UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO DE ESTATÍSTICA E INFORMÁTICA
PROGRAMA DE PÓS–GRADUAÇÃO EM BIOMETRIA E ESTATÍSTICA APLICADA

LUCIANO SERAFIM DE SOUZA

LACKADAISICAL QUANTUM WALK ANALYSES WITH PARTIAL PHASE INVERSION PROPOSAL

RECIFE – PE
2024

LUCIANO SERAFIM DE SOUZA

LACKADAISICAL QUANTUM WALK ANALYSES WITH PARTIAL PHASE INVERSION PROPOSAL

A thesis submitted to the Coordination of the Graduate Program in Biometrics and Applied Statistics of the Department of Statistics and Informatics - DEINFO - Federal Rural University of Pernambuco, as part of the requirements for obtaining a Ph.D.

ADVISOR: Prof. Dr. Tiago A. E. Ferreira

RECIFE – PE

2024

Dados Internacionais de Catalogação na Publicação
Sistema Integrado de Bibliotecas da UFRPE
Bibliotecário(a): Suely Manzi – CRB-4 809

S719l Souza, Luciano Serafim de.
Lackadaisical quantum walk analyses with Partial Phase Inversion Proposal / Luciano Serafim de Souza. – Recife, 2024.
114 f.; il.

Orientador(a): Tiago Alessandro Espinola Ferreira.

Tese (Doutorado) – Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Biometria e Estatística Aplicada, Recife, BR-PE, 2024.

Inclui referências e anexo(s).

1. Computação quântica. 2. Redes Neurais (Computação) . 3. Inteligência artificial. 4. Algoritmos computacionais 5. Algoritmos e estruturas de dados . I. Ferreira, Tiago Alessandro Espinola, orient. II. Título

CDD 519.5

LUCIANO SERAFIM DE SOUZA

LACKADAISICAL QUANTUM WALK ANALYSES WITH PARTIAL PHASE INVERSION PROPOSAL

A thesis submitted to the Coordination of the Graduate Program in Biometrics and Applied Statistics of the Department of Statistics and Informatics - DEINFO - Federal Rural University of Pernambuco, as part of the requirements for obtaining a Ph.D.

Approved on: August 27, 2024.

EXAMINATION BOARD

Prof. Dr. Tiago A. E. Ferreira (Advisor)
Universidade Federal Rual de Pernambuco - UFRPE
Departamento de Informática

Wilson Rosa de Oliveira Junior
Universidade Federal Rural de Pernambuco - UFRPE
Departamento de Informática

Franklin de Lima Marquezino
Universidade Federal do Rio de Janeiro - UFRJ
Laboratório Nacional de Computação Científica

Adenilton José da Silva
Universidade Federal de Pernambuco - UFPE
Centro de Informática

José Ferraz de Moura Nunes Filho
Universidade Federal Rural de Pernambuco - UFRPE
Departamento de Física

Aos meus filhos, esposa, pais e irmãos.

Agradecimentos

Agradeço imensamente a Deus pela criação e por cuidar de minha família durante todo este tempo que estive ausente, por ser nas horas de desânimo e dificuldade a minha esperança e fortaleza. Agradeço aos meus pais, Josefa (***In memoriam***) e Mario (***In memoriam***) e minha avó materna, Lurdes (***In memoriam***), por todo amor e apoio que foi dado a mim em vida. Aos meus irmãos Alexandre (***In memoriam***), Mário e Jônatas e irmãs Alexandra, Ana Paula e Luciana e os seus familiares por fazerem parte da minha vida.

Agradeço com todo meu amor à minha esposa, Maria, por estar sempre ao meu lado, dando forças e segurança. Por ter cuidado de nossos filhos enquanto estive longe para que eu pudesse buscar a realização de mais este sonho. Estas poucas palavras são incapazes de expressar o tamanho e a dimensão de toda gratidão que tenho a você. Agradeço aos meus filhos Ana e Lucas que abdicaram da minha presença, sendo bons filhos, compreensivos e amorosos e vendo em mim além do que sou e mereço. Esta conquista não é apenas minha mas de todos vocês.

Agradeço ao Professor Dr. Tiago A. E. Ferreira, pela orientação. Pela compreensão, paciência, amizade, por toda ajuda e respeito. Por ser um exemplo. Por ter contribuído para que eu me tornasse uma pessoa melhor do que era antes. Ao meu amigo Jonathan pelo companheirismo e por toda contribuição. Ao Professor Dr. Henrique C. T. Santos pela sua amizade e contribuição.

Agradeço a Universidade Federal Rural de Pernambuco, ao Programa de Pós Graduação em Biometria e Estatística Aplicada e a todos da coordenação, do corpo docente, técnicos e corpo discente.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Com a sabedoria se edifica a casa, e com a inteligência ela se estabelece.

(Provérbios 24:3)

Abstract

Quantum walks have been a theme of great interest in quantum computing. The development of quantum search algorithms represents the main application of these quantum walks. Based on quantum walks, these search algorithms have shown promising results, providing efficient solutions to complex problems. However, some theoretical and practical challenges still need to be overcome. This thesis comprises four scientific articles that use quantum walks, defined in a discrete-time space, as search algorithms. Initially, we applied a quantum walk on a complete graph to develop a quantum search procedure capable of finding a set of synaptic weights that train a classical artificial neural network. Some advantages of using quantum walk as a search procedure for a set of synaptic weights can be highlighted: avoiding stagnation in local minima, knowing in advance the number of iterations required to obtain a solution, and guaranteeing a valid solution by the end of the procedure. However, to apply the quantum walk on the complete graph to search for synaptic weights, it is necessary to map its vertices onto an n-dimensional grid to define their locations. This is because, by definition, in a quantum walk on a complete graph, the vertices are indistinguishable, incurring an additional cost. To reduce this cost, we decided to analyze the lackadaisical quantum walk on the hypercube, as it can be reduced to a walk on a line, thereby decreasing its complexity. Since the goal is to search for multiple weights and because the lackadaisical quantum walk depends on a weight value for the self-loop, we proposed an ideal weight value for searching multiple vertices, achieving maximum success probabilities close to 1. Furthermore, the results show that if the vertices constituting the solution are also adjacent, the maximum success probability is compromised. Observing this behavior regarding the adjacency of solutions, we investigated the use of multiple self-loops at each vertex of the hypercube and proposed a new search algorithm based on the lackadaisical quantum walk with partial phase inversion. Initially, we applied this proposal to search for non-adjacent vertices. With this new approach, we also proposed two new weights, achieving maximum success probabilities close to 1. Subsequently, we applied the algorithm in two scenarios: a search for adjacent and non-adjacent vertices, and a search exclusively for adjacent vertices. We achieved maximum success probabilities close to 1 even in cases where the solution vertices were adjacent.

Keywords: Quantum Computing. Artificial Neural Networks Training. Quantum Walk.

Quantum Search Algorithm. Lackadaisical Quantum Walk. Multiple Self-loops. Partial Phase Inversion. Adjacent Marked Vertices

Resumo

As caminhadas quânticas têm sido um tema de muito interesse na computação quântica. O desenvolvimento de algoritmos quânticos de busca representa a principal aplicação dessas caminhadas quânticas. Com base em caminhadas quânticas, esses algoritmos de busca têm mostrado resultados promissores, fornecendo soluções eficientes para problemas complexos. Contudo, alguns desafios teóricos e práticos ainda precisam ser superados. Esta tese compreende quatro artigos científicos que usam caminhadas quânticas, definidas em um espaço de tempo discreto, como algoritmos de busca. Inicialmente, aplicamos uma caminhada quântica em um grafo completo para desenvolver um procedimento quântico de busca capaz de encontrar um conjunto de pesos sinápticos que treinam uma rede neural artificial clássica. Algumas vantagens da utilização das caminhadas quânticas como procedimento de busca para um conjunto de pesos sinápticos podem ser destacadas: evitar estagnação em mínimos locais, o conhecimento prévio do número de iterações necessárias para obter uma solução, e garantir uma solução válida ao final do procedimento. No entanto, para aplicar a caminhada quântica em um grafo completo com o objetivo de buscar pesos sinápticos, é necessário mapear seus vértices em uma grade n-dimensional para definir suas localizações. Isso ocorre porque, por definição, em uma caminhada quântica no grafo completo, os vértices são indistinguíveis, acarretando um custo adicional. Para reduzir este custo, decidimos analisar a caminhada quântica lackadaisical no hipercubo, pois ela pode ser reduzida a uma caminhada em uma reta, diminuindo assim sua complexidade. Como o objetivo é a busca por múltiplos pesos e, devido à dependência da caminhada quântica lackadaisical de um valor de peso para o self-loop, propusemos um valor de peso ideal para buscar múltiplos vértices, alcançando probabilidades máximas de sucesso próximas de 1. Além disso, os resultados mostram que, caso os vértices que constituem a solução sejam adjacentes, a probabilidade máxima de sucesso é comprometida. Observando esse comportamento em relação à adjacência das soluções, investigamos o uso de múltiplos self-loops em cada vértice do hipercubo e propusemos um novo algoritmo de busca baseado na caminhada quântica lackadaisical com inversão parcial de fase. Inicialmente, aplicamos essa proposta na busca por vértices não adjacentes. Com essa nova abordagem, também propusemos dois novos pesos, alcançando probabilidades máximas de sucesso próximas de 1. Posteriormente, aplicamos o algoritmo em dois cenários: uma busca por vértices adjacentes e não adjacentes, e uma busca exclusivamente por vértices adjacentes.

Alcançamos probabilidades máximas de sucesso próximas de 1, mesmo em casos em que os vértices da solução eram adjacentes.

Palavras-chave: Computação quântica. Treinamento em Redes Neurais Artificiais. Caminhada Quântica. Algoritmo de Busca Quântica. Caminhada Quântica Lackadaisical. Múltiplos self-loops. Inversão de Fase Parcial. Vértices Marcados Adjacentes

Summary

Introduction

Introduced by Aharonov, Davidovich and Zagury (1993), quantum walks are one of the main tools for the development of quantum algorithms. We can highlight some works that summarize how quantum walks have evolved since their initial proposal. The first work is considered pioneering. As proposed by Shenvi, Kempe and Whaley (2003), a quantum search algorithm was developed based on a quantum walk on the hypercube similar to Grover's search algorithm to search a marked vertex. Both algorithms start in the equal superposition state, make use of Grover's diffusion operator, can be seen as a rotation in a two-dimensional subspace, use an oracle that marks the target state with a phase of -1, and have a time of execution of $O(\sqrt{N})$.

However, there are two important differences. First, Grover's algorithm is usually mapped onto a two-dimensional space, where each point in the search space represents a possible solution. The quantum walk-based search algorithm can be mapped only approximately in a two-dimensional subspace because of the nature of the search space due to the use of the hypercube, i.e., the quantum walk algorithm proposed by Shenvi, Kempe and Whaley (2003) needs a more complex representation not can be mapped directly into a two-dimensional space. Second, the final state of the search algorithm based on quantum walking in the hypercube is not exactly the pure tagged state as it is in Grover's algorithm, it is a linear combination of states that is mainly composed of the marked state but also has small contributions from its nearest neighbors.

Another critical difference is the locality of the unitary transformations used during the algorithm, i.e., the coin operator shifts amplitude only between nearest neighbors, this means that the coin operator changes amplitude only in n-dimensional coin space. Therefore, all operations in an iteration are local. In Grover's algorithm, the reflection operator used is highly non-local. The use of Grover's diffusion operator is another difference between the two algorithms. In Grover's algorithm, this operator is applied to every two-dimensional search space, however, in the quantum walk algorithm, it is used as the quantum coin and acts only in the n-dimensional coin space. In practical terms, this can be advantageous in certain physical implementations of quantum computers. In general, according to Shenvi, Kempe and Whaley (2003), a similar methodology can be applied to any regular graph.

The second work presents the proposal of the quantum walk developed by Wong (2015) called Lackadaisical Quantum Walk based on the lazy random walk. The lazy random walk is a type of classical random walk where the particle has a certain probability

of staying at the current vertex instead of moving to an adjacent vertex. It is often used to model phenomena in different fields of knowledge, such as physics, mathematics, and computer science. Similarly, in the lackadaisical quantum walk, $m > 0$ self-loops are added to each vertex of a complete graph to search for one marked vertex, where, m belongs to the integers. Its effects were investigated in Grover's algorithm when elaborated as a search. This approach can make the quantum walk have a more stable behavior, improving the search efficiency. Later, in a third work, Wong (2017) proposed a modification in this algorithm causing the number of self-loops to be reduced to $m = 1$ with non-integer weight l. This search algorithm has been applied to a large variety of graphs (WONG, 2015; WONG, 2017; RHODES; WONG, 2020; SOUZA; CARVALHO; FERREIRA, 2021b; RHODES; WONG, 2019; CARVALHO et al., 2023; ZHANG; XIANG; SUN, 2018; SAHA et al., 2022; TANAKA; SABRI; PORTUGAL, 2022; QU et al., 2022).

Both search algorithms were also applied to problems involving the training of classical neural networks. Silva (2014) proposed an alternative quantum approach to the classical approach in training a perceptron. In this approach, the adaptation of the synaptic weights is done through the Grover Search Algorithm promoting an upper limit for the perceptron convergence. Souza, Carvalho and Ferreira (2021a) proposed a computational procedure that applies a lackadaisical quantum walk as a complete graph search algorithm to find all synaptic weights of a classical artificial neural network. This work is presented in Chapter and largely represents the motivation that guided all the rest of the research activities developed in this thesis work.

This approach used the search algorithm proposed by Wong (2015) to build a quantum procedure to train classical artificial neural networks. Since the lackadaisical quantum walk on the complete graph considers two types of vertices, a as a solution and b as a non-solution, these vertices are indistinguishable. Therefore, we consider mapping each vertex of the complete graph to a vertex of an n-dimensional grid. In this way, the possible sets of weights that train the classical neural network are mapped onto the states of the quantum system. Although the proposed methodology is theoretical, the procedure guarantees the training of the neural network with high probability.

The quantum walk in the complete graph used in the procedure, in addition to needing a structure to map the set of weights, also depends on a change of basis, which causes an additional cost. Therefore, with the goal of optimizing the procedure, we decided

to study other structures that represented the location of the walker in the search space without the need for additional mapping and that did not depend on a base change. According to Shenvi, Kempe and Whaley (2003), the hypercube quantum walk can be reduced to a line walk, therefore we decided to investigate the lackadaisical quantum walk on the hypercube. However, when adding weighted self-loops to the quantum walk, an immediate concern arises, defining what is the optimal value for the l weight.

Several works investigated weight values for the most varied structures (RHODES; WONG, 2020; CARVALHO et al., 2021; NAHIMOVS; SANTOS, 2021). Rhodes and Wong (2020) proposed an ideal weight for all transitive graphs with a single marked vertex so that the ideal weight of the self-loop is equal to $l = d/N$, where, d represents the degree of the loopless graph divided by the number total number of vertices N. In the investigation of the lackadaisical quantum walk on the hypercube we defined an ideal weight to search for multiple marked vertices. As noted previously, the final state of the search algorithm based on the quantum walk in the hypercube is not exactly the marked pure state as it is in Grover's algorithm, however, with the weight $l = (d/N) \cdot k$, where, k represents the number of marked vertices it was possible to amplify the probability amplitudes to values close to 1 for the case where all marked vertices are non-adjacent.

Other results found in this work indicate that, if the marked vertices as a solution are adjacent, the probability of success is compromised. In this case, two scenarios were investigated. The first scenario only searched adjacent marked vertices. As the number of marked vertices k increased, the probability of success also increased, however, it reached a maximum probability of success of 0.945. The second scenario searched for adjacent and non-adjacent marked vertices. Again, the probability of success is proportional to the number of marked vertices and reached a maximum of 0.976. From these results, we decided to revisit the addition of multiple self-loops at each vertex in the hypercube and partial phase inversion in a try to improve the chances of success in cases where the set of marked vertices contains adjacent vertices. For this, we propose a modification in Grover's oracle so that the edges that represent the self-loops are considered individually. We named this approach Multi-self-loop Lackadaisical Quantum Walk with Partial Phase Inversion – MSLQW - PPI.

Initially, we investigated the application of MSLQW - PPI to search multiple non-adjacent marked vertices in order to prevent any interference caused when there are

adjacent marked vertices. Using the l weights proposed by Rhodes and Wong (2020) and Souza, Carvalho and Ferreira (2021b) for the self-loops and a distribution strategy that divides the weight l/m, where $1 \leqslant m \leqslant 30$ we apply the oracle that marks the entire destination state with a phase of -1 and we observed that the result was similar when using a single weighted self-loop at each vertex. However, as the self-loops $m > 1$ are redundant we decided to investigate the inversion of the set of self-loops partially.

Preliminary results indicate that the phase inversion of a single self-loop is sufficient to achieve maximum success probabilities analogous to those presented by Rhodes and Wong (2020) and Souza, Carvalho and Ferreira (2021b). Thus, we propose two new weights by adding an exponent in n^{α}, where, $\alpha = 2$. With this, we can improve the maximum probability of success to values close to 1 for the weight proposed by Rhodes and Wong (2020). It was also possible to maintain the maximum probability of success with the weight proposed by Souza, Carvalho and Ferreira (2021b) at values close to 1 using $m = s \cdot n$ self-loops at each vertex, where s is the self-loops with their phase inverted and n is the degree of hypercube. We also investigate the application of MSLQW – PPI to search for multiple adjacent marked vertices. The results show that it was possible to improve the maximum probability of success in cases where there are adjacent marked vertices to values close to 1 with the use of the two new weight values proposed for the MSLQW - PPI.

In summary, this document is divided into six chapters as follows. Chapter 2 describes the proposal of a computational procedure that uses a quantum walk in the complete graph for the training of artificial neural networks. In Chapter 3 we present the application of lackadaisical quantum walk on the hypercube to search for multiple solutions. Chapter 4 presents the Multi-self-loop Lackadaisical Quantum Walk With Partial Phase Inversion proposal. Chapter 5 shows the application of MSLQW - PPI to search multiple adjacent marked vertices on the hypercube. Chapter 6 presents the general conclusions about the results, contributions, and implications for the development of new quantum search algorithms based on quantum walks.

Classical Artificial Neural Network Training Using Quantum Walks as a Search Procedure

Article published in IEEE Transactions on Computers

DOI: 10.1109/TC.2021.3051559

A preprint can be found at: https://arxiv.org/pdf/2108.12448

CLASSICAL ARTIFICIAL NEURAL NETWORK TRAINING USING QUANTUM WALKS AS A SEARCH PROCEDURE

A PREPRINT

Luciano S. de Souza*
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
Recife, Brasil
lucianoserafimdesouza@gmail.com

Jonathan H. A. de Carvalho
Centro de Informática
Universidade Federal de Pernambuco
Recife, Brasil
hcarvalho.jon@gmail.com

Tiago A. E. Ferreira
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
Recife, Brasil
tiago.espinola@ufrpe.br

November 18, 2024

ABSTRACT

This paper proposes a computational procedure that applies a quantum algorithm to train classical artificial neural networks. The goal of the procedure is to apply quantum walk as a search algorithm in a complete graph to find all synaptic weights of a classical artificial neural network. Each vertex of this complete graph represents a possible synaptic weight set in the w-dimensional search space, where w is the number of weights of the neural network. To know the number of iterations required *a priori* to obtain the solutions is one of the main advantages of the procedure. Another advantage is that the proposed method does not stagnate in local minimums. Thus, it is possible to use the quantum walk search procedure as an alternative to the backpropagation algorithm. The proposed method was employed for a XOR problem to prove the proposed concept. To solve this problem, the proposed method trained a classical artificial neural network with nine weights. However, the procedure can find solutions for any number of dimensions. The results achieved demonstrate the viability of the proposal, contributing to machine learning and quantum computing researches.

Keywords Artificial Neural Networks Training · Quantum Computing · Quantum Walk · Search Algorithm.

1 Introduction

The need for increased computing power and the miniaturization of components at scales where quantum effects cannot be ignored [Powell, 2008] support the idea that quantum physics can redefine a new frontier for computing problems by taking an essential role in the computational race [Singh and Singh, 2016]. Quantum effects can provide computational gains and algorithms that are mostly more efficient than their classical counterparts for many problems. Quantum computing seeks through the use of aspects of quantum mechanics to expand computational horizons [Lloyd et al., 2013, Yanofsky et al., 2008]. An example of the computational power of quantum computing is performing a search for elements in a disordered database in just $\mathbf{O}(\sqrt{N})$ [Grover, 1996], where N is the size of the database. The gain is quadratically faster than its classical analog.

*95, R. Manuel de Medeiros, 35 - Dois Irmãos, Recife - PE.

In this perspective, efforts are employed in the search for more efficient algorithms. In particular, there is a branch of research called quantum machine learning (QML) [Wittek, 2014], where machine learning methods and artificial intelligence are integrated into the world of quantum computing in the hope to find more efficient procedures than classical algorithms. According to Dunjko et al. [2016], the quantum processing of information from quantum machine learning is interactively configured in two parts called the agent-environment. These parts are grouped into four categories: CC, CQ, QC, and QQ, where C means *Classical*, Q means *Quantum*. The first letter is referent to the agent and the second to the environment. CC encompasses classical machine learning. CQ analyzes how classical learning techniques can help in quantum tasks. QC represents the quantum variants of classical learning algorithms facing a classical environment, where this work is classified. QQ is the quantum systems world.

Papers grouped within the QC category are developed to improve classical procedures with the quantum information processing (QIP) paradigm. Aïmeur et al. [2013] showed that it is possible to accelerate unsupervised learning algorithms by quantizing some of their subroutines. Zheng et al. [2018] designed an algorithm to train a perceptron using Grover's Algorithm [Grover, 1996]. Schuld et al. [2015] developed a procedure that uses a quantum phase estimation algorithm in the classical neural network training process. Date et al. [2019] presents a Classical-Quantum Hybrid Approach for training unsupervised probabilistic machine learning models.

Machine learning (ML) studies techniques to give machines the ability to learn from past experiences. Its core tasks include classification or regression in supervised learning and density estimation or clustering in unsupervised learning, for example. Usually, in the ML context, the training of a machine is performed using a learning algorithm that uses as input a training data set [Aïmeur et al., 2006] to extract information, adjust its parameters, and solve a given problem.

One of the ML techniques widely employed for many problems is the Artificial Neural Network (ANN), wherein the classical (non-quantum) world is a Classical ANN. Classical ANNs are one of the best-known classifiers and predictors. Classical ANNs have proven to be very competitive in solving real-world problems compared to other conventional data analysis methodologies [Prieto et al., 2016]. Its optimization is observed from various perspectives, but in general, classical ANN training is performed mainly using a gradient descent algorithm. However, optimization methods based on the descending gradient algorithm have limitations. Aspects such as weight initialization, network architecture, activation functions, meta parameters, and learning environment can influence the optimization process [Haykin, 2001, Ojha et al., 2017]. In this way, the training algorithm may not obtain a set of weights that train the neural network, staying stagnated at local minimums.

Therefore, a neural network's training algorithm based on the descending gradient method can be seen as a search problem and seeks to minimize an error function. This ANN training is a search problem for an appropriate weight configuration that allows learning the network [Biamonte et al., 2017]. On the other hand, in the quantum computing branch, some search algorithms are more efficient than their classical analogue. For example, Grover's quantum search algorithm [Grover, 1996] and quantum walks [Wong, 2015, Nahimovs and Rivosh, 2015, Wong, 2018, Lovett et al., 2019, Nahimovs, 2019]. In this perspective, quantum search algorithms have already been used for the training process of classical ANN [Schuld et al., 2015, Zheng et al., 2018].

Based on the incipient work developed by Souza et al. [2019] with extreme learning machines, this article generalizes and extends the quantum walk proposal to train a classical ANN. Here, a full graph lackadaisical quantum walk algorithm [Wong, 2015, 2018, Nahimovs, 2019] is applied as a search method to find all the synaptic weights that optimize the learning procedure of a classical ANN.

This paper is organized as follows. Section 2 introduces some concepts of the one-dimensional and the generalization for n-dimensional quantum walks, and it also presents the quantum walk on a complete graph. Section 3, it shows the computational procedure used in this paper. Section 4, it shows the experiments performed. Section 5 discusses the results obtained. Finally, Section 6 is the conclusion of the work.

2 Quantum Walk

The simplest model of the classical random walk can be described by a particle's classical movement in a straight line [Portugal, 2013]. Let the s the particle probability of going to the right. Let $(1 - s)$ the probability of going to left. Therefore, the direction of the particle is conditioned by tossing a coin. This process is probabilistic, so it is impossible to know with certainty where the particle will be at any given time. However, it is possible to calculate the probability p that particle is at a point n at time t, as shown in Equation 1 for the case $s = 1/2$.

$$p(t, n) \simeq \frac{2}{\sqrt{2\pi t}} e^{-\frac{n^2}{2t}} \tag{1}$$

The quantum walks generalize the concept of a classical random walk, *i.e.*, the quantum mechanical counterpart of classical random walks [Venegas-Andraca, 2012]. It is assuming a walker represented by a normalized vector in Hilbert space. The quantum walk evolves in the Hilbert space $\mathcal{H}_M \otimes \mathcal{H}_P$, where $\mathcal{H}_M$ is the coin space that controls the walker's movement, and $\mathcal{H}_P$ defines the position of the walker [Portugal, 2013].

Suppose the quantum walk takes place in a one-dimensional space. In this case, the coin space needs two degrees of freedom. Therefore, the coin space $\mathcal{H}_M$ is generated by the computational base $\{|0\rangle, |1\rangle\}$. A qubit can represent this quantum information. The possible states that one qubit can assume are represented by the state vectors $|0\rangle$ and $|1\rangle$ described in Equations 2 and 3, respectively.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{2}$$

and

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{3}$$

Walker space $\mathcal{H}_P$ is generated by base $\{|n\rangle : n \in \mathbb{Z}\}$ which represents all integers of one-dimensional space. Consider an operator S that, when applied to the system formed here by a coin and a walker, will shift the position of the individual to $|n+1\rangle$ or $|n-1\rangle$ depending on the state of the coin according to Equation 4.

$$\begin{aligned} S\,|0\rangle\,|n\rangle &= |0\rangle\,|n+1\rangle \\ S\,|1\rangle\,|n\rangle &= |1\rangle\,|n-1\rangle \end{aligned} \tag{4}$$

A unitary transformation describes the evolution of a closed quantum system. This evolution depends on the application of an operator U shown in Equation 5 to the system over time [Nielsen and Chuang, 2002],

$$U = S(H \otimes I) \tag{5}$$

where H is the Hadamard operator and I is the identity.

For an operator U of a quantum system to be unitary, it must satisfy the condition described in Equation 6, where $U^\dagger$ is the adjunct of U. This condition is necessary for the norm of the vectors to be maintained, for this we must calculate $\parallel u \parallel = \sqrt{\langle u \mid u \rangle}$, where $\langle u \mid u \rangle$ is the inner product.

$$UU^\dagger = U^\dagger U = I \tag{6}$$

The evolution of the quantum walk system is performed by the Equation 7.

$$|\Psi(t)\rangle = U^t\,|\Psi(0)\rangle \tag{7}$$

where $|\Psi(0)\rangle$ is the initial state of the quantum system.

Consider the initial state of the quantum system given by the Equations 8 and 9. Asymmetrical and symmetrical state, respectively. We can obtain the asymmetrical and symmetrical probability distributions after one hundred applications of U operator, shown in Figures 1 and 2. Unlike the classical case, where the distribution is an origin-centered Gaussian, for both case asymmetrical and symmetrical, the quantum walk has a large spread with an interval of $-t/\sqrt{2}$ to $t/\sqrt{2}$.

$$|\Psi(0)\rangle = |0\rangle\,|n=0\rangle \tag{8}$$

$$|\Psi(0)\rangle = \frac{|0\rangle - i\,|1\rangle}{\sqrt{2}}\,|n=0\rangle \tag{9}$$

For example, if the quantum walk starts in the initial state described in Equation 8. Apply the Hadamard operator H as the coin of the quantum walk system and then applies the shift operator S. We have the initial evolution state

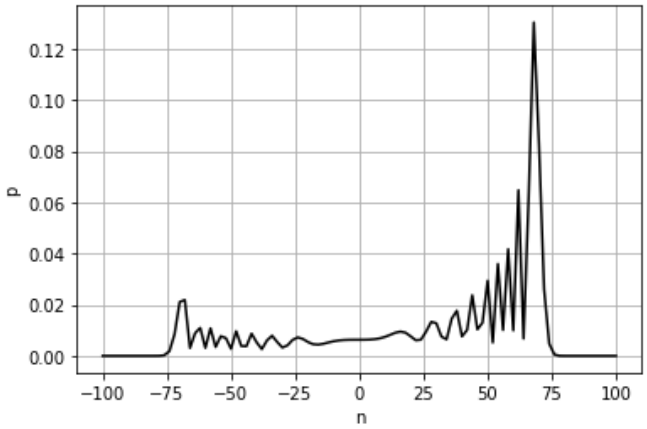


Figure 1: Probability distribution of the one-dimensional quantum walk after 100 steps. The initial state is described by the Equation 8. The points with zero value were ignored.

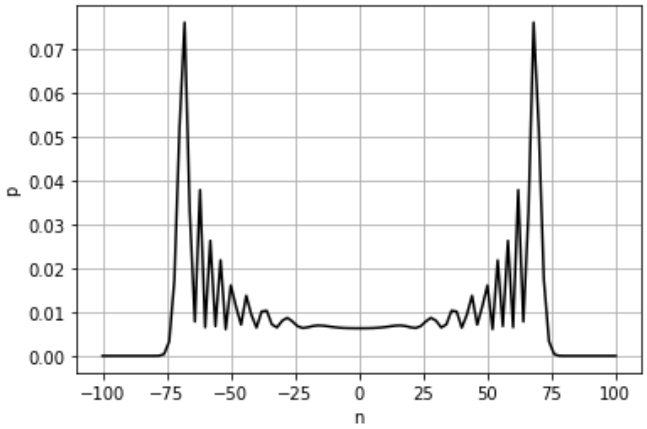


Figure 2: Probability distribution of the one-dimensional quantum walk after 100 steps. The initial state is described by the Equation 9. The points with zero value were ignored.

presented in Equation 10 at the end of the first step. The Hadamard operator application in the computational base vectors generates a superposition state, *i.e*, one of the most well-known quantum effects that qubits are at the same time in distinct states.

$$|\Psi(1)\rangle = \frac{1}{\sqrt{2}}(|0\rangle\,|1\rangle + |1\rangle\,|-1\rangle) \tag{10}$$

With the successive application of the evolution operator U described in Equation 5, at the end of the third stage, we can observe in Equation 11 that the state $|\Psi(3)\rangle$ is asymmetrical about the origin. This asymmetry will keep for all system evolution, as can be observed in Figure 1.

$$\begin{aligned} |\Psi(2)\rangle &= \frac{1}{2}(|0\rangle\,|2\rangle + (|1\rangle + |0\rangle)\,|0\rangle - |1\rangle\,|-2\rangle) \\ |\Psi(3)\rangle &= \frac{1}{2\sqrt{2}}(|0\rangle\,|3\rangle + (2\,|0\rangle + |1\rangle)\,|1\rangle \\ &\quad - |0\rangle\,|-1\rangle + |1\rangle\,|-3\rangle) \end{aligned} \tag{11}$$

2.1 One-dimensional quantum walk

This section will describe the one-dimensional quantum walk model analytically and recursively. The generic state for a one-dimensional quantum walk model is described in Equation 12 [Portugal, 2013].

$$|\Psi(t)\rangle = \sum_{n=-\infty}^{\infty} \left(\alpha_n(t)\,|0\rangle + \beta_n(t)\,|1\rangle\right)|n\rangle\,, \tag{12}$$

where the coefficients $\alpha_n(t)$ and $\beta_n(t)$ satisfy the condition described in Equation 13.

$$\sum_{n=-\infty}^{\infty} |\alpha_n(t)|^2 + |\beta_n(t)|^2 = 1 \tag{13}$$

Applying the operator $H \otimes I$ to state $|\Psi(t)\rangle$, we find recursive formulas involving the coefficients α and β in Equations 14 and 15.

$$\alpha_n(t+1) = \frac{\alpha_{n-1}(t) + \beta_{n-1}(t)}{\sqrt{2}} \tag{14}$$

$$\beta_n(t+1) = \frac{\alpha_{n+1}(t) - \beta_{n+1}(t)}{\sqrt{2}} \tag{15}$$

As already shown in Figures 1 and 2, the quantum walk probability distributions are dependent on the initial state. Therefore, the initial state of the system can generate a walk with symmetrical or asymmetrical probability distribution about the origin. The probability distribution can be calculated using the Equation 16.

$$p(t,n) = |\alpha_n(t)|^2 + |\beta_n(t)|^2 \tag{16}$$

2.2 n-dimensional quantum walk

It is possible to generalize the concept presented in Section 2.1 to any number of dimensions. Consider the quantum walk in an infinite n-dimensional grid with the associated Hilbert space $\mathcal{H}_M \otimes \mathcal{H}_P$, whose $\mathcal{H}_P$'s computational base is $\{|x, y, \ldots, n\rangle : x, y, \ldots, n \in \mathbb{Z}\}$, and the Coin space $\mathcal{H}_M$'s computational basis is $\{|i_x, i_y, \ldots, i_n\rangle : i_x, i_y, \ldots, i_n \in \{0,1\}^n\}$. The generic state for this quantum walk model at time t is presented in Equation 17.

$$|\Psi(t)\rangle = \sum_{i_x,i_y,\ldots,i_n=0}^{1} \sum_{x,y,\ldots,n=-\infty}^{\infty} \psi_{i_x,i_y,\ldots,i_n;x,y,\ldots,n}(t)\,|i_x, i_y, \ldots, i_n\rangle\,|x, y, \ldots, n\rangle \tag{17}$$

where $\psi_{i_x,i_y,\ldots,i_n;x,y,\ldots,n}(t)$ are complex functions that satisfy the condition shown in Equation 18 for all time t.

$$\sum_{i_x,i_y,\ldots,i_n=0}^{1} \sum_{x,y,\ldots,n=-\infty}^{\infty} \left|\psi_{i_x,i_y,\ldots,i_n;x,y,\ldots,n}(t)\right|^2 = 1 \tag{18}$$

Moreover, it is possible to calculate the distribution of probabilities using Equation 19.

$$p_{x,y,\ldots,n}(t) = \sum_{i_x,i_y,\ldots,i_n=0}^{1} \left|\psi_{i_x,i_y,\ldots,i_n;x,y,\ldots,n}(t)\right|^2 \tag{19}$$

Applying the standard evolution operator $U = S(C \otimes I)$ (C is the coin operator) to the generic state described in Equation 17 and making the expansions, we obtain Equation 20 which is the walker evolution equation [Portugal, 2013].

$$\psi_{i_x,i_y,\ldots,i_n;x,y,\ldots,n}(t+1) = \sum_{j_x,j_y,\ldots,j_n=0}^{1} C_{i_x,i_y,\ldots,i_n;j_x,j_y,\cdots,j_n}\,\psi_{j_x,j_y,\ldots,j_n;x+(-1)^{i_x},y+(-1)^{i_y},\ldots,n+(-1)^{i_n}}(t) \tag{20}$$

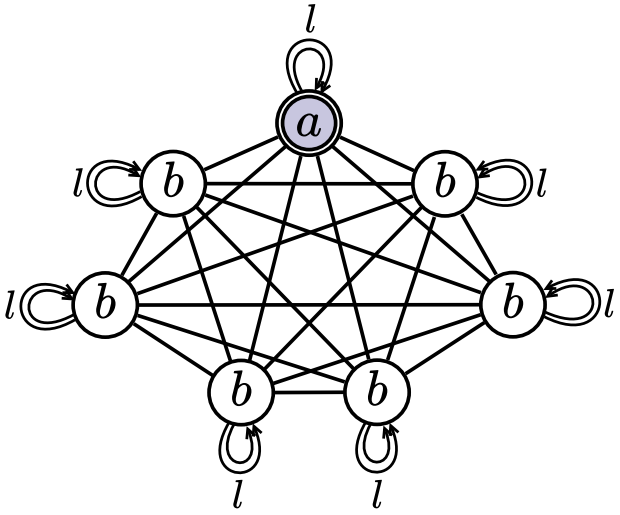


Figure 3: Complete graph with $N = 7$ vertices. The single vertex marked as a is indicated by the double circle shaded. Adapted from Wong's work [Wong, 2015]. Grover search with lackadaisical quantum walks.

2.3 Lackadaisical Quantum Walk on Complete Graph

The quantum walks presented in the previous sections are the basis for other techniques that can be used in other search spaces. One such variation is the quantum walk in a complete graph developed by Wong [2015], represented in Figure 3.

There are two types of vertex, a and b, marked as a solution and non-solution, respectively. Each one vertex has l self-loops. This approach considers the walker's movement into the complete graph to create the states of the new computational basis.

If the walker is on an a vertex, there are two movement options. It can move to a vertex that is a solution ($a \to a$) or to a vertex that is not a solution ($a \to b$). Defining the quantum states $|a\rangle \otimes |a \to a'\rangle$ and $|a\rangle \otimes |a \to b\rangle$. Similarly, if the walker is on any vertex b. It can move to a vertex that is a solution ($b \to a$) or to a vertex that is not a solution ($b \to b'$). Defining the quantum states $|b\rangle \otimes |b \to a\rangle$ and $|b\rangle \otimes |b \to b'\rangle$.

Note that the states $|a\rangle$ or $|b\rangle$ represent the graph vertices and $|a \to b\rangle$, for example, represents the edges where $|a\rangle$ is the walker's current state and $|b\rangle$ is the state to which the walker will move to it. Equation 21 shows the states $|AA\rangle, |AB\rangle, |BA\rangle,$ and $|BB\rangle$ of the new quantum states for the situation where there is only one solution (only one vertex labeled as a). N is the total number of vertices. Since there is only one solution, $k = 1$, and the number of self-loops is greater than zero, $l > 0$.

$$
\begin{aligned}
|AA\rangle &= \frac{1}{\sqrt{l}} |a\rangle \otimes |a \to a'\rangle \\
|AB\rangle &= \frac{1}{\sqrt{N-1}} \sum_{b} |a\rangle \otimes |a \to b\rangle \\
|BA\rangle &= \frac{1}{\sqrt{N-1}} \sum_{b} |b\rangle \otimes |b \to a\rangle \\
|BB\rangle &= \frac{1}{\sqrt{(N-1)(N+l-2)}} \sum_{b} \sum_{b'} |b\rangle \otimes |b \to b'\rangle
\end{aligned}
\tag{21}
$$

Equation 22 defines the initial state $|\Psi_0\rangle$. This state is described as the uniform superposition $N^{-1} \sum_{x,y} |x\rangle |x \to y\rangle$ expressed in terms of the states AA, AB, BA and BB.

$$|\Psi_0\rangle = \frac{1}{\sqrt{N(N+l-1)}}(\sqrt{l}\,|AA\rangle + \sqrt{N-1}\,|AB\rangle + \sqrt{N-1}\,|BA\rangle + \sqrt{(N-1)(N+l-2)}\,|BB\rangle) \tag{22}$$

The lackadaisical quantum walk is accomplished by successive applications of a unitary operator U, defined in Equation 23 which inverts the sign of solution states $|a\rangle\,|a \to x\rangle$ using an oracle and swaps vertices on each edge, $|x\rangle\,|x \to y\rangle \to |y\rangle\,|y \to x\rangle$ as described in Wong [2015].

$$U = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ 0 & 0 & -\cos\phi & \sin\phi \\ -\sin\theta & -\cos\theta & 0 & 0 \\ 0 & 0 & \sin\phi & \cos\phi \end{pmatrix} \tag{23}$$

where θ is defined such that,

$$\cos\theta = \frac{N-l-1}{N+l-1}$$

and

$$\sin\theta = \frac{2\sqrt{l(N-1)}}{N+l-1}$$

and ϕ is defined such that,

$$\cos\phi = \frac{N-l-3}{N+l-1}$$

and

$$\sin\phi = \frac{2\sqrt{N+l-2}}{N+l-1}$$

Thus, the evolution of the system occurs in a four-dimensional subspace, and each state of the new representation is formed by overlapping vertices and edges [Wong, 2015].

