover is equivalent to colouring the edges. As a result, we can use some known results from graph theory.

Theorem (Th.7, Ch.7 [21]): A graph G of maximal degree Δ has edge-chromatic number Δ or $\Delta + 1$.

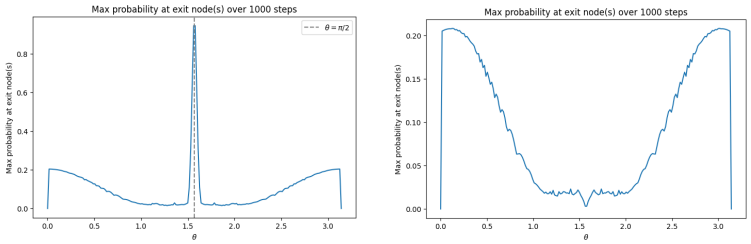
We know that for a welded tree, $\Delta = 3$, therefore, we can tessellate our graph with 4 colours at most.

We define the initial state $|\psi_0\rangle = |s\rangle$, where $|0\rangle$ is the left root. Then, we compute the evolution operator according to Section II, using equation 21 and apply it to the initial state according to equation 1.

Moreover, in order to obtain the efficiency of the search algorithm based on U, we performed a numerical analysis, using the following two-part procedure. Firstly, we fix a depth for the welded tree and plot the probability of finding the walker in the marked vertex, let us call it $|e\rangle$, in terms of θ. This probability is given by equation 29.

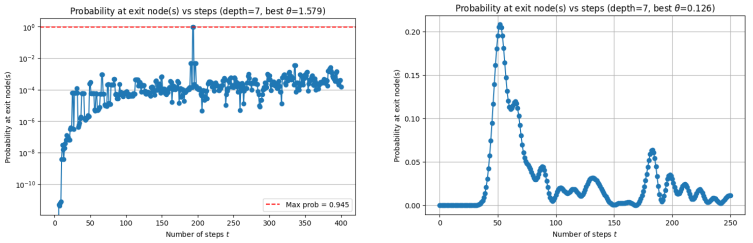
$$p(t,\theta) = |\langle e| U(\theta)^t |s\rangle|^2. \tag{29}$$

In order to define the probability of each θ, we fix a maximum number of steps for a fixed θ and select the highest value for $p(t,\theta)$ as in figure 7. Secondly, we identify what is the best θ ($\theta_{\max}$) and analyse $p(t,\theta_{\max})$ as a function of steps t as in figure 8.



(a) (b)

FIG. 7: Maximum probability in the marked vertex over 1000 steps as a function of θ for two different seeds of the random cycle.



(a) (b)

FIG. 8: Maximum probability in the marked vertex for the best θ as a function of the number of steps, for two different seeds of the random cycle.

There are a few remarks to make about the application of the SQW model to the welded tree problem. First, the random cycle can largely affect the tessellation cover of the graph and, consequently, the evolution of the states. As a result, the results for different seeds that generate the random cycle between trees vary significantly. This can be seen in figures 7 and 8. In figure 7a, the best result is for $\theta = \frac{\pi}{2}$, while in figure 7b, the best values of θ are those that are really close to 0 or π. As a consequence, using the best value of θ, we can have two very different evolutions of the probability of finding the marked vertex as we can observe in figures 8a and 8b.

This problem by itself would be quite interesting to further study. However, due to not being our main goal, we will proceed to our next problem.

B. Random Weighted Graph

In this section, we numerically analyse the extension suggested. Let $G(V,E,w)$ be a simple weighted graph. The procedure to create the evolution operator is the one described in Section II B. We start by applying the clique–insertion operator to our graph, creating G'. Then, we define the tessellation cover of G', that will be completely analogous to figure 4. Moreover, we compute the evolution operator. Note that the marked vertex will become a *marked clique*. As a consequence, when defining the R operator according to equation 26, the set M will be the marked clique.

In this case, the initial state will be:

$$|\psi_0\rangle = \frac{1}{\sqrt{|V|}} \sum_{v\in V} |v\rangle\,, \tag{30}$$

which is a uniform superposition of states. Due to M being a marked clique, equation 29 will change to

$$p(t,\theta) = \sum_{v\in M} |\langle v| U(\theta)^t |\psi_0\rangle|^2. \tag{31}$$

The graphs used in this analysis were generated first by randomly generating the points and their connections and then randomly assigning a value between 0 and 1 to each edge to assign a weight. Notice that equation 28 ensures that every state is normalised, whether the weights incident in each vertex sum to 1 or not. In Appendix E, an example of a randomly generated weighted graph and its transformation is shown in figure 17.

The following procedure will be similar to the previous one. We will start by computing the value of θ that maximizes the probability of finding the marked vertex. Then, once we find the θ_{max}, we will analyse the evolution of the system according to that angle for the evolution operator.

From figure 9, we note that the θ that maximizes probability is $\theta_{\max} = \frac{\pi}{2}$, which recovers 20. This is fairly interesting because the extension from equation 21 seems to work the best when there is some kind of symmetry as in Section III A or as in Ref. [22]. Additionally, in figure 10, we observe fairly high probabilities and the expected quasi-periodicity in the evolution of our initial state, as described in Appendix D.

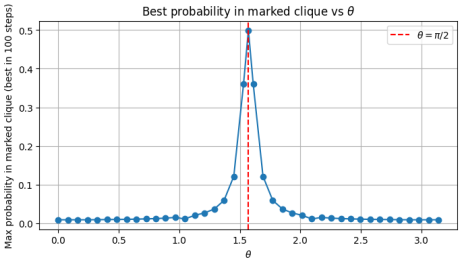


FIG. 9: Maximum probability in the marked vertex over 100 steps as a function of θ for a randomly generated weighted graph with $N = 100$.

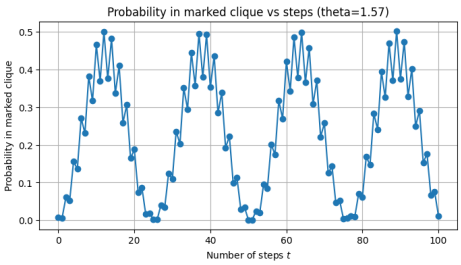


FIG. 10: Maximum probability in the marked vertex for the best θ as a function of the number of steps for the same graph from figure 9.