2.3.1 Quantum walk with self-loops for k solutions

The previous section introduced the quantum walk in a complete graph and describes the approach to the case of a single solution ($k = 1$) and the number of self-loops $l > 0$. Now, consider a number of solutions $k > 1$ and only one self-loop ($l = 1$) per vertex.

The solution set has the number k of vertices, and the non-solution set has $N - k$ vertices, where N is the total number of vertices. We will use the previous idea that considered the walker's movement to define the states of the new multiple solutions computational basis.

If the walker is in a vertex a moving to another vertex a', there will exist k edges of the type $|a \to a'\rangle$. If the walker is in a vertex a moving to a vertex b, there will exist $N - k$ edges of the type $|a \to b\rangle$. Similarly, if the walker is in a vertex b moving to a vertex a, there will exist k edges of the type $|b \to a\rangle$. And if walker is in a vertex b moving to another vertex b', there will exist $N - k$ edges of the type $|b \to b'\rangle$.

The number of vertices marked as solution must be of the order $o(N)$ because if $k = O(N)$, then $k = cN$ (in the limit of $N \to \infty$ and c is a finite constant). In this last case, the search for one solution could be performed classically in an efficient way, *i.e.*, in a constant number of guesses [Wong, 2015]. Thus, the new quantum states are redefined in Equation 24.

$$
\begin{aligned}
|AA\rangle &= \frac{1}{k}\sum_{a}\sum_{a'}|a\rangle \otimes |a \to a'\rangle \\
|AB\rangle &= \frac{1}{\sqrt{k(N-k)}}\sum_{a}\sum_{b}|a\rangle \otimes |a \to b\rangle \\
|BA\rangle &= \frac{1}{\sqrt{k(N-k)}}\sum_{b}\sum_{a}|b\rangle \otimes |b \to a\rangle \\
|BB\rangle &= \frac{1}{N-k}\sum_{b}\sum_{b'}|b\rangle \otimes |b \to b'\rangle
\end{aligned}
\tag{24}
$$

The initial state of the system is rewritten, as presented in Equation 25.

$$
|\Psi_0\rangle = \frac{1}{N}(k\,|AA\rangle + \sqrt{k(N-k)}\,|AB\rangle + \sqrt{k(N-k)}\,|BA\rangle + (N-k)\,|BB\rangle) \tag{25}
$$

A modification to the evolution operator U (Equation 23) is made so that the number of solutions is included in the definitions of θ and ϕ [Wong, 2015]. Therefore, θ is redefined according to Equations 26 and 27,

$$
\cos\theta = \frac{N-2k-l+1}{N+l-1} \tag{26}
$$

$$
\sin\theta = \frac{2\sqrt{(N-k)(k+l-1)}}{N+l-1} \tag{27}
$$

and ϕ is redefined according to Equations 28 and 29,

$$
\cos\phi = \frac{N-2k+l-1}{N+l-1} \tag{28}
$$

$$
\sin\phi = \frac{2\sqrt{k(N-k+l-1)}}{N+l-1} \tag{29}
$$

The maximum success probability value is reached after the number of steps t defined in Equation 30 [Wong, 2015]. Success is defined as the measurement of some state $|AA\rangle$ or the state $|AB\rangle$. Both states represent the set of vertices marked as a solution.

$$
t = \frac{\pi}{\sqrt{2(2k+l-1)}}\sqrt{N} \tag{30}
$$

2.3.2 A Toy Example

Consider the particular case of the lackadaisical quantum walk in a complete graph with $N = 8$ vertices, $k = 2$ solutions, and $l = 1$ self-loops at each vertex. For illustrative purposes, these vertices have also been marked with sub-indices, so a_1 and a_2 are the solutions whereas vertices $b_1, \ldots, b_6$ are not solutions. In practice, however, sub-indices or whatever kind of information that can distinguish the solutions from one another, or the non-solutions from one another, are not available. The vertices are marked only with a or b, exclusively.

The quantum state $|AB\rangle$, for example, is formed by all vertices that are solution combined with their respective edges for non-solution vertices. Therefore, the state $|AB\rangle$ for the given example is defined according to Equation 31.

$$|AB\rangle = \frac{1}{\sqrt{k(N-k)}} \sum_a \sum_b |a\rangle \otimes |a \to b\rangle = \frac{1}{\sqrt{12}} \Big[\Big(|a_1\rangle |a_1 \to b_1\rangle + |a_1\rangle |a_1 \to b_2\rangle + |a_1\rangle |a_1 \to b_3\rangle + |a_1\rangle |a_1 \to b_4\rangle + |a_1\rangle |a_1 \to b_5\rangle + |a_1\rangle |a_1 \to b_6\rangle \Big) + \Big(|a_2\rangle |a_2 \to b_1\rangle + |a_2\rangle |a_2 \to b_2\rangle + |a_2\rangle |a_2 \to b_3\rangle + |a_2\rangle |a_2 \to b_4\rangle + |a_2\rangle |a_2 \to b_5\rangle + |a_2\rangle |a_2 \to b_6\rangle \Big) \Big] \tag{31}$$

The analysis for the quantum states $|AA\rangle$, $|BA\rangle$ and $|BB\rangle$ is analogous. With the new quantum states prepared, the initial state of the system $|\Psi_0\rangle$ can be defined according to Equation 32. This initial state is also normalized.

$$|\Psi_0\rangle = \frac{1}{8} \Big(2\,|AA\rangle + \sqrt{12}\,|AB\rangle + \sqrt{12}\,|BA\rangle + 6\,|BB\rangle \Big) \tag{32}$$

Making only three ($t = 3$) successive applications of the evolution operator U, represented by the matrix of Equation 33, the probability of success (the measurement of a solution state) tends to 1. A measurement made in $|\Psi_3\rangle$ makes the system to collapse to the state $|AA\rangle$. All energy of the system is concentrated at this state $|AA\rangle$.

$$U = \frac{1}{2} \begin{pmatrix} 1 & -\sqrt{3} & 0 & 0 \\ 0 & 0 & -1 & \sqrt{3} \\ -\sqrt{3} & -1 & 0 & 0 \\ 0 & 0 & \sqrt{3} & 1 \end{pmatrix} \tag{33}$$

3 Proposal Procedure

Quantum walks are algorithms that can be applied to search problems [Shenvi et al., 2003, Lovett et al., 2019, Wong, 2018], where the proof of its correctness can be found in [Feng et al., 2007]. Thus, it is possible to idealize this algorithm's application to find the set of synaptic weights that train a classical artificial neural network. Based on the concept of quantum information processing called agent-environment presented in Section 1, the objective is to replace the classical algorithm with a quantum search algorithm in a neural network training process.

In a classical environment, we replace the backpropagation algorithm for a quantum walk to search synaptic weights. The evolution of the quantum walk occurs by applying a unitary operator U over an initial state. Subsequently, a measurement of the walker state is performed. If the measurement is performed at each step, then the quantum walk falls in the classical case. In this situation, the correlations between the different positions of the walker are lost.

For the correlations between the walker's positions to be maintained, the measurement should not occur at every single step. Thus, the measurement process only will occur after a predetermined time t. Once the correlations between positions hold, constructive and destructive interference occurs [Portugal, 2013]. The interference caused by these relationships between positions generates the probability amplitudes for each position. Depending on the initial state of the system, the probability distribution may be asymmetrical or symmetrical according to equations 8 and 9, respectively, as viewed in Section 2. As observed in Figures 1 and 2, according to the probability distribution, the chances of finding the walker on the extremities are higher than finding it in another position. In this way, it is possible to obtain these extreme positions of the walker with a high probability of measurement.

Therefore, it is possible to determine with a high probability the walker's extreme position after a given number of steps. If this position does not contain a solution, the quantum walk search will not succeed. It is necessary to know the position where there is a solution and guarantee an amplitude amplification for this position. It is also necessary to guarantee that the solution state at the time of measurement has a high probability of measurement. The complete-graph quantum walk proves to be adequate for these proposals. The complete-graph quantum walk employed here considers only the information about the *solution* or *no solution* label (a or b) contained in all vertices to execute the quantum walk evolution. However, to define these labels, the information about the weights used to train the classical ANN is necessary. Thus, besides the label for *solution* ($|a\rangle$) and *no solution* ($|b\rangle$), each vertex will also have the information about the associated weights used to train the classical ANN.

Consider a w-dimensional lattice. Let a classical ANN with the number of weights equal to the lattice's dimension, w weights. In the discrete representation, each intersection of lattice lines can represent an ANN w weights configuration. Figure 4 represents this idea for the 2-dimensional case. In this way, each point i in this w-dimensional lattice will

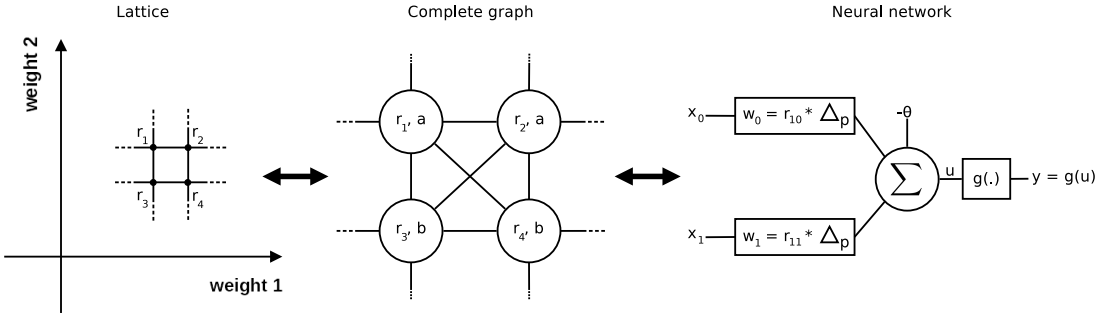


Figure 4: On the left side, a discrete schematic representation of the set of weight values of an artificial neural network. This representation generates a grid, represented here by a lattice. Each intersection point i of the lattice has the information about the set of weight values r_i. With an oracle operator, the state solution label (a for a solution and b for no solution) is determined. On the central figure, the complete graph representation of the grid. On the right side, a neural network with one neuron demonstrates the relationship between the labels on the grid, the complete graph, and the definition of synaptic weights w_0 and w_1, where $g(\cdot)$ is the activation function, and $u = x_0 \cdot \mathrm{w}_0 + x_1 \cdot \mathrm{w}_1 - \theta$.

be a vertex in a complete graph, where it was labeled as a if its value configuration of weights r_i is a solution for the ANN. Otherwise, it was labeled as b. The labels a and b are created by applying the oracle to the grid. Change the grid representation to the complete graph creates a new representation, where all i grid point is a vertex with the information r_i (the weight set) and the label a or b. Wong's quantum walk [Wong, 2015] is applied in the complete graph, where now each vertex has the label a or b, and the associated weight set (r_i) used to train the classical ANN. Then, the search procedure is done, where a vertex labeled by $|a\rangle$ is searched.

Thus, to recover the synaptic weights that trained the ANN after the quantum walk evolution, it is necessary to obtain the specific weight values configuration r_i of the state measured. For this reason, a modification is proposed to the original procedure, including the weight vector $|\vec{r_i}\rangle$ in the base states definition. All quantum walk search procedure is the same, where the state with label a is sought. However, with this new associated information about the weight vector $|\vec{r_i}\rangle$, it is possible to determine the walker's position on the grid (or lattice) in the final measurement procedure in the final state found by the quantum walk algorithm. In this way, the information about the set of weight values used to train the classical ANN can be recovered. The new definition of states is presented in Equation 34. With this modification, it is proposed the search procedure presented in Algorithm 1.

$$
\begin{aligned}
|AA\rangle &= \frac{1}{k} \sum_{a} \sum_{a'} |\vec{r}_a\rangle |a\rangle \otimes |a \to a'\rangle \\
|AB\rangle &= \frac{1}{\sqrt{k(N-k)}} \sum_{a} \sum_{b} |\vec{r}_a\rangle |a\rangle \otimes |a \to b\rangle \\
|BA\rangle &= \frac{1}{\sqrt{k(N-k)}} \sum_{b} \sum_{a} |\vec{r}_b\rangle |b\rangle \otimes |b \to a\rangle \\
|BB\rangle &= \frac{1}{N-k} \sum_{b} \sum_{b'} |\vec{r}_b\rangle |b\rangle \otimes |b \to b'\rangle
\end{aligned}
\tag{34}
$$

Initially, it is necessary to define some initial parameters of the Algorithm 1. A w-dimensional grid (or a lattice) will define the possible weight configurations for the walker, where each dimension represents an ANN weight set. Therefore, the user defines the number of grid points, N, and the distance between adjacent points, Δp. In this way, N will be the number of possibles positions state of the system for the walker, represented by the complete graph, and Δp will define the granularity of the weights representation. If Δp is small, the weight representation will have high resolution, but a significant N value is necessary to search for a solution in practice. If Δp is large, the weight search will have low resolution, but a small N value is sufficient to cover a given search space. The algorithm employs an oracle, which defines the vertices that are solution and non-solution. The oracle employed here is described in Section 4.5. In this way, it is considered that the oracle used in this work is independent of the algorithm proposed. An w-dimensional sparse matrix with N elements represents classically the oracle, where the element marked with number one denotes a solution and with zero a non-solution. The user also defines the number of self-loops per vertex, l. However, the number of self-loops per vertex here always was $l = 1$ for all experiments.

After defining the initial parameters, the proposed algorithm performs a quantum count to estimate the number of solutions k in the search space, as indicated in line 3 of the Algorithm 1. It is necessary to know the quantity k of solutions to determine the initial state, the shift operator U, and the number t of iterations. It is possible to define the number of solutions by combining the phase estimation technique based on the Fourier Quantum Transform with the Grover iteration [Nielsen and Chuang, 2002]. Alternatively, it is also possible to apply the amplitude estimation to the problem of approximate counting [Brassard et al., 2000], or to use techniques inspired by Shor's celebrated quantum factorization algorithm and combines them with Grover's algorithm [Boyer et al., 1998]. Note that the number of solutions is known at this moment, but their search space positions are not known.

With the number k of solutions determined, the states will be constructed according to Equation 34. The quantum walk occurs in a complete graph, as shown in Figure 3, where the quantum walk will be performed in a four-dimensional space according to in Section 2.3, reducing the search space. Thus, the search space is represented by the superposition of vertices and edges already presented in Equation 34.

Following the Algorithm 1 in line 4, the initial state preparation is performed considering the space size, the number of solutions, and the relations between vertices and edges, according to Equation 25.

After preparing the system, the quantum walk is performed. Line 5 of the Algorithm 1 defines the total number of steps t, according to Equation 30. At each step j, the evolution operator U is applied to the quantum system $|\Psi_{j-1}\rangle$, where $j = 1, 2, \ldots, t$, as shown in line 6.

Once the evolution is completed, a measurement in the basis $|x\rangle\,|x \to y\rangle$ is performed, as presented in line 8. Thus, the states $|a\rangle\,|a \to a'\rangle$ or the states $|a\rangle\,|a \to b\rangle$, which both have the solutions, is obtained with high probability. It is worth noting that the vertex information where the walker stays defines the found solution at the time of the measurement. The direction where the walker points for the next quantum movement (defined by its edge) is only relevant to define the quantum walk evolution, not to define the solution state at the measurement time.

After recovering the states $|a\rangle\,|a \to a'\rangle$ or $|a\rangle\,|a \to b\rangle$ with high probability after the measurement, in the line 9, the algorithm initializes the classical neural network weights as shown on the right side of Figure 4.

Algorithm 1: TRAINING ALGORITHM.

1 **begin**
2 Set the parameters: Δp, N and l
3 Quantum count execution
4 Preparation of the initial state
5 **for** $j \leftarrow 1$ **to** $\frac{\pi}{\sqrt{2(2k+l-1)}}\sqrt{N}$ **do**
6 $|\Psi_j\rangle \leftarrow U\,|\Psi_{(j-1)}\rangle$
7 **end**
8 Make the measurement
9 Initialize the weights of the Artificial Neural Network
10 **end**

4 Experiment Setup

A simple classification problem was performed to assess the concept of the proposed algorithm. An artificial neural network of MLP type (Multilayer Perceptron Type) was employed to solve the "EXCLUSIVE-OR" classification problem. The neural network was initialized with the weights generated by the procedure proposed in this work.

4.1 Exclusive-OR Function

The EXCLUSIVE OR (XOR) problem, a simple but nonlinearly separable problem, was used as the function to be learned by the neural network. The XOR function, also known as exclusive disjunction, is an operation on two binary values, x_0 and x_1, where if only one of these binary values is equal to 1, then the function returns 1, otherwise returns 0 [LeCun et al., 2015].

Consider the problem of classifying points in the unitary hypercube. The EXCLUSIVE OR can be understood as a particular case of this problem. In this case, it is sufficient to consider only the four vertices of the unit square corresponding to the points $\{(0,0),(0,1),(1,0),(1,1)\}$. Each set of patterns determine outputs that are called classes. The inputs $\{(0,0),(1,1)\}$ generate outputs $x_0 \oplus x_1 = 0$, where it will be called class 0. The inputs $\{(0,1),(1,0)\}$

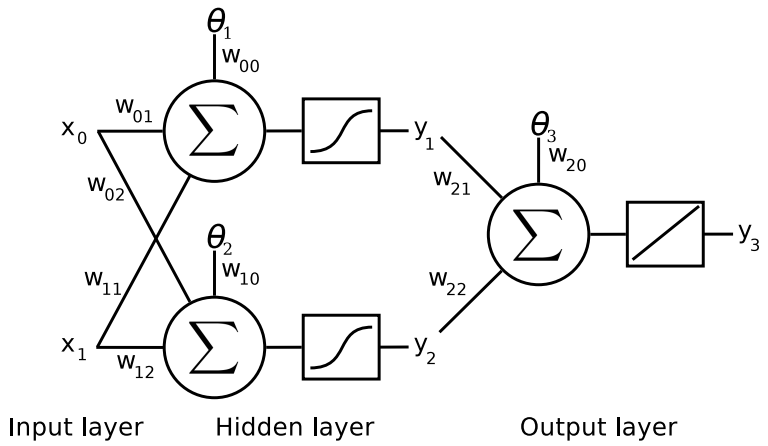


Figure 5: Configuration of a multilayer neural network. The letters θ_1, θ_2 and θ_3 mean the bias and the letters y_1, y_2 and y_3 mean the outputs of neurons.

generate outputs $x_0 \oplus x_1 = 1$, class 1 [Haykin, 2001]. Therefore, it is a pattern classification problem that consists of associating an input pattern (x_0, x_1) with one of the previously defined classes $\{0, 1\}$ [Da Silva et al., 2017].

4.2 Neural Network Architecture

The EXCLUSIVE OR problem can be solved by a Multi-Layer Perceptron (MLP) neural network with three neurons, where two of them are in the hidden layer, and the other neuron is in the output layer. The neural network used in this work follows this architecture.

The input layer has two values $\{\mathsf{x}_0, \mathsf{x}_1\}$ that are inputs data. The hidden layer has two neurons. The output layer has one neuron. Each neuron has a bias. Thus, the neural network has nine synaptic weights. Six weights are in the hidden layer $\{\omega_{00}, \omega_{01}, \omega_{02}, \omega_{10}, \omega_{11}, \omega_{12}\}$ and three are in the output layer $\{\omega_{20}, \omega_{21}, \omega_{22}\}$ according to Figure 5. The sigmoid logistic, described by Equation 35, is the activation function for all neurons of the hidden layer and the neuron of the output layer is linear,

$$f(x) = \frac{L}{1 + e^{-\lambda(x-\zeta_0)}} \tag{35}$$

where $\lambda = 1$ is the declivity of the curve, $L = 1$ is the maximum value of the curve, and $\zeta_0 = 0$ is the value of x at the midpoint of the curve.

4.3 Hardware and Software Setup

The simulations were performed using the following hardware configurations. For simulate neural network training with classical backpropagation algorithm, it was used,

- Operational System: Debian GNU/Linux 10 Buster;
- Memory: 4 GiB;
- Processor: Intel Core i3-5005U CPU @ 2.00 GHz x 4;
- OS type 64-bit;
- HD: 1 terabytes.

To simulate the procedure using a quantum walk in the complete graph, it was used

- Operational system: Debian GNU/Linux Jessie 8.11;
- Memory: 16 GiB;
- Processor: Xeon Intel 5th gen CPU @ 3.6 GHZ x 8;
- OS type 64-bit;

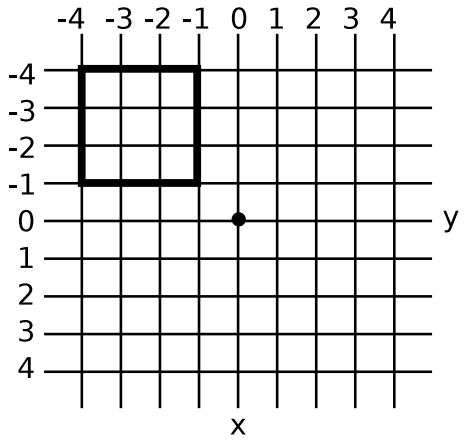


Figure 6: Illustration of a search in an infinite two-dimensional grid using a window with a finite set of points.

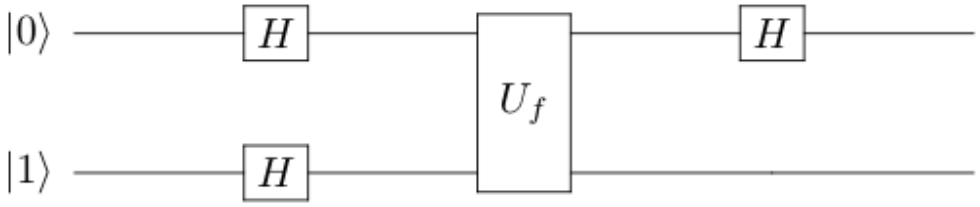


Figure 7: The circuit that implements the Deutsch algorithm [Yanofsky et al., 2008].

- HD: 4 terabytes.

The programming languages used to write the algorithms were Python 3.6 with the open-source machine learning framework PyTorch[2].

4.4 Search Space

Theoretically, the quantum walk is performed in an infinite space. However, in practice, because of memory and hardware limitations, we have determined sub-spaces or windows. Once the window size is defined, the procedure performs the search within it.

Geometrically, the windows employed here will always be hyper-cubes in the search space. For example, in the 2-dimensional case, a window is a square with $N = z^2$ points, where z is the number of points in a dimension. For the d-dimensional case, a window is a hyper-cube whit $N = z^d$ points, where there are z points in each dimension.

If there is no solution in the sub-region defined by the windows, the windows will be shifted. By applying offsets, the windows are moved by performing the search in the infinite search space regions. To illustrate, consider an infinite two-dimensional search space as illustrated in Figure 6. These window shifts in infinite space are carried out until at least one solution is found in the current window. There are many forms to define the shift of the windows. However, the simplest way is to sum an offset of size z for each dimension with respect to the current window, which was the approach used in this work.

4.5 Representation of the Oracle

An oracle is a structure capable of generating answers to binary questions. The circuit shown in Figure 7 represents the implementation of the Deutsch algorithm. Conditional port U_f implements the NOT-controlled port with control bit $f(x)$ and acts as $U_f : |x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$.

Port U_f is a black box with no explicit implementation, often called an oracle [Sasaki and Nakahara, 2013]. In this way, a quantum oracle is a "black box" operator that, when applied to a system state, return if this specific state is a

[2] `https://pytorch.org/`

solution or is not a solution. An example of oracle implementation can be seen in the work of Zheng et al. [2018], which uses Grover's oracle as a central part of the circuit that implements their proposal of quantum perceptron models [Kapoor et al., 2016].

In this way, it was necessary to create a representation of an oracle to perform the simulations. The solution employed uses a sparse matrix with the size of the search space. The positions filled with 1 indicate the points that are a solution and 0 otherwise. In practice, the oracle answers whether a given point in the walker space, after converted to synaptic weights, correctly classifies the input patterns or not. However, a formal conception of this quantum oracle for this purpose is beyond the scope of this work. Therefore, we considered it existing.

4.6 Weight Generation

The quantum walk performs a search in quantum states that represent points in an integer space. At the end of the process, it is necessary to convert these points into synaptic weights values represented by real numbers. Thus, a real value was defined, Δp, where it will multiply each measured point at the end of the process. That is, the walker grid space is in Δp units. The walker starts at the window center, and the walker will move by the positive and negative integer indices of each dimension of the grid.

Therefore, when the proposed algorithm makes the measurement at the walker position in the grid $|r_i\rangle$ (where i is the label for each point in the w-dimensional search space), the integer components of $r_i = (n_1, n_2, \ldots, n_w)$ for each dimension are converted to the real values $\Delta p * n_j$ ($j = 1, 2, \ldots, w$), generating the synaptic weights for the neural network. In this way, the search is performed only among the factors of Δp. Also, the value defined for Δp establishes the search refinement level.

4.7 Measurement Process

Through unitary transformation, closed quantum systems evolve. In order to be able to access information that is in the $|\Psi\rangle$ state, an observation must be made. In practice, measurements are made in laboratories using physical devices, such as lasers, magnets, scales, and stopwatches, but in theory, this measurement process is described mathematically [Portugal, 2013]. According to the postulate of quantum mechanics, the probability of a state occurs is,

$$p(m) = \langle \Psi \mid M_m^\dagger M_m \mid \Psi \rangle , \tag{36}$$

where M is a measurement operator, † is the symbol used to describe the conjugate transpose operation, and the index m represents the results that can occur in the experiment after the measurement. The state $|\Psi'\rangle$ described in Equation 37 is the state of the system just after the measurement [Nielsen and Chuang, 2002].

$$|\Psi'\rangle = \frac{M_m |\Psi\rangle}{\sqrt{\langle \Psi \mid M_m^\dagger M_m \mid \Psi \rangle}} \tag{37}$$

As an example we are going to measure the state $|\Psi\rangle$ whose states are described in Equation 34. Considering the system with four possible results we will define the operators $\{|AA\rangle \langle AA| , |AB\rangle \langle AB| , |BA\rangle \langle BA| , |BB\rangle \langle BB|\}$. So when applying Equations 36 and 37 we have,

$$p(m) = |\lambda_m|^2$$

$$|\Psi'\rangle = \frac{\lambda_m}{|\lambda_m|} |m\rangle$$

where $m = AA, AB, BA, BB$.

In this way, the measurement is a probabilistic process, where theoretically, the probability of measuring a given state $|m\rangle$ will be $|\lambda_m|^2$. Thus, after the quantum walk evolution, it is possible to observe from the final quantum state the quantity $|\lambda_m|^2$ for each state component $|m\rangle$.

Statistically, as we can see in Section 5, the chances of measuring the states that contain a solution were much more significant than measuring a state that did not contain a solution.

Table 1: Classical Backpropagation Simulations. The column lr presents the investigated learning rate, Epochs limit shows the experiment number that reached the maximum number of epochs, and Successful the number of experiments with zero classification error.

lr	Epochs limit	Successful
0.5000	None	1200
0.1000	None	1200
0.0100	452	748
0.0010	467	733
0.0001	726	474

4.8 Comparison against Backpropagation Algorithm

Since the search space contains solutions to the problem, the approach proposed in this paper can find a solution with high probability after the number of iterations defined in Equation 30. It implies that, in addition to ensuring a successful search, the number of iterations required to reach the artificial neural network training is known *a priori*.

The classical backpropagation algorithm seeks to minimize an error function based on the gradient descent methodology. For this reason, it can stagnate in local minimums. The practical implication is the possibility that the algorithm runs indefinitely and yet does not generate correct outputs. These limitations of a gradient descent methodology are overcome with the proposed quantum algorithm.

Furthermore, many experiments were done with both approaches to compare the proposed algorithm with the traditional backpropagation algorithm. The criterion to measure the performance was the number of iterations needed to find an ANN weight set that solves the XOR problem. The number of epochs for the backpropagation and the number of iterations for the proposed algorithm were observed.

Comparing the number of epochs of the backpropagation algorithm with the number of iterations of the procedure developed in this work was applied to measure how much better one approach is than the other. The efficiency of the proposed procedure is the order of $O(\sqrt{N/k})$. However, it was not possible to compare the efficiencies in terms of running time between our procedure and the backpropagation algorithm.

5 Experimental Results and Discussions

A set of experiments with classical backpropagation was done to define a comparative baseline. Employing the same ANN architecture $2-2-1$ (two inputs, two hidden neurons, and one output – according to Section 4.2), five different learning rates $\eta = \{0.5, 0.1, 0.01, 0.001, 0.0001\}$ were investigated, where 1200 simulations were computed for each one. The maximum number of epochs of 150000, the training stagnation, and the zero classification error were the stopping conditions employed. The network was considered in a stagnation training situation if its MSE error did not decrease by 1000 consecutive epochs. The classification error is zero when the ANN can classify all the four XOR inputs correctly.

For these conditions, none of the classical backpropagation experiments reached a stagnation situation. The experiments with learning rates of 0.5 and 0.1 never reached the maximum number of epochs, always reaching the perfect classification. The summarization of this experimental behavior can be viewed in Table 1, where *lr* indicates the learning rates, *Epochs limit* is the number of times that the experiment reached the maximum number of epochs stop condition, and *Successful* is the number of experiments that obtained zero classification error.

Table 2 presents the descriptive statistics for the baseline experiments. For each learning rate, it is presented the minimum, mean, maximum value of the Epochs number, and its standard deviation (Std.).

Table 3 presents the results of the simulations performed using the proposed computational procedure described in Algorithm 1. Here, k represents the number of solutions, N the number of vertices, l the number of self-loops (for all experiments $l = 1$), and t the number of iterations. $\Delta p = 0.5$ was utilized for all experiments. Here, it was investigated three configurations of N ($2^9 = 512$; $4^9 = 262144$; $8^9 = 134217728$) in five experiments. Two experiments with $N = 512$, 2 points per dimension. Two experiments with $N = 262144$, 4 points per dimension, and one experiment with $N = 134217728$, 8 points per dimension. The search spaces are sub-regions (windows) of an infinite ω-dimensional grid, as presented in Section 4.4. However, the initialization of these sub-regions can occur in different positions of the infinite grid. If there are no solutions in the current sub-region, the sub-region is moved to another region of the infinite grid. All simulations had their sub-region initiated randomly around the origin. The

Table 2: Descriptive statistics. Parameters of the backpropagation algorithm experiment for the number of epochs. lr means the learning rate. Std. means Standard Deviation.

Epoch Statistics				
lr	Minimum	Mean	Maximum	Std.
0.5000	1	33.60	319	35.68
0.1000	3	433.84	3279	463.78
0.0100	2	5277.48	132199	17927.67
0.0010	9	12949.18	148256	22451.79
0.0001	295	46987.00	149644	36780.22

Table 3: Weight search experiment by the quantum walk procedure. K means the number of solutions. N is the total number of vertices. The number of iterations is represented for letter t. For all experiments, the number of self-loops was $l = 1$.

Experiment	k	N	t	
			Theoretical	Simulated
1	12	512	10.26	11
2	12	512	10.26	11
3	17	262144	195.83	196
4	20	262144	179.83	180
5	80295	134217728	64.22	65

shift of the sub-region occurs until a region with at least one solution is found. With a few interactions, the sub-space converged to a region with solutions in all cases studied here.

Coincidentally, although the experiments 1 and 2 used distinct search windows, the proposed procedure converged to sub-spaces with 12 solutions (see Table 3). As $N = 512$ and $k = 12$ for those experiments, the iteration number t is also equal, being $t = 11$, which was theoretically defined to 10.26 by Equation 30. For the simulations 3 and 4, the proposed procedure converged to sub-spaces with 17 and 20 solutions, respectively. Thus, the iteration number t is different for those simulations, being 196 and 180, theoretically defined to 195.83 and 179.83, respectively. For the simulation 5, the procedure converged to a sub-space with more than eighty thousand solutions (80295), which implied in $t = 65$ iterations (theoretical number of iterations of 64.22).

The procedure developed using the quantum search algorithm, even in spaces with a high number of vertices, was able to amplify the amplitudes in a relatively low number of iterations compared with the results for the classical backpropagation procedure. In the best case, the maximum iteration number for the quantum walk algorithm was 11, while the mean number of epochs obtained by the backpropagation algorithm (in the best case) was 33.

It is also possible to see in Table 2 that the minimum number of epochs of backpropagation is less than the number of the interactions of the quantum walk procedure, see Table 3. Although it appears that the classical backpropagation algorithm has an advantage over the quantum walk procedure, it is essential to observe that the number of interactions t is known before start the search in the quantum walk algorithm, but it is not for the classical backpropagation. Thus, the critical measure to characterize the practical cost expectation is the mean number of epochs for the classical backpropagation. Prior knowledge of the amount of interaction required for the algorithm to converge is a great advantage of the quantum walk algorithm.

Therefore, observing the mean result of epochs required for network training by the backpropagation (Table 2) and the values of t in Table 3, the computational proposed procedure is more efficient on average. Depending on the size of the search space of the proposed algorithm, the network training using the backpropagation algorithm, in some cases, performs a smaller number of iterations. However, on average, the iterations number of the proposed algorithm is decidedly smaller than the backpropagation procedure. As seen in the backpropagation simulations, the random initialization of weights is one factor that influences the result of time convergence for the network. However, because of the impossibility of determination for an excellent region to initialize the weights, the mean value and standard deviation are the information statistically relevant for backpropagation experiments, where low performance and high variation in training epochs number results are obtained when compared with the proposed quantum algorithm.

Table 4: Measurement experiment — the percentage measure observed for each quantum. The states $|AA\rangle$ and $|AB\rangle$ are the solutions.

Experiment	$\|AA\rangle$	$\|AB\rangle$	$\|BA\rangle$	$\|BB\rangle$
1	95.48%	3.07%	1.40%	0.05%
2	95.03%	3.67%	1.26%	0.04%
3	**100.0%**	0.00%	0.00%	0.00%
4	**99.99%**	0.00%	0.01%	0.00%
5	**99.88%**	**0.10%**	0.02%	0.00%

Nevertheless, the proposed algorithm is quantum. Table 4 shows the measurement probabilities for the five simulations for neural network training. It was considered both states $|AA\rangle$ and $|AB\rangle$ are solutions, given that for these states, the walker is in a vertex a. At least 98.55% of the measurements find a solution (experiment 1), reaching 100% for the experiment 3. At mean, the proposed quantum procedure find a solution in 99.44% of the measurements.

6 Conclusion

Training based on the backpropagation algorithm (or descendant gradient algorithms) may fall to local minimums. Many factors can influence this result and carry training for many epochs until the network converges to a solution or even stagnates. Therefore, it is impossible to say whether the training will stop or that good accuracy can be guaranteed in a prior way.

The non-stagnating in local minimums and the knowing in advance the number of iterations required to obtain a solution are some advantages of the proposed procedure. It is not possible to guarantee that the solution obtained is optimal because the probability is equal between all weight set solutions within the state $|AA\rangle$ and the state $|AB\rangle$. However, with a high probability, there will have a valid solution at the end of the procedure. So, the procedure proposed here guarantees the neural network training with high probability. However, the proposed methodology is a theoretical proposal. In practice, the proposed algorithm needs a quantum computer, which does not exist yet.

Another critical point is the oracle. In a quantum system, an oracle is an operator that can answer if a given state is or is not marked, *i.e.*, if a given state is or is not a solution. For the proposed methodology, an oracle would be an operator capable of determining whether a given state would train a network or not. Here, an oracle was simulated by a simple matrix of zeros and ones. Each position in the oracle matrix is a possible state in the quantum system. If the value is 1, then the state is a solution. Otherwise, the state is not a solution. The creation of a real quantum oracle operator is a future research.

As seen before, the execution time of the proposed algorithm is of the order of $O(\sqrt{N/k})$, *i.e.*, there a quadratic gain when compared with the classical analog. In this case, the search problem grows exponentially, the window's search space is $\mathbf{O}(P^{\omega})$, where $P = \sqrt{(N)}$, N is the number of points in the search window (a squared window), and ω is the number of ANN weights. The number of qubits required to represent ω synaptic weights is equal to $\omega * \log_2(N)$. The proof of correctness of the quantum walk algorithm can be found in [Feng et al., 2007].

Finally, classical simulations showed a significant gain in training an artificial neural network using the procedure that applies a quantum walk to find the set of weights compared with the use of the backpropagation algorithm.

Acknowledgments

Acknowledgments to the Science and Technology Support Foundation of Pernambuco (FACEPE) Brazil, Brazilian National Council for Scientific and Technological Development (CNPq), and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 by their financial support to the development of this research.

References

James R Powell. The quantum limit to moore's law. *Proceedings of the IEEE*, 96(8):1247–1248, 2008. doi:10.1109/JPROC.2008.925411.

Jasmeet Singh and Mohit Singh. Evolution in quantum computing. In *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*, pages 267–270. IEEE, 2016. doi:10.1109/SYSMART.2016.7894533.

Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *Quantum Physics*, 2013.

Noson S Yanofsky, Mirco A Mannucci, and Mirco A Mannucci. *Quantum computing for computer scientists*, volume 20. Cambridge University Press, Cambridge, UK, 2008.

Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, Philadelphia, PA, 1996. ACM. doi:10.1145/237814.237866.

Peter Wittek. *Quantum machine learning: what quantum computing means to data mining*. Academic Press, 2014. doi:10.1016/C2013-0-19170-2.

Vedran Dunjko, Jacob M Taylor, and Hans J Briegel. Quantum-enhanced machine learning. *Physical review letters*, 117(13):130501, 2016. doi:10.1103/PhysRevLett.117.130501.

Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Quantum speed-up for unsupervised learning. *Machine Learning*, 90(2):261–287, 2013. doi:10.1007/s10994-012-5316-5.

Yu Zheng, Sicong Lu, and Re-Bing Wu. Quantum circuit design for training perceptron models. *arXiv preprint arXiv:1802.05428*, 2018.

Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. Simulating a perceptron on a quantum computer. *Physics Letters A*, 379(7):660–663, 2015. doi:10.1016/j.physleta.2014.11.061.

Prasanna Date, Catherine Schuman, Robert Patton, and Thomas Potok. A classical-quantum hybrid approach for unsupervised probabilistic machine learning. In *Future of Information and Communication Conference*, pages 98–117. Springer, Cham, 2019.

Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Machine learning in a quantum world. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 431–442. Springer, 2006. doi:10.1007/11766247_37.

Alberto Prieto, Beatriz Prieto, Eva Martinez Ortigosa, Eduardo Ros, Francisco Pelayo, Julio Ortega, and Ignacio Rojas. Neural networks: An overview of early research, current frameworks and new challenges. *Neurocomputing*, 214: 242–268, 2016. doi:10.1016/j.neucom.2016.06.014.

Simon Haykin. Neural networks: principles and practice. *Bookman*, 11:900, 2001.

Varun Kumar Ojha, Ajith Abraham, and Václav Snášel. Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, 60:97–116, 2017. doi:10.1016/j.engappai.2017.01.013.

Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195, 2017. doi:10.1038/nature23474.

Thomas G Wong. Grover search with lackadaisical quantum walks. *Journal of Physics A: Mathematical and Theoretical*, 48(43):435304, 2015. doi:10.1088/1751-8113/48/43/435304.

Nikolajs Nahimovs and Alexander Rivosh. Exceptional configurations of quantum walks with grover's coin. In *International Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, pages 79–92. Springer, 2015. doi:10.1007/978-3-319-29817-7_8.

Thomas G Wong. Faster search by lackadaisical quantum walk. *Quantum Information Processing*, 17(3):68, 2018. doi:10.1007/s11128-018-1840-y.

Neil B Lovett, Matthew Everitt, Robert M Heath, and Viv Kendon. The quantum walk search algorithm: Factors affecting efficiency. *Mathematical Structures in Computer Science*, 29(3):389–429, 2019. doi:10.1017/S0960129518000051.

Nikolajs Nahimovs. Lackadaisical quantum walks with multiple marked vertices. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 368–378. Springer, 2019. doi:10.1007/978-3-030-10801-4_29.

Luciano S. Souza, Jonathan H. A. Carvalho, and Tiago A. E. Ferreira. Quantum walk to train a classical artificial neural network. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 836–841. IEEE, 2019. doi:10.1109/BRACIS.2019.00149.

Renato Portugal. *Quantum walks and search algorithms*. Springer Science & Business Media, Switzerland, AG, 2013. doi:10.1007/978-1-4614-6336-8.

Salvador Elías Venegas-Andraca. Quantum walks: a comprehensive review. *Quantum Information Processing*, 11(5): 1015–1106, 2012.

Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. AAPT, Cambridge, UK, 2002.

Neil Shenvi, Julia Kempe, and K Birgitta Whaley. Quantum random-walk search algorithm. *Physical Review A*, 67 (5):052307, 2003.

Yuan Feng, Runyao Duan, Zhengfeng Ji, and Mingsheng Ying. Proof rules for the correctness of quantum programs. *Theoretical Computer Science*, 386(1-2):151–166, 2007.

Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *arXiv preprint quant-ph/0005055*, 2000.

Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik: Progress of Physics*, 46(4-5):493–505, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015. doi:10.1038/nature14539.

Ivan Nunes Da Silva, Danilo Hernane Spatti, Rogerio Andrade Flauzino, Luisa Helena Bartocci Liboni, and Silas Franco dos Reis Alves. *Artificial neural networks: A Practical Course*. Springer, Switzerland, AG, 2017. doi:10.1007/978-3-319-43162-8.

Yoshitaka Sasaki and Mikio Nakahara. *Quantum Information and Quantum Computing (Kinki University Series on Quantum Computing)*. World Scientific, 2013. doi:10.1119/1.1463744.

Ashish Kapoor, Nathan Wiebe, and Krysta Svore. Quantum perceptron models. In *Advances in Neural Information Processing Systems*, pages 3999–4007, 2016.

Lackadaisical Quantum Walk in the Hypercube to Search for Multiple Marked Vertices

Article published in Brazilian Conference on Intelligent Systems

DOI: 10.1007/978-3-030-91702-9_17 A preprint can be found at:

https://arxiv.org/pdf/2108.09399

LACKADAISICAL QUANTUM WALK IN THE HYPERCUBE TO SEARCH FOR MULTIPLE MARKED VERTICES

A PREPRINT

Luciano S. de Souza*
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
Recife, Brasil
luciano.serafim@ufrpe.br

Jonathan H. A. de Carvalho
Centro de Informática
Universidade Federal de Pernambuco
Recife, Brasil
jhac@cin.ufpe.br

Tiago A. E. Ferreira
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
Recife, Brasil
tiago.espinola@ufrpe.br

November 18, 2024

ABSTRACT

Adding self-loops at each vertex of a graph improves the performance of quantum walks algorithms over loopless algorithms. Many works approach quantum walks to search for a single marked vertex. In this article, we experimentally address several problems related to quantum walk in the hypercube with self-loops to search for multiple marked vertices. We first investigate the quantum walk in the loopless hypercube. We saw that neighbor vertices are also amplified and that approximately $1/2$ of the system energy is concentrated in them. We show that the optimal value of l for a single marked vertex is not optimal for multiple marked vertices. We define a new value of $l = (n/N) \cdot k$ to search multiple marked vertices. Next, we use this new value of l found to analyze the search for multiple marked vertices non-adjacent and show that the probability of success is close to 1. We also use the new value of l found to analyze the search for several marked vertices that are adjacent and show that the probability of success is directly proportional to the density of marked vertices in the neighborhood. We also show that, in the case where neighbors are marked, if there is at least one non-adjacent marked vertex, the probability of success increases to close to 1. The results found show that the self-loop value for the quantum walk in the hypercube to search for several marked vertices is $l = (n/N) \cdot k$.

Keywords Quantum Computing · Quantum Walk · Quantum Search Algorithm.

1 Introduction

According to Shenvi et al. [2003], quantum walks provide one of the most promising features, an intuitive framework for building new quantum algorithms. They were pioneers in designing a quantum search algorithm on the hypercube based on quantum random walks [Potoček et al., 2009]. Recent works have used the quantum walks to search weights and train artificial neural networks [Souza et al., 2019, 2021].

The topology of the structure where the walk is applied considerably affects the evolution of the walker [Wang et al., 2017]. Therefore, many works are developed to improve the performance of quantum walks, quantum search algorithms

*95, R. Manuel de Medeiros, 35 - Dois Irmãos, Recife - PE

in different structures: one-dimensional, two-dimensional, and multidimensional grids, complete and bipartite graphs, among others [Bezerra et al., 2021, Carvalho et al., 2020, Nahimovs et al., 2019, Rhodes and Wong, 2019].

Quantum walk modification proposals are also made to improve their performance. For example, Wong [2018] added to each vertex of a two-dimensional grid a self-loop, so the walker has some probability of staying put, achieving an improvement over the algorithm without self-loop [Ambainis et al., 2004].

Rhodes and Wong [2020] proposed an ideal weight for all vertex-transitive graphs with a single marked vertex such that the ideal self-loop weight is equal to the degree of the loopless graph divided by the total number of vertices. Potoček et al. [2009] observed that the nearest neighbors are also presented with high probability and Nahimovs [2019] that adjacent vertices can be hard to find by quantum walks.

In this way, we investigate whether the optimal value of $l = (d/N)$ for a single marked vertex is optimal for multiple marked vertices, where d is the degree of the loopless vertex and N is the number of vertices. We analyzed the quantum walk on hypercube without self-loop and with self-loop. We analyzed the quantum walk on the hypercube for multiple marked adjacent and non-adjacent vertices. Finally, we find an optimal value of l for a quantum walk in the hypercube with multiple marked vertices.

This paper is organized as follows. In Section 2, we present some concepts about quantum walks and specifically the quantum walk on the hypercube. In Section 3, we characterize the probability distribution along with the space, adjust the self-loop weight for multiple marked vertices, and search for adjacent marked vertices. Finally, in Section 4 is the conclusion.

2 Quantum Walk

The processing of quantum information is governed by quantum mechanics or quantum physics [Singh and Singh, 2016]. Quantum computing study the processing of this information [Nielsen and Chuang, 2002, Yanofsky and Mannucci, 2008, McMahon, 2007]. Quantum walks are the quantum counterpart of classical random walks. Discrete and continuous-time quantum walks are the advanced tools used to build quantum algorithms [Aharonov et al., 1993, Ambainis et al., 2012]. The main feature that differentiates these two types of quantum walks is the timing used to applying the evolution operators. In the quantum walk in continuous time, the evolution operator is applied at any time, whereas the quantum walks in discrete time, the evolution operator is applied in discrete time steps [Venegas-Andraca, 2012]. The quantum walk evolution in the discrete-time process occurs by the successive applications of a unitary evolution operator U that acts on the Hilbert space

$$\mathcal{H} = \mathcal{H}^C \otimes \mathcal{H}^S.$$

The coin space $\mathcal{H}^C$ is the Hilbert space associated with a quantum coin, and the walker's space $\mathcal{H}^S$ is the Hilbert space associated with the position of the nodes in a graph, for example. The evolution operator U is defined in Equation 1.

$$U = S(C \otimes I_N) \tag{1}$$

where, S is the shift operator, i.e., a permutation matrix that acts in the walker's space based on the state of the coin space. The unitary matrix C is the coin operator [Shenvi et al., 2003]. Therefore, the equation of evolution represented by a quantum walk at time t is given by

$$|\Psi(t)\rangle = U^t |\Psi(0)\rangle$$

.

2.1 Quantum walk on the hypercube

According to Venegas-Andraca [2012], the hypercube is defined as an undirected graph of degree n and $N = 2^n$ nodes. Each node is represented by an n-bit binary string. Two nodes $\vec{x}$ and $\vec{y}$ are connected by an edge if the Hamming distance between them is 1, i.e., $|\vec{x} - \vec{y}| = 1$. This means that $\vec{x}$ and $\vec{y}$ only differ in a single bit. The expression $|\vec{x}|$ is the Hamming weight of $\vec{x}$. The Hilbert space associated with the quantum walk on the hypercube is

$$\mathcal{H} = \mathcal{H}^n \otimes \mathcal{H}^{2^n},$$

where $\mathcal{H}^n$ is the Hilbert space associated with the quantum coin space, and $\mathcal{H}^{2^n}$ is the Hilbert space associated with nodes on the hypercube.

According to Shenvi et al. [2003], in a d-dimensional hypercube, the d directions specify the coin state. Kempe [2002] defines that directions can be labeled by the n base-vectors $\{|0\rangle, |1\rangle, \ldots, |n-1\rangle\}$ on the hypercube which corresponding to the n vectors of Hamming weight 1. These n vectors are represented by the states $\{|e_0\rangle, |e_1\rangle, \ldots, |e_{n-1}\rangle\}$, where e_d has a 1 in the d-th bit. The shift operator S described in Equation 2 acts mapping a state $|d, \vec{x}\rangle \rightarrow |d, \vec{x} \oplus \vec{e_d}\rangle$.

$$S = \sum_{d=0}^{n-1} \sum_{\vec{x}} |d, \vec{x} \oplus \vec{e_d}\rangle \langle d, \vec{x}| \tag{2}$$

The initial state of the quantum walk in the hypercube is defined according to Equation 3 as an equal superposition over all N nodes and n directions.

$$|\Psi(0)\rangle = \frac{1}{\sqrt{n}} \sum_{d=0}^{n-1} |d\rangle \otimes \frac{1}{\sqrt{N}} \sum_{\vec{x}} |\vec{x}\rangle \tag{3}$$

According to Rhodes and Wong [2020], the hypercube was the first graph in which quantum walks were researched. In their work, Shenvi et al. [2003] presented a quantum search algorithm based on the random walk quantum architecture. In this article, we are based on the approach used by Wong [2018]. The pure quantum walk (without search) evolves by repeated applications from the evolution operator described in Equation 1, where C is Grover's "diffusion" operator on the coin space and is given by

$$C = 2 |s^C\rangle \langle s^C| - I_n \tag{4}$$

where, I_n is the identity operator, n is the vertex degree loopless, and $|s^C\rangle$ is the equal superposition over all n directions [Moore and Russell, 2002, Shenvi et al., 2003], i.e.,

$$|s^C\rangle = \frac{1}{\sqrt{n}} \sum_{d=0}^{n-1} |d\rangle . \tag{5}$$

We include a query to the "Grover oracle", described in Equation 6, at each step of the quantum walk.

$$U' = U \cdot (I_n \otimes Q) \tag{6}$$

where, $Q = I_N - 2 |\omega\rangle \langle\omega|$, and $|\omega\rangle$ means the marked vertex. The system is initiated according to the initial state presented in Equation 3.

3 Analyzing the quantum walk on the hypercube

In this section, we experimentally analyze the quantum walk on the hypercube searching for multiple marked vertices. The simulations and the obtained results are detailed in the following subsections.

3.1 Characterizing the probability distribution along the space

Previous works showed there is an amplification in the solution neighborhood, which interferes with the amplification of the solutions by the quantum walk on the hypercube [Shenvi et al., 2003, Potoček et al., 2009, Nahimovs et al., 2019]. Initially, it is necessary to understand how the probability amplitudes are distributed in the search space and how the quantum walk evolves in the hypercube over time considering the impacts caused by the solution neighborhood.

Figure 1 shows the probability of success after one hundred steps for the quantum walk in the hypercube with one, two, three, and four arbitrarily marked vertices. Although the search algorithm is able to amplify the probability amplitudes of the marked vertices, if a measurement is performed, the probability of finding one of the solutions is still unsatisfactory. Another interesting aspect that can be observed is that as the number of marked vertices increases, the speed of amplification the probability amplitudes also increases. However, it is necessary to increase the probability amplitudes of the marked vertices.

Figure 2 shows the probability distributions of the marked vertices only after the number of iterations necessary to reach the maximum value of the probability amplitude close to $1/2$. As Potoček et al. [2009] noted in their work, we also note that the set of neighbors have a high probability. If we add the amplitudes of the neighbor's vertices, the values

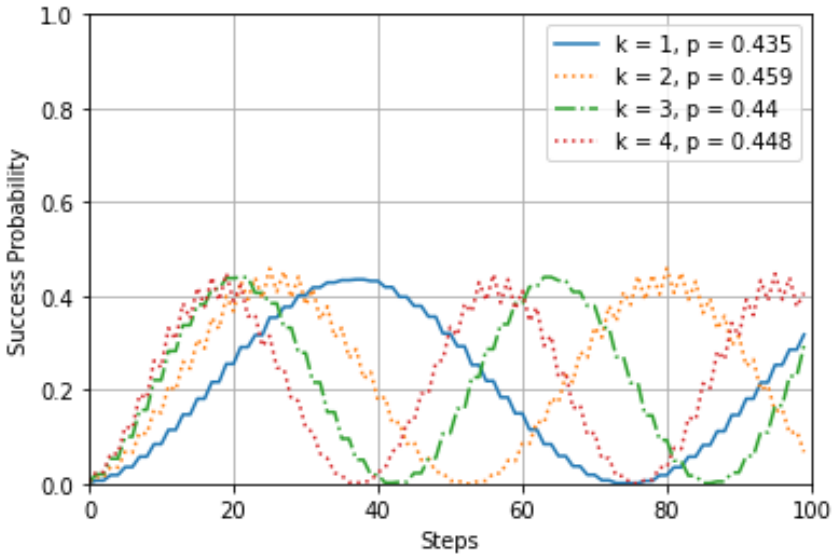


Figure 1: Success probability after 100 steps in a hypercube with 1024 nodes. The solid blue curve is the success probability for one solution. The dotted orange curve is the success probability for two solutions. The dot-dashed green curve is the success probability for three solutions. The dotted red curve is the success probability for four solutions.

are compatible with the amplitude value of the marked vertex. We conclude that a considerable part of the energy, approximately $1/2$, is retained in the neighbors of the marked vertices. Figure 2d, shows the probability distribution of four marked vertices. Note that the amplitudes of each vertex have their maximum and a neighborhood region. The x-axis distribution is the relative position of the position on the hypercube. It explains why even increasing the number of marked vertices, the success probabilities do not reach values above $1/2$.

Figure 3 shows the success probability for the quantum walk with one and four marked vertices after one hundred steps. Figure 3a shows the behavior of the success probability of one marked vertex, the solid blue curve, and its neighbors, which is the dotted orange curve. If a measurement is performed, the probability of getting a neighbor vertex is greater than getting a marked vertex. With probability above 90%, you get the solution or a vertex that is one step away from the solution. Figure 3b shows the behavior of the success probability of four marked vertices, the solid blue curve, and their neighbors, the dotted orange curve. Note that in a step when the probability of success of the marked vertices is high, the probability of success of the neighbors decreases, and in the next step, when the probability of success of the neighbors is high, the probability of success of the marked vertices decreases. Because of this behavior, if a measurement is performed, the probability of getting a neighbor is high. This happens in Figure 3a but more smoothly.

Observing these results, we must consider the probability p of obtaining a marked vertex and the probability $p' = (1-p)$ of obtaining an unmarked vertex which is the sum of the probabilities of the $(N-k)$ vertices, where k is the number of marked vertices. These results are shown in Table 1. Note the column of the value of p', which is composed of the value of the amplitudes of the neighbors and the amplitude of the vertices that are neither neighbors nor marked. The probability of the walker finding a region is high because the energy is concentrated in the neighboring region. It is concluded that the amplification of the neighborhood around the marked vertices interferes with the probability of success of finding a target vertex.

3.2 Adjusting the self-loop weight for multiple marked vertices

Many works have been proposed with the purpose of improving the search capacity of quantum algorithms. According to Wong [2015], adding a self-loop to each vertex boosts the success probability from $1/2$ to 1. A modification to the initial state in the Equation 3 and to Grover's coin in the Equation 4 is needed so that the self-loop can be added. The

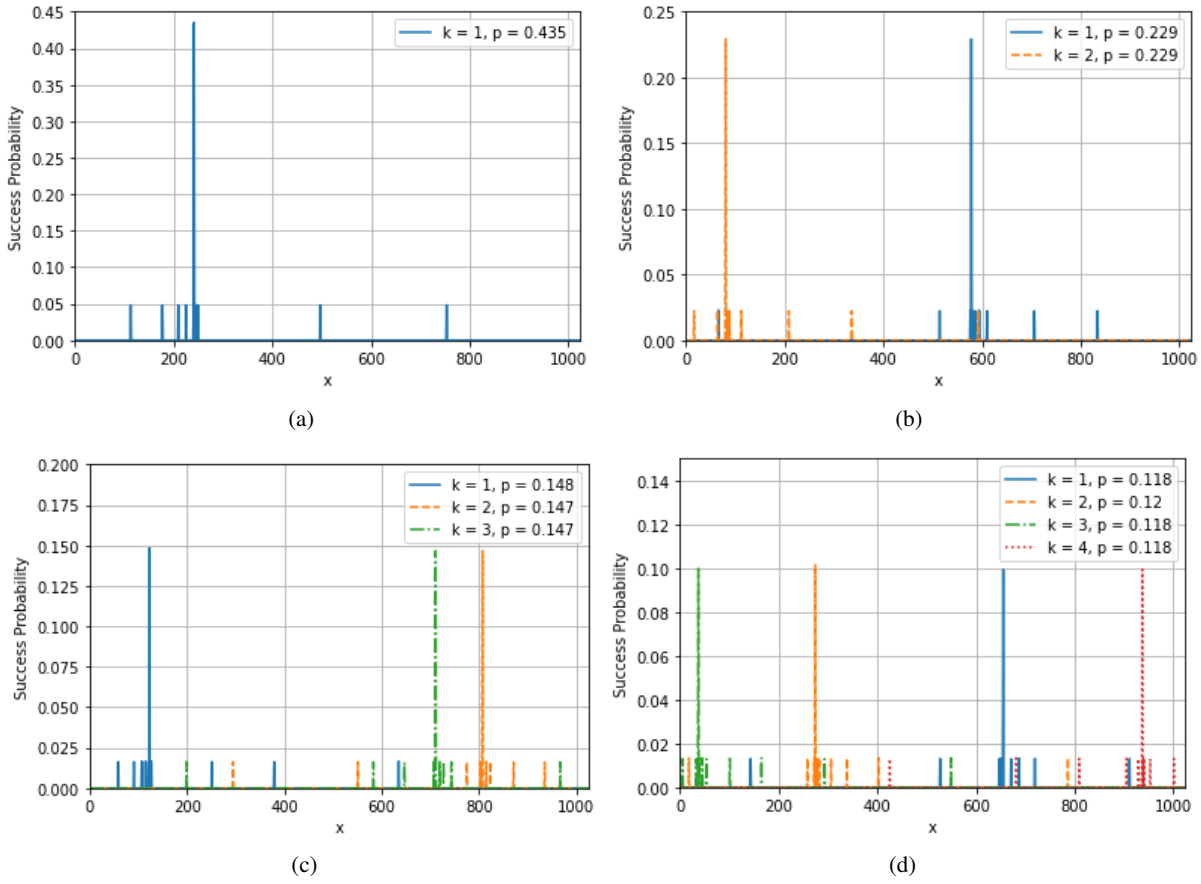
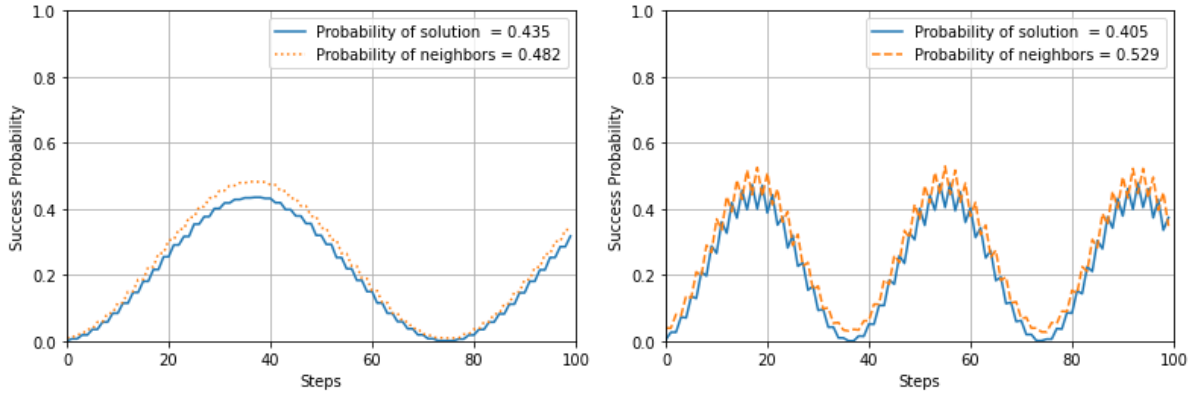


Figure 2: Probability distribution of the quantum walk after the number of iterations necessary to reach the maximum value of the probability amplitude with $n = 10$ and $N = 1024$ vertices. The y-axis values are at different ranges to improve visualization. (a) solid blue bar show the probability distribution for one marked vertex. (b) solid blue bar and orange dashed bar show the probability distribution for two marked vertices. (c) solid blue bar, orange dashed bar and green dash-dot bar show the probability distribution for three marked vertices. (d) solid blue bar, orange dashed bar, green dash-dot bar and dotted red bar show the probability distribution for four marked vertices.

Table 1: Probabilities of success of marked and unmarked vertices.

Probabilities of success			
Figure	p	$p' = (1-p)$	
		Neighbors	Neither
2a	43.5%	48.2%	8.3%
2b	45.8%	45.5%	8.7%
2c	44.2%	48.4%	7.4%
2d	47.4%	44.5%	8.1%
3a	43.5%	48.2%	8.3%
3b	40.5%	52.9%	6.6%



(a) (b)

Figure 3: Probability of success after 100 steps with $n = 10$ and $N = 1024$ vertices. (a) shows the probability of success for one marked vertex and its neighbors. (b) shows the probability of success for four marked vertices and their neighbors.

addition of the self-loop is described in Equation 7. Thus, the coin space is now an $(n + 1)$-dimensional space [Rhodes and Wong, 2020].

$$|s^C\rangle = \frac{1}{\sqrt{n+l}} \left(\sqrt{l}\,|\circlearrowleft\rangle + \sum_{d=0}^{n-1} |d\rangle \right) \tag{7}$$

One of the concerns when adding a self-loop at each vertex is knowing the best self-loop value. More specifically, in the case of the quantum walk on the hypercube, Rhodes and Wong [2020] proposed an optimal self-loop value

$$l = \frac{d}{N}, \tag{8}$$

where d is equal to the degree of the loopless graph and N is the number of vertices in the hypercube. Recently, two works showed that inserting the number of marked vertices in calculating the self-loop value optimizes quantum walks. Carvalho et al. [2021] shows that the optimal value of the self-loop for quantum walks in D-dimensional grids with multiple marked vertices is

$$l = \frac{2Dm}{N},$$

where $2D$ is the number of movements the walker can do, not counting the self-loop, m the number of marked vertices, and N the number of vertices of the grid. Nahimovs and Santos [2021] shows that for different types of two-dimensional grids - triangular, rectangular, and honeycomb the optimal self-loop value is also,

$$l = \frac{m \cdot d}{N}$$

where d is the degree of the vertex, m is the number of marked vertices, and N is the number of vertices of the grid.

Figure 4a shows the probability of success after two hundred steps for one marked vertex. Here, the values of l were the same as used by Rhodes. The dashed red curve has the optimum value of l. Our interest was to investigate whether the value of l described in Equation 8 also improved the walk results for a number ($k > 1$) of marked vertices. For this, we performed three more experiments where we increased the number of marked vertices up to four. As we added the marked vertices the success probability of the dashed red curve decreased to 88.7% (4b) while the success probability of the dotted purple curve increased to 99.8% (4b) but then also decreased to 96.2% (4c) and 89.3% (4d). It indicates that a new value of l is required when the number of marked vertices increases. To find the optimal self-loop for multiple marked vertices, we defined a set of values in the form $l' = (\alpha \cdot l)$, where $\alpha \in \mathbb{N}$.

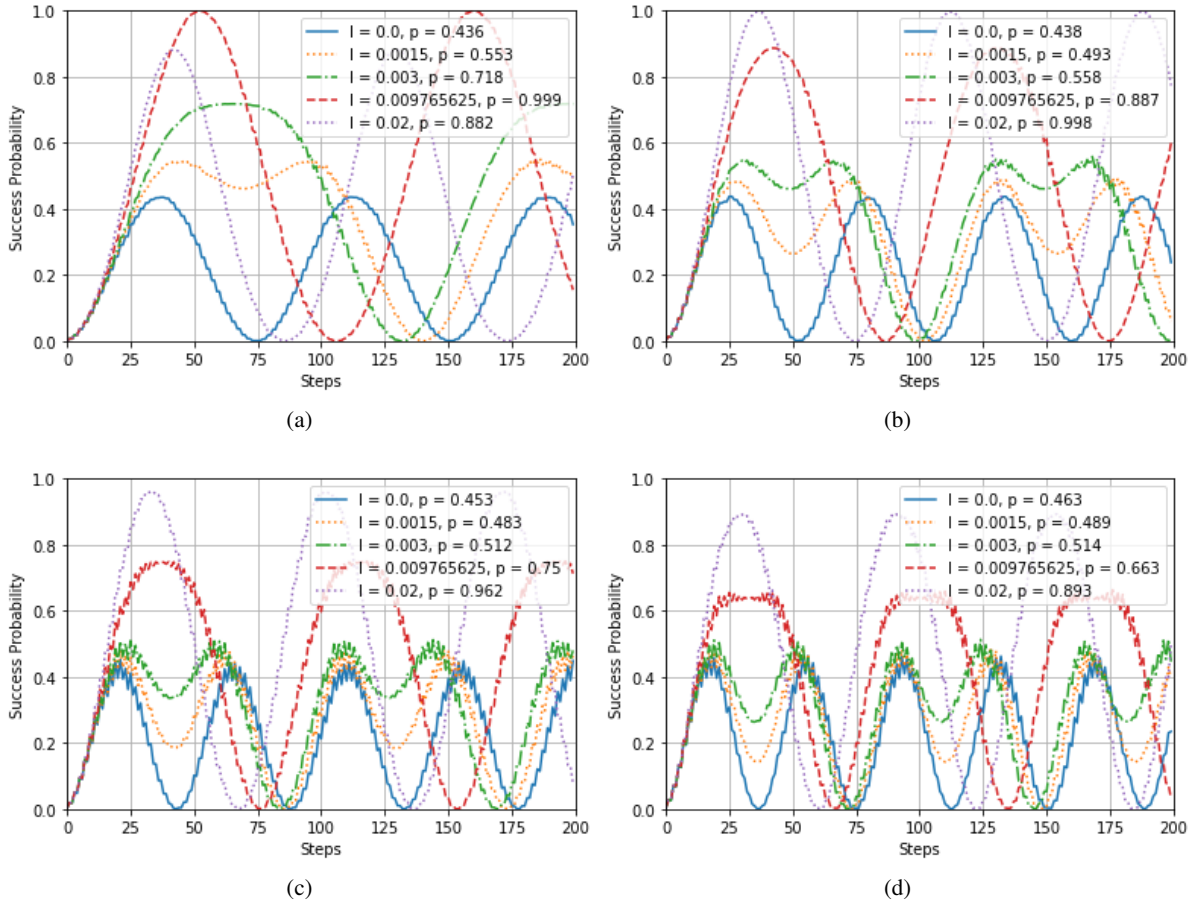


Figure 4: Comparison between multiple self-loops values and $l = (n/N)$. (a) shows the success probability for one marked vertex. (b) shows the success probability for two marked vertices. (c) shows the success probability for three marked vertices. (d) shows the success probability for four marked vertices.

Figure 5 compares the probability of success for a set of marked vertices, $k = \{2, 3, 5, 14, 17\}$, these vertices were chosen randomly as well as their number. The self-loop values for these vertex numbers are $\alpha \cdot l$, where $l = (d/N)$ and $\alpha = \{1, 2, 3, ...\}$. Note that the curves have their maximum points exactly at the locations on the x-axis where the l' values are. We can conclude that the value of $(\alpha = k)$. Therefore, we can set the value of l for multiple marked vertices for the quantum walk in the hypercube,

$$l' = \frac{n}{N} \cdot k \tag{9}$$

where n is equal to the degree of the loopless vertex of the hypercube, N the number of vertices in the hypercube, and k the number of marked vertices. The self-loop value shown by Nahimovs and Santos [2021] for the quantum search in various types of two-dimensional grids coincides with the optimal self-loop value for the search for a quantum walk in the hypercube.

Figure 5 shows that, as the values of l approach the optimal value, the probability of success of the curve also approaches its maximum value. We can observe this behavior in Table 2 which shows the probability of success for multiple values of l and multiple marked vertices. Consider the values of the main diagonal, which are the maximum success probabilities for each $l = (n/N) \cdot k$.

Table 2 shows the relationship between the self-loop value and the number of marked vertices. We observe the relationship between the self-loop value and the number of marked vertices. Note that when the values of l approach the optimal values for each number of marked vertices, there is an improvement in the probability amplitude. Figure 6

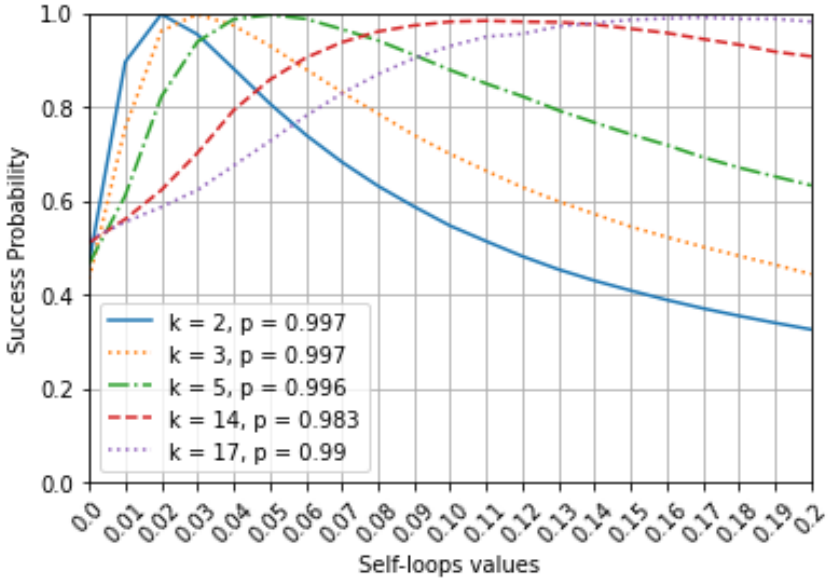


Figure 5: Investigation to set the value of l for multiple marked vertices.

Table 2: Probability of success for multiple values of l.

$l = (n/N) \cdot k$	Number of marked vertices									
	1	2	3	4	5	6	7	8	9	10
(n/N)*1	**0.999**	0.888	0.75	0.663	0.775	0.592	0.575	0.576	0.589	0.55
(n/N)*2	0.888	**0.998**	0.958	0.886	0.815	0.9	0.705	0.672	0.639	0.624
(n/N)*3	0.749	0.959	**0.998**	0.976	0.934	0.886	0.941	0.792	0.857	0.727
(n/N)*4	0.64	0.888	0.978	**0.998**	0.975	0.954	0.922	0.885	0.847	0.813
(n/N)*5	0.555	0.816	0.937	0.986	**0.996**	0.989	0.966	0.943	0.912	0.883
(n/N)*6	0.49	0.75	0.888	0.958	0.989	**0.996**	0.991	0.973	0.953	0.928
(n/N)*7	0.438	0.691	0.84	0.926	0.969	0.992	**0.996**	0.99	0.978	0.983
(n/N)*8	0.395	0.641	0.794	0.888	0.944	0.895	0.991	**0.993**	0.99	0.988
(n/N)*9	0.361	0.596	0.75	0.852	0.915	0.957	0.978	0.993	**0.994**	0.982
(n/N)*10	0.331	0.554	0.711	0.816	0.888	0.935	0.966	0.982	0.99	**0.996**

shows the probability of success after two hundred steps for multiple marked vertices. We can conclude that for cases where there is more than one marked vertex, the optimal value of $l = (n/N) \cdot k$.

3.3 Searching for adjacent marked vertices

The results found in the previous sections refer to the search for non-adjacent marked vertices, i.e., $|\vec{\omega_i} - \vec{\omega_j}| \neq 1$ the Hamming distance from vertex $\vec{\omega_i}$ and all other marked vertices is different from 1. Nahimovs et al. [2019] shows in their work that for quantum walks in the hypercube if the search space contains marked neighbors vertices, the search can be drastically affected. The authors considered two sets, one with two adjacent marked vertices and the other with two non-adjacent marked vertices. In the first case, the two adjacent marked vertices are $M = \{0, 1\}$. The absolute value of the overlap remained close to 1, and the probability remains close to the initial state probability. In the second case, the two non-adjacent marked vertices are $M = \{0, 3\}$. The behavior on this one is different, the same behavior as the solid blue curve in Figure 3a.

As the addition of self-loop in the quantum walk in the hypercube improved the search for multiple non-adjacent marked vertices, we investigated the case where the marked vertices are adjacent. We consider ten sets of vertices, $M = [\{0, 1\}, \{0, 1, 2\}, \cdots, \{0, 1, 2, 4, 8, \cdots, 256, 512\}]$, i.e., all vertices adjacent to the vertex 0. We add one more vertex to the set of marked vertices on each new walk until the number of vertices in M is equal to the degree n of the vertex.

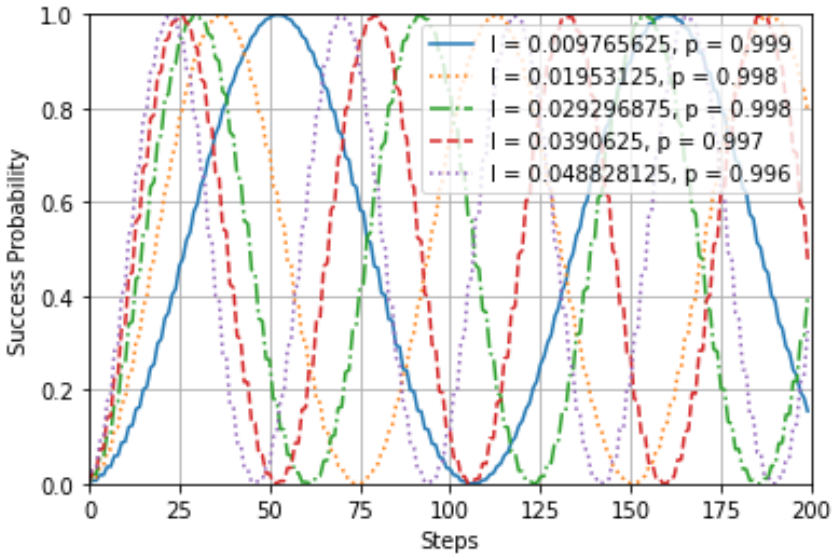
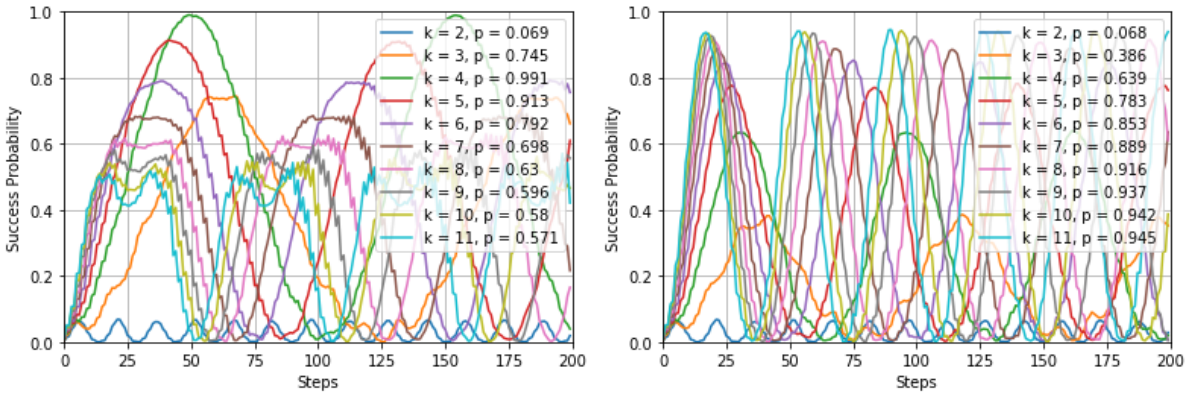


Figure 6: Probability of success after 200 steps. Solid blue curve, $k = 1$. Dotted orange curve, $k = 2$. Green dash-dot curve, $k = 3$. Red dashed curve, $k = 4$. Dotted purple curve, $k = 5$



(a) (b)

Figure 7: Probability of success after 200 steps with $n = 10$ and $N = 1024$ vertices. Shows the probability of success for k adjacent marked vertices. (a) shows for $l = (n/N)$ and (b) for $l = (n/N) \cdot k$.

Figure 7 shows the probability of success after two hundred steps. Figure 7a shows the result for the value of $l = (n/N)$. The probability reaches its maximum when the number of vertices reaches $k = 4$ with a probability of success of 99.1%. Then the probability starts to decrease as k increases. Figure 7b shows the result for the value of $l = (n/N) \cdot k$. The probability reaches its maximum when the number of vertices reaches $k = 11$ with a success probability of 94.5%. Although the probability increases with a slower speed when $k = 5$, it already reaches 78.3%. This behavior is interesting for search spaces where the marked vertex density is high. Note the probability of the solid cyan curve. This behavior was found in work done by Nahimovs et al. [2019] and was repeated here in our experiments. According to the authors, this is because the quantum walk has a stationary state.

Figure 8 shows the comparison between what happens to the success probabilities in Figure 7 when the number of k increases. Note the dotted orange curve, the probability of success grows to its maximum value when the value of $l = (n/N) \cdot k$. The same does not happen when $l = (n/N)$.

We considered before that the marked vertices were neighbors. Now, let us analyze the possibility that in addition to having marked vertices in the neighborhood, there are also marked vertices that are not neighbors. We run ten

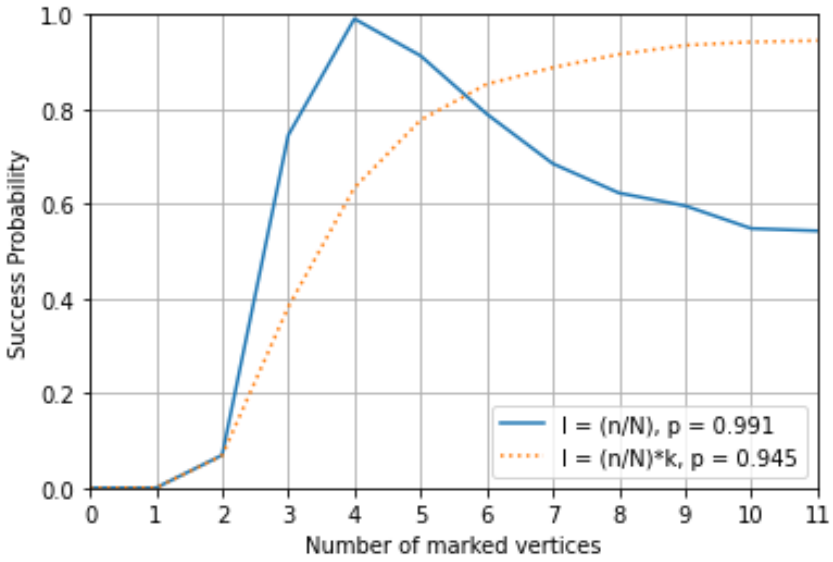
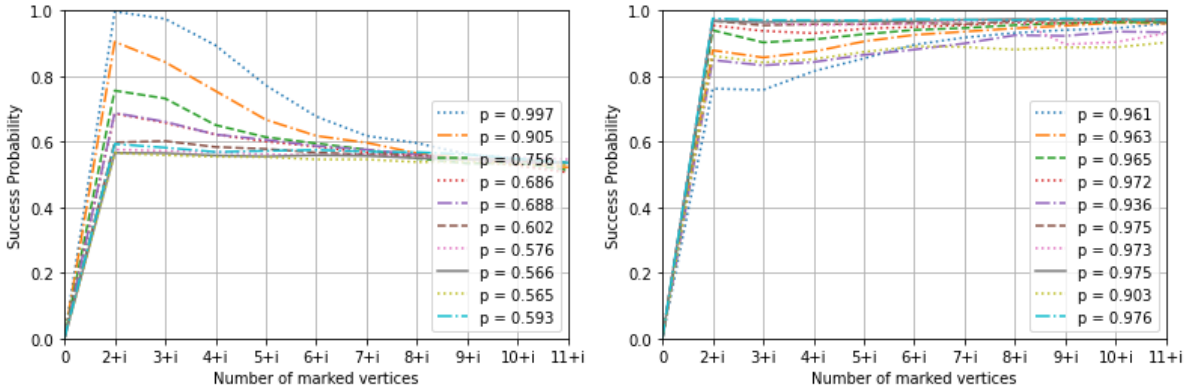


Figure 8: Maximum probability reached for each number of marked vertices in the neighborhood after one hundred steps with $n = 10$ and $N = 1024$ vertices. Evaluating the interference of the number of adjacent marked vertices in the value of l.