In order to compare this model with the classical counterpart, we analysed how the number of steps necessary to have a high probability of finding the marked vertex scaled compared to the size of our graph G.

We know that the number of operations for a classical random algorithm scales according to $O(N)$, whereas, for certain conditions, it has been proven, for example, in Ref. [9], that it is possible to achieve a quadratic speedup ($O(\sqrt{N})$). In figure 11, our model seems to also scale in the same way. Additionally, in Appendix F, figure 18 shows the probabilities for each N. As we can check, $p \approx 0.49$.

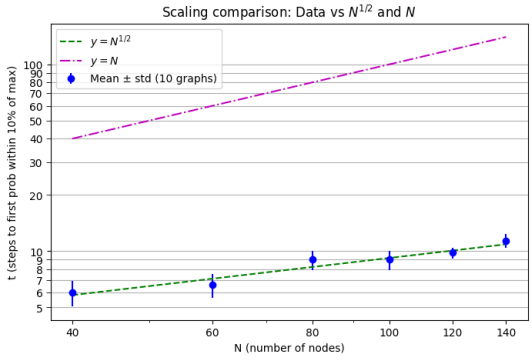


FIG. 11: Number of steps necessary to reach the first probability within 10% of its maximum in terms of the number of nodes in the graph, where the error bars are the standard deviation of the 10 tests for each N.

Nevertheless, for the problem of the marked vertex, the addition of weights is not of much interest as its consequence is to create a bias for certain paths. As a result, although the results seem promising at first-hand, they seem no different than testing for non-weighted graphs. Moreover, since weighted graphs make the most sense for problems such as *pathfinding*, we decided to dive into that in the next section.

C. Pathfinding Problem

1. *Problem*

The graph to be studied is shown in figure 12. The reason for studying this specific problem is due to Ref. [16] having already done an analysis of this graph for its approach. As a consequence, we thought it would be interesting to apply our own model.

We start by creating a graph G', which will have three welded trees inside and an s node (which will be the initial state) and a e node (which will be the marked vertex). Afterwards, we weld each graph by combining the node e_k with the node s_{k+1}, where k refers to the k-th graph. As a result, the final graph will have the blue vertex as its initial state and the red vertex as its marked vertex.

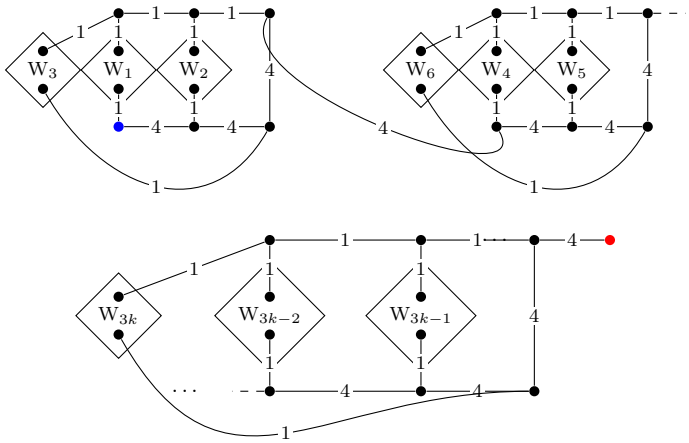


FIG. 12: Graph G, which is a result of welding n smaller weighted graphs G' with welded trees inside

The goal of this problem is to output the shortest path $e - t$. Naturally, this is a test problem, the solution of which is obviously the path that does not traverse any welded tree. We also have to point out that every edge inside the welded tree will have $w = 1$. Furthermore, it is also assumed that we have access to an oracle $\mathcal{O}$ that, in addition to evaluating if it is a marked vertex, if we are in v, it also gives access to $\Gamma(v)$, which can also be called the *adjacency list* in this case.

2. *Procedure*

In our analysis, we start by approaching this problem as a *marked vertex* problem. Then, we propose how we can use use it and adapt to a *pathfinding* problem. The procedure will be the following:

Step 1: For every edge outside a welded tree, apply the *clique-insertion operator*.

Step 2: Compute the tessellation cover of the graph. For that we will consider six tessellations. Two tessellations will be for the

edges and vertices outside of the welded trees. The process for this will be analogous to that described in Section II B. The other four tessellations will be used for the welded trees, as was described and studied in Section III A. Naturally, the two tessellations that belong "outside the welded trees" will only be 1-cliques inside each welded tree (the same thing will happen for the four other tessellations outside the welded trees). An example of such is shown in figure 13.

Step 3: Compute the evolution operator, which, once again, will be as described in the previous sections. For this problem, we will consider $\theta = \frac{\pi}{2}$. The evolution operator will be:

$$U = W_{\text{cyan}} W_{\text{brown}} W_{\text{orange}} W_{\text{green}} W_{\text{blue}} W_{\text{red}} R, \qquad (32)$$

where W_{colour} is defined according to equation 20 and R according to equation 26.

Step 4: Apply the evolution operator to our initial state.

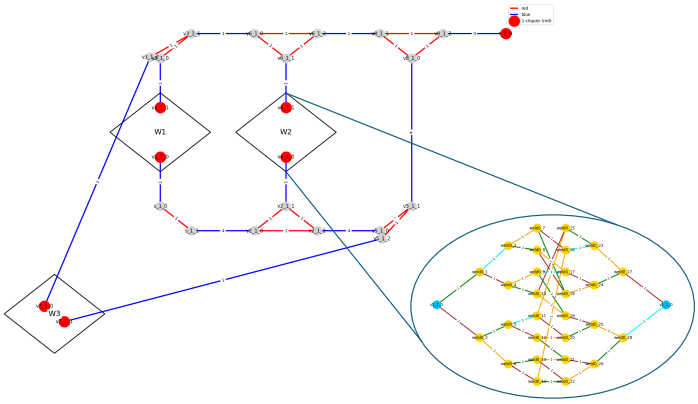


FIG. 13: Graph G' after the clique-insertion operator outside the welded trees with its tesselation cover

3. *Results and Pathfinding proposal*

Firstly, I will give an idea on how we can turn a *marked vertex* approach into a *pathfinding* solution. Using ref. [20], we can use the hypothesis that when choosing the edge that minimizes the path, that edge will *probably* be the one that maximizes the probability of finding the marked vertex in the least number of steps possible. As a result, we will analyse whether the shortest path for our problem satisfies this hypothesis. For that, we use the following procedure:

Step 1: First, we shall assume that we are on one vertex that is connected to the other two. We choose the *right path*, which in our graph from figures 12 and 13 corresponds to the vertex that is not a node from a welded tree.

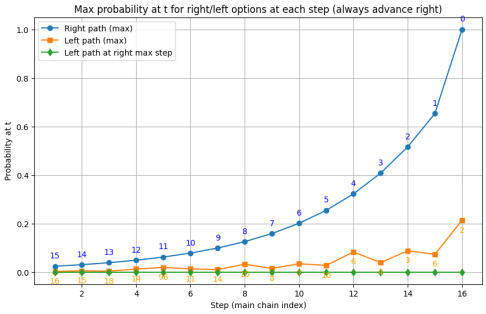
Step 2: Erase the edge that we just traversed and update our tessellations and evolution operator accordingly. Our initial state will be a uniform superposition of the clique that we moved to.

Step 3: Analyse the evolution of the initial state. We store the step at which the probability of finding the marked vertex is at its maximum and the probability itself.

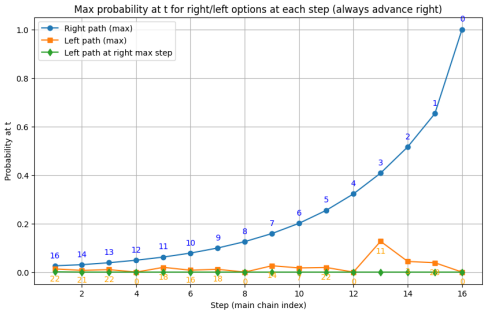
Step 4: Repeat Step 1, but we take the *left path* instead. In other words, we move to the vertex that belongs to a welded tree. Then, we repeat Step 2 for this case. Our initial state will now be just the first vertex of the welded tree.

Step 5: Analyse the evolution of the initial state. In this case, we will store two things: the step at which the probability of finding the marked vertex is at its maximum and the probability itself; the probability of finding the marked vertex for the best step from the right path previously stored.

For reference, in Appendix G, there is an example of the evolution of the graph for each cycle and each path. In addition, there are also examples of plots of the probabilities in terms of steps for different cycles and paths taken.



(a)



(b)

FIG. 14: Evolution of the probability of finding a marked vertex in terms of the cycle in our path. The values at each point are those of the best step for each path. The blue curve is the probability of finding the marked vertex for the best step in the right path; the orange curve is the same probability for the best step in the left path; the green curve is the probability of finding the marked vertex in the left path at the best step from the right path. (a) Graph with 4 welded graphs and welded trees of depth 6; (b) Graph with 4 welded graphs and no welded trees.

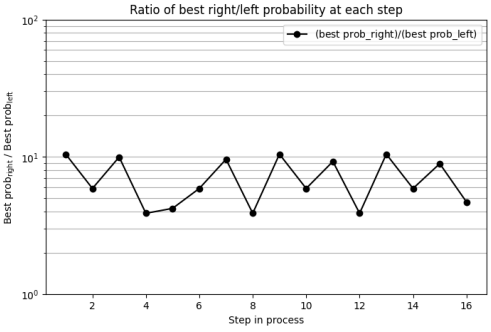


FIG. 15: Ratio $\frac{p_{\text{right}}(t_{\text{right}})}{p_{\text{left}}(t_{\text{left}})}$ in terms of the step of the process for G with 4 welded graphs and welded trees with depth 6.

Let us denote t_{right} and t_{left} as the best step for the right and left path, respectively. As we can observe in figure 14a, the probabilities of finding the marked vertex for the right path are not only superior, but also have a smaller value for the step that reaches the highest probability. In other words, we have $p_{\text{right}}(t_{\text{right}}) > p_{\text{left}}(t_{\text{left}})$ and $t_{\text{right}} < t_{\text{left}}$. Moreover, we can also observe in figure 15 that $p_{\text{right}}(t_{\text{right}})$ is usually one order of magnitude higher than $p_{\text{left}}(t_{\text{left}})$. Additionally, when we evaluate the probability of finding the marked vertex in the left path for t_{right}, we have that $p_{\text{left}}(t_{\text{right}}) \approx 0$.

In addition to the analysis of the problem for the graph in figure 12, we decide to test the same graph but without welded trees. The reason for this is due to the knowledge that welded trees are purposely difficult to solve in comparison to more general graphs, as verified in the previous two subsections. As a result, the right path would, expectedly, always perform better. The results for the exact same graph without the welded trees can be found in figure 14b. As we can see, there is still an obvious bias towards the path with the highest weights (which can be thought as the path of less resistance as in Section I B 3). As a result, the results are consistent with our hypothesis.

Algorithm: Quantum algorithm to find an $e-t$ path in the graph G

Input: Graph $G=(V,E)$, $x,y\in V$ and $i=1$.
Output: an x-y path

1. Given the name of the vertex x, the adjacency list oracle $\mathcal{O}$ returns the names of the two neighbours of x, that is u_1,u_2. Without loss of generality, choose one of the two neighbours u_1 and define its clique as the initial state $|\alpha_1\rangle$.

2. Let U' be the modified evolution operator by removing the previous edge that was traversed. Run the DTQW $(U')^t|\alpha_1\rangle$ for a certain time $\tau(d,n)$. Measure the resulting state on the computational basis and get the name of an outcome vertex v. Compute if it is a marked vertex.