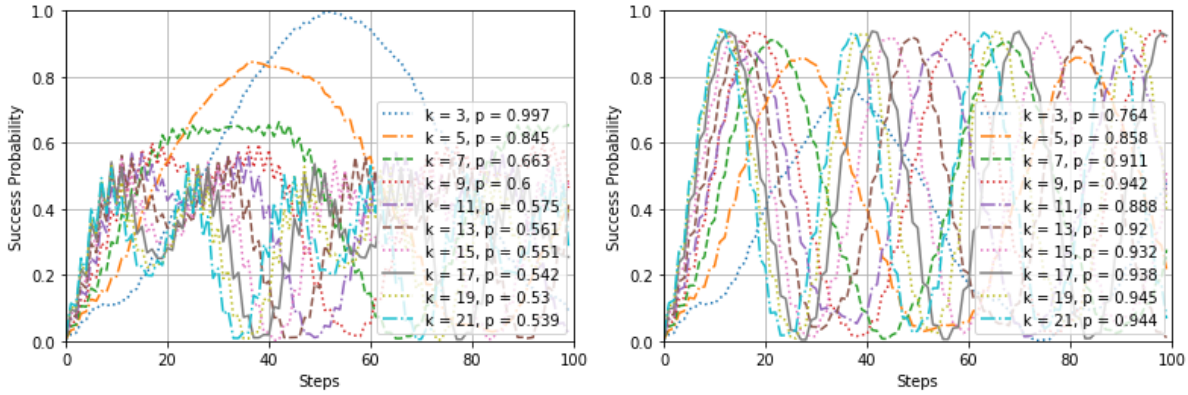


(a) (b)

Figure 9: Maximum probability reached for each number of marked vertices after one hundred steps with $n = 10$ and $N = 1024$ vertices. (a) shows the probability of success for k adjacent and non-adjacent marked vertices for $l = (n/N)$. (b) shows the probability of success for k adjacent and non-adjacent marked vertices for $l = (n/N) \cdot k$.

experiments, and each one starts with two adjacent marked vertices $M = \{0, 1\}$. In each experiment, a $i = \{1, 2, 3, \cdots\}$ non-adjacent vertex is randomly marked and the next marked neighbor, i.e., $M = \{0, 1, 2, ...\}$. Therefore, in the tenth experiment, there will be eleven adjacent and ten non-adjacent vertices.

Figure 9 shows the behavior of probability amplitudes when for each set of adjacent vertices, a number of non-adjacent vertices are marked. Figure 9a shows that as new non-adjacent vertices are marked the probability is affected. Note that the behavior seen in the solid blue curve in Figure 8 when there were no non-adjacent vertices is similar, i.e., as the density of the marked vertices increases, the probabilities decrease, even adding the vertices non-adjacent. The same can be seen in the case of the dotted orange curves in Figure 8 and Figure 9b, i.e., when the density of the marked vertices increases, the probability also increases, this tells us that the value of $l = (n/N) \cdot k$ is optimal for high marked vertex densities.



(a) (b)

Figure 10: Probability of success after 100 steps with $n = 10$ and $N = 1024$ vertices. (a) shows the probability of success for k adjacent and non-adjacent marked vertices for $l = (n/N)$. (b) shows the probability of success for k adjacent and non-adjacent marked vertices for $l = (n/N) \cdot k$.

Figure 10 shows the probability of success for the search of marked adjacent and non-adjacent vertices in the search space. We performed an experiment, where, at every hundred steps, an adjacent vertex and a non-adjacent vertex were marked, i.e., for each M set of adjacent vertices a vertex $i \notin M$ was marked randomly, then, $M' = \{0, 1, i_0\}, \{0, 1, i_0, 2, i_1\}, \cdots, \{0, 1, i_0, 2, i_1, 4, i_2, \cdots, 512, i_{10}\}$. Figure 10a shows the probability of success for $l = (n/N)$ and Figure 10b shows the probability of success for $l = (n/N) \cdot k$. Note that the probability of success above 90% is achieved in a smaller number of steps.

4 Conclusions

Many efforts are applied in order to improve the performance of quantum search algorithms. Quantum walks are the main tool for building these algorithms. We initially analyzed the quantum walk in the hypercube applying Grover's search and came to the conclusion that neighbor vertices affect the search performance, an observation that has been corroborated by other authors. We found that the walk could not improve its results even for a number of marked vertices equal to one. Many authors have developed works for adding self-loops in various types of graphs and grids of different dimensions. In this sense, we decided to investigate how to improve the quantum search in the hypercube using self-loops. Previous works defined the optimal self-loop value as $l = (d/N)$ for one marked vertex to the quantum walk on the hypercube. After performing experiments we saw that this value of l was not optimal for multiple marked vertices. We arrive at a value of $l = (n/N) \cdot k$ for an arbitrary number of vertices. This value is also used when searching in two-dimensional grids. Another aspect of the quantum walk in the hypercube is whether the marked vertex is adjacent or not, this interferes with the search performance. We then analyzed whether the value of $l = (n/N)$ and $l = (n/N) \cdot k$ had any positive effect when applied to the hypercube vertices. The results show that the value of $l = (n/N)$ is not optimal for the quantum walk in the hypercube with multiple marked vertices adjacent or not. It also shows that for a search space where there are marked adjacent vertices, just one non-adjacent marked vertex is sufficient for the value of $l = (n/N) \cdot k$ to be better. According to the results presented here, there is a greater than 90% probability that the measurement will collapse in one of the solutions. Recent works have used the quantum walks to search weights and train artificial neural networks [Souza et al., 2019, 2021]. The quantum walk in the hypercube has an interesting behavior, the amplification of neighbors vertices. In future work, we intend to use this quantum walk to find a set of weights to initialize and train classical artificial neural networks. We also intend to analyze the quantum walk in the hypercube with multiple weighted self-loops.

Acknowledgments

Acknowledgments to the Science and Technology Support Foundation of Pernambuco (FACEPE) Brazil, Brazilian National Council for Scientific and Technological Development (CNPq), and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 by their financial support to the development of this research.

References

Neil Shenvi, Julia Kempe, and K Birgitta Whaley. Quantum random-walk search algorithm. *Physical Review A*, 67(5): 052307, 2003.

V Potoček, Aurél Gábris, Tamás Kiss, and Igor Jex. Optimized quantum random-walk search algorithms on the hypercube. *Physical Review A*, 79(1):012325, 2009.

Luciano S Souza, Jonathan H A Carvalho, and Tiago A E Ferreira. Quantum walk to train a classical artificial neural network. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 836–841. IEEE, 2019.

Luciano S Souza, Jonathan H A Carvalho, and Tiago A E Ferreira. Classical artificial neural network training using quantum walks as a search procedure. *IEEE Transactions on Computers*, 2021.

Huiquan Wang, Jie Zhou, Junjie Wu, and Xun Yi. Adjustable self-loop on discrete-time quantum walk and its application in spatial search. *arXiv preprint arXiv:1707.00601*, 2017.

GA Bezerra, PHG Lugão, and R Portugal. Quantum walk-based search algorithms with multiple marked vertices. *Physical Review A*, 103(6):062202, 2021.

Jonathan H A Carvalho, Luciano S Souza, Fernando M Paula Neto, and Tiago A E Ferreira. Impacts of multiple solutions on the lackadaisical quantum walk search algorithm. In *Brazilian Conference on Intelligent Systems*, pages 122–135. Springer, 2020.

Nikolajs Nahimovs, Raqueline A M Santos, and K R Khadiev. Adjacent vertices can be hard to find by quantum walks. *Moscow University Computational Mathematics and Cybernetics*, 43(1):32–39, 2019.

Mason L Rhodes and Thomas G Wong. Quantum walk search on the complete bipartite graph. *Physical Review A*, 99 (3):032301, 2019.

Thomas G Wong. Faster search by lackadaisical quantum walk. *Quantum Information Processing*, 17(3):1–9, 2018.

Andris Ambainis, Julia Kempe, and Alexander Rivosh. Coins make quantum walks faster. *arXiv preprint quant-ph/0402107*, 2004.

Mason L Rhodes and Thomas G Wong. Search on vertex-transitive graphs by lackadaisical quantum walk. *Quantum Information Processing*, 19(9):1–16, 2020.

Nikolajs Nahimovs. Lackadaisical quantum walks with multiple marked vertices. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 368–378. Springer, 2019.

Jasmeet Singh and Mohit Singh. Evolution in quantum computing. In *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*, pages 267–270. IEEE, 2016.

Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. AAPT, Cambridge, UK, 2002.

Noson S Yanofsky and Mirco A Mannucci. *Quantum computing for computer scientists*. Cambridge University Press, 2008.

David McMahon. *Quantum computing explained*. John Wiley & Sons, 2007.

Yakir Aharonov, Luiz Davidovich, and Nicim Zagury. Quantum random walks. *Physical Review A*, 48(2):1687, 1993.

Andris Ambainis, Artūrs Bačkurs, Nikolajs Nahimovs, Raitis Ozols, and Alexander Rivosh. Search by quantum walks on two-dimensional grid without amplitude amplification. In *Conference on Quantum Computation, Communication, and Cryptography*, pages 87–97. Springer, 2012.

Salvador Elías Venegas-Andraca. Quantum walks: a comprehensive review. *Quantum Information Processing*, 11(5): 1015–1106, 2012.

Julia Kempe. Quantum random walks hit exponentially faster. *arXiv preprint quant-ph/0205083*, 2002.

Cristopher Moore and Alexander Russell. Quantum walks on the hypercube. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 164–178. Springer, 2002.

Thomas G Wong. Grover search with lackadaisical quantum walks. *Journal of Physics A: Mathematical and Theoretical*, 48(43):435304, 2015.

Jonathan H. A. Carvalho, Luciano S Souza, Fernando M Paula Neto, and Tiago A E Ferreira. On applying the lackadaisical quantum walk algorithm to search for multiple solutions on grids. *arXiv preprint quant-ph/2106.06274*, 2021.

Nikolajs Nahimovs and Raqueline AM Santos. Lackadaisical quantum walks on 2d grids with multiple marked vertices. *arXiv preprint arXiv:2104.09955*, 2021.

Multi-self-loop Lackadaisical Quantum Walk with Partial Phase Inversion

A preprint can be found at: https://arxiv.org/pdf/2305.01121

MULTI-SELF-LOOP LACKADAISICAL QUANTUM WALK WITH PARTIAL PHASE INVERSION

A PREPRINT

Luciano S. de Souza*
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
Recife, Brasil
luciano.serafim@ufrpe.br

Jonathan H. A. de Carvalho
Centro de Informática
Universidade Federal de Pernambuco
Recife, Brasil
jhac@cin.ufpe.br

Henrique C. T. Santos
Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco
Recife, Brasil
henrique.santos@recife.ifpe.edu.br

Tiago A. E. Ferreira
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
Recife, Brasil
tiago.espinola@ufrpe.br

November 18, 2024

ABSTRACT

The lackadaisical quantum walk, a quantum analog of the lazy random walk, is obtained by adding a weighted self-loop transition to each state. Impacts of the self-loop weight l on the final success probability in finding a solution make it a key parameter for the search process. The number of self-loops can also be critical for search tasks. This article proposes the quantum search algorithm Multi-self-loop Lackadaisical Quantum Walk with Partial Phase Inversion, which can be defined as a lackadaisical quantum walk with multiple self-loops, where the target state phase is partially inverted. In the proposed algorithm, each vertex has m self-loops, with weights $l' = l/m$, where l is a real parameter. The phase inversion is based on Grover's algorithm and acts partially, modifying the phase of a given quantity $s < m$ of self-loops. On a hypercube structure, we analyzed the situation where $1 \leqslant m \leqslant 30$. We also propose two new weight values based on two ideal weights l used in the literature. We investigated the effects of partial phase inversion in the search for 1 to 12 marked vertices. As a result, this proposal improved the maximum success probabilities to values close to 1 in $O(\sqrt{(n+m)\cdot N})$, where n is the hypercube degree. This article contributes with a new perspective on the use of quantum interferences in constructing new quantum search algorithms.

Keywords Quantum Computing · Quantum Interference · Quantum Walks · Quantum Search Algorithm · Lackadaisical Quantum Walk · Multiple Self-loops · Partial Phase Inversion

1 Introduction

Just as quantum interference plays an essential role in the development of quantum algorithms, for Shenvi et al. [2003], quantum walks provide one of the most promising features, an intuitive framework for building new quantum algorithms. Their pioneering paper designed a quantum search algorithm based on quantum walks. They demonstrated that this quantum search algorithm could be used to find a marked vertex of a hypercube. Although they are distinct algorithms, there are several similarities with Grover's algorithm [Grover, 1996]. Both algorithms start in the state of equal superposition over all states. They make use of Grover's diffusion operator. They can be seen as a rotation in

*R. Dom Manuel de Medeiros, s/n, Dois Irmãos – Recife, Pernambuco – Brasil

a two-dimensional subspace. They have the same running time. The measurement is performed at a specific time to obtain the maximum probability of success. Both algorithms use an oracle that marks the target state with a phase of -1, *i.e.*, they use quantum interference to develop their quantum search algorithms.

Since then, some works have been developed to improve the capacity of the quantum search algorithm in the hypercube [Potoček et al., 2009, Hein and Tanner, 2009]. The addition of self-loops at each vertex is part of many improvement proposals for quantum search algorithms. Hoyer and Meyer [2009] used a directed walk on the line to produce an algorithm with $(N-1)$ self-loops at each vertex, where N is the total number of vertices. In its turn, Wong [2015] proposed a quantum analog of the classical lazy random walk, called the lackadaisical quantum walk - LQW. The quantum walker has a chance to stay at the same vertex by introducing m integer self-loops at each vertex of the graph, and its effects were investigated with Grover's search algorithm. However, Wong [2017] proposed a modification that reduced the number m to a single self-loop with a non-integer weight l.

Recently, some works have investigated the application of the lackadaisical quantum walk on the hypercube structure. Rhodes and Rhodes and Wong [2020] showed that the ideal value for the self-loop weight in the search for a single marked vertex is $l = d/N$, where d is the degree of the vertex and N is the number of nodes in the hypercube. Souza et al. [2021] showed that the optimal self-loop weight value for searching a single marked vertex is not optimal for searching multiple marked vertices on the hypercube. Thus, they defined a new value of l which is the value proposed by Rhodes and Rhodes and Wong [2020] multiplied by the number of marked vertices k, resulting in $l = (d/N) \cdot k$.

Although Souza et al. [2021] have obtained improvements in the search for multiple vertices in the hypercube by proposing a new ideal weight value for the self-loop, in some cases, it is not possible to maintain the performance of the quantum search algorithm using the LQW. The LQW's original proposal used m self-loops with an integer weight l, and later this number was reduced to a single self-loop with a non-integer weight. However, according to Rhodes and Rhodes and Wong [2020], if the weight value l of the self-loop is an integer, it is equivalent to having several unweighted self-loops. We can therefore suggest that there is a relationship between the weight value l and the number m of self-loops.

Therefore, in the described scenario, it is possible to observe three aspects that were discussed, which can interfere with the performance of the LQW. The first is quantum interference, which in practical terms, can be achieved with the phase inversion of a target state. The second is the weight value of the self-loop, and the third is the number of self-loops. Another aspect we must consider is the distribution of the weight value between the multiple self-loops. Some authors use different strategies to define how the weights are distributed in the set of vertices.

In the work developed by Hoyer and Meyer [2009], the amplitude of each self-loop is the same, and they are grouped into a single normalized state. Wang et al. [2017] proposed a model of adjustable self-loops controlled by a real parameter in the coin operator. Rhodes and Rhodes and Wong [2019a] investigated the spatial search in the complete bipartite graph, which can be irregular with N_1 and N_2 vertices in each partition. In this way, self-loops in each set of vertices can have different weights l_1 and l_2, respectively. Rapoza and Rapoza and Wong [2021] defined a self-loop weight value for marked vertices, while the remaining weights for unmarked vertices can be chosen randomly.

Based on the previous information and through experiments, we verified that using a single self-loop at each hypercube vertex with a non-integer weight l is equivalent to using m self-loops where each self-loop has a non-integer weight l/m. Considering Grover's search algorithm used in the LQW, in practical terms, after executing the part of the quantum system responsible for detecting the target state, a phase rotation is applied, *i.e.*, by linearity, it inverts the phase of all the base states related to the target state and applies the diffusion transformation.

According to McMahon [2007], quantum interference plays an important role in the development of quantum algorithms. There are two types of interference, positive (constructive) and negative (destructive), in which the probability amplitudes add constructively or the probability amplitudes add destructively, respectively. For Grover [1996], which developed an algorithm based on quantum interference, the operation that results in the phase shift of the target state is one of the procedures that form the basis of quantum mechanics algorithms and makes them more efficient than classical analog algorithms.

However, as seen previously, in some cases, it is impossible to maintain the performance of the LQW. Since the m self-loops are redundant, the proposal of this work is a different way of performing the phase inversion process. We suggest modifying Grover's search algorithm to make possible the partial inversion of the base states that represent the m self-loops of the marked vertices. Therefore, we propose the Multi-self-loop Lackadaisical Quantum Walk with Partial Phase Inversion - MSLQW-PPI. We will show and analyze the effects of using multiple real-valued weighted self-loops at each vertex of an n-dimensional hypercube with Grover's search algorithm using a different strategy in the phase inversion operation that only acts on $s < m$ self-loops.

The LQW algorithm highly depends on the self-loop weight value. The ideal weight composition in several structures, including the hypercube, considers the vertex degree, the total number of vertices, and the number of marked vertices [Wong, 2018, Rhodes and Wong, 2019b, Giri and Korepin, 2020, Carvalho et al., 2023]. Therefore, based on the weights proposed by Rhodes and Rhodes and Wong [2020] and Souza et al. [2021] for the use of a single self-loop, we also suggest two new weights for the use of multiple self-loops that explore this relationship between vertex degree, the total number of vertices and the number of marked vertices. We added an integer exponent α equal to 2, restricting the analysis to this value as the initial choice. In this way, we can analyze the performance of the quantum walk in the hypercube, maintaining the ideal weights' composition, just by modifying its scale.

The main contributions of this work are summarized as follows. We present a new quantum search algorithm based on lackadaisical quantum walks. For this, we revisit the use of multiple self-loops per vertex and propose a partial phase inversion of the target states based on a modification in Grover's oracle. We formulate two new weight values l, such that $l = n^2/N$ and $l = (n^2/N) \cdot k$, where n^2 is the degree of the vertex in a n-regular structure 2 times, N is the number of vertices, and k is the number of marked vertices. Each m self-loop is weighted in the form $l' = l/m$. Finally, with this approach, we were able to increase the maximum success probabilities to values close to 1.

This paper is organized as follows. In Section 2, we present some concepts about quantum walks in the hypercube. In Section 3, we present the proposal for this work. In Section 4, the experiments are defined. Section 5 presents the results and discussion. Finally, Section 6 contains the conclusions.

2 Lackadaisical quantum walk on the hypercube

Quantum walks are the quantum counterpart of classical random walks. They are an advanced tool that provides one of the most promising features, an intuitive framework for building new quantum algorithms [Aharonov et al., 1993, Shenvi et al., 2003, Ambainis et al., 2012]. The evolution of the discrete-time quantum walk occurs by successive applications of a unitary evolution operator U that operates in the Hilbert space,

$$\mathcal{H} = \mathcal{H}^C \otimes \mathcal{H}^S \tag{1}$$

where the coin space $\mathcal{H}^C$ is the Hilbert space associated with a quantum coin, and the walker space $\mathcal{H}^S$ is the Hilbert space associated with its position representation. The evolution operator U is defined as follows,

$$U = S(C \otimes I_N) \tag{2}$$

where S is the shift operator that acts in the walker's space based on the state of the coin. I_N is the identity matrix, and the unitary matrix C is the coin operator [Shenvi et al., 2003]. The evolution equation represented by a quantum walk at time t is given by

$$|\Psi(t)\rangle = U^t |\Psi(t=0)\rangle . \tag{3}$$

In general, the study of quantum walks needs a structure to represent the time evolution of the walker. It is possible to use many different structures, such as complete and johnson's graphs [Wong, 2015, 2017, Zhang et al., 2018], grids [Saha et al., 2022, Carvalho et al., 2023], hypercubes [Rhodes and Wong, 2020, Souza et al., 2021], among others [Rhodes and Wong, 2019a, Tanaka et al., 2022, Qu et al., 2022]. Here, it was chosen the hypercube structure. The hypercube was used by Shenvi et al. [2003] as a temporal evolution structure to propose in his pioneering work a search algorithm based on quantum walks. Furthermore, a quantum walk on the hypercube can be reduced to a quantum walk on a line which reduces its complexity. The n-degree hypercube is an undirected graph with 2^n nodes, where each node can be described by a binary string of n bits. In this way, two nodes $\vec{x}$ and $\vec{y}$ in the hypercube are adjacent if $\vec{x}$ and $\vec{y}$ differ by only a single bit, *i.e.*, if $|x - y| = 1$, where $|x - y|$ is the Hamming distance between $\vec{x}$ and $\vec{y}$ [Venegas-Andraca, 2012].

Let us define the Hilbert space associated with the quantum walk on the hypercube. According to Equation 1, the Hilbert space associated with the quantum coin space is $\mathcal{H}^C$, and the Hilbert space associated with the walker's position is $\mathcal{H}^S$. Then, the Hilbert space associated with the quantum walk in the hypercube is

$$\mathcal{H} = \mathcal{H}^n \otimes \mathcal{H}^{2^n} \tag{4}$$

where $\mathcal{H}^n$ is the Hilbert space associated with the quantum coin space, and $\mathcal{H}^{2^n}$ is the Hilbert space associated with nodes in the hypercube, which represents the walker's position. In an n-dimensional hypercube, the i directions define the states of the coin and can be labeled by the n base vectors $\{|0\rangle, |1\rangle, \ldots, |n-1\rangle\}$. Each one of these n base vectors can be represented by $\{|e_0\rangle, |e_1\rangle, \ldots, |e_{n-1}\rangle\}$, where e_i is a binary string of n bits with 1 in the i-th position [Kempe, 2002, Shenvi et al., 2003]. The shift operator S, described in Equation 5, acts mapping a state $|i, \vec{x}\rangle \rightarrow |i, \vec{x} \oplus \vec{e_i}\rangle$.

$$S = \sum_{i=0}^{n-1} \sum_{\vec{x}} |i, \vec{x} \oplus \vec{e_i}\rangle \langle i, \vec{x}| \tag{5}$$

The pure quantum walk (without search) evolves by repeated applications of the evolution operator described in Equation 2, where C is Grover's "diffusion" operator on the coin space, and it is given by

$$C = 2 |s^C\rangle \langle s^C| - I_n \tag{6}$$

where I_n is the identity operator, and $|s^C\rangle$ is the equal superposition over all n directions [Moore and Russell, 2002, Shenvi et al., 2003], *i.e.*,

$$|s^C\rangle = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} |i\rangle . \tag{7}$$

Now, consider the quantum walk with search. A query to the "Grover oracle", described in Equation 8, is included in each step of the quantum walk.

$$U' = U \cdot (I_n \otimes Q) \tag{8}$$

where $Q = I_N - 2 |\omega\rangle \langle\omega|$, and $|\omega\rangle$ is the marked vertex. The initial state of the quantum walk in the hypercube is defined according to Equation 9 as an equal superposition for all N nodes and n directions.

$$|\Psi(t=0)\rangle = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} |i\rangle \otimes \frac{1}{\sqrt{N}} \sum_{\vec{x}} |\vec{x}\rangle \tag{9}$$

Finally, we will define the lackadaisical quantum walk in the hypercube. The lackadaisical quantum walk is a quantum analog of the lazy random walk. This quantum algorithm is obtained by adding at least one self-loop to each graph vertex [Wong, 2015]. According to the definition presented by Høyer and Yu [2020], considering an n-regular graph with a single marked vertex, by adding a self-loop of weight l to each vertex, the coined Hilbert space becomes

$$\mathcal{H}^{n+1} = \{|e_0\rangle, |e_1\rangle, \ldots, |e_{n-1}\rangle, |\circlearrowleft\rangle\},$$

where $|\circlearrowleft\rangle$ represents the self-loop. Thus weighted self-loop accounting is done by modifying Grover's coin presented in Equation 6, as follows

$$C = 2 |s^C\rangle \langle s^C| - I_{(n+1)} \tag{10}$$

where

$$|s^C\rangle = \frac{1}{\sqrt{n+l}} \left(\sqrt{l} |\circlearrowleft\rangle + \sum_{i=0}^{n-1} |i\rangle \right) . \tag{11}$$

3 Article proposal

Let us present the proposal for this work, the Multi-self-loop Lackadaisical Quantum Walk with Partial Phase Inversion (MSLQW-PPI). It is an alternative to exploring multiple non-integers self-loops in a way that can improve the results of the lackadaisical quantum walk. Considering an n-regular hypercube, we add m self-loops with weights l' at each vertex, *i.e.*, the amplitude of each self-loop is weighted by $\sqrt{l'}$. Therefore, the Hilbert space associated with the coin space becomes

$$\mathcal{H}^{n+m} = \{|e_0\rangle, |e_1\rangle, \ldots, |e_{n-1}\rangle, |\circlearrowleft_0\rangle, |\circlearrowleft_1\rangle, \ldots, |\circlearrowleft_{\mathrm{m}-1}\rangle\}.$$

To account for the m weighted self-loops, a new modification was made to Grover's coin described in Equation 10, as follows,

$$C = 2\,|s^C\rangle\,\langle s^C| - I_{(n+m)} \tag{12}$$

where

$$|s^C\rangle = \frac{1}{\sqrt{n+l}}\left(\sqrt{l'}\sum_{j=0}^{m-1}|\circlearrowleft_j\rangle + \sum_{i=0}^{n-1}|i\rangle\right) \tag{13}$$

and $l' = l/m$. In this way, according to Equation 13, we consider that the weight l has its value equally distributed to the m self-loops. The MSLQW-PPI system in the hypercube starts as follows,

$$|\Psi(t=0)\rangle = |s^C\rangle \otimes \frac{1}{\sqrt{N}}\sum_{\vec{x}}|\vec{x}\rangle\,. \tag{14}$$

Substituting Equation 13 into Equation 14 and applying the expansions, we obtain the initial state described in Equation 15.

$$|\Psi(t=0)\rangle = \frac{\sqrt{l'}}{\sqrt{n+l}\times\sqrt{N}}\sum_{j=0}^{m-1}\sum_{\vec{x}}|\circlearrowleft_j,\vec{x}\rangle + \frac{1}{\sqrt{n+l}\times\sqrt{N}}\sum_{i=0}^{n-1}\sum_{\vec{x}}|i,\vec{x}\rangle \tag{15}$$

To properly explore those multiple self-loops at each vertex, we propose a modification to Grover's oracle described previously. Note that, in Equation 8, a query $(I_n \otimes Q)$ is included at each step of the quantum walk, where $Q = I_N - 2\,|\omega\rangle\,\langle\omega|$ and ω is the marked vertex. Thus, by linearity, when we apply the oracle to all $|i\rangle\,|\vec{x}\rangle$ states of the superposition of vertices and edges, there are two possibilities. The first possibility is that $\vec{x}$ is not the target state, *i.e.*, $\omega \neq \vec{x}$. As $\langle * \rangle\,\omega\vec{x} = 0$, the state stays unchanged. The second possibility is that $\vec{x}$ is the target state, *i.e.*, $\omega = \vec{x}$. In this case, as $\langle * \rangle\,\omega\vec{x} = 1$, we have the phase inversion of the target state.

As can be seen, this oracle depends exclusively on the vertex in question. The edge is not considered for state phase inversion. All states associated with the vertex in question, independent of the edge, are also inverted. Here, we propose a partial inversion of the states related to the marked vertices. According to Rhodes and Wong [2020], the number of self-loops is a parameter that adjusts the probability of a walker staying put. The idea here is to be able to invert the phase of a target self-loops $\circlearrowleft_\tau$ and all edges that are not self-loops ϵ of the hypercube of a target vertex and investigate their effects. Inspired by Hoyer and Meyer [2009], we identify each edge of the hypercube by assigning a basis vector, as follows, $|\circlearrowleft_j,\vec{x}\rangle$ and $|i,\vec{x}\rangle$, where $0 \leqslant j \leqslant m-1$ and $0 \leqslant i \leqslant n-1$. Hence, each state $|\mathbf{x}\rangle$, which represents a walker's position, is a linear combination of the states,

$$|\mathbf{x}\rangle = |\circlearrowleft_0,\vec{x}\rangle + \cdots + |\circlearrowleft_{\mathrm{m}-1},\vec{x}\rangle + |0,\vec{x}\rangle + \cdots + |n-1,\vec{x}\rangle \tag{16}$$

which denotes the superposition of all edges [Yu, 2018]. In the case where $|\mathbf{x}\rangle$ contains the target state, it will have the phase of the components $|\circlearrowleft_\tau,\vec{x}\rangle$ and $|\epsilon,\vec{x}\rangle$ changed. It requires an oracle that identifies the state's components. Consider, again, Grover's oracle described in Equation 8. The proposed modification of the oracle described in Equation 17 makes it possible to identify the components of the target state.

$$Q = I_{(n+\mathrm{m})\cdot N} - 2\sum_{\omega}\sum_{\epsilon=0}^{n-1} |\epsilon,\omega\rangle\langle\epsilon,\omega| - 2\sum_{\omega}\sum_{\tau} |\circlearrowleft_{\tau},\omega\rangle\langle\circlearrowleft_{\tau},\omega| \tag{17}$$

where $|\omega\rangle$ represents the marked vertex, ϵ represents an edge that is not a self-loop, and $\circlearrowleft_{\tau}$ are the self-loops that will have their phases inverted. In this way, Q acts in the coin and vertex space described in Equation 1, contrary to Equation 8, where Q acts in the vertex space. Consider an arbitrary state $|\mathbf{x}\rangle$. When applying the proposed oracle, the phase of the components that represent their edges is considered individually. Considering Equation 18, at each step of the quantum walk an oracle query is applied to each edge of the state by linearity.

$$Q|\mathbf{x}\rangle = Q|\circlearrowleft_0,\vec{x}\rangle + Q|\circlearrowleft_1,\vec{x}\rangle + \cdots + Q|\circlearrowleft_{\mathrm{m}-1},\vec{x}\rangle + Q|0,\vec{x}\rangle + Q|1,\vec{x}\rangle + \cdots + Q|n-1,\vec{x}\rangle \tag{18}$$

Here also there are two possibilities. The first possibility is that $|\mathbf{x}\rangle$ does not contain the target state. The second possibility is that $|\mathbf{x}\rangle$ contains the target state. Finally, it is necessary to define the self-loop to have its phase inverted. Here, the phase of s self-loops is inverted, where experiments with $1 \leqslant s \leqslant m$ were done. The simpler situation where only one single self-loop is inverted can be used to illustrate the process. Consider the states $|\circlearrowleft_{\tau=j}\rangle$ as the target self-loops without loss of generality. Therefore, the description of the oracle is done as follows,

$$Q = I_{(n+\mathrm{m})\cdot N} - 2\sum_{\omega}\sum_{\epsilon=0}^{n-1} |\epsilon,\omega\rangle\langle\epsilon,\omega| - 2\sum_{\omega}\sum_{\tau} |\circlearrowleft_{\tau=j},\omega\rangle\langle\circlearrowleft_{\tau=j},\omega| \tag{19}$$

For the case where $|\mathbf{x}\rangle$ does not contain the target state, the phase of the states remains unchanged. For the case where $|\mathbf{x}\rangle$ contains the target state, we have the partial phase inversion of the components $|\circlearrowleft_{\tau=j},\vec{x}\rangle$ and $|i,\vec{x}\rangle$, while the components $|\circlearrowleft_{\tau\neq j},\vec{x}\rangle$ stays unchanged. Additional details on the application of the new oracle in three possible scenarios can be found in Appendix A.

4 Experiment setup

According to the definitions of the hypercube, two marked vertices are adjacent if the Hamming distance between them is 1. A set of non-adjacent marked vertices have a Hamming distance of at least 2 from any other marked vertex, *i.e.*, they are mutually non-adjacent. In the experiments performed in this paper, we consider only the scenario where the marked vertices are non-adjacent. The search for adjacent marked vertices constitutes a separate scenario and will not be addressed in this work.

Experiments were performed to analyze the behavior of a Lackadaisical Quantum Walk with multiple self-loops at each vertex, where, the quantity s $(1 \leqslant s \leqslant m)$ of self-loops is inverted. The case $s = m$ is the conventional oracle operation situation. Initially, we used the weight values proposed by Rhodes and Wong [2020] and Souza et al. [2021]. These experiments were also made for the weight values proposed in this work. To evaluate the relative dispersion behavior of the mean success probability, we used Pearson's coefficient of variation (the ratio between the standard deviation and the mean value).

4.1 Definition of vertex sets and simulations

To determine how the simulations are performed, it is necessary to define how the marked vertices are divided. For each number of marked vertices k, γ simulations are performed varying the position of the k marked vertices. Therefore, the marked vertices are divided into groups of $M_{k,\gamma}$ samples. Here we define $1 \leqslant k \leqslant 12$ and $\gamma = 100$. In this way, we have a set of twelve hundred samples. This set is divided into twelve groups of one hundred samples as follows: $M_{1,100}, M_{2,100}, M_{3,100}, \cdots, M_{12,100}$. Each sample was made without replacement following a uniform distribution, *i.e.*, each one of them has k distinct vertices. For each group of one hundred samples, we fixed the k number of marked vertices and vary their location, for example, when $k = 2$, as shown below,

$$M_{2,100} = [\{254, 1498\}_1, \{969, 3520\}_2, \ldots, \{410, 1121\}_{100}].$$

The values shown, for example, $\{254, 1498\}_1$, are the 1st sample from a total of 100 with 2 marked vertices, where its binary representation is $\{000011111110, 010111011010\}_1$ in the hypercube. In this way, for every new group of

Table 1: Client machine settings.

Node	System RAM	System Processor
Node 1 and 2	32 GB	Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz
Node 3	8 GB	Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz
Node 4 and 5	32 GB	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
Node 6	16 GB	Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz

simulations $M_{k,\gamma}$, $k \cdot 100$ non-adjacent vertices are marked and divided into 100 samples of k vertices. Each one of these samples with k vertices is a computational experiment. For instance, $k = 3$

$$M_{3,100} = [\{3034, 1616, 2438\}_1, \{2059, 2745, 3686\}_2, \ldots, \{3017, 3484, 1773\}_{100}].$$

As a first investigation, the experiments employ $s = 1$ inverted self-loop. Experiments were also performed for full-phase inversion where $s = m$. In the second investigation, the experiments employ $2 \leqslant s < m$. Therefore, $1 \leqslant s \leqslant 30$ and $1 \leqslant m \leqslant 30$. For each sample $M_{k,\gamma}$, all possible combinations of $s \leqslant m$ are done. For example, when $m = 1$, then $s = 1$. When $m = 2$, then $s = 1$, and $s = 2$, and so on until $m = 30$, the $s = \{1, 2, \cdot, 30\}$. This way, a total of 558000 simulations were performed. A simulation stops after each quantum walk has reached the number of iterations necessary to obtain the maximum value of the probability amplitude in the k marked vertices. Considering $s = 1$ and $m = 1$ we have the same conditions as Rhodes and Wong [2020] and Souza et al. [2021], *i.e.* the use of a single self-loop.

4.2 Hardware and software setup

The simulations were performed using the Parallel Experiment for Sequential Code - PESC [Santos et al., 2023]. The PESC is a computational platform for distributing computer simulations on the resources available on a network by packaging the user code in containers that abstract all the complexity needed to configure these execution environments, allowing any user to benefit from this infrastructure. All client nodes that participated in the simulations use the Ubuntu 18.04.6 LTS (Bionic Beaver) operational system and have an HD of 500 GB. Other machine settings are shown in Table 1.

The PESC platform provides a web interface where the user configures a request to execute a simulation. In this request, the user can inform the number of times the simulation must be repeated. Each instance of the simulation receives via parameter the instance identification, called a rank. The rank is used to initialize variables and parameterize other processes. The programming language used to write the algorithms was Python 3.7.

After receiving the request, the PESC platform distributes the simulations to the available client nodes. The distribution is based on each client's workload, where each client's performance factor is informed when connecting to the server. Before starting the execution, each client creates and configures the environment necessary to run the received code. The user informs the execution environment needs at the request moment.

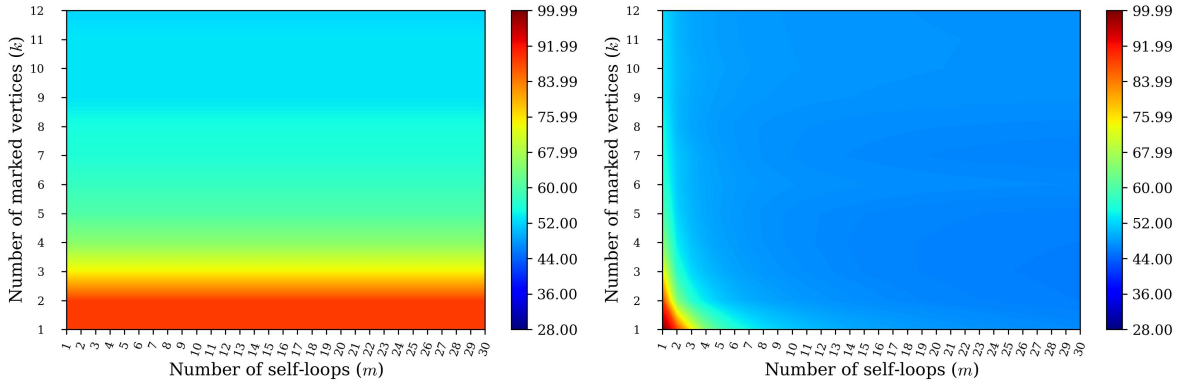
Using the platform simplified the simulation execution process as it manages the status and life cycle of the request, restarting simulation instances in case of failure on any of the clients, with the possibility of moving the instance to another client node depending on the type of failure detected.

All this was very important considering the nature of the proposed study due to the long time required to finish all simulations and then collect the data that supports this study's results. This tool was developed for the instrumentation and optimization of computational studies conducted by Prof. Tiago A. E. Ferreira's research group.

5 Results and discussion

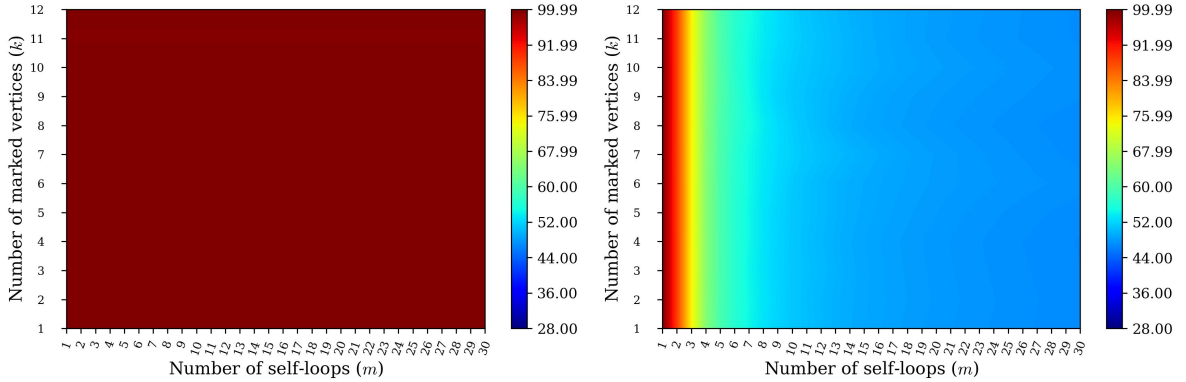
Initially, our discussions focus on cases where only one self-loop is inverted, *i.e.* $s = 1$ and $1 \leqslant m \leqslant 30$, and are compared with approaches where all self-loops are inverted or $s = m$. Fig. 1 shows the success probabilities for the lackadaisical quantum walk and the MSLQW-PPI in the hypercube for the weights $l = n/N$ and $l = (n/N) \cdot k$.