3. Repeat Step 2 $q(d,n)$ times or until the measured vertex is equal to e.

4. If the measured vertex v is equal to t, then collect the edge (x,u_1) as an x-y path edge and let $x=u_1$. Otherwise, collect the edge (x,u_2) as an x-y path edge and let $x=u_2$.

5. Update the graph G by deleting the selected edge. Repeat all the above steps until $x=y$, then output all the selected edges as an x-y path.

Based on our results, we propose the aforementioned algorithm. However, more work is still needed to determine the values of the *running time* $\tau(d,n)$ and the *number of queries* $q(d,n)$, where, for this problem, d is the depth of each welded tree and n the number of welded graphs.

IV. CONCLUSIONS

In this report, I briefly introduced some discrete-time Quantum Walk models that were relevant to the Staggered Quantum Walk framework. Afterwards, we proposed our original contribution on how we could extend this model so that we could tackle problems for weighted graphs.

In order to study the last problem, a preliminary study was done on the application of the SQW model to one of the most popular problems in graph theory, the welded tree. As our results showed, the behaviour of our model was quite erratic, being largely dependent on the random cycle that connected the two binary trees. As a consequence, despite the results being fairly satisfactory, due to randomness, we can not argue some speedup compared to the classical case. Moreover, to test our extension to a normal problem of a marked vertex, we generated random weighted graphs and applied our model. The results seem to agree with the quadratic speedup, as the number of operations needed to have a high probability of finding a marked vertex seemed to scale according to $O(\sqrt{N})$, where N is the number of nodes.

Moreover, we also tested an hypothesis that could be beneficial to a pathfinding proposal, which was that the edges from the best path between $s-e$ were the ones that presented the lowest optimal number of steps and the highest probability of finding the marked vertex e. After verifying that hypothesis, we proposed an algorithm to solve this pathfinding problem.

Naturally, this last part will be the future directions of our work. Our goal is now to perform a formal and analytical analysis to determine $\tau(d,n)$ and $q(d,n)$. Afterwards, we also need to do a complexity analysis so we could compare our model to not only the classical counterpart, but also other discrete-time Quantum Walk and even continuous-time Quantum Walk models.

Overall, our proposal may represent a step further toward generalizing the power of quantum walks into a more complex class of graphs and new types of problems. As a result, we will be able to tackle new challenges, which is an extremely valuable addition for real-world applications.

REFERENCES

[1] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Journal on Computing **26**, 1484–1509 (1997).

[2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2000).

[3] L. K. Grover, A fast quantum mechanical algorithm for database search (1996), arXiv:quant-ph/9605043 [quant-ph].

[4] G. Almeida, R. Santos, L. Janiurek, and Y. Omar, Quantum walks on arbitrary spatial networks with Rydberg atoms (2025).

[5] R. Portugal, *Quantum Walks and Search Algorithms*, 2nd ed., Quantum Science and Technology (Springer, 2018).

[6] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, Exponential algorithmic speedup by a quantum walk, in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, STOC03 (ACM, 2003) p. 59–68.

[7] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, and J. Watrous, Onedimensional quantum walks, in *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC '01)* (ACM, 2001).

[8] A. Nayak and A. Vishwanath, Quantum walk on the line (2000), arXiv:quant-ph/0010117 [quant-ph].

[9] M. Szegedy, Quantum speed-up of markov chain based algorithms, in *45th Annual IEEE symposium on foundations of computer science* (IEEE, 2004) pp. 32–41.

[10] R. Portugal, Establishing the equivalence between Szegedy's and coined quantum walks using the staggered model, Quantum Information Processing **15**, 1387–1409 (2016).

[11] A. Belovs, Quantum walks and electric networks (2013), arXiv:1302.3143 [quant-ph].

[12] A. Belovs, Global phase helps in quantum search: Yet another look at the welded tree problem (2024), arXiv:2404.19476 [quant-ph].

[13] S. Piddock, Quantum walk search algorithms and effective resistance (2019), arXiv:1912.04196 [quant-ph].

[14] S. Apers and S. Piddock, Elfs, trees and quantum walks (2025), arXiv:2211.16379 [quant-ph].

[15] S. Jeffery and S. Zur, Multidimensional quantum walks, with application to k-distinctness, TheoretiCS **Volume 4**, 10.46298/theoretics.25.7 (2025).

[16] J. Li and S. Zur, Multidimensional electrical networks and their application to exponential speedups for graph problems, Quantum **9**, 1733 (2025).

[17] R. Portugal, R. A. M. Santos, T. D. Fernandes, and D. N. Gonçalves, The staggered quantum walk model, Quantum Information Processing **15**, 85–101 (2015).

[18] R. Portugal, Staggered quantum walks on graphs, Physical Review A **93**, 10.1103/physreva.93.062335 (2016).

[19] R. Portugal, M. C. de Oliveira, and J. K. Moqadam, Staggered quantum walks with Hamiltonians, Physical Review A **95**, 10.1103/physreva.95.012328 (2017).

[20] J. Li, Exponential speedup of quantum algorithms for the pathfinding problem (2024), arXiv:2307.12492 [quant-ph].

[21] B. Bollobás, *Modern Graph Theory*, 1st ed., Graduate Texts in Mathematics, Vol. 184 (Springer, New York, NY, 1998).

[22] R. Portugal and T. D. Fernandes, Quantum search on the two-dimensional lattice using the staggered model with Hamiltonians, Physical Review A **95**, 10.1103/physreva.95.042341 (2017)

Appendix A: Graph Theory

When studying quantum walks, many problems can be naturally formulated using graph theory. Therefore, it is essential to introduce some fundamental concepts that will aid in understanding the various quantum walk models.

1. Basic definitions

A *simple graph* $G(V, E)$ is defined by a set $V(G)$ of vertices or nodes and a set $E(G)$ of edges so that each edge links two vertices and two vertices are linked by at most one edge [21]. Two vertices linked by an edge are called *adjacent* or *neighbours*. Two edges that share a common vertex are also called *adjacent*. A *loop* is an edge whose endpoints are equal. *Multiple edges* are edges having the same pair of endpoints. A simple graph has neither multiple edges nor loops, such as the one in figure 16.

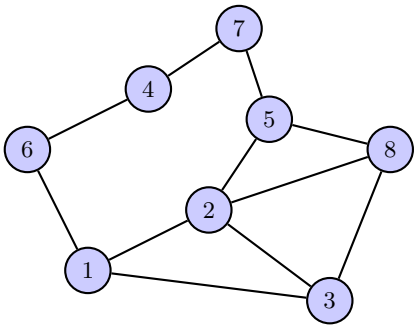


FIG. 16: Simple Graph

In simple graphs, the edges can be named by the endpoints like an unordered set $\{v, v'\}$, where v and v' are vertices. The degree of vertex v is the number of edges incident to the vertex and is denoted by $d(v)$. A graph is *d-regular* if all vertices have degree d, that is, each vertex has exactly d neighbours.

A subgraph $G'(V', E')$, where $V' \subset V$ and $E' \subset E$, is an *induced subgraph* of $G(V, E)$ if it has exactly the edges that appear in G over the same vertex set. If two vertices are adjacent in G, they are also adjacent in the induced subgraph.

A graph is *connected* when there is a path between every pair of vertices. Otherwise, it is called *disconnected*.

The *complete graph* is a simple graph in which every pair of distinct vertices is connected by an edge.

A bipartite graph is a graph whose vertex set V is the union of two disjoint sets X and X' so that no two vertices in X are adjacent and no two vertices in X' are adjacent.

A *clique* is a subset of vertices of a graph such that its induced subgraph is complete. A *maximal clique* is a clique of maximum possible size. A clique of size d is called a *d-clique*. A clique graph $K(G)$ of a graph G is a graph such that every vertex represents a maximal clique of G and two vertices of $K(G)$ are adjacent if and only if the corresponding maximal cliques in G share at least one vertex in common.

Another useful tool for the following sections is the *clique–insertion operator* This operator replaces each vertex v of a graph G by a maximal $d(v)$-clique, creating a new graph $CI(G)$.

2. Weighted Graphs and Electrical Networks

A *Weighted Graph* is a connected graph $G = (V, E, w)$ with a vertex set V, an (undirected) edge set E and some weight function $w : E \to \mathbb{R}_{>0}$ [16]. A *directed graph* is a graph whose edges have a direction associated with them.

A weighted graph is sometimes called a *network* [16]. Since edges are undirected, we can equivalently describe the edges by some set $\vec{E}$ such that all $(v, u) \in \vec{E}$, exactly one of (u, v) or (v, u) is in $\vec{E}$. The choice of edge directions is arbitrary. Then we can view the weights as a function $w : \vec{E} \to \mathbb{R}_{>0}$, and for all $(u, v) \in \vec{E}$, define $w_{u,v} = w_{v,u}$. For convenience, we will define $w_{u,v} = 0$ for every pair of vertices such that $(u, v) \notin E$. For an implicit network G, and $u \in V$, we will let $\Gamma(u)$ denote the neighbourhood of u:

$$\Gamma(u) := \{v \in V : (u, v) \in E\} \tag{A1}$$

We use the following notation for the out- and in-neighbourhoods of $u \in V$:

$$\Gamma(u)^+ := \{v \in V : (u, v) \in \vec{E}\} \tag{A2a}$$
$$\Gamma(u)^- := \{v \in V : (v, u) \in \vec{E}\} \tag{A2b}$$

The following definitions are quite useful and important when analysing weighted graphs as electrical networks. Additionally, it is also essential for the Electrical Networks QW model.

Definition A.1: (Flow, Circulation)[16]. A *flow* on a network $G = (V, E, w)$ is a real-valued function $\theta : \vec{E} \to \mathbb{R}$, extended to edges in both directions by $\theta_{u,v} = \theta_{v,u}$ for all $(u, v) \in \vec{E}$. For any flow θ on G, vertex $u \in V$, and subset $A \subseteq V$ we define $\theta_u = \sum_{v \in \Gamma(u)} \theta_{u,v}$ as the flow coming out of u. If $\theta_u = 0$, we say flow is *conserved* at u. If flow is conserved at every vertex, we call θ a *circulation*. If $\theta_u > 0$, we call u a *source*, and if $\theta_u < 0$ we call u a *sink*. A flow with a unique source s and unique sink t (satisfying $\theta_s = -\theta_t = -1$) is called an *(unit) $s - t$ flow*. The energy of any flow θ is:

$$\mathcal{E}(\theta) := \sum_{(u,v) \in \vec{E}} \frac{\theta_{u,v}^2}{w_{u,v}} \tag{A3}$$

The *effective resistance* $\mathcal{R}_{s,t}$ is given by the minimal energy $\mathcal{E}(\theta)$ over all unit flows θ from s to t. The $s - t$ electrical flow is the unique unit $s-t$ flow that achieves this minimal energy.

Definition A.2: (Electrical Network)[16]. Given a network $G = (V, E, w)$ with a weight function w, we can interpret every edge $(u, v) \in E$ as a resistor with resistance $\frac{1}{w_{u,v}}$. This allows G to be modelled as an electrical network.

Definition A.3: (Kirchhoff's Law)[16]. For any $s - t$ flow on an electrical network $G = (V, E, w)$ with $s, t \in V$, the amount of electrical flow that enters any $u \in V \backslash \{s, t\}$is equal to the amount of flow that exits u, that is $\sum_{v \in \Gamma(u)} \theta_{u,v} = 0$.