Fig. 1a shows the probability of success for the full phase inversion and Fig. 1b shows the probability of success for the partial phase inversion, both for weight values $l = n/N$. When we invert the phase of all self-loops at each marked vertex, it is equivalent to using a single weighted self-loop at each marked vertex on the hypercube [Wong, 2015]. In these cases, the results are similar to those obtained in works by Rhodes and Wong [2020] with a probability of success of approximately 99% to search for one marked vertex.



(a) Total Inversion $l = n/N$

(b) Partial Inversion $l = n/N$



(c) Total Inversion $l = (n/N) \cdot k$

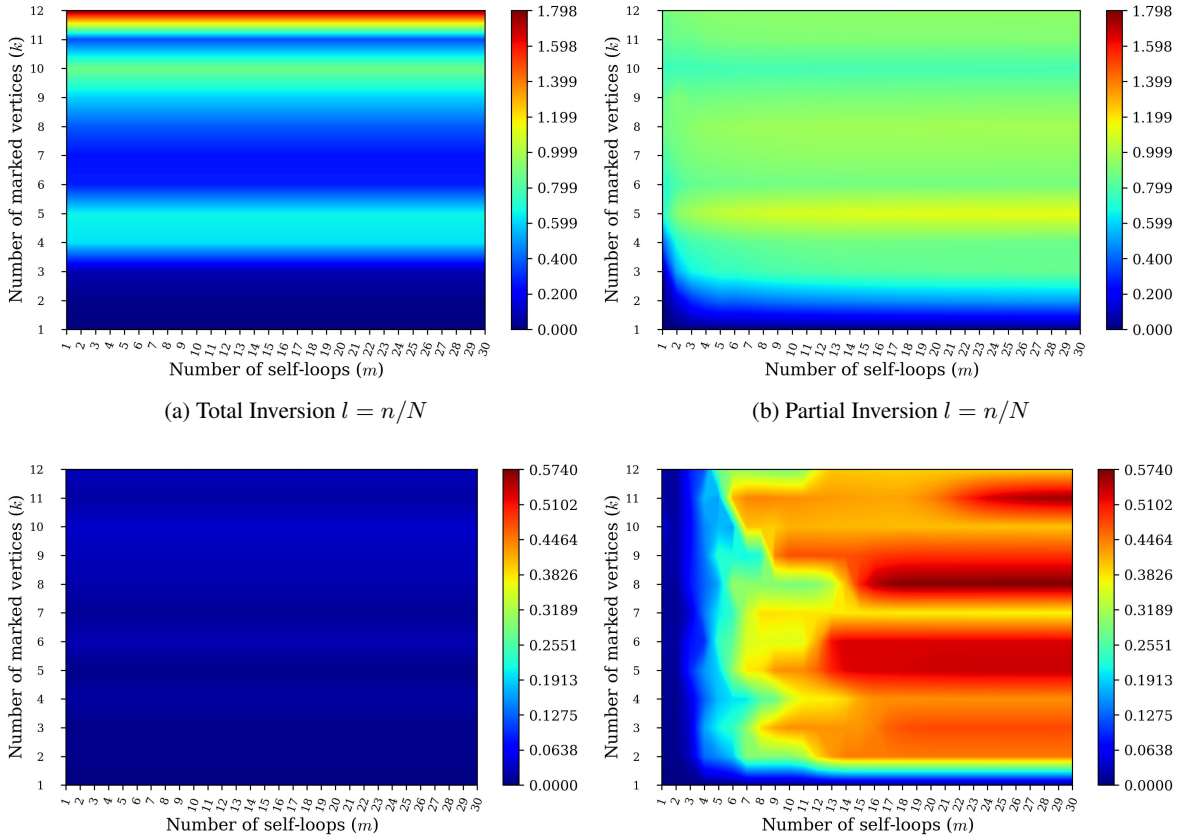
(d) Partial Inversion $l = (n/N) \cdot k$

Figure 1: The success probability of the Lackadaisical Quantum Walk with full phase inversion in Figures (a) and (c), and the MSLQW-PPI in Figures (b) and (d) in the hypercube to search for non-adjacent marked vertices with $n = 12$ and $N = 4096$ vertices. Figures (a) and (b) for the weight value $l = n/N$, proposed by Rhodes and Wong [2020]. Figures (c) and (d) for the weight value $l = (n/N) \cdot k$, proposed by Souza et al. [2021]

Fig. 1c shows the probability of success for the full phase inversion and Fig. 1d shows the probability of success for the partial phase inversion, both for weight values $l = (n/N) \cdot k$. In Fig. 1c, where the full phase inversion occurs, the probability of success stays at values close to 1, even using multiple self-loops, with a small variation if we observe its coefficient of variation in Fig. 2c. As we can see in Fig. 1d, the probabilities of success present results similar to those obtained by Souza et al. [2021].

Fig. 2 shows the coefficient of variation ($\sigma_w/\bar{w}$ – standard deviation normalized by the mean value) to measure the level of dispersion presented in the results obtained with the total and partial phase inversion of the results presented in Fig 1. The level of dispersion presented in the results obtained with the partial phase inversion in Figures 2b and 2d describes a behavior where the maximum probabilities of success present a smaller coefficient of variation. Unlike the results presented in Fig. 2a obtained with full inversion where the level of dispersion presented is greater for lower probabilities. Analyzing Fig. 2c, although the result for the coefficient of variation appears stable, there are subtle variations concerning the maximum probabilities of success presented in Fig. 1c.

As we see in Fig. 1, the weight values proposed by Rhodes and Wong [2020] and Souza et al. [2021] are ideal to use in approaches with a single self-loop at each vertex. Therefore, for our approach, it is necessary to define the best weight value for using multiple self-loops. Most of the weights defined in previous works for structures such as a complete graph, hypercube, one-dimensional grid, and two-dimensional grid consider three main parameters: the vertex degree, the total number of vertices, and the number of marked vertices. Two weight values were already defined by Rhodes and Wong [2020] and Souza et al. [2021] as ideal for searching for one and multiple marked vertices in the hypercube,



(a) Total Inversion $l = n/N$

(b) Partial Inversion $l = n/N$

(c) Total Inversion $l = (n/N) \cdot k$

(d) Partial Inversion $l = (n/N) \cdot k$

Figure 2: The results for coefficients of variation. Their values are represented in percentage terms. (a) and (b) weight value $l = n/N$. (c) and (d) weight value $l = (n/N) \cdot k$.

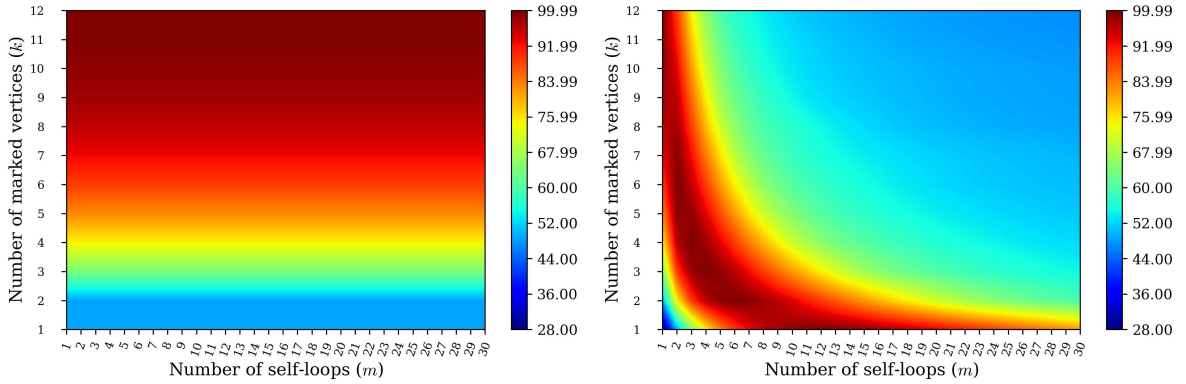
respectively: $l = n/N$ and $l = (n/N) \cdot k$, where n is the degree of the vertices, N the number of vertices, and k the number of marked vertices.

The idea here is to propose two new weight values based on those previous ideal weight values used to search for one and multiple vertices in the hypercube. We introduce a new parameter in the weight composition: the exponent in the numerator. Considering that the weights $l = n/N$ and $l = (n/N) \cdot k$ have an exponent equal to 1 in the parameter n^1, we can consider other values for this exponent. We can modify the scale of weight values through this new exponent. So we have a new proposal for the two weights

$$l = \left(\frac{n^2}{N}\right), \text{ and } l = \left(\frac{n^2}{N}\right) \cdot k.$$

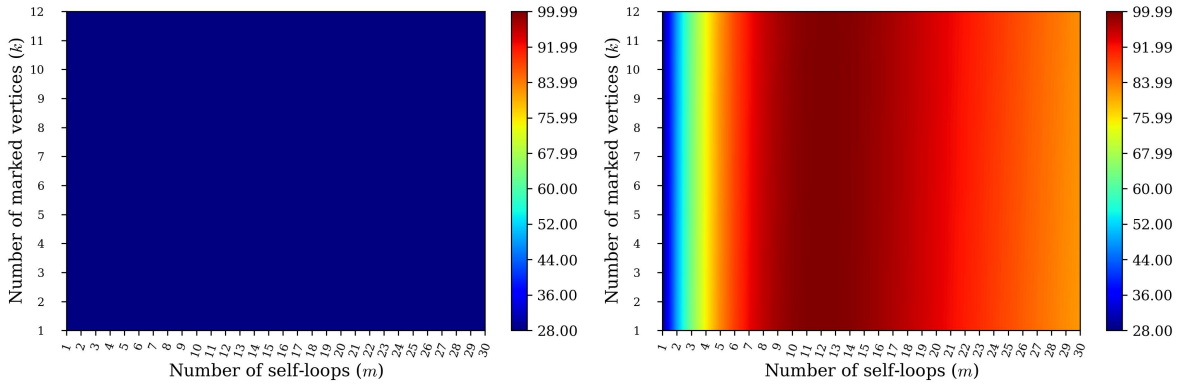
Fig. 3 shows the probability of success of the lackadaisical quantum walk with full phase inversion and MSLQW-PPI using the two new weight values to search multiple marked vertices. More one time, the first column (Figures 3a and 3c) presents the results for the full phase inversion and the second column (3b and 3d) the results for the partial phase inversion to the simplest situation of $s = 1$ and $1 \leqslant m \leqslant 30$.

Figures 3a and 3b show the maximum probability of success for the weight value $l = n^2/N$. In the Fig. 3a, the phase of all components of the target state representing the m self-loops is inverted. In this case, the probability of success is affected reaching a maximum probability of approximately $p \approx 1$ only when $k = 11, 12$. The Fig. 3b shows the maximum probability of success for the partial inversion. In this case, we obtain success probabilities close to 1. For $k = 2, 3, 4$ the maximum success probability is reached (with $m = 6, 4, 3$) self-loops, respectively, and for $k = 5, 6, 7, 8$ (with $m = 2$). Fig. 3c shows the maximum probability of success where the phase of all components of



(a) Total Inversion $l = n^2/N$

(b) Partial Inversion $l = n^2/N$



(c) Total Inversion $l = (n^2/N) \cdot k$

(d) Partial Inversion $l = (n^2/N) \cdot k$

Figure 3: The success probability of the Lackadaisical Quantum Walk with full phase inversion in Figures (a) and (c), and the MSLQW-PPI in Figures (b) and (d) in the hypercube to search for non-adjacent marked vertices with $n = 12$ and $N = 4096$ vertices. Figures (a) and (b) for the weight value $l = n^2/N$. Figures (c) and (d) for the weight value $l = (n^2/N) \cdot k$.

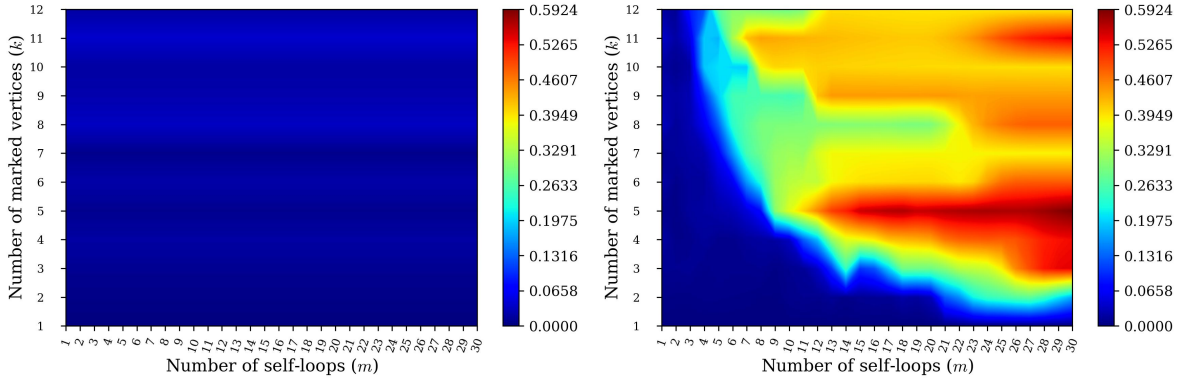
the target state is inverted and Fig. 3d shows the maximum probability of success for the partial inversion, both for the weight value $l = (n^2/N) \cdot k$. This scenario showed a considerable gain in the maximum probability of success from $p = 0.28$ (with m $= 1$) to $p \approx 1$ (with m $= 12$) self-loops to any number k of the marked vertices.

Figure 4 shows the coefficient of variation for the results described in Fig. 3. Analyzing the results, we observe a different behavior between the complete and partial phase inversions. With the partial phase inversion, the small coefficient of variation coincides with the maximum probabilities of success, however, with the complete phase inversion the variation is also smaller but does not coincide with the maximum probabilities of success. Table 2 shows that in most of the results applying partial phase inversion, there was an improvement in the maximum probability of success and a smaller coefficient of variation.

Continuing the comparisons between the results obtained with the different weight values, again, we observed that it was possible to increase the probability of success with more than one self-loop and partial phase inversion. Fig. 1d shows the probability of success for the weight $l = (n/N) \cdot k$ with total phase inversion. Compared to the results presented in Fig. 3d that shows the probability of success for the weight $l = (n^2/N) \cdot k$ with partial phase inversion, the probabilities of success are similar and close to 1, and what differs is the number of self-loops per vertex. In practical terms, using a single self-loop in this scenario is better. However, it is important to note that there exists a relationship between the weight and the number of self-loops per vertex. Note that the number of self-loops may change depending on the weight value. In the case of Fig. 3d, the probability of success is maximized when the number of self-loops increases to 12.

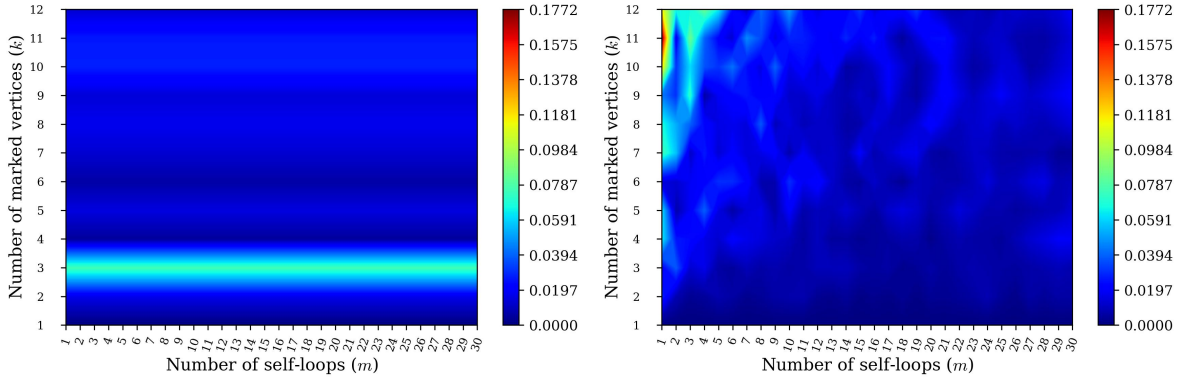
Table 2: Comparison between the probability of success and number of self-loops for two scenarios. The column for Fig. 3a represents the results where the phase of all self-loops is reversed. The column for Fig. 3b is for the case where the phase of only one self-loop is inverted. Both using the weight $l = n^2/N$. The acronym cv means the coefficient of variation.

	Figures					
	3a			3b		
k	p	m	cv	p	m	cv
2	0.48	1	5.234e-05	0.99	6	3.536e-04
3	0.64	1	1.050e-04	0.99	4	2.177e-04
4	0.75	1	1.922e-04	0.99	3	9.214e-05
5	0.83	1	9.214e-05	0.99	2	1.922e-04
6	0.88	1	2.177e-04	0.99	2	1.922e-04
7	0.92	1	8.130e-05	0.99	2	1.922e-04
8	0.95	1	3.536e-04	0.97	2	1.922e-04
9	0.97	1	2.477e-04	0.97	1	1.050e-04



(a) Total Inversion $l = n^2/N$

(b) Partial Inversion $l = n^2/N$



(c) Total Inversion $l = (n^2/N) \cdot k$

(d) Partial Inversion $l = (n^2/N) \cdot k$

Figure 4: The results for coefficients of variation. Their values are represented in percentage terms. (a) and (b) weight value $l = n^2/N$. (c) and (d) weight value $l = (n^2/N) \cdot k$.

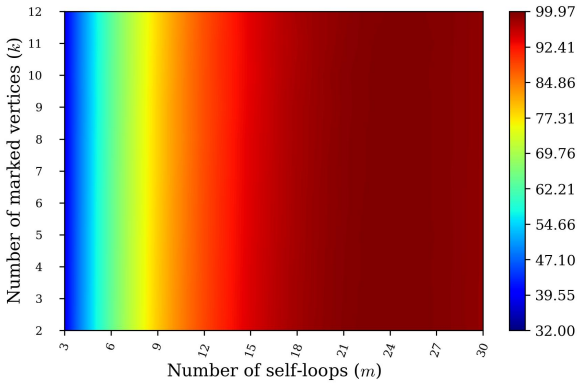


Figure 5: The success probability of MSLQW-PPI for weight value $l = (n^2/N) \cdot k$ for $s = 2$ inverted self-loops and $3 \leqslant m \leqslant 30$.

Table 3: Comparison between probabilities of success for searching non-adjacent marked vertices and the number m of self-loops. The results presented here refer to the search using partial phase inversion of the target state. From $k = 4$ marked vertices, the success probabilities for weight $l = n/N$ remain below $p = 0.66$, while for the other three weight values, the probability of success stays above $p = 0.99$. The acronym cv means the coefficient of variation.

	Self-loop weights											
	$l = n/N$			$l = (n/N) \cdot k$			$l = n^2/N$			$l = (n^2/N) \cdot k$		
k	p	m	cv	p	m	cv	p	m	cv	p	m	cv
2	0.887	1	1.643e-04	0.999	1	8.259e-05	0.999	6	5.234e-05	0.999	12	2.810e-04
3	0.750	1	7.037e-04	0.999	1	1.144e-04	0.999	4	1.050e-04	0.999	12	2.095e-04
4	0.663	1	3.882e-03	0.999	1	1.221e-04	0.998	3	1.922e-04	0.999	12	6.324e-04

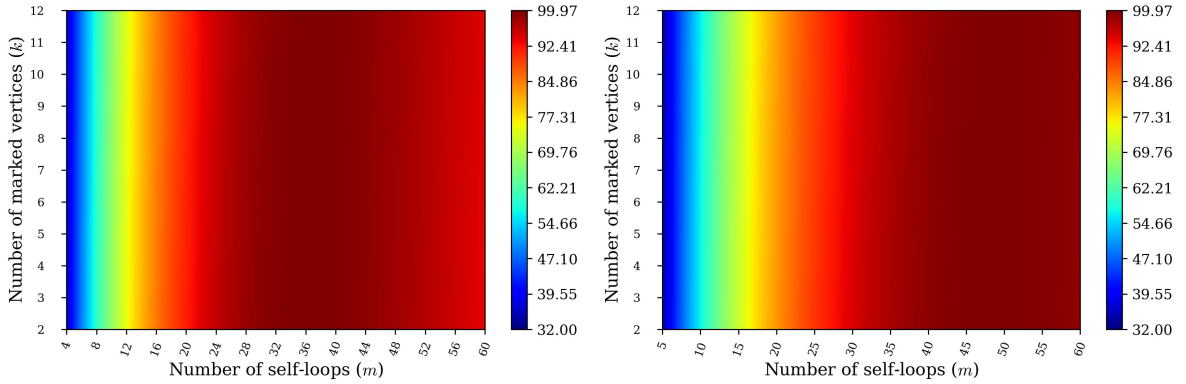
Finally, let us compare the results obtained from the partial inversion using all four weights. The results presented in Figures 1b and 1d show that the weights proposed by Rhodes and Wong [2020] and Souza et al. [2021], $l = n/N$ and $l = (n/N) \cdot k$ are not the ideal weights for MSLQW - PPI. However, with the use of the weights proposed in this work, $l = n^2/N$ and $l = (n^2/N) \cdot k$, we achieved the best results. We highlight the weight value $l = (n^2/N) \cdot k$. With this weight value, we obtained a stable behavior in most of the results. Table 3 shows the results of this comparison for the probability of success, number of self-loops, and coefficient of variation for all weights used in the MSLQW - PPI. Note that by modifying the weight scale and using various self-loops and partial phase inversion, it is possible to increase the probability of success.

Now, our discussions focus on cases where more than one self-loop is inverted. Preliminary results indicate that the phase inversion of a single self-loop $\circlearrowleft_{\tau=j}$ is sufficient to obtain the results presented in this work. However, the results also showed that it is possible to achieve maximum success probabilities close to 1 by inverting $1 < s < m$ self-loops.

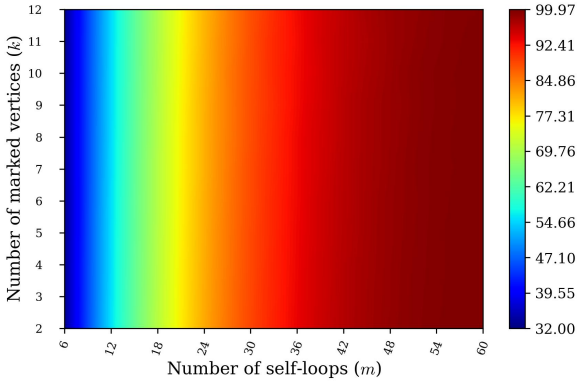
The results of the experiments show that as the number of inverted self-loops s increases, the number of self-loops m needed to achieve maximum success probabilities $p \approx 1$ also increases. However, it is also possible to find the required number of self-loops m to achieve maximum success probabilities close to 1.

The quantity of m is calculated as follows: $m = s \cdot n$, where s is the number of self-loops with its phases inverted, and n is the degree of the hypercube. New experiments were performed to confirm this hypothesis. Fig. 5 shows the result of MSLQW-PPI for $2 \leqslant s \leqslant 5$, and Fig. 6 shows the result of MSLQW-PPI for $s = 2, 3, 4, 5$. For the number of inverted self-loops $s = 2, 3, 4, 5$ and $n = 12$, $m = 24, 36, 48, 60$ self-loops were needed to achieve maximum success probabilities.

To obtain the complexity of the proposed algorithm, two analyses were performed. The first analysis is about knowing how the runtime complexity behaves as $N = 2^n$ is changed. The second analysis is about knowing how the runtime complexity behaves as m self-loops are added at each vertex of the hypercube. In both analyses, the weight $l = (n^2/N) \cdot k$ was used. Fig. 7 shows the results of the quantum walk applied to eleven hypercubes of degrees $n =$



(a) $s = 3, m = 36$ (b) $s = 4, m = 48$



(c) $s = 5, m = 60$

Figure 6: The success probability of MSLQW-PPI for weight value $l = (n^2/N) \cdot k$. In Figures (a), (b) and (c) $s = 3, 4, 5$ and $4 \leqslant m \leqslant 60$, respectively.

$\{10, ...20\}$ respectively. Fig. 7a presents the results obtained using only one self-loop, and Fig. 7b shows the results obtained with the MSLQW-PPI.

Considering the use of a single self-loop at each vertex of the hypercube we have a cost of square root. The adjustments show that the running time t is

$$t = c_1 \cdot \sqrt{((n + \mathrm{m}) \cdot N)^{c_2}} + c_3,$$

where c_1, c_2 and $c_3 \in \mathbb{R}$. Then, numerical simulations suggest that the running time is $O(\sqrt{((n + m) \cdot N)}$. This also occurs when using multiple self-loops and partial phase inversion. Maximum success probabilities stay close to 1 as n changes. Note that when we fix the number of self-loops the running time of the algorithm does not on the dimensions of the hypercube. The fit curves are

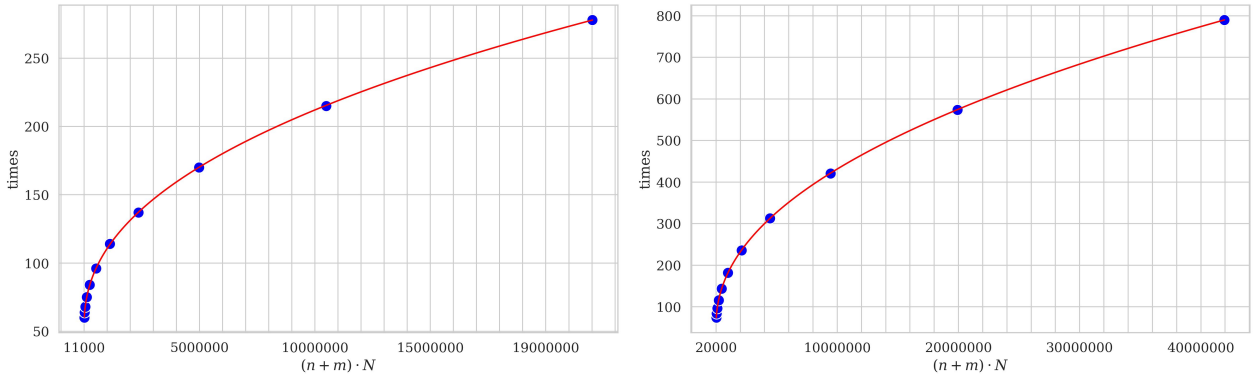
$$t_a = 0.1408 \cdot \sqrt{((n + \mathrm{m}) \cdot N)^{0.873}} + 52.0675$$

and

$$t_b = 0.2105 \cdot \sqrt{((n + \mathrm{m}) \cdot \mathrm{N})^{0.9299}} + 53.0933.$$

Where t_a is the cost for simulations with only one self-loop – Fig. 7a, and t_b is the cost of simulation with the best self-loop quantity (between 1 to 30) for each hypercube size – Fig. 7b.

Once the number of vertices of the hypercube is defined, the complexity is logarithmic as we can see. Fig. 8 shows the results of the MSLQW-PPI applied to five hypercubes of degrees $n = \{12, ...16\}$ respectively. On each of these



(a) One self-lop.

(b) Multiple self-loops.

Figure 7: The time complexity of the algorithm relative to the size of the hypercube. The solid red line represents the estimated curve and the blue dots are the numerical simulation values of the quantum walk.

Table 4: Algorithm complexity adjustments, m is the number of self-loops.

$c_1 \cdot \log((n+m)\cdot N + c_2) + c_3$			
n	c_1	c_2	c_3
12	11.6237	-49296.2873	-28.8548
13	17.0371	-105453.4444	-81.3999
14	23.5898	-227612.0627	-148.5875
15	34.2267	-484336.2147	-268.0318
16	49.4194	-1024918.6020	-451.1252

hypercubes, thirty quantum walks were performed. The results show how many times the evolution operator needs to be applied to obtain the maximum probability of success concerning the number of self-loops for each of the hypercube sizes. The adjustments show that the running time t is

$$t = c_1 \cdot \log((n+m)\cdot N + c_2) + c_3,$$

where c_1, c_2 and $c_3 \in \mathbb{R}$. Then, numerical simulations suggest that the running time is $O(\log((n+\mathrm{m})\cdot N))$, where n is a constant representing the dimension of the hypercube in use, m is the number of self-loops, and N is also a constant representing the total number of vertices of the hypercube. The run times are shown in Table 4. However, if all constants are discarded, the computational cost for this situation can be approximated to $O(\log(m))$.

6 Conclusions

The lackadaisical quantum walk has a strong dependence on the self-loop weight value. We can see this fact in the results presented in all figures. For the cases where the phases of all self-loops are flipped, using a single self-loop per vertex is better. We can affirm the same for the results shown in Fig. 1.

When Wong [2015] proposed the lackadaisical quantum walk, he included m integer self-loops at each vertex in the complete graph, which according to Rhodes and Rhodes and Wong [2020] is equivalent to having m unweighted self-loops. However, Wong [2017] redefined the lackadaisical quantum walk, where each vertex with m integer self-loops can be reduced to a quantum walk where each vertex has a single self-loop of real weight l. Note that the number of self-loops and the weight value l are the main characteristics of this quantum walk. Our strategy differs in that we apply partial phase inversion and consider a real-valued weight l and equally distribute it in m self-loops in each vertex, *i.e.*, m self-loops of non-integer weight value. In addition to using the two existing weight values in the literature to perform our experiments, we can contribute to the proposal of two new weight values. As the vertex degree is one of the parameters used in many works to define the weights assigned to self-loops, we suggest a modification that adds an exponent 2 to the numerator.

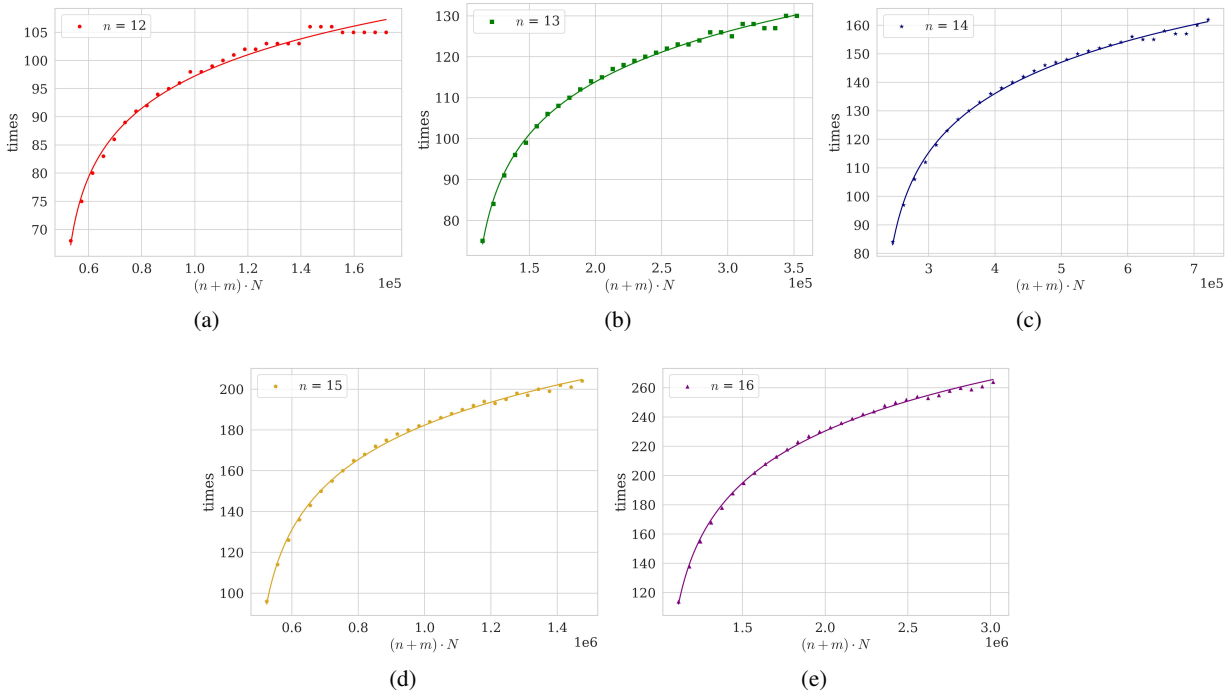


Figure 8: The algorithm's time complexity concerning the number of self-loops at each vertex. The solid lines represent the estimated curves. The points are the values from numerically simulating the quantum walk. For each figure, n is a constant.

Quantum interference is essential in developing quantum algorithms [McMahon, 2007]. The interference caused by the phase inversion operation, jointly with the average inversion operation, amplifies the target states' amplitudes. It allows Grover's search algorithm a certain probability of finding the desired state [Grover, 1996]. The self-loops are redundant elements within the structure that compose the states of the quantum system.

When we interfered jointly by flipping the phase of the m self-loops, making them indistinguishable, we saw that it was equivalent to using a single self-loop. However, when we invert the phase of one of the m self-loops, we empirically observe that the behavior caused by the constructive and destructive interference between the m self-loops are analogous to those observed between the states that represent the vertices as a whole, *i.e.*, most of the energy of the m self-loops is retained in the phase-inverted self-loop. The results indicate that the phase inversion of a single self-loop $\circlearrowleft_{\tau=j}$ is sufficient to obtain the results presented in this work. However, it is possible to find maximum success probabilities close to 1 by inverting $1 < s < m$ self-loops and using a total of m $= s * n$ self-loops.

To obtain the average behavior based on the relative position of the marked vertices, each simulation was performed on a sample that contains a number k of distinctly marked vertices, that is, without replacement. Thus, verifying whether the relative position of non-adjacent marked vertices influences the results is possible. According to the results, the relative position of non-adjacent marked vertices does not significantly affect considering a numerical precision of four digits. Furthermore, the coefficient of variation indicates that the number m of self-loops and the number k of marked vertices influence the results. We used the coefficient of variation to analyze the results' dispersion level. We observe that for MSLQW-PPI when we have the slightest standard deviations, the maximum probabilities of success scale to values close to 1. The percentage variations around the mean are not significant. However, the behavior is stable for MSLQW-PPI, where there are slight variations. The results are more influenced by the number of marked vertices and the number of self-loops than by the relative position of the non-adjacent marked vertices. As we can see from the results, the multiple self-loops are an essential tool to improve the success probability of searching multiple marked vertices on the hypercube. Parameters such as weight value, weight distribution strategies, and phase inversion operation contributed to the results of this work.

In this work, the marked vertices are all non-adjacent. According to Souza et al. [2021], the type of marked vertices can interfere with the result of the quantum walk on the hypercube. Therefore, as a supplementary work, we have used the MSLQW-PPI to search for multiple adjacent marked vertices on the hypercube. In this sense, some preliminary

results can be found in the pre-print [Souza et al., 2023]. We also intend to apply this methodology to evaluate the MSLQW-PPI in other d-regular structures, for example, the Johnson graph [Peng et al., 2024]. We intend to analyze other exponent values α for the composition of the weights $l = \{n^\alpha/N, (n^\alpha/N) \cdot k\}$, including real values, *i.e.*, $\{\alpha \in \mathbb{R} \mid \alpha \neq 1\}$. It is possible to use an evolutionary search algorithm to define the best exponent α value capable of maximizing the probability of success and minimizing the number of self-loops influencing the current proposal. We intend to propose other strategies to distribute weight values, such as providing distinct self-loop weight values for marked vertices. Another possible path for deeper investigation is to consider the partial inversion of edges i that are not self-loops. Furthermore, proposing a variation of Grover's search algorithm could also be a promising direction for future efforts.

Acknowledgments

Acknowledgments to the Science and Technology Support Foundation of Pernambuco (FACEPE)- Brazil, The Brazilian National Council for Scientific and Technological Development (CNPq), and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 by their financial support to the development of this research.

References

Neil Shenvi, Julia Kempe, and K Birgitta Whaley. Quantum random-walk search algorithm. *Physical Review A*, 67 (5):052307, 2003. doi:https://doi.org/10.1103/PhysRevA.67.052307.

Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996. doi:https://doi.org/10.1145/237814.237866.

V Potoček, Aurél Gábris, Tamás Kiss, and Igor Jex. Optimized quantum random-walk search algorithms on the hypercube. *Physical Review A*, 79(1):012325, 2009. doi:https://doi.org/10.1103/PhysRevA.79.012325.

Birgit Hein and Gregor Tanner. Quantum search algorithms on the hypercube. *Journal of Physics A: Mathematical and Theoretical*, 42(8):085303, 2009. doi:https;//doi.org/10.1088/1751-8113/42/8/085303.

Stephan Hoyer and David A Meyer. Faster transport with a directed quantum walk. *Physical Review A*, 79(2):024307, 2009. doi:https://doi.org/10.1103/PhysRevA.79.024307.

Thomas G Wong. Grover search with lackadaisical quantum walks. *Journal of Physics A: Mathematical and Theoretical*, 48(43):435304, 2015. doi:https://doi.org/10.1088/1751-8113/48/43/435304.

Thomas G Wong. Coined quantum walks on weighted graphs. *Journal of Physics A: Mathematical and Theoretical*, 50(47):475301, 2017. doi:https://doi.org/10.1088/1751-8121/aa8c17.

Mason L Rhodes and Thomas G Wong. Search on vertex-transitive graphs by lackadaisical quantum walk. *Quantum Information Processing*, 19(9):1–16, 2020. doi:https://doi.org/10.1007/s11128-020-02841-z.

Luciano S Souza, Jonathan H A Carvalho, and Tiago A E Ferreira. Lackadaisical quantum walk in the hypercube to search for multiple marked vertices. In André Britto and Karina Valdivia Delgado, editors, *Brazilian Conference on Intelligent Systems*, pages 249–263, Cham, 2021. Springer International Publishing. doi:https://doi.org/10.1007/978-3-030-91702-9_17.

Huiquan Wang, Jie Zhou, Junjie Wu, and Xun Yi. Adjustable self-loop on discrete-time quantum walk and its application in spatial search. *arXiv preprint*, 2017. doi:https://doi.org/10.48550/arXiv.1707.00601.

Mason L Rhodes and Thomas G Wong. Quantum walk search on the complete bipartite graph. *Physical Review A*, 99 (3):032301, 2019a. doi:https://doi.org/10.1103/PhysRevA.99.032301.

Jacob Rapoza and Thomas G Wong. Search by lackadaisical quantum walk with symmetry breaking. *Physical Review A*, 104(6):062211, 2021. doi:https://doi.org/10.1103/PhysRevA.104.062211.

David McMahon. *Quantum computing explained*. John Wiley & Sons, Hoboken, New Jersey, 2007.

Thomas G Wong. Faster search by lackadaisical quantum walk. *Quantum Information Processing*, 17(3):1–9, 2018. doi:https://doi.org/10.1007/s11128-018-1840-y.

Mason L Rhodes and Thomas G Wong. Search by lackadaisical quantum walks with nonhomogeneous weights. *Physical Review A*, 100(4):042303, 2019b. doi:https://doi.org/10.1103/PhysRevA.100.042303.

Pulak Ranjan Giri and Vladimir Korepin. Lackadaisical quantum walk for spatial search. *Modern Physics Letters A*, 35(08):2050043, 2020. doi:https://doi.org/10.1142/S0217732320500431.

Jonathan H A Carvalho, Luciano S Souza, Fernando M Paula Neto, and Tiago A E Ferreira. On applying the lackadaisical quantum walk algorithm to search for multiple solutions on grids. *Information Sciences*, 622:873–888, 2023. doi:https://doi.org/10.1016/j.ins.2022.11.142.

Yakir Aharonov, Luiz Davidovich, and Nicim Zagury. Quantum random walks. *Physical Review A*, 48(2):1687, 1993. doi:https://doi.org/10.1103/PhysRevA.48.1687.

Andris Ambainis, Artūrs Bačkurs, Nikolajs Nahimovs, Raitis Ozols, and Alexander Rivosh. Search by quantum walks on two-dimensional grid without amplitude amplification. In *Conference on Quantum Computation, Communication, and Cryptography*, pages 87–97. Springer, 2012. doi:https;//doi.org/10.1007/978-3-642-35656-8_7.

Jingyuan Zhang, Yonghong Xiang, and Weigang Sun. A discrete random walk on the hypercube. *Physica A: Statistical Mechanics and its Applications*, 494:1–7, 2018. doi:https://doi.org/10.1016/j.physa.2017.12.005.