Definition A.4: (Ohm's Law)[16]. Let θ be the $s - t$ electrical flow on an electrical network $G = (V, E, w)$ with $s, t \in V$. Then there exists a potential vector p such that the potential difference between the two end points of any edge $(u, v) \in E$ is equal to the amount of electrical flow $\theta_{u,v}$ along this edge multiplied with the resistance $\frac{1}{w_{u,v}}$, that is

$$p_u - p_v = \frac{\theta_{u,v}}{w_{u,v}} \tag{A4}$$

Appendix B: Coined Quantum Walk on more general graphs

Naturally, the CQW can be applied to more general graphs. Let $G(V,E)$ be a finite d-regular graph with $N = |V|$ vertices. The labels of the vertices are 0 to $N-1$, and the labels of the edges are 0 to $d-1$. The Hilbert space for G will be $\mathcal{H} = \mathcal{H}^d \otimes \mathcal{H}^N$. As a result, we can define the computational basis of $\mathcal{H}$ as the set of vectors $\{|a,v\rangle : 0 \le a \le d-1,\ 0 \le n \le N-1\}$. The evolution operator will be:

$$U = S(C \otimes I_N) \tag{B1}$$

where C is a general *coin operator* and S is the *flip-flop shift operator* ($S^2 = I_{dN}$) and is defined by:

$$S\,|a,v\rangle = |a,v'\rangle \tag{B2}$$

where v and v' are adjacent and incident to edge a, that is the, the label of $\{v,v'\}$ is a.

Moreover, let $G(V,E)$ be an arbitrary simple graph with vertex set V and edge set E and let $N = |V|$ be the number of vertices. An edge is denoted by an unordered set $\{v,v'\}$, where v and v' are adjacent. An arc is denoted by an ordered pair (v,v'), where v is the *tail* and v' is the *head*. Let $\vec{G}(V,A)$ be a directed graph such that (v,v') and (v',v) are in $A(G)$ if and only if $\{v,v\} \in G$. $\vec{G}$ and G have the same vertex set, and G is a *symmetric digraph*, whose underlying graph is G.

Appendix C: Evolution operator and Application of the Electrical Networks Quantum Walk

Let us consider the following two subspaces of $\mathcal{H}$. Let

$$\mathcal{A} = \text{span}\{|\psi\rangle \in \mathcal{H} : \langle u,v|\psi\rangle = -\langle v,u|\psi\rangle\,, \forall\,|u,v\rangle \in \mathcal{H}\}$$

be the *antisymmetric subspace* of $\mathcal{H}$. Moreover, $\mathcal{B} = \text{span}\{|\psi_u\rangle \in \mathcal{H} : u \in V\backslash\{s,t\}\}$ is the star space of $\mathcal{H}$. The quantum walk operator $U_{\mathcal{AB}}$ is defined as

$$U_{\mathcal{AB}} = (2\Pi_{\mathcal{A}} - I)(2\Pi_{\mathcal{B}} - I) \tag{C1}$$

We also need to define the state

$$|\psi_u^+\rangle := \sqrt{2}(I - \Pi_{\mathcal{A}})\,|\psi_u\rangle = \frac{I + \text{SWAP}}{\sqrt{2}}\,|\psi_u\rangle \tag{C2}$$

where SWAP acts as $\text{SWAP}\,|u,v\rangle = |v,u\rangle$.

Then according to Ref. [13, 14, 16], by performing *phase-estimation* on the initial state $|\psi_s^+\rangle$ with the operator $U_{\mathcal{AB}}$ and precision $O\left(\frac{\varepsilon^2}{\sqrt{\mathcal{R}_{s,t} w_s ||p||}}\right)$, the phase-estimation algorithm outputs "0" with probability $\Theta\left(\frac{1}{\sqrt{\mathcal{R}_{s,t} w_s}}\right)$, leaving a state $|\hat{\theta}\rangle$ satisfying

$$\frac{1}{2}\Big\|\,|\hat{\theta}\rangle\langle\hat{\theta}| - |\theta\rangle\langle\theta|\,\Big\|_1 \le \varepsilon,$$

where

$$\|X\|_1 = \text{tr}\,|X| \equiv \text{tr}\,\sqrt{X^\dagger X} \equiv \sum_i \sqrt{\lambda_i(X^\dagger X)}\ .$$

In terms of application, basically, we can estimate the state $|\theta\rangle$. Then, we do the measurement, which will output an edge. We can use the oracle, which will be explained in section II A 2, to infer if we have found the marked vertex. As we can check, this part is fundamentally different from the CQW, SzQW and SQW evolution.

Appendix D: Quasi-periodicity and Dynamics [5]

In this section, we address some aspects of the evolution of out quantum walk. Firstly, let $|\psi(0)\rangle$ be the initial state and $|\psi(t)\rangle$ will evolve according to equation 1:

$$|\psi(t)\rangle = U^t\,|\psi(0)\rangle\,. \tag{D1}$$

One very important observation is that for DTQW, $|\psi(t)\rangle$ does not tend to some stationary state. In other words, $\lim_{t\to\infty}|\psi(t)\rangle$ does not exist. This limit does not exist because the norm $\Big\|\,|\psi(t+1)\rangle - |\psi(t)\rangle\,\Big\|$ is constant for all t, in fact,

$$\frac{1}{2}\Big\|\,|\psi(t+1)\rangle - |\psi(t)\rangle\,\Big\|^2 = \frac{1}{2}\Big\|\,U^t(U-I)\,|\psi(0)\rangle\,\Big\|^2 = \\ = 1 - \text{Re}\{\langle\psi(0)|\,U\,|\psi(0)\rangle\}$$

The result is time-independent because operator U is unitary. Since the real part of $\langle\psi(0)|\,U\,|\psi(0)\rangle$ is constant and strictly smaller than 1 for a given non-trivial evolution operator U, the above norm is a non-zero constant. The time-dependent state $|\psi(t)\rangle$ cannot tend toward a stationary state because the left-hand side would approach zero, which is a contradiction.

Quasi-periodic dynamic in the quantum walk literature is used with the meaning that there are an infinite number of time steps such that the quantum state is close to the initial state. Besides, the repetitive structure is over varying timescales. Since this concept is an extension of the periodic behaviour, let us start by defining the latter.

Definition D.1 (Periodic dynamics): The dynamic based on the repeated action of an evolution operator is *periodic* if there is a fundamental period $t_0 \in \mathbb{Z}^+$ and an angle α such that $U^{t_0} = e^{i\alpha} I$.