Amit Saha, Ritajit Majumdar, Debasri Saha, Amlan Chakrabarti, and Susmita Sur-Kolay. Faster search of clustered marked states with lackadaisical quantum walks. *Quantum Information Processing*, 21(8):1–13, 2022. doi:https://doi.org/10.1007/s11128-022-03606-6.

Hajime Tanaka, Mohamed Sabri, and Renato Portugal. Spatial search on johnson graphs by continuous-time quantum walk. *Quantum Information Processing*, 21(2):1–13, 2022. doi:https://doi.org/10.1007/s11128-022-03417-9.

Dengke Qu, Samuel Marsh, Kunkun Wang, Lei Xiao, Jingbo Wang, and Peng Xue. Deterministic search on star graphs via quantum walks. *Physical Review Letters*, 128(5):050501, 2022. doi:https://doi.org/10.1103/PhysRevLett.128.050501.

Salvador Elías Venegas-Andraca. Quantum walks: a comprehensive review. *Quantum Information Processing*, 11(5): 1015–1106, 2012. doi:https://doi.org/10.1007/s11128-012-0432-5.

Julia Kempe. Quantum random walks hit exponentially faster. *arXiv preprint*, 2002. doi:https://doi.org/10.48550/arXiv.quant-ph/0205083.

Cristopher Moore and Alexander Russell. Quantum walks on the hypercube. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 164–178, Berlin, Heidelberg, 2002. Springer. doi:https://doi.org/10.1007/3-540-45726-7_14.

Peter Høyer and Zhan Yu. Analysis of lackadaisical quantum walks. *Quantum Information and Computation*, 20 (13-14):1138–1153, 2020. doi:https://doi.org/10.26421/QIC20.13-14-4.

Zhan Yu. Searching faster using self-loops in quantum walks. *University of Calgary*, 2018. `https://prism.ucalgary.ca/server/api/core/bitstreams/406e4407-a7fb-4c51-a259-6ff3e0e8bd5f/content`. Accessed 5 May 2023.

Henrique C T Santos, Luciano S Souza, Jonathan H A Carvalho, and Tiago A E Ferreira. Pesc–parallel experiment for sequential code. *arXiv preprint*, 2023. doi:https://doi.org/10.48550/arXiv.2301.05770.

Luciano S Souza, Jonathan H A de Carvalho, Henrique C T Santos, and Tiago A E Ferreira. Search for multiple adjacent marked vertices on the hypercube by a quantum walk with partial phase inversion. *arXiv preprint*, 2023. doi:https://doi.org/10.48550/arXiv.2305.19614.

Fangjie Peng, Meng Li, and Xiaoming Sun. Lackadaisical discrete-time quantum walk on johnson graph. *Physica A: Statistical Mechanics and its Applications*, 635:129495, 2024. doi:https://doi.org/10.1016/j.physa.2024.129495.

A Evaluation of the Oracle application in three possible scenarios

Let us evaluate the application of the oracle in the three possible scenarios with respect to the edges. For each scenario, we also apply the oracle to unmarked vertices to show its general effectiveness.

Scenario 1 *The first scenario shows the oracle applied in states $|\circlearrowleft_j, \vec{x}\rangle$ which represents the self-loop to be inverted.*

• $\omega = \vec{x}$, $\epsilon \neq \circlearrowleft_j$, *and* $\{\circlearrowleft_\tau = \circlearrowleft_j\}$.

$$
\begin{aligned}
Q\sum_{j=0}^{s-1}|\circlearrowleft_j,\vec{x}\rangle &= (I_{(n+m)\cdot N}-2\,|\epsilon,\omega\rangle\langle\epsilon,\omega|-2\,|\circlearrowleft_{\tau=j},\omega\rangle\langle\circlearrowleft_{\tau=j},\omega|)\sum_{j=0}^{s-1}|\circlearrowleft_j,\vec{x}\rangle \\
&= \sum_{j=0}^{s-1}|\circlearrowleft_j,\vec{x}\rangle-2\,|\epsilon,\omega\rangle\langle\epsilon,\omega|\sum_{j=0}^{s-1}|\circlearrowleft_j,\vec{x}\rangle-2\,|\circlearrowleft_{\tau=j},\omega\rangle\langle\circlearrowleft_{\tau=j},\omega|\sum_{j=0}^{s-1}|\circlearrowleft_j,\vec{x}\rangle \\
&= \sum_{j=0}^{s-1}|\circlearrowleft_j,\vec{x}\rangle-2\,|\epsilon,\omega\rangle\cdot 0-2\,|\circlearrowleft_{\tau=j},\omega\rangle\cdot 1 \\
&= -\sum_{j=0}^{s-1}|\circlearrowleft_j,\vec{x}\rangle
\end{aligned}
\tag{20}
$$

• $\omega \neq \vec{x}$, $\epsilon \neq \circlearrowleft_j$, *and* $\{\circlearrowleft_\tau = \circlearrowleft_j\}$.

$$
\begin{aligned}
Q\sum_{j=0}^{s-1}|\circlearrowleft_j,\vec{x}\rangle &= \sum_{j=0}^{s-1}|\circlearrowleft_j,\vec{x}\rangle-2\,|\epsilon,\omega\rangle\langle\epsilon,\omega|\sum_{j=0}^{s-1}|\circlearrowleft_j,\vec{x}\rangle-2\,|\circlearrowleft_{\tau=j},\omega\rangle\langle\circlearrowleft_{\tau=j},\omega|\sum_{j=0}^{s-1}|\circlearrowleft_j,\vec{x}\rangle \\
&= \sum_{j=0}^{s-1}|\circlearrowleft_j,\vec{x}\rangle-2\,|\epsilon,\omega\rangle\cdot 0-2\,|\circlearrowleft_{\tau=j},\omega\rangle\cdot 0 \\
&= \sum_{j=0}^{s-1}|\circlearrowleft_j,\vec{x}\rangle
\end{aligned}
\tag{21}
$$

Scenario 2 *The second scenario shows the application of the oracle in the jth self-loop $|\circlearrowleft_j,\vec{x}\rangle$, where $\tau \neq j$.*

• $\omega = \vec{x}$, $\epsilon \neq \circlearrowleft_j$, *and* $\{\circlearrowleft_\tau \neq \circlearrowleft_j\}$.

$$
\begin{aligned}
Q\sum_{j=s}^{m-1}|\circlearrowleft_j,\vec{x}\rangle &= (I_{(n+m)\cdot N}-2\,|\epsilon,\omega\rangle\langle\epsilon,\omega|-2\,|\circlearrowleft_{\tau=j},\omega\rangle\langle\circlearrowleft_{\tau=j},\omega|)\sum_{j=s}^{m-1}|\circlearrowleft_j,\vec{x}\rangle \\
&= \sum_{j=s}^{m-1}|\circlearrowleft_j,\vec{x}\rangle-2\,|\epsilon,\omega\rangle\langle\epsilon,\omega|\sum_{j=s}^{m-1}|\circlearrowleft_j,\vec{x}\rangle-2\,|\circlearrowleft_{\tau=j},\omega\rangle\langle\circlearrowleft_{\tau=j},\omega|\sum_{j=s}^{m-1}|\circlearrowleft_j,\vec{x}\rangle \\
&= \sum_{j=s}^{m-1}|\circlearrowleft_j,\vec{x}\rangle-2\,|\epsilon,\omega\rangle\cdot 0-2\,|\circlearrowleft_{\tau=j},\omega\rangle\cdot 0 \\
&= \sum_{j=s}^{m-1}|\circlearrowleft_j,\vec{x}\rangle
\end{aligned}
\tag{22}
$$

• $\omega \neq \vec{x}$, $\epsilon \neq \circlearrowleft_j$, *and* $\{\circlearrowleft_\tau \neq \circlearrowleft_j\}$.

$$
\begin{aligned}
Q\sum_{j=s}^{m-1}|\circlearrowleft_j,\vec{x}\rangle &= \sum_{j=s}^{m-1}|\circlearrowleft_j,\vec{x}\rangle-2\,|\epsilon,\omega\rangle\langle\epsilon,\omega|\sum_{j=s}^{m-1}|\circlearrowleft_j,\vec{x}\rangle-2\,|\circlearrowleft_{\tau=j},\omega\rangle\langle\circlearrowleft_{\tau=j},\omega|\sum_{j=s}^{m-1}|\circlearrowleft_j,\vec{x}\rangle \\
&= \sum_{j=s}^{m-1}|\circlearrowleft_j,\vec{x}\rangle-2\,|\epsilon,\omega\rangle\cdot 0-2\,|\circlearrowleft_{\tau=j},\omega\rangle\cdot 0 \\
&= \sum_{j=s}^{m-1}|\circlearrowleft_j,\vec{x}\rangle
\end{aligned}
\tag{23}
$$

Scenario 3 *The third scenario shows the application of the oracle on the ith non-loop edge $|i, \vec{x}\rangle$, i.e., which is not a self-loop.*

• $\omega = \vec{x}$, $\epsilon = i$, *and* $\{\circlearrowleft_\tau \neq i\}$.

$$\begin{aligned} Q\,|i,\vec{x}\rangle &= (I_{(n+m)\cdot N} - 2\,|\epsilon,\omega\rangle\,\langle\epsilon,\omega| - 2\,|\circlearrowleft_{\tau=j},\omega\rangle\,\langle\circlearrowleft_{\tau=j},\omega|)\,|i,\vec{x}\rangle \\ &= |i,\vec{x}\rangle - 2\,|\epsilon,\omega\rangle\,\langle\epsilon,\omega|i,\vec{x}\rangle - 2\,|\circlearrowleft_{\tau=j},\omega\rangle\,\langle\circlearrowleft_{\tau=j},\omega|i,\vec{x}\rangle \\ &= |i,\vec{x}\rangle - 2\,|\epsilon,\omega\rangle\cdot 1 - 2\,|\circlearrowleft_{\tau=j},\omega\rangle\cdot 0 \\ &= -\,|i,\vec{x}\rangle \end{aligned} \tag{24}$$

• $\omega \neq \vec{x}$, $\epsilon = i$, *and* $\{\circlearrowleft_\tau \neq i\}$.

$$\begin{aligned} Q\,|i,\vec{x}\rangle &= |i,\vec{x}\rangle - 2\,|\epsilon,\omega\rangle\,\langle\epsilon,\omega|i,\vec{x}\rangle - 2\,|\circlearrowleft_{\tau=j},\omega\rangle\,\langle\circlearrowleft_{\tau=j},\omega|i,\vec{x}\rangle \\ &= |i,\vec{x}\rangle - 2\,|\epsilon,\omega\rangle\cdot 0 - 2\,|\circlearrowleft_{\tau=j},\omega\rangle\cdot 0 \\ &= |i,\vec{x}\rangle \end{aligned} \tag{25}$$

Consider Equations 20 and 21. Note that the phase of the self-loops $|\circlearrowleft_j\rangle$ is inverted only when $\omega = \vec{x}$, *i.e.*, if $|\mathbf{x}\rangle$ contains the target state. According to Equations 22 and 23, the jth self-loop, where $\tau \neq j$, stay unchanged even if $|\mathbf{x}\rangle$ contains the target state. The same behavior observed in Scenario 1 occurs with the ith non-loop edge in Scenario 3 according to Equations 24 and 25. The phase will only be inverted if $|\mathbf{x}\rangle$ contains the target state. As we can see, the oracle is able to partially invert the phase of the target state, based on position and edge information.

Search for Multiple Adjacent Marked Vertices on the Hypercube by Quantum Walk with Partial Phase Inversion

A preprint can be found at: https://arxiv.org/pdf/2305.19614

SEARCH FOR MULTIPLE ADJACENT MARKED VERTICES ON THE HYPERCUBE BY A QUANTUM WALK WITH PARTIAL PHASE INVERSION

A PREPRINT

Luciano S. de Souza*
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
Recife, Brasil
luciano.serafim@ufrpe.br

Jonathan H. A. de Carvalho
Centro de Informática
Universidade Federal de Pernambuco
Recife, Brasil
jhac@cin.ufpe.br

Henrique C. T. Santos
Instituto Federal de Educação, Ciência e
Tecnologia de Pernambuco
Recife, Brasil
henrique.santos@recife.ifpe.edu.br

Tiago A. E. Ferreira
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
Recife, Brasil
tiago.espinola@ufrpe.br

November 18, 2024

ABSTRACT

Quantum walks provide an efficient tool for the construction of new quantum search algorithms. In this paper, we analyze the application of the Multiself-loop Lackadaisical Quantum Walk on the hypercube that uses partial phase inversion of the target state to search for multiple adjacent marked vertices. We consider two scenarios and one of them evaluates the influence of the relative position of non-adjacent marked vertices on the search results. The use of self-loops and the composition of their weights are an essential part of the construction process of new quantum search algorithms based on lackadaisical quantum walks, however, other aspects have been considered, such as, for example, the type of marked vertices. It is known that part of the energy of a quantum system is retained in states adjacent to the target state. This behavior causes the amplification of these states where the sum of probability amplitudes reaches values equivalent to those of the target state, reducing their chances of being observed. Here we show experimentally that with the use of partial phase inversion of the target state, it is possible to amplify its probability amplitudes even in scenarios with adjacent marked vertices reaching maximum success probabilities with values close to 1. We also show that the relative position of the non-adjacent marked vertices did not significantly influence the results. The lackadaisical quantum walk generalization in using multiple self-loops to only a single self-loop and the ideal composition of a weight value was sufficient to obtain advances to quantum search algorithms based on quantum walks. However, the results presented here show that many other aspects need to be taken into account for the construction of new quantum algorithms. It was possible to add gains in the maximum probabilities of success compared to other results found in the literature. In one of the most significant cases, the probability of success increased from $p \approx 0.38$ to $p > 0.99$. Therefore, the use of partial phase inversion of target states brings new contributions to the development of new quantum search algorithms based on quantum walks and the use of multiple self-loops.

*R. Dom Manuel de Medeiros, s/n, Dois Irmãos – Recife, Pernambuco – Brasil

Keywords Quantum Computing · Quantum Walks · Quantum Search Algorithm · Lackadaisical Quantum Walk · Multiple Self-loops · Adjacent Marked Vertices · Partial Phase Inversion

1 Introduction

Many advances have been achieved since the publication of the article by Aharonov et al. [1993], which is considered the first in quantum walks. One of the first quantum search algorithms based on quantum random walks was designed by Shenvi et al. [2003], which defined the quantum walks as one of the most promising resources and an intuitive framework for building new quantum algorithms. Many other works on quantum walks have been developed since this moment [Ambainis et al., 2004, Potoček et al., 2009, Hein and Tanner, 2009, Ambainis et al., 2012].

Amongst many proposed works in quantum walks, Wong [2015] developed a quantum search algorithm called lackadaisical quantum walks - LQW, an analog of classical lazy random walks in which the quantum walker has a chance to stay at the current vertex position by introducing m self-loops of integer weight l at each vertex of the complete graph. This proposal was altered by Wong [2017] where the m self-loops were generalized to one self-loop of non-integer weight. In turn, Souza et al. [2023] proposed a new quantum search algorithm based on the LQW called Multiself-Loop Lackadaisical Quantum Walk - MSLQW, which uses m self-lops in each vertex on the hypercube with weight value $l = l' \cdot m$, and the partial phase inversion of the target state to research multiple marked vertices. The weight value $l' \in \mathbb{R}$ and $m \in \mathbb{Z}$.

However, some other studies indicate that the type of marked vertices influences the results of quantum search algorithms, in particular, the adjacent marked vertices. According to Potoček et al. [2009], the final state of the algorithm designed by Shenvi et al. [2003] is mainly composed of the marked state but also has small contributions from its nearest neighbors, i.e., part of the probability amplitude is retained in adjacent vertices. Another behavior of quantum walks on the hypercube referring to adjacent marked vertices is the formation of stationary states [Nahimovs et al., 2019]. Souza et al. [2021] experimentally showed that adjacent marked vertices interfere with the results of the search for multiple marked vertices. Although they have proposed a new ideal value of weight $l = (d/N) \cdot k$, when there are adjacent marked vertices in the set of solutions occurs a decrease in the maximum probability of success.

Therefore, this work objective is to apply MSLQW-PPI to research multiple marked vertices in two scenarios. The first scenario analyzes research by multiple adjacent and no-adjacent vertices to verify that the relative position of non-adjacent vertices interferes with the search results. The second scenario analyzes research by multiple adjacent vertices. Based on the methodology used by Souza et al. [2023], the results presented in this work are promising. Comparing the results of this work with the results obtained by Souza et al. [2021] there was a gain in the maximum probability of success to values close to 1. Before some of the success probabilities reached only $p \approx 0.59$ and $p \approx 0.38$, the first and second scenarios respectively.

In their proposal, Souza et al. [2023] used the search for non-adjacent marked vertices with the phases of $1 \leqslant s < m$ inverted self-loops and $1 \leqslant m \leqslant 30$. The results indicated that, by inverting the phase of only one self-loop, it is possible to achieve maximum probabilities of success close to 1. Based on the results of Souza et al. [2023] we applied the particular case of MSLQW-PPI with $s = 1$ inverted self-loop. Compared to the results obtained in this work, the maximum success probabilities remain close to 1. The coefficient of variation was also used to evaluate the dispersion around the average relative position of the non-adjacent marked vertices. The coefficient of variation was also used to evaluate the dispersion around the average maximum probability according to the relative position of the non-adjacent marked vertices. The results indicate that the variation around the maximum mean probability of success is not significant. These results are important because they show that the partial inversion of the target state based on the use of multiple self-loops provides a new perspective of advances in the development of new quantum search algorithms.

This paper is organized as follows. In Section 2 we present some concepts about Multiself-loop Lackadaisical Quantum Walks in the hypercube. Section 3 the experiments are defined. Section 4 presents the results and discussion. Finally, in Section 5 are the conclusions.

2 Multi-self-loop lackadaisical quantum walk on the hypercube

The lackadaisical quantum walk is the quantum counterpart of the classical lazy random walk. This quantum algorithm was proposed by Wong [2015] and is obtained by adding a self-loop to each vertex of the graph. Then, the Hilbert space associated with the lackadaisical quantum walk in the hypercube is

$$\mathcal{H} = \mathcal{H}^{n+1} \otimes \mathcal{H}^{2^n}$$

where $\mathcal{H}^{n+1}$ is the Hilbert space associated with the quantum coin space, and $\mathcal{H}^{2^n}$ is the Hilbert space associated with nodes in the hypercube. According to Høyer and Yu [2020], in a n-regular graph by adding a self-loop of weight l to each vertex, the coined Hilbert space becomes

$$\mathcal{H}^{n+1} = \{|e_0\rangle, |e_1\rangle, \ldots, |e_{n-1}\rangle, |\circlearrowleft\rangle\}.$$

where e_i is a binary string of n bits with 1 in the i-th position [Kempe, 2002, Shenvi et al., 2003], and $|\circlearrowleft\rangle$ is the self-loop. Weighted self-loop accounting is done by modifying Grover's coin as follows

$$C = 2\,|s^C\rangle\,\langle s^C| - I_{(n+1)} \tag{1}$$

where

$$|s^C\rangle = \frac{1}{\sqrt{n+l}}\left(\sqrt{l}\,|\circlearrowleft\rangle + \sum_{i=0}^{n-1}|i\rangle\right). \tag{2}$$

The Lackadaisical Quantum Walk system in the hypercube starts as follows

$$|\Psi(0)\rangle = \frac{1}{\sqrt{N}}\sum_{\vec{x}}|\vec{x}\rangle \otimes |s^C\rangle. \tag{3}$$

Substituting Equation 2 into Equation 3 we obtain the initial state described in Equation 4.

$$|\Psi(0)\rangle = \frac{\sqrt{l}}{\sqrt{N}\times\sqrt{n+l}}\sum_{\vec{x}}|\vec{x},\circlearrowleft\rangle + \frac{1}{\sqrt{N}\times\sqrt{n+l}}\sum_{\vec{x}}\sum_{i=0}^{n-1}|\vec{x},i\rangle \tag{4}$$

The Multiself-loop lackadaisical quantum walk was proposed by Souza et al. [2023]. This quantum algorithm is obtained by adding m self-loops at each vertex of the hypercube and a partial phase inversion of the target state is applied. The Hilbert space associated with the lackadaisical quantum walk in the hypercube is

$$\mathcal{H} = \mathcal{H}^{n+m} \otimes \mathcal{H}^{2^n}.$$

Then, the Hilbert space associated with the coin space becomes

$$\mathcal{H}^{n+m} = \{|e_0\rangle, |e_1\rangle, \ldots, |e_{n-1}\rangle, |\circlearrowleft_0\rangle, |\circlearrowleft_1\rangle, \ldots, |\circlearrowleft_{m-1}\rangle\}.$$

To account for the weighted auto-loop, a modification is made to the Grover coin described in Equation 1 as follows

$$C = 2\,|s^C\rangle\,\langle s^C| - I_{(n+m)} \tag{5}$$

where

$$|s^C\rangle = \frac{1}{\sqrt{n+l}}\left(\sqrt{l'}\sum_{j=0}^{m-1}|\circlearrowleft_i\rangle + \sum_{i=0}^{n-1}|i\rangle\right) \tag{6}$$

and $l = l' \cdot m$. The Multiself-loop lackadaisical quantum walk system on the hypercube is also started according to Equation 3. Substituting Equation 6 into Equation 3, we obtain the initial state described in Equation 7.

$$|\Psi(t=0)\rangle = \frac{\sqrt{l'}}{\sqrt{n+l}\times\sqrt{N}}\sum_{j=0}^{m-1}\sum_{\vec{x}}|\circlearrowleft_j,\vec{x}\rangle + \frac{1}{\sqrt{n+l}\times\sqrt{N}}\sum_{i=0}^{n-1}\sum_{\vec{x}}|i,\vec{x}\rangle \tag{7}$$

The proposed modification of the oracle described in Equation 8 makes it possible to identify the components of the target state.

$$Q = I_{(n+\mathrm{m})\cdot N} - 2\sum_{\omega}\sum_{\epsilon=1}^{n}|\epsilon,\omega\rangle\,\langle\epsilon,\omega| - 2\sum_{\omega}\sum_{\tau}|\circlearrowleft_\tau,\omega\rangle\,\langle\circlearrowleft_\tau,\omega| \tag{8}$$

where $|\omega\rangle$ represents the marked vertex, ϵ represents an edge that is not a self-loop, and $\circlearrowleft_\tau$ is the self-loop that will have its phase inverted. Consider an arbitrary state which denotes the superposition of all edges

$$|\mathbf{x}\rangle = |\vec{x}, \circlearrowleft_0\rangle + |\vec{x}, \circlearrowleft_1\rangle + |\vec{x}, \circlearrowleft_2\rangle + \cdots + |\vec{x}, \circlearrowleft_{m-1}\rangle + |\vec{x}, 0\rangle + |\vec{x}, 1\rangle + |\vec{x}, 2\rangle + \cdots + |\vec{x}, n-1\rangle\,. \tag{9}$$

Consider the states $|\circlearrowleft_\tau\rangle$ as the target self-loops and $s = 1$. Applying the phase inversion operator, represented by Equation 10.

$$Q = I_{(n+\mathrm{m})\cdot N} - 2\sum_{\omega}\sum_{\epsilon=0}^{n-1} |\epsilon, \omega\rangle \langle \epsilon, \omega| - 2\sum_{\omega}\sum_{\tau} |\circlearrowleft_{\tau=0}, \omega\rangle \langle \circlearrowleft_{\tau=0}, \omega| \tag{10}$$

where I_{n+m} is the identity operator of dimension $n + m$ and $|\circlearrowleft_{\tau=0}\rangle$ as the target self-loop, we have

$$|\vec{\mathbf{x}}\rangle = -|\vec{x}, \circlearrowleft_0\rangle + |\vec{x}, \circlearrowleft_1\rangle + |\vec{x}, \circlearrowleft_2\rangle + \cdots + |\vec{x}, \circlearrowleft_{m-1}\rangle - |\vec{x}, 0\rangle - |\vec{x}, 1\rangle - |\vec{x}, 2\rangle - \cdots - |\vec{x}, n-1\rangle\,. \tag{11}$$

3 Experiment setup

According to Shenvi et al. [2003], Potoček et al. [2009], part of the probability amplitude of a quantum walk on the hypercube is retained at vertices adjacent to a marked vertex, and if two marked vertices on the hypercube are adjacent stationary states are formed [Nahimovs, 2019]. Souza et al. [2021] showed that adjacent marked vertices interfere with the search performance. Therefore, the experiments performed in this work are divided into the following two scenarios. In the first scenario, we consider both adjacent and non-adjacent marked vertices. In the second scenario, we consider only the adjacent marked vertices.

3.1 Definition of marked vertex samples

According to the definitions of the hypercube, two vertices are adjacent if the Hamming distance between them is 1. Non-adjacent vertices are those that have a Hamming distance of at least 2 from any other vertex. In this way, we define the set of marked vertices to execute the simulations. The set of marked vertices is divided into $M_{k,\gamma}$ groups of samples with k vertices and j samples.

The first set is formed by the vertices that are both adjacent and non-adjacent. This set is divided into twelve groups of one hundred samples. For each sample of k adjacent vertices, other $k-1$ non-adjacent vertices are also marked, and thirty MSLQW-PPI are performed. Therefore, thirty-six hundred simulations are performed. Every hundred simulations we preserve the same k adjacent marked vertices and vary the locations of the $k-1$ non-adjacent marked vertices, for example, if $k = 3$ we have two adjacent marked vertices and one non-adjacent marked vertex,

$$M_{3,100} = [\{0, 1, 1128\}_1, \{0, 1, 2950\}_2, \ldots, \{0, 1, 1470\}_{100}]$$

Every one hundred new simulations, k new adjacent vertices and $(k-1)\cdot 100$ new non-adjacent vertices are marked and added to the group until $k + (k-1) = 5, 7, \ldots, 25$.

The second set is formed by the adjacent vertices. This set is divided into twelve groups of one sample that contain between 2 and 13 marked vertices. To search for adjacent vertices, twelve simulations are performed. Initially, we have two marked vertices and at each new simulation a new vertex is marked and added to the new group as follows

$$M_{2,1} = \{0, 1\}, M_{3,1} = \{0, 1, 2\}, \ldots, M_{13,1} = \{0, 1, 2, \ldots, 1024, 2048\}$$

until all adjacent vertices are marked.

The samples have k distinct vertices, i.e., without replacement. The simulations performed in the set of the first scenario were necessary so that we could obtain the average behavior based on the relative position of the non-adjacent marked vertices and verify their influence on the results. In each simulation, thirty MSLQW-PPI are performed. The stop condition for a simulation occurs after each of the thirty walks obtains the maximum value of the probability amplitude. In each quantum walk, a number m of self-loops per vertex was defined, which varies between 1 and 30. The weight l is distributed by dividing its value between m equal parts.

3.2 Hardware and software setup for the simulations

The simulations were performed using the Parallel Experiment for Sequential Code - PESC [Henrique et al., 2023], to perform computational simulations distributed over a network. The platform provides a web interface for configuring simulation requests and manages the status and lifecycle of the request. The use of the platform simplified the simulations execution process, which was important to support the collect the study data. The tool is being developed for the instrumentation and optimization of the research group's computational experiments. The programming language used to write the algorithms was Python 3.7. All machines that were used in the simulations utilize the operational system, Ubuntu 18.04.6 LTS (Bionic Beaver), and have an HD of 500 GiB. Table 1 shows the machines' settings.

Table 1: Machine hardware configuration.

Adapted from: [Souza et al., 2023].

Machines	System RAM	System Processor
1	8 GiB	Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz
1	16 GiB	Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
2	32 GiB	Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz
2	32 GiB	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz

4 Results and discussion

As previously defined, the experiments are divided into two scenarios according to the type of marked vertices. In the first scenario, we have adjacent and non-adjacent marked vertices. As we analyzed the relative positional of the non-adjacent marked vertices, thirty-six thousand simulations were performed, which are divided into twelve groups with one hundred samples from k vertices, and to each sample was performed thirty quantum walks MSLQW. Then, the variability of the results was also analyzed and is represented in Fig. 2. In the second scenario, we only have adjacent marked vertices. Each node has the same number of adjacent vertices as the hypercube's degree number, therefore, twelve simulations were realized, and in each simulation, thirty MSLQW-PPI were also performed. The quantum walks vary according to the number of self-loops from one to thirty. The results are represented in Figures 1 and 3 respectively. They present the maximum probability of success according to the number of self-loops and marked vertices.

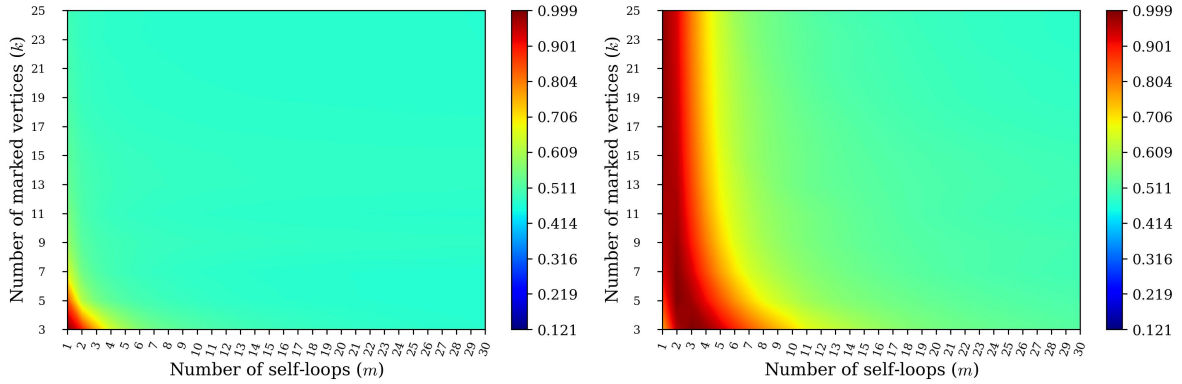
4.1 Analyzing the search with adjacent and non-adjacent marked vertices

Fig. 1a shows the probability of success for the weight $l = n/N$. Rhodes and Wong [2020] proposed this weight value to search a single vertex while Souza et al. [2021] used it to search multiple vertices, however, the results showed that this weight value is not ideal in this case. Souza et al. [2023] used this weight value and applied MSLQW-PPI to search for multiple non-adjacent marked vertices but there was no increase in the maximum probability of success. In this article, the maximum average probability obtained $p = 0.999$ with three marked vertices (two adjacent vertices and one non-adjacent vertex) and a single self-loop, which is a result close to that achieved by Souza et al. [2021] of $p = 0.997$. In both cases, as the number of marked vertices increases, the maximum probability of success decreases.

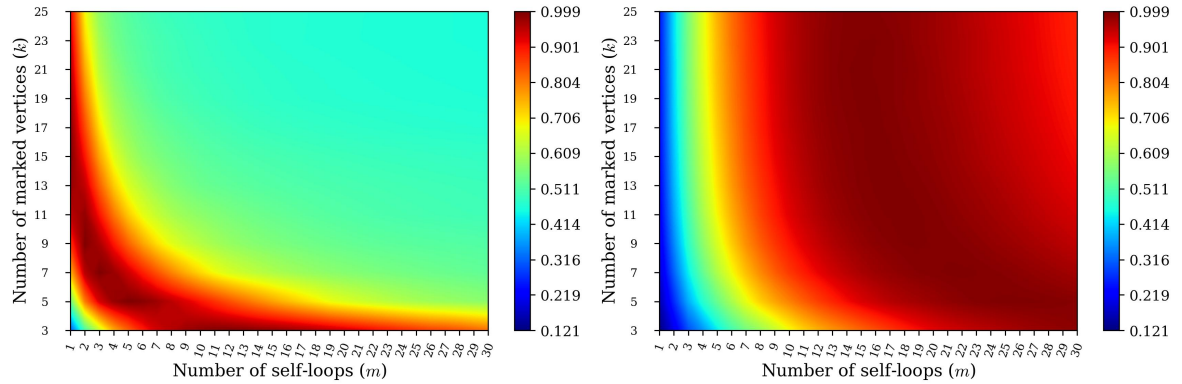
Fig. 1b shows the success probability using the weight $l = (n/N) \cdot k$. In cases where there are only non-adjacent marked vertices, for this weight value, only a single self-loop is needed [Souza et al., 2021, 2023]. Although, when there are adjacent marked vertices, it is necessary to increase the number of self-loops to obtain success probabilities close to 1 as we can see in Table 2. In some cases, we can observe that there was an improvement in the probability of success compared to the results obtained with the use of only one self-loop.

Comparing columns A and C of Table 2, we can see that from a certain quantity of self-loops, it is possible to obtain more significant probabilities than those achieved with the use of a single self-loop. It is necessary to use the least number of self-loops to obtain these results. Now, comparing columns B and C with at least two self-loops, it is possible to improve the maximum probability of success. However, for this self-loop weight value, as the number of marked vertices increased, only a single self-loop is needed to achieve probabilities of approximately $p \approx 0.98$.

Fig. 1c shows the probability of success in the search for adjacent and non-adjacent vertices using the weight value $l = n^2/N$. This weight is proposed by Souza et al. [2023] and is composed of the weight value proposed by Rhodes and Wong [2020] to search for one marked vertex plus an exponent in the element that represents the degree of the vertex in the numerator. Compared with the results found by Souza et al. [2021] and with the results shown in Fig. 1a,



(a) $l = n/N$ (b) $l = (n/N) \cdot k$



(c) $l = n^2/N$ (d) $l = (n^2/N) \cdot k$

Figure 1: The probability of success of the MSLQW-PPI to search for adjacent and non-adjacent marked vertices with $n = 12$ and $N = 4096$ vertices. (a) weight value $l = n/N$. (b) weight value $l = (n/N) \cdot k$. (c) weight value $l = n^2/N$. (d) weight value $l = (n^2/N) \cdot k$.

there was a significant improvement in the maximum probability of success for numbers $k > 3$ marked vertices. In this scenario, the success probabilities depend on the inversely proportional relationship between the number k of marked vertices and the number m of Self-loops. This means that as the number of marked vertices increases, the number of self-loops decreases and the other way around, however, the maximum probability of success continues above $p = 0.97$.

Another analysis of the results for the weight $l = n^2/N$ was performed. Two different scenarios are compared and some results are shown in Table 3. The results presented in column A refer to the scenario with only non-adjacent marked vertices obtained by Souza et al. [2023]. In the scenario presented in column B where we have both types of marked vertices for each k adjacent vertices, we have $k - 1$ non-adjacent vertices. Although there are adjacent marked vertices in the sample, the use of multiple self-loops guarantees, in some cases, maximum success probabilities close to 1.

Note that, in the case where there are only non-adjacent marked vertices, as the number of marked vertices increases the number of self-loops decreases. However, when there are marked adjacent vertices, a larger number of self-loops is needed to maintain the success probability close to the maximum. Comparisons made between different scenarios and the same weights show that the type of marked vertex influences the search result. However, although there are adjacent marked vertices in the sample, partial state inversion guarantees, in some cases, maximum success probabilities close to 1.

Now, we are going to analyze the case where the scene is the same but the weights are different. Considering the behavior of the probability of success in Figures 1c and 1d, we can see that not only the type of marked vertices influences the probability of success, but also the weight value. Note that the difference in weight composition, in this case, is the number of marked vertices. We can see that after the increase in the number of self-loops, overall we

Table 2: Cases for searching adjacent and non-adjacent marked vertices where more than one self-loop is required to obtain a maximum probability close to 1 using the weight $l = (n/N) \cdot k$ proposed by Souza et al. [2021]. Rows with (-) in column A mean that the same values are reached in the same rows in column B.

	A		B		C	
k	p	m	p	m	p	m
3	0.794	8	0.999	3	0.754	1
5	0.911	4	0.996	2	0.863	1
7	0.931	3	0.993	2	0.921	1
9	-	-	0.981	2	0.948	1
11	-	-	0.970	2	0.964	1

Table 3: Comparison between the probability of success and number of self-loops for two different scenarios for weight value $l = n^2/N$. Column A represents the results found by Souza et al. [2023] to search for non-adjacent marked vertices and column B to search for adjacent and non-adjacent marked vertices.

	A		B	
k	p	m	p	m
3	0.999	4	0.999	12
5	0.990	2	0.997	5
7	0.992	2	0.996	3
9	0.978	1	0.996	2
11	0.996	1	0.983	2

had a significant improvement in the probability of success. We can better see these results in Table 4 with shows the maximum probabilities of success and the number of self-loops according to the number of marked vertices and weight value. Comparing the results described in Table 4a and Table 4b, it is important to realize that the weight composition is very relevant. Although the type of marked vertices can influence the probability of success, with an ideal weight value along with an ideal number of self-loops, it is possible to improve the results. The exception was $k = 3$, where there was a reduction in the probability of success but still close to 1. The other bold lines show the cases with the more expressive improvements in the probability of success. In general, there was a significant increase in the number of self-loops.

Table 4: Ideal number of self-loops and maximum probability of success for searching adjacent and non-adjacent marked vertices. (4a) weight value $l = n^2/N$. (4b) weight value $l = (n^2/N) \cdot k$.

(a)

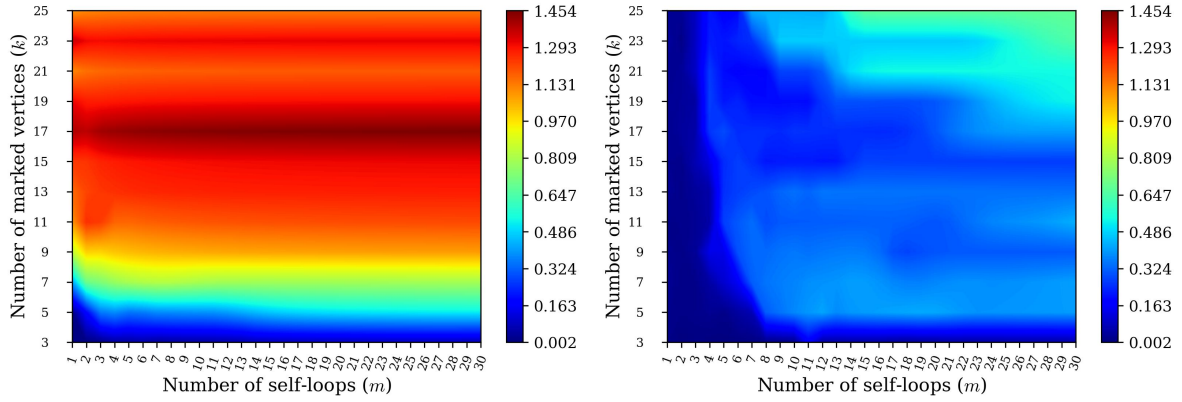
k	p	m	k	p	m
3	**0.999**	**12**	15	0.996	1
5	0.997	5	17	0.991	1
7	0.996	3	**19**	**0.980**	**1**
9	0.996	2	**21**	**0.960**	**1**
11	0.983	2	**23**	**0.941**	**1**
13	0.983	1	**25**	**0.920**	**1**

(b)

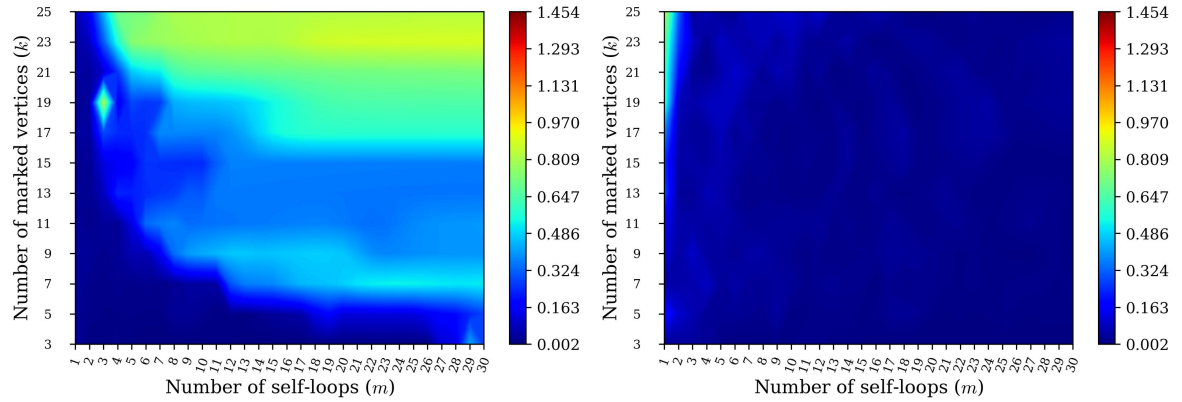
k	p	m	k	p	m
3	**0.988**	**30**	15	0.997	16
5	0.997	25	17	0.996	16
7	0.997	22	**19**	**0.997**	**15**
9	0.996	19	**21**	**0.997**	**15**
11	0.996	18	**23**	**0.996**	**16**
13	0.996	16	**25**	**0.995**	**15**

As in Souza et al. [2023], we analyzed whether the relative position of the non-adjacent marked vertices influences the results of the maximum probability of success. We also used the coefficient of variation to analyze the level of dispersion of the results. The relative position of the non-adjacent marked vertices also did not show significant influence considering a numerical precision of four digits. Fig. 2 shows the coefficient of variation for the results presented in Fig. 1. Variations around the mean value are small, however, the behavior shown is stable. The maximum

success probabilities close to 1 coincide with these small variations. Considering the weight value of the self-loop, in general, the weight $l = (n^2/N) \cdot k$ indicated minor variability.



(a) $l = n/N$ (b) $l = (n/N) \cdot k$



(c) $l = n^2/N$ (d) $l = (n^2/N) \cdot k$

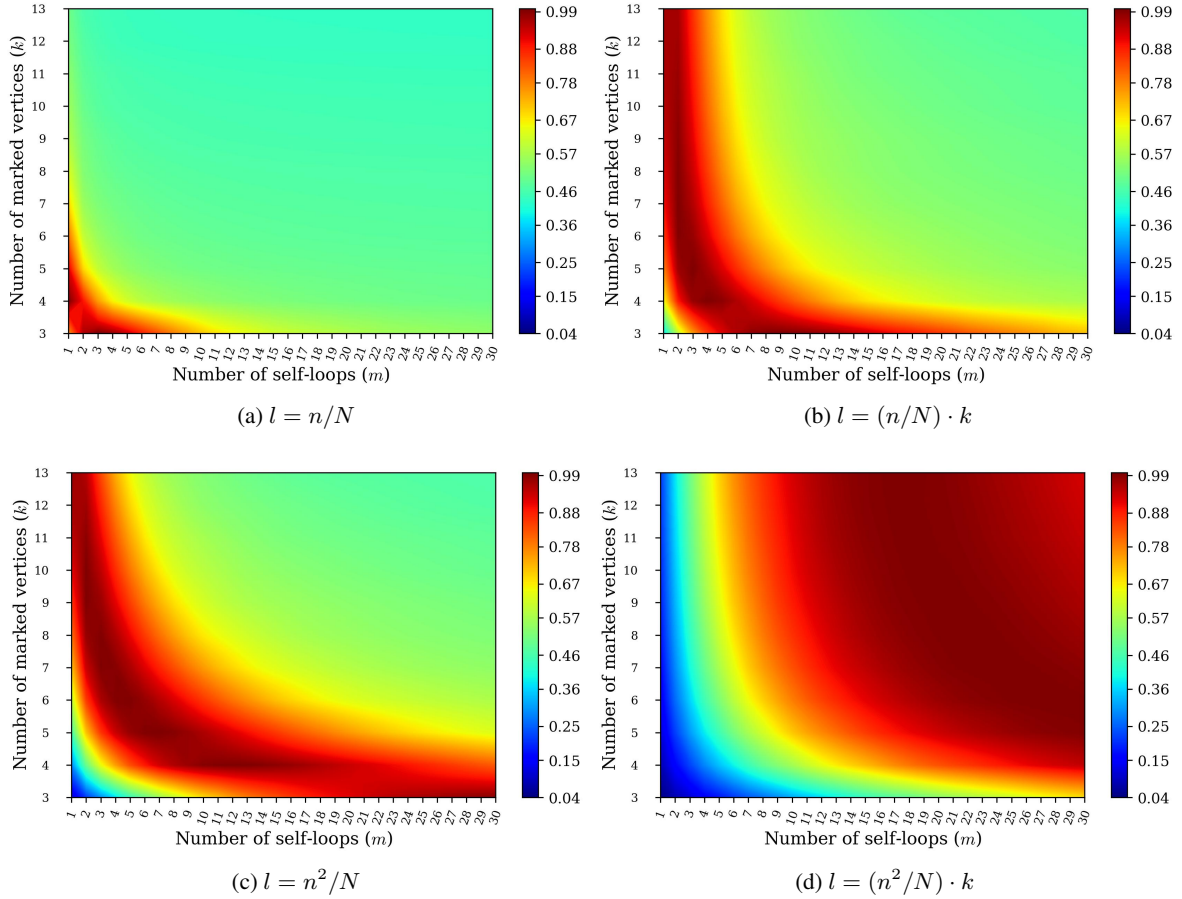
Figure 2: The coefficient of variation of the MSLQW-PPI to search for adjacent and non-adjacent marked vertices. The results are represented in percentage terms. (a), (b), (c), and (d) represents the coefficient of variation of the results presented in Figures 1a, 1b, 1c, and 1d for the weight values $l = n/N$, $l = (n/N) \cdot k$, $l = n^2/N$, and $l = (n^2/N) \cdot k$, respectively.

4.2 Analyzing the search with adjacent marked vertices

The simulations performed in the samples of the previous scenario were necessary so that we could obtain the average behavior based on the relative position of the non-adjacent marked vertices. In this scenario, let us analyze only adjacent vertices. According to Nahimovs et al. [2019], when there are two adjacent marked vertices, a stationary state occurs. In this case, the maximum probability of success obtained in our simulations was approximately $p = 0.02$ for all weights. Now, consider $k \geqslant 3$. Fig. 3a shows the probability of success for the weight $l = n/N$. Comparing the results presented by Souza et al. [2021] and Souza et al. [2023] to search for multiple marked vertices and a single self-loop we had an improvement in the success probability for $k = 3$ marked vertices which were $p = 0.745$ and evolved to $p = 0.999$ with $m = 3$ self-loops.

Fig. 3b shows the probability of success for the weight $l = (n/N) \cdot k$. Compared to the results obtained by Souza et al. [2021] for searching multiple adjacent marked vertices using a single self-loop, there was an improvement in the probability of success. As we can see in Table 5, two results are significant, the search for $k = 3$ marked vertices, with 9 self-loops allowed to increase the probability from $p = 0.386$ to $p = 0.999$. For $k = 4$ marked vertices, with 4 self-loops allowed to increase the probability from $p = 0.639$ to $p = 0.996$. Fig. 1b shows the results where both adjacent and non-adjacent vertices are marked. Analyzing the cases where the number of marked vertices is the same,

i.e., $k = \{3, 5, 7, 9, 11\}$ with $m = 2$ self-loops, although the scenarios are different, the maximum probability of success remained above $p = 0.99$.



(a) $l = n/N$ (b) $l = (n/N) \cdot k$

(c) $l = n^2/N$ (d) $l = (n^2/N) \cdot k$

Figure 3: The probability of success of the MSLQW-PPI to search for adjacent marked vertices with $n = 12$ and $N = 4096$ vertices. (a) weight value $l = n/N$. (b) weight value $l = (n/N) \cdot k$. (c) weight value $l = n^2/N$. (d) weight value $l = (n^2/N) \cdot k$.

Again, let us analyze two different scenarios for the same weight values. Fig. 3c shows the success probabilities for searching only adjacent marked vertices while Fig. 1c shows the success probabilities for searching adjacent and non-adjacent marked vertices both using the weight $l = n^2/N$. Note that the behaviors are very similar, however, in the scenario where there are only adjacent marked vertices, which is the case in Fig. 3c, a greater number of self-loops is necessary when the density of adjacent marked vertices is small. Table 6 shows the comparison between the success probabilities and the number of self-loops for the cases where the number of marked vertices is the same. Again, note that the results are similar except for $k = 3$, where there was a significant increase in the number of self-loops.

Now, consider Fig. 3d. It shows the success probabilities to search for adjacent marked vertices using the self-loop weight $l = (n^2/N) \cdot k$. Compared with the results of the scenario presented in Fig. 1d we notice a very similar behavior where for a small density of marked vertices a greater number of self-loops is necessary, however, when this density of marked vertices increases, the number of self-loops decreases to the point of approaching the results presented by Souza et al. [2023] for the same weight value $l = (n^2/N) \cdot k$. Table 7 shows the number of self-loops needed to obtain the maximum probabilities of success. Comparing with the results presented in Table 4b for the same numbers of marked vertices, it is possible to see that, a greater number of self-loops are required to achieve success probabilities close to 1 when there are only adjacent marked vertices. However, for $k = 3$, $m = 30$ was insufficient.

To obtain the complexity of the algorithm proposed by Souza et al. [2023]. applied to the search for adjacent vertices, two analyses were performed. The first analysis described in Fig. 4 shows the runtime complexity when $N = 2^n$ is changed. The second analysis described in Fig. 5 shows how the runtime complexity behaves when m self-loops

Table 5: Comparison between the success probabilities and the number of self-loops to search for adjacent marked vertices with the weight $l = (n/N) \cdot k$. (5a) shows the results obtained by Souza et al. [2021] using a single self-loop. (5b) shows the results in this work using multiple self-loops.

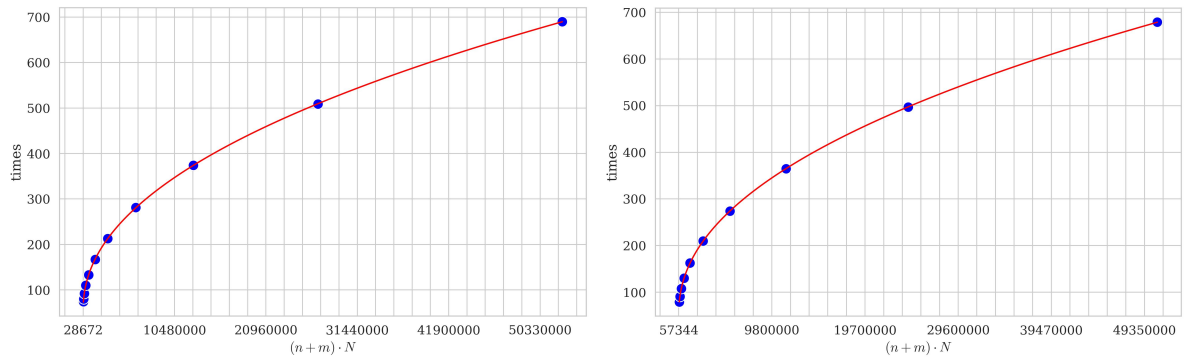
(a)

k	p
3	**0.386**
4	**0.639**
5	0.783
6	0.853
7	0.889
8	0.916
9	0.937
10	0.942
11	0.945

(b)

k	p	m
3	**0.999**	**9**
4	**0.996**	**4**
5	0.997	3
6	0.994	2
7	0.997	2
8	0.994	2
9	0.990	2
10	0.982	2
11	0.975	2

are added at each vertex of the hypercube. As in Souza et al. [2023], the computational cost when we consider the hypercube size is $O(\sqrt{((n+m)}))$. The computational cost where m self-loops varies is $O(\log(m))$.



(a) Adjacents.

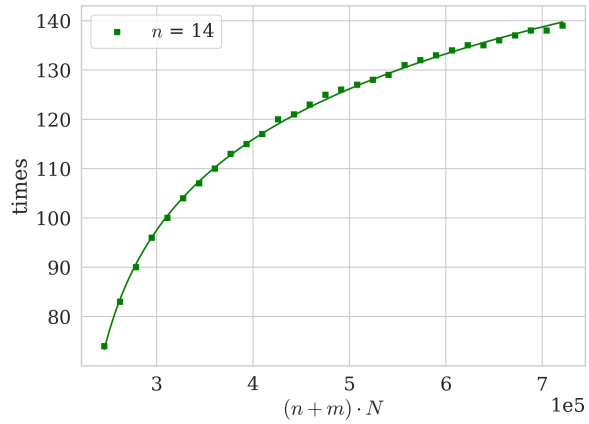
(b) Adjacents and non adjacents.

Figure 4: The time complexity of the algorithm relative to the size of the hypercube. The solid red line represents the estimated curve and the blue dots are the numerical simulation values of the quantum walk.

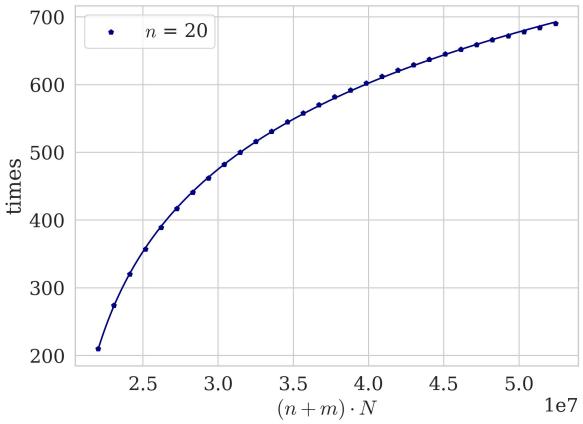
5 Conclusions

In this work, we analyzed the application of MSLQW-PPI in two scenarios based on the type of marked vertices: adjacent and non-adjacent. In the first scenario, the two types of marked vertices were searched. Here, we analyzed the relative position of the non-adjacent marked vertices to verify your influence on the results about adjacent vertices, for this, the coefficient of variation was also used to verify the dispersion of results around the maximum success probability mean value as Souza et al. [2023]. In the second scenario, we analyzed only the adjacent marked vertices. The dependence on the self-loop weight value is inherent to the lackadaisical quantum walk. Therefore, the composition of the weights is important. Thus, all analyses were made considering the four weight values. However, when applied to MSLQW-PPI a strategy of weight distribution is equally necessary because due to the use of the multiple self-loop. The strategy of weight distribution in this work was the same used by Souza et al. [2023], i.e., the weight $l = l'/m$.

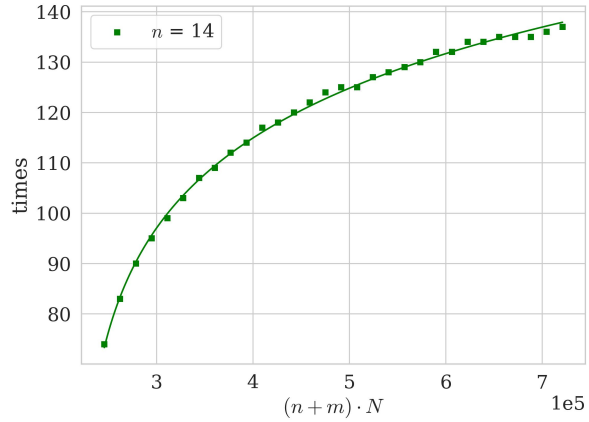
In the first scenario, to search for adjacent and non-adjacent vertices, according to the results, the relative position of non-adjacent marked vertices does not have a significant influence, considering a numerical precision of four digits. The results obtained by Souza et al. [2021, 2023] to search for non-adjacent marked vertices indicate that the weight values influenced the maximum success probabilities according to weight values $l = (n/N) \cdot k$ and $l = (n^2/N) \cdot k$.



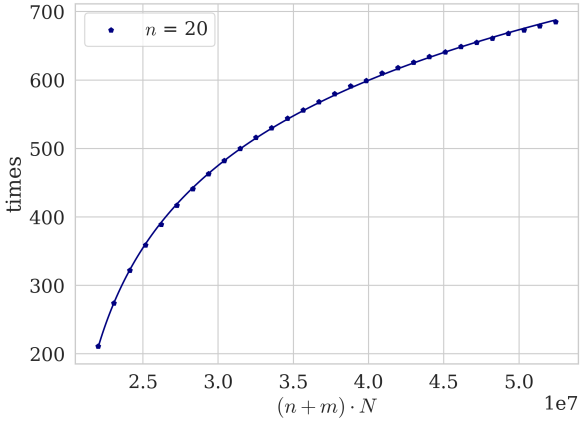
(a) Adjacents



(b) Adjacents and non adjacents



(c) Adjacents



(d) Adjacents and non adjacents

Figure 5: The algorithm's time complexity concerning the number of self-loops at each vertex. The solid lines represent the estimated curves. The points are the values from numerically simulating the quantum walk. For each figure, n is a constant.

Table 6: Comparison between the probability of success and number of self-loops for two different scenarios. The column for 1c represents the results of searching for adjacent and non-adjacent vertices and the column for 3c to search for adjacent vertices.

	Figures			
	1c		3c	
k	p	m	p	m
3	0.999	12	0.991	30
5	0.997	5	0.998	7
7	0.996	3	0.995	3
9	0.996	2	0.995	2
11	0.983	2	0.986	2

Table 7: Maximum success probability and number of self-loops to search for adjacent marked vertices with the weight $l = (n^2/N) \cdot k$.

k	p	m
3	**0.681**	**30**
4	**0.943**	**30**
5	0.995	30
6	0.999	28
7	0.998	24
8	0.998	21
9	0.998	20
10	0.996	19
11	0.997	18
12	0.997	18
13	0.997	17

Moreover, when we analyze the coefficient of variation, we saw that a minor variability coincides with the maximum probability of success. Fig. 2 shows that the number of marked vertices influences the results causing a more significant variability. In the second scenario, the results obtained in the search for adjacent marked vertices present very similar results to the search for adjacent and non-adjacent marked vertices, except in some cases. For example, when the weight $l = (n^2/N) \cdot k$, the number of self-loops increases considerably.

In summary, we conclude that for MSLQW-PPI, there is a dependence between the weight value, the vertex type, the number of marked vertices, and the number of self-loops needed to obtain success probabilities close to 1. In the search for adjacent and non-adjacent marked vertices, the weight that presented the best results was $l = (n^2/N) \cdot k$. In the search for adjacent marked vertices, three of four weights presented the best results: $l = (n/N) \cdot k$, $l = (n^2/N)$, and $l = (n^2/N) \cdot k$. The results presented in Fig. 3d show that from a certain k, the self-loops converge to a certain quantity. In future works, we intend to apply this methodology to evaluate the MSLQW-PPI in other d-regular structures with samples that contain adjacent marked vertices. We intend to verify the convergence of the number of multiple self-loops for a specific m from a certain k for the weight value $l = (n^2/N) \cdot k$ to search for adjacents marked vertices.

Acknowledgments

Acknowledgments to the Science and Technology Support Foundation of Pernambuco (FACEPE) Brazil, Brazilian National Council for Scientific and Technological Development (CNPq), and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 by their financial support to the development of this research.

References

Yakir Aharonov, Luiz Davidovich, and Nicim Zagury. Quantum random walks. *Physical Review A*, 48(2):1687, 1993. doi: \href{10.1103/PhysRevA.48.1687}.

Neil Shenvi, Julia Kempe, and K Birgitta Whaley. Quantum random-walk search algorithm. *Physical Review A*, 67(5): 052307, 2003. doi: 10.1103/PhysRevA.67.052307.

Andris Ambainis, Julia Kempe, and Alexander Rivosh. Coins make quantum walks faster. arXiv:quant-ph/0402107, 2004. URL `https://arxiv.org/abs/quant-ph/0402107`.

V Potoček, Aurél Gábris, Tamás Kiss, and Igor Jex. Optimized quantum random-walk search algorithms on the hypercube. *Physical Review A*, 79(1):012325, 2009. doi: 10.1103/PhysRevA.79.012325.

Birgit Hein and Gregor Tanner. Quantum search algorithms on the hypercube. *Journal of Physics A: Mathematical and Theoretical*, 42(8):085303, 2009. doi: 10.1088/1751-8113/42/8/085303.

Andris Ambainis, Artūrs Bačkurs, Nikolajs Nahimovs, Raitis Ozols, and Alexander Rivosh. Search by quantum walks on two-dimensional grid without amplitude amplification. In *Conference on Quantum Computation, Communication, and Cryptography*, pages 87–97. Springer, 2012. doi: https;//doi.org/10.1007/978-3-642-35656-8_7.

Thomas G Wong. Grover search with lackadaisical quantum walks. *Journal of Physics A: Mathematical and Theoretical*, 48(43):435304, 2015. doi: 10.1088/1751-8113/48/43/435304.

Thomas G Wong. Coined quantum walks on weighted graphs. *Journal of Physics A: Mathematical and Theoretical*, 50 (47):475301, 2017. doi: 10.1088/1751-8121/aa8c17.

Luciano S. Souza, Jonathan H. A. Carvalho, Henrique C. T. Santos, and Tiago A. E. Ferreira. Multiself-loop lackadaisical quantum walk with partial phase inversion. arXiv:2305.01121, 2023. URL `https://arxiv.org/abs/2305.01121`.

Nikolajs Nahimovs, Raqueline A M Santos, and K R Khadiev. Adjacent vertices can be hard to find by quantum walks. *Moscow University Computational Mathematics and Cybernetics*, 43(1):32–39, 2019.

Luciano S Souza, Jonathan H A Carvalho, and Tiago A E Ferreira. Lackadaisical quantum walk in the hypercube to search for multiple marked vertices. In André Britto and Karina Valdivia Delgado, editors, *Brazilian Conference on Intelligent Systems*, pages 249–263, Cham, 2021. Springer International Publishing. doi: 10.1007/978-3-030-91702-9_17.

Peter Høyer and Zhan Yu. Analysis of lackadaisical quantum walks. *Quantum Information and Computation*, 20 (13-14):1138–1153, 2020. doi: 10.26421/QIC20.13-14-4.

Julia Kempe. Quantum random walks hit exponentially faster. *arXiv preprint*, 2002. doi: 10.48550/arXiv.quant-ph/0205083.

Nikolajs Nahimovs. Lackadaisical quantum walks with multiple marked vertices. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 368–378. Springer, 2019.

C. T. Henrique, Luciano S. Souza, Jonathan H. A. Carvalho, and Tiago A. E. Ferreira. Pesc–parallel experiment for sequential code. *cs.DC*, 2023. doi: 10.48550/arXiv.2301.05770. URL `https://arxiv.org/abs/2301.05770`.

Mason L Rhodes and Thomas G Wong. Search on vertex-transitive graphs by lackadaisical quantum walk. *Quantum Information Processing*, 19(9):1–16, 2020. doi: 10.1007/s11128-020-02841-z.

General Conclusions

This thesis investigated the application of quantum walks in different contexts, addressing theoretical and practical challenges through four independent but interrelated studies. Each study represents a contribution that expands the possibilities for applying quantum walks.

In the paper "Classical Artificial Neural Network Training Using Quantum Walks as a Search Procedure", the application of quantum walks in the training of classical neural networks is presented. The integration of quantum computing and classical algorithms can offer significant computational advantages. This work contributed to the field of hybrid computing, demonstrating the potential of quantum walks as a tool to enhance machine learning techniques.

In the paper "Lackadaisical Quantum Walk in the Hypercube to Search for Multiple Marked Vertices", the lackadaisical quantum walk on the Hypercube was analyzed for searching multiple solutions. An ideal self-loop weight value for multiple solution searches on the hypercube was proposed. It was found that the presence of adjacent vertices among the solutions affects the algorithm's efficiency. This highlights a limitation of quantum walks and provides an understanding of scenarios where such algorithms can be applied effectively.

To mitigate the limitations observed in the previous work, an original approach was proposed in the paper "Multi-Self-Loop Lackadaisical Quantum Walk with Partial Phase Inversion" (MSLQW-PPI). This method is based on the lackadaisical quantum walk for partial phase inversion of multiple self-loops, and addresses the search for both adjacent and non-adjacent vertices. This proposal represents a methodological advance and expands the possibilities for manipulating systems based on quantum walks. The lackadaisical quantum walk shows a strong dependency on self-loop weights. Thus, two new weights were proposed for use with this novel approach.

In the paper "Search for Multiple Adjacent Marked Vertices on the Hypercube by Quantum Walk with Partial Phase Inversion", the proposed method was applied to the problem of search for adjacent solution vertices. The results showed that the proposed approach can overcome the limitations identified earlier, highlighting its potential to solve more complex and specific problems.

The studies are interconnected by their objectives to explore and push the limits of quantum walk applications in different contexts. The first work demonstrates how quantum

computing can complement classical algorithms, while the subsequent papers deepen the theoretical comprehension of quantum walks and propose advances to overcome practical limitations. These studies not only reinforce the flexibility of quantum walks but also provide a foundation for future hybrid and theoretical applications. The progression of the works reflects continuous development, culminating in a robust solution for search problems, even in contexts where solutions are adjacent.

Based on the results found in this thesis, several perspectives for future work can be outlined. Wong's quantum walk on the complete graph depends on a transformation that reduces an N-dimensional graph to a 4-dimensional space. Thus, it is possible to develop a quantum base-change operator that transforms an N-dimensional graph into an n-dimensional state. Additionally, it is necessary to describe a quantum oracle capable of determining whether a set of weights trains a classical artificial neural network. Since part of a quantum system's energy on a hypercube is retained in vertices adjacent to a solution, a potential future direction could be the analysis and proposal of a procedure for initializing synaptic weights in a set of vertices adjacent to a vertex that is a solution.

In this thesis, the MSLQW-PPI methodology was applied to the hypercube. Future work could apply MSLQW-PPI to other d-regular structures and evaluate its behavior. It was observed that the composition of self-loop weights takes into account the components characterizing the structures that define the space where the walk occurs. In this context, a possible future endeavor would be proposing a class of self-loop weights based on the value of the exponent α, as well as evaluating the inclusion of other parameters in defining self-loop weights, such as the number s of inverted self-loops or the total number m of self-loops required to achieve maximal success probabilities close to 1.

In the annexes, the article titled "On the Application of the Lackadaisical Quantum Walk Algorithm for Searching Multiple Solutions in Grids", published in Information Sciences and co-authored by the author, provides an analysis of several issues related to the lackadaisical quantum walk algorithm applied to n-dimensional grids. Premature stopping conditions, spatial density, and relative distance of solutions, as well as the appropriate weight adjustment for the self-loop, can compromise the search for multiple solutions. This article is closely related to the results found in this thesis and provides several insights on weight adjustments and the influence of solution adjacency on search results.

We believe this work has contributed to the continuous advancement of quantum

walks. Furthermore, these advances allowed for the observation of other relationships affecting the behavior and results of quantum walks, such as the impacts caused by the adjacency of solutions in search spaces, the reuse of multiple self-loops associated with weight distribution strategies among self-loops, and, especially, the partial application of phase inversion. Finally, this thesis offers promising perspectives for new approaches to quantum search algorithms based on quantum walks.

References

AHARONOV, Y.; DAVIDOVICH, L.; ZAGURY, N. Quantum random walks. *Physical Review A*, APS, v. 48, n. 2, p. 1687, 1993.

CARVALHO, J. H. A. et al. On applying the lackadaisical quantum walk algorithm to search for multiple solutions on grids. *arXiv preprint quant-ph/2106.06274*, 2021.

CARVALHO, J. H. A. et al. On applying the lackadaisical quantum walk algorithm to search for multiple solutions on grids. *Information Sciences*, Elsevier, v. 622, p. 873–888, 2023.

NAHIMOVS, N.; SANTOS, R. A. Lackadaisical quantum walks on 2d grids with multiple marked vertices. *arXiv preprint arXiv:2104.09955*, 2021.

QU, D. et al. Deterministic search on star graphs via quantum walks. *Physical Review Letters*, APS, v. 128, n. 5, p. 050501, 2022.

RHODES, M. L.; WONG, T. G. Quantum walk search on the complete bipartite graph. *Physical Review A*, APS, v. 99, n. 3, p. 032301, 2019.

RHODES, M. L.; WONG, T. G. Search on vertex-transitive graphs by lackadaisical quantum walk. *Quantum Information Processing*, Springer, v. 19, n. 9, p. 1–16, 2020.

SAHA, A. et al. Faster search of clustered marked states with lackadaisical quantum walks. *Quantum Information Processing*, Springer, v. 21, n. 8, p. 275, 2022.

SHENVI, N.; KEMPE, J.; WHALEY, K. B. Quantum random-walk search algorithm. *Physical Review A*, APS, v. 67, n. 5, p. 052307, 2003.

SILVA, R. R. da. *Utilizaçao do Algoritmo de Grover para o Treinamento de Redes Neurais Artificiais Clássicas*. 93 p. Dissertation (Mestrado em Ciência da Computação) — Universidade Federal Rural de Pernambuco, Recife, PE, 2014.

SOUZA, L. S.; CARVALHO, J. H. A.; FERREIRA, T. A. E. Classical artificial neural network training using quantum walks as a search procedure. *IEEE Transactions on Computers*, IEEE, 2021.

SOUZA, L. S.; CARVALHO, J. H. A.; FERREIRA, T. A. E. Lackadaisical quantum walk in the hypercube to search for multiple marked vertices. In: BRITTO, A.; DELGADO, K. V. (Ed.). *Brazilian Conference on Intelligent Systems*. Cham: Springer International Publishing, 2021. p. 249–263.

TANAKA, H.; SABRI, M.; PORTUGAL, R. Spatial search on johnson graphs by discrete-time quantum walk. *Journal of Physics A: Mathematical and Theoretical*, IOP Publishing, v. 55, n. 25, p. 255304, 2022.

WONG, T. G. Grover search with lackadaisical quantum walks. *Journal of Physics A: Mathematical and Theoretical*, v. 48, n. 43, p. 435304, 2015.

WONG, T. G. Coined quantum walks on weighted graphs. *Journal of Physics A: Mathematical and Theoretical*, IOP Publishing, v. 50, n. 47, p. 475301, 2017.

ZHANG, J.; XIANG, Y.; SUN, W. A discrete random walk on the hypercube. *Physica A: Statistical Mechanics and its Applications*, Elsevier, v. 494, p. 1–7, 2018.

ANNEXES

arXiv:2106.06274v2 [quant-ph] 9 Jan 2023

On Applying the Lackadaisical Quantum Walk Algorithm to Search for Multiple Solutions on Grids

Jonathan H. A. de Carvalho*[1], Luciano S. de Souza[2], Fernando M. de Paula Neto[1], and Tiago A. E. Ferreira[2]

[1]Centro de Informática, Universidade Federal de Pernambuco, Recife, Pernambuco, Brazil
{*jhac,fernando*}*@cin.ufpe.br*
[2]Departamento de Estatística e Informática, Universidade Federal Rural de Pernambuco, Recife, Pernambuco, Brazil
{*luciano.serafim,tiago.espinola*}*@ufrpe.br*

Abstract

Quantum computing promises to improve the information processing power to levels unreachable by classical computation. Quantum walks are heading the development of quantum algorithms for searching information on graphs more efficiently than their classical counterparts. A quantum-walk-based algorithm standing out in the literature is the lackadaisical quantum walk. The lackadaisical quantum walk is an algorithm developed to search graph structures whose vertices have a self-loop of weight l. This paper addresses several issues related to applying the lackadaisical quantum walk to search for multiple solutions on grids successfully. Firstly, we show that only one of the two stopping conditions found in the literature is suitable for simulations. In the most discrepant case shown here, a stopping condition is prematurely satisfied at the step $T = 288$ with a success probability $Pr = 0.593276$, while the suitable condition captures the actual amplification that occurred until $T = 409$ with $Pr = 0.878178$. We also demonstrate that the final success probability depends on both the space density of solutions and the relative distance between solutions. For instance, we show here that decreases in the density of solutions can even take a success probability of 0.849178 to 0.961896. In contrast, increases in the relative distances can even take a success probability of 0.871665 to 0.940301. Furthermore, this work generalizes the lackadaisical quantum walk to search for multiple solutions on grids of arbitrary dimensions. In addition, we propose an optimal adjustment of the self-loop weight l for such d-dimensional grids. It turns out other fits of l found in the literature are particular cases. Our experiments demonstrate that successful searches for multiple solutions with higher than two dimensions are possible by achieving success probabilities such as 0.999979, with the value of l proposed here, where it would be 0.637346, with the value of l proposed in previous works. Finally, we observe a two-to-one relation between the steps of the lackadaisical quantum walk and Grover's algorithm, which requires modifications in the stopping condition. That modified stopping condition can escape intermediary fluctuations that would produce premature stops at $T = 6$ with $Pr = 0.000878$ where the system can evolve until $T = 354$ with $Pr = 0.99999$, as an example that we show here. In conclusion,

*Corresponding author

this work deals with practical issues one should consider when applying the lackadaisical quantum walk, besides expanding the technique to a broader range of search problems.

1 Introduction

Quantum computing is expected to demonstrate supremacy [1] over classical computing through the exploration of inherently quantum phenomena such as superposition and entanglement [2]. The opportunities that emerge from the quantum realm have attracted significant efforts in research areas like information security [3], decision making [4], artificial neural networks [5], and optimization [6]. Regarding optimization, the scientific community is actively developing quantum or even quantum-inspired meta-heuristics of search, such as genetic algorithms and particle swarm optimization [7, 8, 9]. Searching is one of the tasks where quantum computing has the most known examples of speedup over classical counterparts, mainly due to the algorithm proposed by Grover [10]. Grover's algorithm can successfully search for a single element within a disordered array of N items in $O(\sqrt{N})$ steps, which is a quadratic speedup over the classical analogs.

However, if the task is a spatial search, Benioff [11] showed that a quantum robot using Grover's algorithm is no more efficient than a classical robot because both require $O(N \log \sqrt{N})$ steps to search 2-dimensional grids of size $\sqrt{N}$ x $\sqrt{N}$. It makes room for employing other techniques to search for information on physical regions modeled as connected graphs [12], also known as spatial search problems. Then, in pursuit of the speedup that Grover's algorithm failed to provide, researchers addressed that type of problem using quantum walks. First, Childs and Goldstone [13] addressed a 2D spatial search problem using a continuous-time quantum walk but failed to provide substantial speedup. On the other hand, Ambainis et al. [14] proposed an algorithm capable of finding the solution in $O(\sqrt{N} \log N)$ steps using a discrete-time quantum walk. Childs and Goldstone [15] showed later that a continuous-time model of quantum walks can also achieve this same speedup.

Over time, quantum walks for other graph structures have been developed [16, 17, 18, 19, 20, 21, 22], but the attempts to improve the search on 2D grids also continued. In particular, the lackadaisical quantum walk (LQW) developed in [23] has been drawing attention because it improved the 2D spatial search by making a simple modification to the algorithm proposed in [14]. The modification was to attach a self-loop of weight l at each vertex of the 2D grid. Adding a new degree of freedom to enable staying at the same position had already been studied for the quantum walk on the line [24], where analysis of time scaling [25], entanglement entropy and temperature [26], and decoherence [27] have been made recently. The LQW, in turn, added a weighted edge that points to the same vertex on the 2D grid. When the weight l of the self-loop is optimally adjusted, the LQW can find the solution to the search problem in $O(\sqrt{N \log N})$ steps, which is an $O(\sqrt{\log N})$ improvement over that loopless version presented in [14].

The LQW improvement was achieved by fitting the self-loop weight to $l = 4/N$, where N is the total number of vertices. This optimal value is only one instance of a general observation about the LQW searching vertex-transitive graphs with $m = 1$ solutions. For these cases, the optimal value of l equals the degree of the graph without loops divided by N [28]. An analytical proof of this conjecture is given in [29] using the fact that the quantum interpolated walk can approximate the LQW. However, that conjecture about the adjustment of l does not hold when the grid's number of solutions m is higher than 1. Thus, another adjustment of l is required. Nahimovs [30] proposed two adjustments of l for arbitrary placements of the solutions, both in the form $l = \frac{4(m-O(m))}{N}$. After that, Giri and Korepin [31] showed that one of these m solutions can be obtained with sufficiently high probability in $O(\sqrt{\frac{N}{m} \log \frac{N}{m}})$ steps. Saha et al. [32] showed

that $l \approx \frac{4}{N(m+1)}$ is the optimal value for the exceptional configuration of m solutions arranged as a block of $\sqrt{m}$ x $\sqrt{m}$ within the grid.

This paper is a solid continuation of the incipient conference paper presented in [33]. Figure 1 shows a general flowchart of our complete research, which is an extensive experimental analysis of the LQW search algorithm. The main flow represents the studies performed here, linked by connectors. Previous knowledge and the knowledge discovered in this research are linked to the main flow by arrows. Each study is decomposed into minor activities with their intermediary contributions, which compose all the knowledge provided by this research to the scientific community.

From the previous efforts related to the search on 2D grids by the LQW algorithm, one of the ideas that can be learned is that the simulations should stop according to two interchangeable stopping conditions. However, we demonstrate through a convergence analysis that LQW simulations should stop according to the stopping condition used in [23] only because that condition captures the maximum amplification of the system. In contrast, the stopping condition used in [30] is satisfied prematurely, i.e., when the system can evolve even further. After, we investigate the impacts of solution setups on the final success probabilities achieved by the LQW algorithm. As a result, we show that the final success probability is inversely proportional to the space density of solutions and directly proportional to the relative distance between solutions. Those results have already been corroborated and extended in the literature. Nahimovs and Santos [34] showed that the success probability is inversely proportional to the density of solutions not only on rectangular 2D grids but also on triangular and honeycomb 2D grids.

We finally consolidate the work by addressing the search for multiple solutions on d-dimensional grids. As the first step in this direction, we generalize the LQW algorithm to such a new scenario by rewriting its mathematical formalism. In this way, we enable the LQW to search for solutions with arbitrary dimensions. Retrieving higher than two quantities per solution is supposed to enlarge the spectrum of applications. For example, one application we glimpse is to use the LQW to search all weights and biases of artificial neural networks, inspired by works that applied a lackadaisical quantum walk for complete graphs [35, 36]. As quantum information routing by quantum walks can benefit from high dimensional cases [37], spatial information search by quantum walks can also. For example, spatial search by quantum walks on sufficiently high dimensions can allow the full $O(\sqrt{N})$ speedup, which is unfeasible in low dimensions [12, 13]. Thus, the solution to the search problem may be retrieved even faster.

Proceeding to the application of the generalized LQW, a new optimal self-loop weight becomes necessary to achieve success in d-dimensional grids. We then propose a generalized adjustment for the value of l. It turns out that other fits reported in the literature [23, 28, 30] are particular cases. The adjustment of l proposed here has already been used in the literature as evidence to consider the number of solutions in the value of l when searching hypercubes [38]. Finally, we observe a two-to-one relation between the steps of the LQW and the steps of Grover's algorithm, which requires a modification in the stopping condition. In conclusion, this work addresses several practical issues that are critical to the successful application of the LQW algorithm when searching for multiple solutions on grids.