It follows from this definition that $|\langle\psi(nt_0)|\psi(0)\rangle|^2 = 1$ for all positive integers n and for any choice of the initial state $|\psi(0)\rangle$.

Definition D.2 (Quasi-periodic dynamics): The dynamic based on the repeated action of an evolution operator is *quasi-periodic* if for any positive number ε there is a time step t such that $\|U^t - I\| \le \varepsilon$.

Using the norm of linear operators as follows

$$A = \max_{\|\psi\|=1} \langle\psi|A|\psi\rangle \tag{D2}$$

this definition implies that, for any fixed positive ε, there is a time step t such that $|\langle\psi(t)|\psi(0)\rangle| \ge 1-\varepsilon$.

Definition D.2 also implies that there are an infinite number of time steps such that $|\langle\psi(t)|\psi(0)\rangle| \ge 1-\varepsilon$.

In fact, if there is one such t, then choose a new ε, for instance,

$$\varepsilon' = \frac{1 - |\langle\psi(t)|\psi(0)\rangle|^2}{2}.$$

By Definition D.2, there is a time step $t' \neq t$ such that $|\langle\psi(t')|\psi(0)\rangle| \ge 1-\varepsilon'$. This process can be repeated over and over. If t is the smallest time step such that $|\langle\psi(t)|\psi(0)\rangle| \ge 1-\varepsilon$, then $t' > t$.

One such example of quasi-periodicity is the figure 10.

Appendix E: Auxiliary example from Section III B

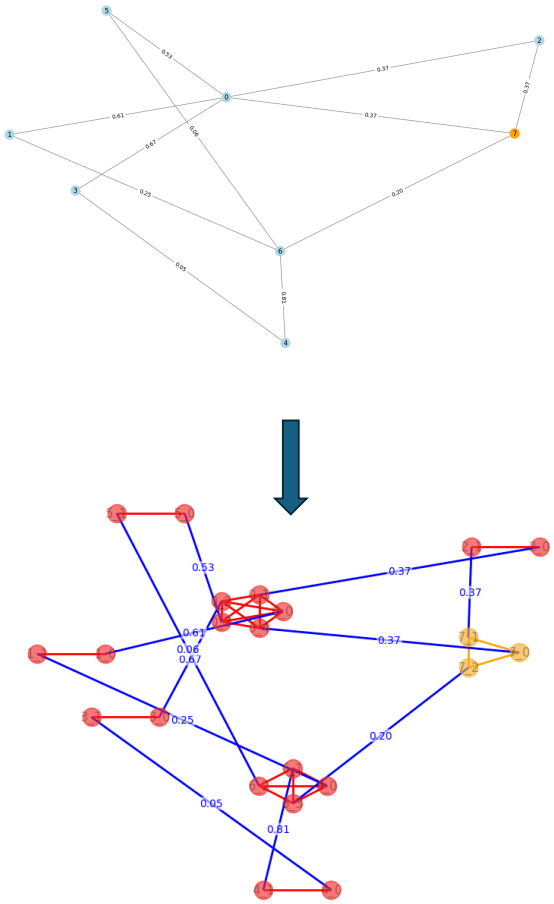


FIG. 17: Randomly generated weighted graph with 8 nodes and the application of the clique-insertion operator along with its tessellation cover

Appendix F: Auxiliary results from Section III B

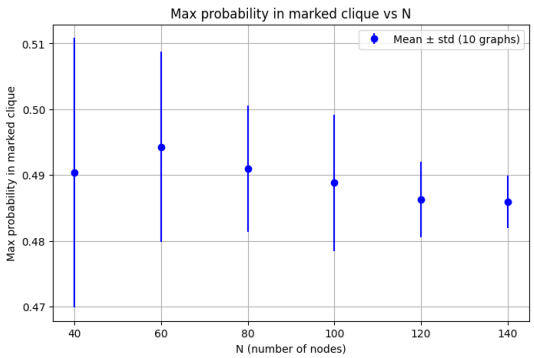
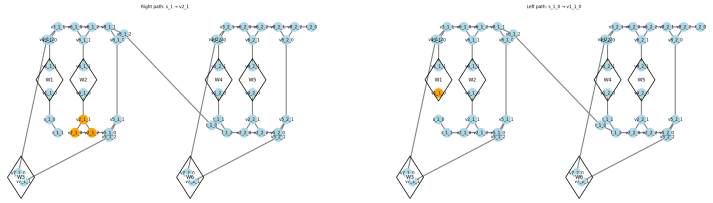


FIG. 18: Probability of finding the marked vertices for the best step and best θ in terms of the number of nodes N of the weighted graph.

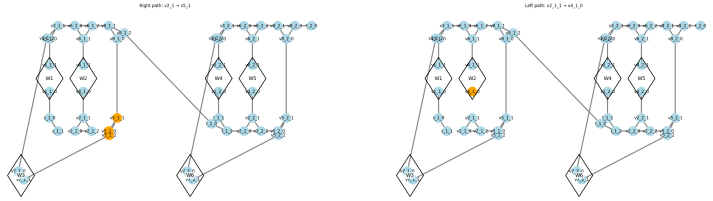
Appendix G: Auxiliary results from Section III C 3

In order to visualize the process described in the procedure from section III C 3, we present the following figures for the simple case of a welded graph with $n = 2$. The orange vertices represent the initial state. The marked vertex, in this case, is 't_2_0'.



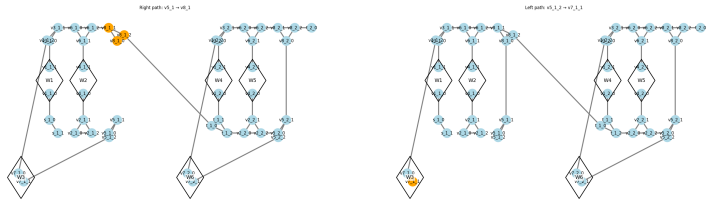
(a) (b)

FIG. 19: Step 1: (a) Right path and (b) Left path



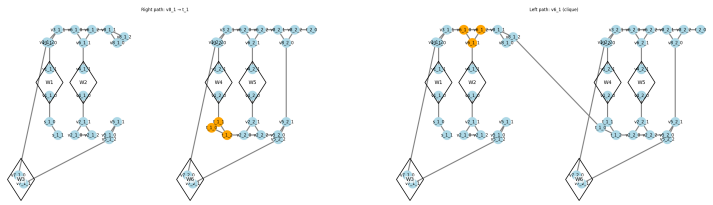
(a) (b)

FIG. 20: Step 2: (a) Right path and (b) Left path



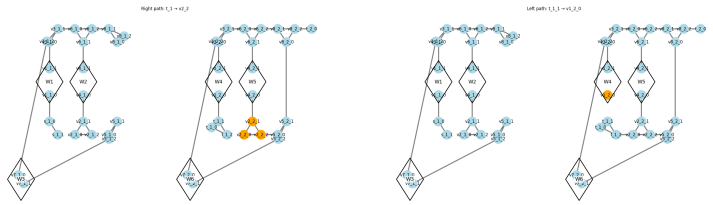
(a) (b)

FIG. 21: Step 3: (a) Right path and (b) Left path



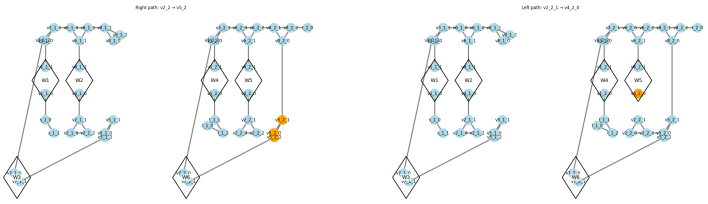
(a) (b)

FIG. 22: Step 4: (a) Right path and (b) Left path



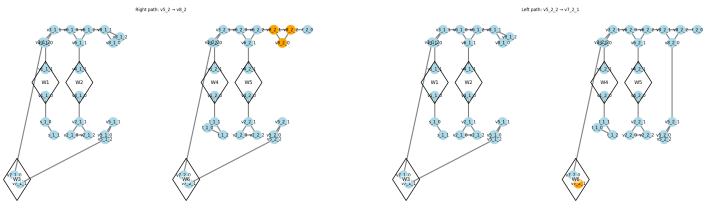
(a) (b)

FIG. 23: Step 5: (a) Right path and (b) Left path



(a) (b)

FIG. 24: Step 6: (a) Right path and (b) Left path



(a) (b)

FIG. 25: Step 7: (a) Right path and (b) Left path

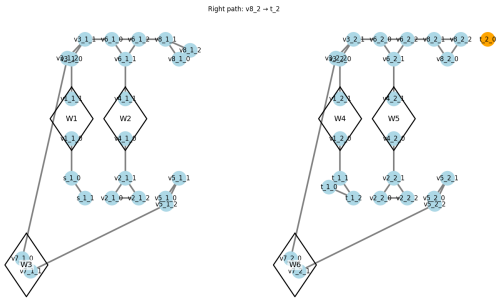
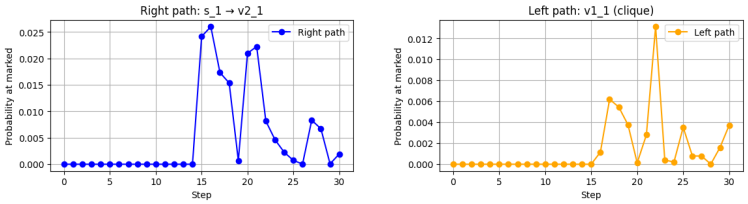


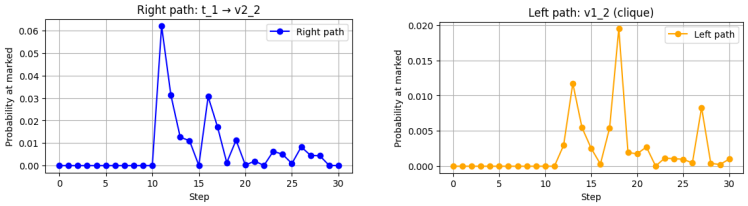
FIG. 26: Step 8

In addition to the plot of the graphs, here it is a few examples of the plots for the probabilities of finding the marked vertex in function of the steps. This is the case for 4 welded graphs and each welded tree with depth 0. This figure are results from the graph relative to figure 14b.



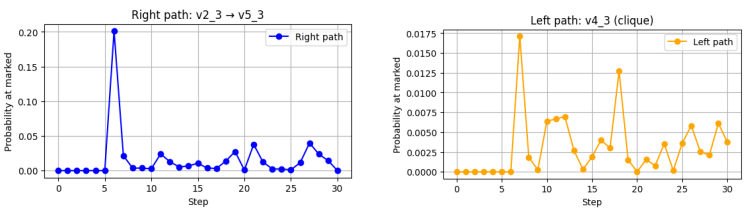
(a) (b)

FIG. 27: Step 1: (a) Right path and (b) Left path



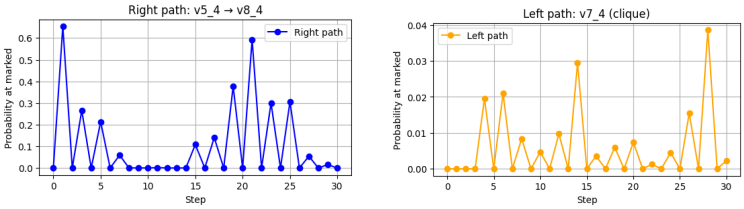
(a) (b)

FIG. 28: Step 5: (a) Right path and (b) Left path



(a) (b)

FIG. 29: Step 10: (a) Right path and (b) Left path



(a) (b)

FIG. 30: Step 15: (a) Right path and (b) Left path

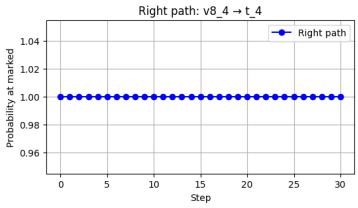


FIG. 31: Step 16: Right path