This paper is organized as follows. Section 2 presents the theoretical background about the task of search on 2D grids by the LQW algorithm. Here, the reader is expected to be familiar with the basics of quantum computing. If it is not the case, knowledge from the basic to the advanced levels can be obtained in [2, 39]. Section 3 compares the different stopping conditions used in previous works. After, Section 4 relates the impacts on the final success probability to both the space density of solutions and the relative distance between solutions. Section 5 includes generalizing the LQW to grids of arbitrary dimensions with multiple solutions, finding a new optimal value of l, and modifying the stopping condition to tolerate a meaningful kind of

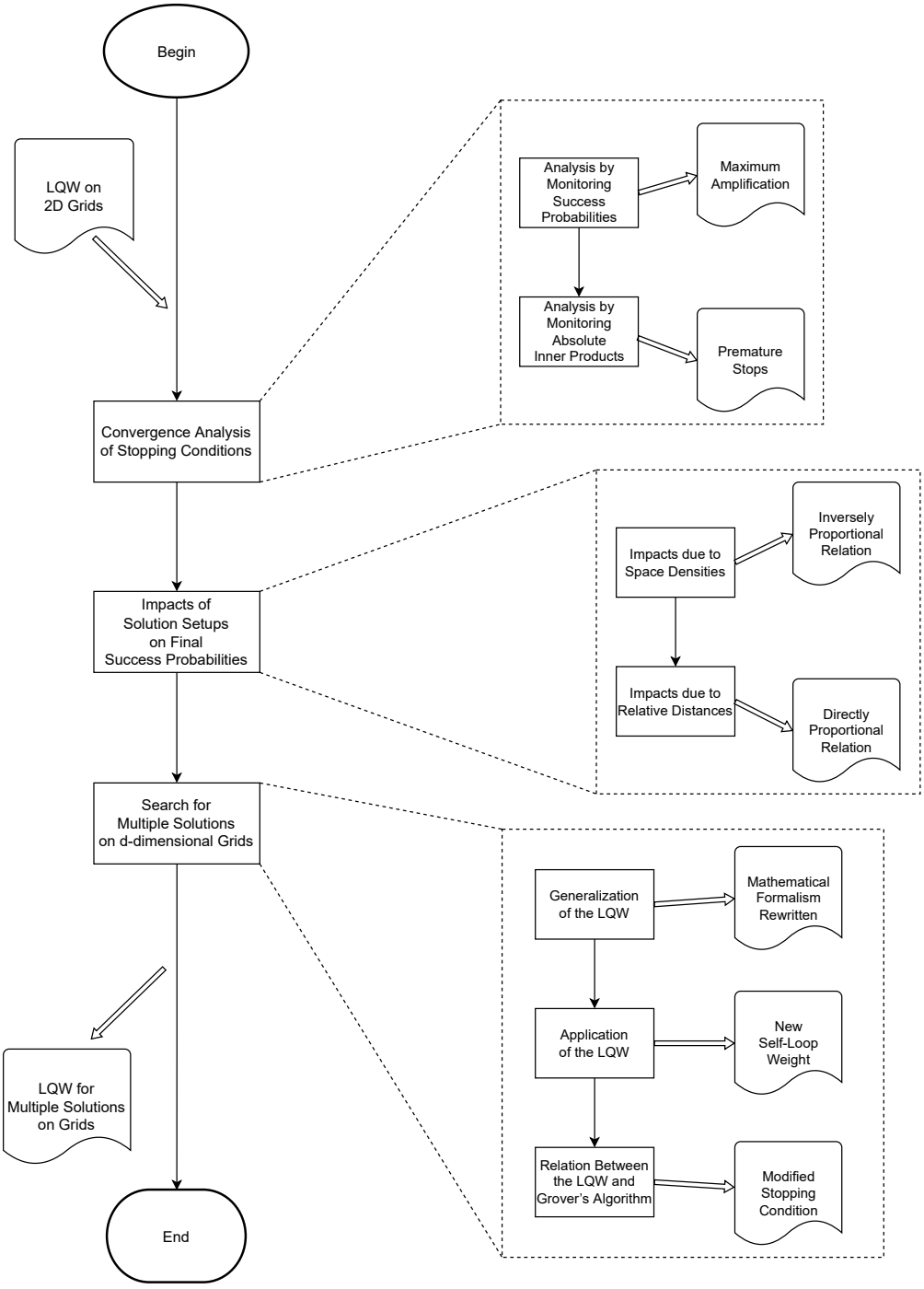


Figure 1: General flowchart representing the studies contemplated in this work, linked by connectors. Information that enters the process as previous knowledge or leaves the process as contributions are linked to the main flow through arrows. Each study is subdivided into minor activities in a more detailed view, where intermediary contributions are also presented.

fluctuation. Finally, Section 6 presents concluding remarks.

2 Search with the Lackadaisical Quantum Walk

The classical random walk is a probabilistic movement in which a particle jumps to its adjacent positions based on the outcome of a non-biased random variable at each step [40]. Generally, the random variable is a fair coin with one degree of freedom for each possible direction of movement in the space at hand. This simple concept can also support sophisticated approaches to practical problems, such as random-walk-based recommendations of potential lenders in peer-to-peer lending [41].

The quantum walk, in turn, is a generalized concept compared to the classical random walk. That high-level idea of conditioned movements remains, but quantum operations are responsible for evolving the system. In this context, quantum properties such as interference and superposition allow the quantum walk to spread quadratically faster than the classical one [40]. This advantage, therefore, can be used to search spatial regions more efficiently [42].

2.1 Spatial Search with a Quantum Walk

Ambainis et al. [14] proposed a quantum walk algorithm to search a single vertex, also called the marked vertex, in the 2-dimensional grid of $L \times L = N$ vertices. In that work, the process evolved on the Hilbert space $\mathcal{H} = \mathcal{H}_C \otimes \mathcal{H}_P$, where $\mathcal{H}_C$ is the 4-dimensional coin space, spanned by $\{|\uparrow\rangle, |\downarrow\rangle, |\leftarrow\rangle, |\rightarrow\rangle\}$, and $\mathcal{H}_P$ represents the N-dimensional space of positions, spanned by $\{|x, y\rangle : x, y \in [0, \ldots, L-1]\}$.

Firstly, the coin toss is accomplished by the operator C presented in Equation 1, which combines the coin operators C_0 and C_1 in such a way that C_1 is applied only to the marked state $|v\rangle$, while C_0 is applied to the others. This idea of different evolution regimes for marked and unmarked vertices was introduced in [17]. Particularly, Ambainis et al. [14] defined C_0 as the Grover diffusion coin: $C_0 = 2|s\rangle\langle s| - I_4$, where $|s\rangle = \frac{1}{2}(|\uparrow\rangle + |\downarrow\rangle + |\leftarrow\rangle + |\rightarrow\rangle)$ and I_4 denotes the 4-dimensional identity operator. Finally, C_1 was defined as $-I_4$. Thus, $-I_4$ is applied to the marked state, while the Grover diffusion coin is applied to the others.

$$C = C_0 \otimes (I_4 - |v\rangle\langle v|) + C_1 \otimes |v\rangle\langle v| \tag{1}$$

Then, the flip-flop shift operator S_{ff} is applied to move the quantum particle while inverting the coin state, as presented in Equation 2. This shift works $\mod \sqrt{N} = L$ because the grid has periodic boundary conditions. Finally, the quantum walk is a repeated application of the operator $U = S_{ff} \cdot C$ to the quantum system $|\psi\rangle$, which begins in the state $|\psi(0)\rangle = \frac{1}{\sqrt{N}} \sum_{x,y=0}^{\sqrt{N}-1} |s\rangle \otimes |x, y\rangle$.

$$\begin{aligned} S_{ff}|\rightarrow\rangle|x, y\rangle &= |\leftarrow\rangle|x+1, y\rangle \\ S_{ff}|\leftarrow\rangle|x, y\rangle &= |\rightarrow\rangle|x-1, y\rangle \\ S_{ff}|\uparrow\rangle|x, y\rangle &= |\downarrow\rangle|x, y+1\rangle \\ S_{ff}|\downarrow\rangle|x, y\rangle &= |\uparrow\rangle|x, y-1\rangle \end{aligned} \tag{2}$$

As a result, the marked vertex can be obtained at the measurement with a probability $O(1/\log N)$ after $T = O(\sqrt{N \log N})$ steps. To achieve a success probability near to 1, amplitude amplification [43] was applied, which implied additional $O(\sqrt{\log N})$ steps. Hence, the total running time of this quantum-walk-based search algorithm is $O(\sqrt{N} \log N)$.

2.2 Improved Running Time by the Lackadaisical Quantum Walk

The LQW search algorithm [23] is an approach strictly based on the algorithm designed in [14] that we just discussed. The main modification is to attach a self-loop of weight l at each vertex of the 2D grid, which implies other changes in the loopless technique. First, $\mathcal{H}_C$ is spanned now by $\{|\uparrow\rangle, |\downarrow\rangle, |\leftarrow\rangle, |\rightarrow\rangle, |\circlearrowleft\rangle\}$ because of the new degree of freedom. However, no changes are required for $\mathcal{H}_P$.

Regarding the coin operator, C_0 was defined as the Grover diffusion coin for weighted graphs [44], so $C_0 = 2|s_c\rangle\langle s_c| - I_5$, where $|s_c\rangle$ is the non-uniform distribution presented in Equation 3, and I_5 denotes the 5-dimensional identity operator. Also, better results were found when $C_1 = -C_0$, outperforming that choice of $C_1 = -I$ used in [14]. About the shift operator S_{ff}, it works like an identity operator when applied to $|\circlearrowleft\rangle|x, y\rangle$. Finally, the quantum system $|\psi\rangle$ begins in a uniform distribution between all vertices with their edges in the weighted superposition $|s_c\rangle$ presented in Equation 3 instead of the uniform $|s\rangle$.

$$|s_c\rangle = \frac{1}{\sqrt{4+l}}(|\uparrow\rangle + |\downarrow\rangle + |\leftarrow\rangle + |\rightarrow\rangle + \sqrt{l}\,|\circlearrowleft\rangle) \tag{3}$$

As a result, the LQW with $l = 4/N$ finds the marked vertex with a success probability close to 1 after $T = O(\sqrt{N \log N})$ steps. It is an $O(\sqrt{\log N})$ improvement over the loopless algorithm. More sophisticated approaches can also achieve this improvement in the running time, like in [45], but the LQW is a significantly simpler and equally capable technique. Moreover, the success probability converges closer and closer to 1 if the number of vertices N increases when using that optimal l.

Those numerical results reported in [23] were found by simulations that stopped when the first peak in the success probability occurred. For that, the stopping condition monitored the success probability at each step. When the current value was smaller than the immediately previous one for the first time, the simulation stopped, and this immediately previous result was reported as the maximum found.

2.3 Lackadaisical Quantum Walk with Multiple Solutions

If there are multiple marked vertices, i.e., multiple solutions in the search space, the LQW results for the case with only one solution do not hold. Such cases with multiple solutions require new optimal choices of the self-loop weight l. In this way, Nahimovs [30] optimally adjusted the value of l for the cases where m solutions are randomly sampled within the 2D grid.

This multiple-solution adjustment of l occurred by searching for new optimal values in the form $l = \frac{4}{N} \cdot a$, where a is a modifiable multiplicative factor. Thus, l was adjusted as a factor of the optimal value for $m = 1$ reported in [23], which was $l = 4/N$. As a result, two adjustments were proposed: $l = \frac{4m}{N}$, for small values of m, and $l = \frac{4(m-\sqrt{m})}{N}$, for large values of m. To find these optimal values of l, the m solutions were arranged following the M_m set presented in Equation 4. However, random placements of solutions yielded similar results.

$$M_m = \{(0, 10i) \mid i \in [0, m-1]\} \tag{4}$$

Regarding the simulations performed in [30], a different stopping condition was used rather than monitoring the success probability at each step. Alternatively, the inner product $|\langle\psi(t)|\psi(0)\rangle|$ was monitored until its minimum was achieved, so the simulation stopped when this inner product became close to 0 in absolute value for the first time.

Table 1: Convergence step T and final success probability Pr, as the number of solutions m increases, for the different stopping conditions used in previous works with $l = \frac{4(m-\sqrt{m})}{N}$.

m	Stopping Conditions			
	Marked Vertices		$\lvert\langle\psi(t)\rvert\psi(0)\rangle\rvert$	
	T	Pr	T	Pr
1	399	0.140828	420	0.138489
5	409	0.878178	288	0.593276
10	297	0.867440	249	0.704010
15	290	0.835395	254	0.747045
20	288	0.818635	268	0.778724

3 Comparison between Different Stopping Conditions

It is possible to find two stopping conditions in the literature regarding the LQW simulations. One of them is to monitor the success probability until its maximum is achieved [23], where "success probability" refers to the probability of measuring a marked vertex (a solution to the search problem). Here, we name this stopping condition as "Marked Vertices". The other stopping condition found in the literature is to monitor the inner product $|\langle\psi(t)|\psi(0)\rangle|$ until its minimum is achieved [30]. However, as the convergence of those stopping conditions has not been compared, their interchangeability became an open question. Therefore, before conducting further experimental analysis of the LQW search algorithm, we verified whether those conditions converge to the same points from equal initial settings.

We conducted experiments using a setup equal to the one used in [30], i.e., a grid of 200 x 200 vertices with the m solutions following the M_m set. Constrained by this M_m scheme, up to 20 solutions can be placed in that space, since $M_{20} = \{(0,0),(0,10),\ldots,(0,180),(0,190)\}$. Placing more than 20 solutions in the 200 x 200 grid would require an organization other than the M_m set so that the grid limits would not be extrapolated. However, as we used the M_m scheme, we made experiments with 1, 5, 10, 15 and 20 solutions in the grid.

As a result, the stopping conditions converged to the same points for $l = \frac{4m}{N}$, suggesting that the stopping conditions are equivalent. However, it is not the case for $l = \frac{4(m-\sqrt{m})}{N}$. Table 1 contrasts the results obtained for $l = \frac{4(m-\sqrt{m})}{N}$ when monitoring both the marked vertices and the inner product $|\langle\psi(t)|\psi(0)\rangle|$. As can be seen, the results tend to converge to the same points as m increases. Nevertheless, the conditions were not equivalent because each was satisfied at a different step T, which implied different final success probabilities Pr. Also, monitoring the inner product $|\langle\psi(t)|\psi(0)\rangle|$ generated lower success probabilities in all cases.

Step by step, the system evolution was stored to investigate the divergence carefully. Figure 2 shows the evolution of the $m = 5$ case, which has the most significant discrepancy in Table 1. The solid black line represents the condition that monitors the marked vertices, while the dashed blue line represents the one that monitors the inner product $|\langle\psi(t)|\psi(0)\rangle|$.

As reported in Table 1, the condition that monitors the inner product in absolute value is satisfied prematurely at the step $T = 288$. It is said premature because the success probability continues increasing until $T = 409$. After the step $T = 288$, though, the curves have a similar growth damping, which raised a question about monitoring the real value of the inner product rather than its absolute value.

Figure 3 shows the system evolution during 1000 steps. As before, the monitoring of the marked vertices is represented by the solid black line. At this time, the inner product is monitored without calculating its absolute value, represented by the dashed green line. Note that both

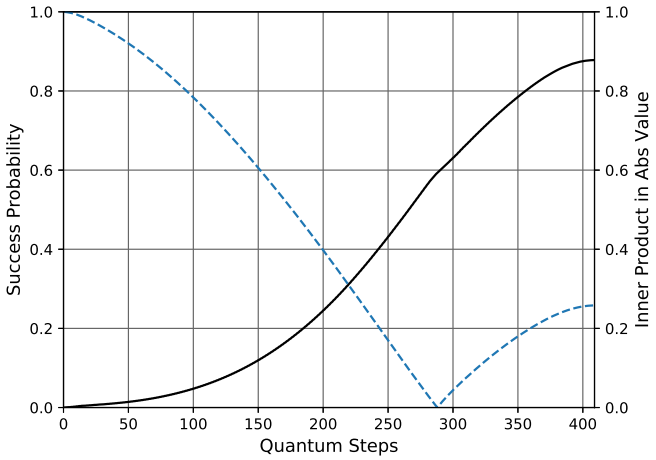


Figure 2: System evolution step by step until the condition that monitors the marked vertices is satisfied. The solid black line monitors that condition, while the dashed blue line monitors the inner product in absolute value.

curves have the same behavior. Therefore, it is possible to conclude that the stopping conditions used in previous works are equivalent if, and only if, the inner product is considered without calculating its absolute value. Otherwise, only the condition that monitors the marked vertices leads to the real amplitude amplification achieved by the LQW search algorithm.

In this manner, all results discussed in this work from now on were found using the stopping condition that monitors the marked vertices. Since the objective is to measure the quantum system when the maximum amplification in the success probability occurs, monitoring the marked vertices is the most natural choice to define when the simulation should stop.

4 Solution Setups Affecting the Success Probability

After choosing the stopping condition properly in Section 3, we now move forward to addressing factors that affect the final success probability achieved by the technique. Previous works have already demonstrated the considerable dependence of the LQW algorithm on the self-loop weight l [23, 30]. Expanding those analyses, we address the density of solutions and the relative distance between solutions.

4.1 Previous Evaluations of Solution Densities and a Complementary Experiment

The experiments performed by Wong [23] and Nahimovs [30] can reveal some dependence between the success probability and the space density of solutions ρ, where $\rho = \frac{m}{N}$. However, such works did not link the density of solutions and the success probability achieved at the end of the simulation. Here, we briefly discuss these previous experiments identifying the impacts of ρ. Finally, a complementary experiment was performed.

Firstly, Wong [23] investigated the impacts of adding more unmarked vertices in a grid with only one solution. That experiment evaluated how decreases in the density of solutions could

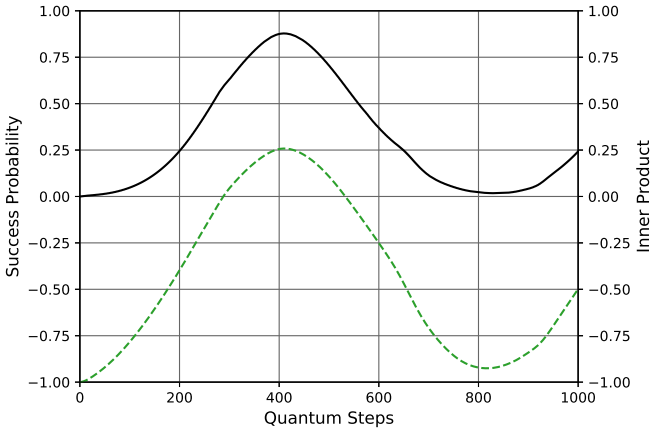


Figure 3: System evolution step by step during 1000 steps. The solid black line monitors the marked vertices, while the dashed green line monitors the inner product without calculating its absolute value.

affect the final success probability. As a result, the success probability tends to improve, even though some disturbed behavior for the first values of N exists.

After that, Nahimovs [30] inserted more and more solutions in the 200 x 200 grid when adjusting the value of l for multiple marked vertices. Since the grid size was fixed, that experiment increased the density of solutions with each new vertex being marked. However, the probability of measuring a marked vertex was smaller when m increased. It would be a counter-intuitive idea in a classical world.

It is possible to comprehend the dependence between the total number of vertices N, the number of solutions m, and the final success probability observing Grover's algorithm [40]. Consider $|\omega\rangle$ as the state where the total energy of the quantum system is equally distributed only between the marked states, so the success probability is 1. The goal of Grover's algorithm is to rotate the system's state $|\psi\rangle$ to get as close to $|\omega\rangle$ as possible. However, this is an iterative process in which $|\psi\rangle$ rotates at each step by an angle θ. As θ is inversely proportional to N, increasing N implies more steps T, but these fine rotations turn $|\psi\rangle$ closer to $|\omega\rangle$ as N increases, explaining the results of Wong [23]. As θ is proportional to m, increasing m implies fewer steps T if $N \gg m$, although $|\psi\rangle$ gets less close to $|\omega\rangle$ at the end, explaining the smaller success probabilities in the experiments of Nahimovs [30] while m increased.

Thus, these previous experiments suggest that the success probability is inversely proportional to the density of solutions within the grid due to having a more refined or less refined angle θ in Grover's rotations, as explained. In this work, a complementary experiment was made to fill the gap not addressed by those previous works: decreasing the density of solutions, like in [23], in a grid with multiple marked vertices, like in [30]. While we conducted this experiment, we also searched for the optimal value of l in the form $l = \frac{4}{N}a$, like in [30] again.

Figure 4 shows the peaks in the success probability, represented by the solid black line, and the optimal a values that generated these peaks, represented by the dashed brown line, both as functions of N. The density of solutions decreased in this case because m was always equal to 10, while N increased by adding unmarked vertices. The optimal values of a were searched with a step size of 0.5, and N varied from 10^4 to 10^6 with the $m = 10$ solutions following the M_{10} set.

Again, there is a disturbed behavior for the first values of N, like in [23]. Afterward, the success probability tends to 1, and the optimal value of a tends to the number of solutions

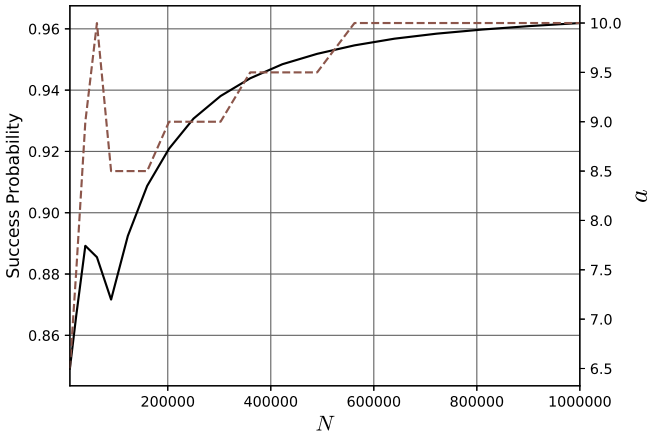


Figure 4: Peaks in the success probability, represented by the solid black line, and respective optimal values of a, represented by the dashed brown line, both as functions of N, with the $m = 10$ solutions located in the 2D grid according to the M_{10} set.

$m = 10$. This experimental result suggests that the construction $l = \frac{4(m-O(m))}{N}$ proposed in [30] is a way of adjusting for the cases where the density of solutions is not small enough. In the best cases of solutions density, a equals m and, consequently, $l = \frac{4m}{N}$.

4.2 A New Set of Solutions Increasing Relative Distances

In the last experiment, the $m = 10$ solutions were located always at the points $\{(0, 0), (0, 10), \ldots, (0, 90)\}$, following the M_{10} set. That solutions distribution did not take advantage of the gradual increment in the total number of vertices N. If the solutions were located far from each other, it would be possible to continue evaluating how the success probability depends on the density of solutions but also on the relative distance between solutions.

Thus, we propose an alternative to the M_m set that is the $P_{L,m}$ set presented in Equation 5. Following this new set, the m solutions are located depending on the number of vertices in each dimension L so that the grid size is better used. For example, $m = 10$ solutions on the 200 x 200 grid would be located at the points $\{(0, 0), (20, 20), \ldots, (180, 180)\}$, following the $P_{200,10}$ set. Hence, the solutions are farther apart using the $P_{L,m}$ set than the M_m set.

$$P_{L,m} = \left\{ \left(\left\lfloor \frac{L}{m} \right\rfloor i, \left\lfloor \frac{L}{m} \right\rfloor i \right) \;\middle|\; i \in [0, m-1] \right\} \tag{5}$$

Then, our complementary experiment that evaluated decreases in the density of solutions with $m = 10$ solutions was redone, but using the $P_{L,m}$ set this time to localize the solutions farther from each other. The results obtained with this new set of solutions are contrasted in Table 2 with the ones obtained previously, which used the M_m set. The success probabilities in Table 2 for the M_m set are precisely the ones already presented in Figure 4 but in terms of L now because L is the variable used to define the $P_{L,m}$ set. It is worth reminding that $L^2 = N$.

For all values of L, the set of solutions $P_{L,m}$ generated better results because the success probability was higher and with fewer steps. Besides this, the disturbed behavior for the first values of L did not appear in the results with the new set of solutions. Finally, it is possible to

Table 2: Number of steps T and final success probability Pr as L increases for different sets of $m = 10$ solutions.

L	$M_{m=10}$		$P_{L,m=10}$	
	T	Pr	T	Pr
100	147	0.849178	109	0.902339
200	293	0.889219	223	0.927680
300	511	0.871665	342	0.940301
400	747	0.908749	460	0.948348
500	965	0.930714	581	0.953927
600	1181	0.943863	700	0.958288
700	1407	0.951843	822	0.961646
800	1623	0.956787	941	0.964317
900	1857	0.959761	1063	0.966613
1000	2097	0.961896	1187	0.968522

Table 3: Values of m that do not have asymptotic and growing behaviors for the success probability as the density of solutions decreases.

L	Pr		
	$m = 3$	$m = 4$	$m = 5$
100	0.991433	0.986119	0.981772
200	0.988165	0.992391	0.990397
300	0.985744	0.993451	0.992697
400	0.984221	0.993206	0.993754
500	0.983418	0.992604	0.994283
600	0.983100	0.991933	0.994585
700	0.983081	0.991282	0.994717
800	0.983252	0.990683	0.994677
900	0.983548	0.990138	0.994557
1000	0.983927	0.989644	0.994392

conclude from these numerical results that the success probability is directly proportional to the relative distance between solutions.

Regardless of whether or not a disturbed behavior exists for the first values of L, the success probability had an asymptotic and growing behavior for higher values of L in all previous cases discussed here. However, that is not true for all values of m. Table 3 shows the same investigation of decreases in the density of solutions with the $P_{L,m}$ set again, but for $m = \{3, 4, 5\}$, and not for $m = 10$ as before.

The qualitative behaviors found for these m values are not equal to the behaviors for both $m = 1$, as reported in [23], and $m = 10$, as shown in Figure 4 and Table 2. In those $m = 1$ and $m = 10$ cases, a disturbed behavior existed during a transition from small to high values of L, and then the success probability improved continuously. However, $m = \{3, 4, 5\}$ can be seen as a kind of transition from small to high values from the perspective of the number of solutions m. It suggests that the asymptotic and growing behavior for the success probability as the density of solutions decreases is only guaranteed for values high enough of both $L = \sqrt{N}$ and m.

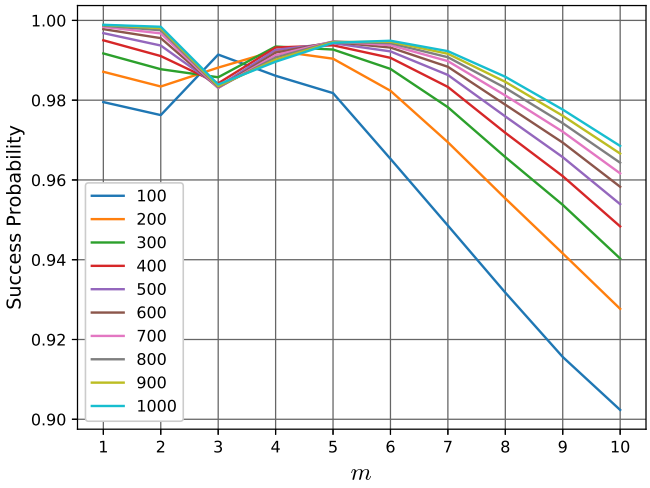


Figure 5: Final success probability as the density of solutions increases with the solutions following the $P_{L,m}$ set. The colored lines represent grids with different numbers L of vertices per dimension.

4.3 Evaluation of Density Increasing with the New Set of Solutions

The density of solutions within the grid affects the final success probability achieved by the LQW algorithm. We already analyzed decreases in the solution density by adding unmarked vertices using both the M_m and $P_{L,m}$ sets. Regarding increases in the density of solutions, we complement this analysis here using the $P_{L,m}$ set since results using the M_m set are found in [30]. Increases in the density of solutions occur by having more marked vertices in a fixed-size grid of L x L.

Figure 5 shows the final success probability as a function of the number of solutions m for grids with different numbers L of vertices per dimension. The colored lines represent the results for grids with L varying from 100 to 1000 and with the number of solutions $m = \{1, 2, \ldots, 10\}$ following the $P_{L,m}$ set.

As expected, because of the inversely proportional relation, the success probability decreases as the density of solutions increases by having more solutions in the grid. However, these results with the $P_{L,m}$ set also had that transitory phenomenon. There are intervals where a disturbed behavior exists for all cases, and then the success probability tends to decrease continuously.

This result constitutes one more perspective that shows some uncertainty about the behavior of the LQW algorithm with small values of some input parameters. Thus, it is more confident to apply the LQW search algorithm in real scenarios where the input parameters are higher to avoid all those disturbed behaviors.

5 Lackadaisical Quantum Walk on d-dimensional Grids

Marking a vertex as a solution in the two-dimensional case means that its coordinates x and y satisfy the search problem when combined. Searching and retrieving only two values can be a limitation because more than two quantities may be required to solve some applications. The idea here is to expand the technique to search for solutions with an arbitrary number of

dimensions, which implies walking through grids with higher than two dimensions. The new capacity is supposed to expand the spectrum of applications of the LQW search algorithm. In the following, the mathematical formalism is redefined, the self-loop weight l is adjusted, and the stopping condition is revisited, all of this considering practical issues that arise in scenarios with arbitrary dimensions.

5.1 Generalization to d-dimensional Grids

Here, the mathematical formalism is rewritten to the search with the LQW algorithm on grids of higher dimensions. Fortunately, the generalization is straightforward from the two-dimensional formulation, demonstrated in the following algebra. Besides that, we discuss some aspects involving classical simulations of the generalized technique in grids of arbitrary dimensions.

Considering a grid with d dimensions, each vertex has $2d+1$ possible directions of movement because each dimension has a positive and a negative direction, and there is a self-loop attached to the vertex. In a generalized way, the coin space $\mathcal{H}_C$ is spanned by $\{|\Uparrow_1\rangle, |\Downarrow_1\rangle, \ldots, |\Uparrow_d\rangle, |\Downarrow_d\rangle, |\circlearrowleft\rangle\}$, where $\Uparrow_i$ and $\Downarrow_i$ represent the movements on the i-th dimension. The computational basis for the space of positions $\mathcal{H}_P$ is $\{|x_1, \ldots, x_d\rangle : x_i \in [0, \ldots, \sqrt[d]{N} - 1]\}$. Thus, the quantum walk evolves on the space $\mathcal{H} = \mathcal{H}_C \otimes \mathcal{H}_P$ with these generalized reformulations.

The action of applying $C_1 = -C_0$ to the solutions and C_0 to the other vertices can be replaced by an oracle that is used first, followed by C_0 acting on all vertices indistinguishably. Since the oracle flips the signs of all m marked vertices, the overall effect is to apply $-C_0$ to the marked vertices and C_0 to the others. Using an oracle, C does not need to be broken down into two different coin operators, so $C = C_0$. In a generalized form, the coin operator is now defined as $C = 2\,|s_c\rangle\langle s_c| - I_{2d+1}$, where $|s_c\rangle$ is the generalized distribution presented in Equation 6.

$$|s_c\rangle = \frac{1}{\sqrt{2d+l}}(|\Uparrow_1\rangle + |\Downarrow_1\rangle + \ldots + |\Uparrow_d\rangle + |\Downarrow_d\rangle + \sqrt{l}\,|\circlearrowleft\rangle) \tag{6}$$

For the purpose of applying that coin operator C, the outer product $|s_c\rangle\langle s_c|$ can be written in the matrix form as follows:

$$\begin{aligned} |s_c\rangle\langle s_c| &= \frac{1}{\sqrt{2d+l}} \begin{pmatrix} 1 \\ \vdots \\ 1 \\ \sqrt{l} \end{pmatrix} \cdot \frac{1}{\sqrt{2d+l}} \begin{pmatrix} 1 & \ldots & 1 & \sqrt{l} \end{pmatrix} \\ &= \frac{1}{2d+l} \begin{pmatrix} 1 & \ldots & 1 & \sqrt{l} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \ldots & 1 & \sqrt{l} \\ \sqrt{l} & \ldots & \sqrt{l} & l \end{pmatrix}. \end{aligned}$$

Let $|\phi\rangle$ be a quantum state of the generalized coin space $\mathcal{H}_C$, i.e., $|\phi\rangle = (\alpha_1, \ldots, \alpha_{2d}, \alpha_{2d+1})^T$. Thus, the application of the coin operator C in that generic quantum state is as follows:

$$(2\left|s_c\right\rangle\left\langle s_c\right| - I_{2d+1})\left|\phi\right\rangle = 2\left|s_c\right\rangle\left\langle s_c\right|\left|\phi\right\rangle - I_{2d+1}\left|\phi\right\rangle$$

$$= 2 \cdot \frac{1}{2d+l} \begin{pmatrix} 1 & \dots & 1 & \sqrt{l} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \dots & 1 & \sqrt{l} \\ \sqrt{l} & \dots & \sqrt{l} & l \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{2d} \\ \alpha_{2d+1} \end{pmatrix} - \begin{pmatrix} 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 1 & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{2d} \\ \alpha_{2d+1} \end{pmatrix}$$

$$= 2 \cdot \frac{1}{2d+l} \begin{pmatrix} \alpha_1 + \ldots + \alpha_{2d} + \sqrt{l} \cdot \alpha_{2d+1} \\ \vdots \\ \alpha_1 + \ldots + \alpha_{2d} + \sqrt{l} \cdot \alpha_{2d+1} \\ \sqrt{l} \cdot (\alpha_1 + \ldots + \alpha_{2d} + \sqrt{l} \cdot \alpha_{2d+1}) \end{pmatrix} - \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{2d} \\ \alpha_{2d+1} \end{pmatrix}.$$

Defining λ as:

$$\lambda = \frac{1}{2d+l}(\alpha_1 + \ldots + \alpha_{2d} + \sqrt{l} \cdot \alpha_{2d+1}),$$

the application of the coin operator C comes down to:

$$(2\left|s_c\right\rangle\left\langle s_c\right| - I_{2d+1})\left|\phi\right\rangle = \begin{pmatrix} 2\lambda - \alpha_1 \\ \vdots \\ 2\lambda - \alpha_{2d} \\ 2\lambda\sqrt{l} - \alpha_{2d+1} \end{pmatrix}.$$

It is precisely the result expected when the weighted Grover diffusion coin is applied. Since λ is not the mean of the coin amplitudes, the inversion is about a mean that takes into account the weight of the graph edges rather than the simple arithmetic mean [44]. All this worked out also in the generalized form.

Regarding classical simulations, the coefficients $\alpha_1, \cdots, \alpha_{2d}$ must be handled in a flexible and generalized way according to the total number of dimensions, besides considering the loop weight appropriately to normalize the quantum state. It turns out that the simulation is similar to the one developed by Wong [23], but considering the generalized relations stated here when it comes to the coin operator.

About the flip-flop shift operator S_{ff}, there is also no restriction to generalize it. The intuition behind this operator is to transfer energy between vertices in a dimension-per-dimension way. At a step, what is happening in one dimension for a vertex does not affect what is happening in another dimension for the same vertex. For example, in each step, a vertex $|v\rangle$ stores on its state $|\Uparrow_i\rangle$ the energy coming from the state $|\Downarrow_i\rangle$ of the vertex immediately following on the direction $\Uparrow_i$, and vice versa. Note that, for that dimension, the others do not cause interference.

Therefore, the operator S_{ff} can be generalized by acting on each dimension separately, as presented in Equation 7. That pattern of dimension-wise energy transfer simplifies an abstraction for a classical simulation. In each step, through a double for-loop, the implementation can replicate the operation in each dimension one by one for each vertex.

$$\begin{aligned} S_{ff}\left|\Uparrow_i\right\rangle\left|x_1, \ldots, x_i, \ldots, x_d\right\rangle &= \left|\Downarrow_i\right\rangle\left|x_1, \ldots, x_i + 1, \ldots, x_d\right\rangle \\ S_{ff}\left|\Downarrow_i\right\rangle\left|x_1, \ldots, x_i, \ldots, x_d\right\rangle &= \left|\Uparrow_i\right\rangle\left|x_1, \ldots, x_i - 1, \ldots, x_d\right\rangle \end{aligned} \tag{7}$$

As before, the energy stored in the self-loop remains unchanged after an application of the flip-flop shift operator, i.e., $S_{ff}\left|\circlearrowleft\right\rangle\left|x_1, \ldots, x_i, \ldots, x_d\right\rangle = \left|\circlearrowleft\right\rangle\left|x_1, \ldots, x_i, \ldots, x_d\right\rangle$. Thus, no con-

Table 4: Number of steps and final success probability for some cases in grids with higher than two dimensions using $l = \frac{4m}{N}$ and the $P_{d,L,m}$ set.

d	L	m	T	Pr
3	32	8	134	0.958805
4	16	4	257	0.888795
5	10	5	285	0.816259
6	8	4	441	0.739591

siderations are needed for generalization. As the space topology is torus-like because of the periodic boundary conditions, the shift operates mod $\sqrt[d]{N}$.

Finally, the system begins in the uniform distribution between each of the N vertices of the d-dimensional grid with the weighted superposition of coin states generalized in Equation 6. The step where the simulation stops depends on a stopping condition. As we already demonstrated in Section 3, the more appropriate stopping condition is to monitor the success probability until achieving its maximum. Now, the LQW algorithm can be simulated on higher-than-two-dimensional grids with multiple solutions.

5.2 Application on d-dimensional Grids

To conduct our experiments on d-dimensional grids, we locate the m solutions according to the $P_{d,L,m}$ set presented in Equation 8, which is a straightforward generalization of the $P_{L,m}$ set already introduced in Equation 5. Therefore, each solution is a d-tuple, and the solutions are equidistant on the grid's main diagonal.

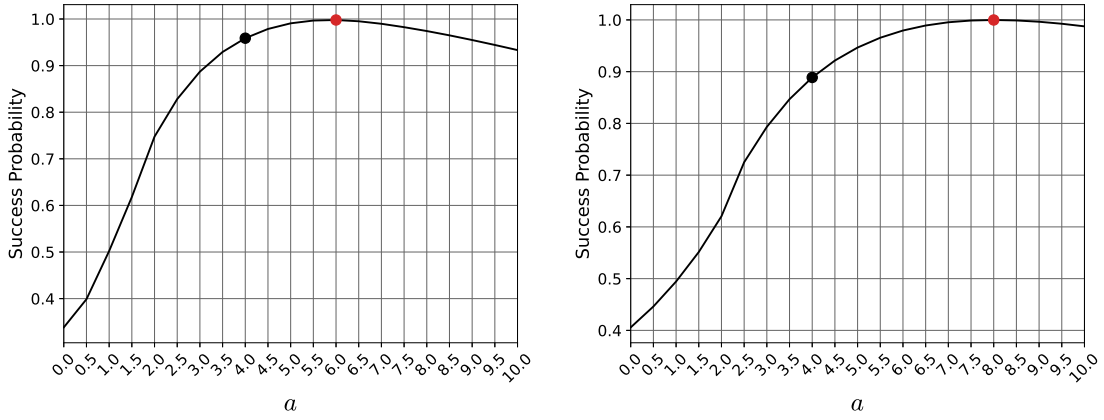
$$P_{d,L,m} = \left\{ \left(\left\lfloor \frac{L}{m} \right\rfloor i, \cdots, \left\lfloor \frac{L}{m} \right\rfloor i \right)_d \;\middle|\; i \in [0, m-1] \right\} \tag{8}$$

Experiments can be performed on grids of higher dimensions since the technique, the stopping condition, and the solution setup are described. The number of steps T and the final success probability Pr for some d-dimensional cases are presented in Table 4. Note that the success probability decreases as the number of dimensions increases, suggesting that the LQW algorithm is ineffective in higher-than-two-dimensional scenarios.

Although L and m varied in the cases presented in Table 4, the density of solutions was small enough and the relative distance between solutions was high enough to not affect substantially at all. Thus, the final success probability deteriorated strictly due to increases in d. Those deteriorated results were found setting $l = \frac{4m}{N}$ for being the optimal l on 2D grids, at least for the best cases of solution densities. It makes room to search for even better adjustments of the self-loop weight since research efforts have already demonstrated how critical adjusting l is [23, 30]. The following experiment aims to verify whether another optimal value of l for d higher than two exists.

As demonstrated in [23], l is inversely proportional to the number of vertices N. At the same time, l is directly proportional to the number of solutions m [30]. Thus, the value of l depends on the density of solutions ρ, where $\rho = \frac{m}{N}$. Preserving the relations found by those works, we search for new fits of l in the form $l = \rho \cdot a$, where a is a multiplicative factor. From those previous works, the value of a would be 4, but we already showed in Table 4 that the LQW deteriorates as d increases with such a value of a.

Figure 6a and Figure 6b show the final success probability as a function of that multiplicative factor a for the 3D and 4D cases presented in Table 4, respectively. The result for $a = 4$ is marked



(a) (b)

Figure 6: Final success probability as a function of a for the self-loop weight in the form $l = \rho \cdot a$. To the left, results for the 3D case with $L = 32$ and $m = 8$. To the right, results for the 4D case with $L = 16$ and $m = 4$. The black dot marks the results for a equal to 4, while the red dot marks the best overall result found.

by the black dot, while the red dot marks the best overall result in the search. In both cases, $a = 4$ is not optimal because other values generated better success probabilities. The best overall result of each case could generate success probabilities near 1, while its distance to the $a = 4$ gets larger when d increases.

Regarding the other cases presented in Table 4, the best overall results in the search generated success probabilities of 0.999933 for the 5D case and 0.999986 for the 6D case. In this way, it is possible to conclude that the LQW algorithm can generate satisfactory results when applied in grids with higher than two dimensions with multiple solutions. The results reported in Table 4 deteriorated because a equal to 4, proposed by previous works, is optimal only in the restricted 2D case.

Moreover, our results revealed a pattern. The best values of a found in the search were 6 for the 3D case, 8 for the 4D case, 10 for the 5D case, and 12 for the 6D case. Thus, the experimental results suggest that the optimal value of a is $2 \cdot d$, so the optimal value of l for d-dimensional grids is $l = \rho \cdot 2d = \frac{2dm}{N}$. In the 2D case, $l = \rho \cdot 2 \cdot 2 = \frac{4m}{N}$, as proposed in previous works. More experimental evidence is presented in Table 5, which compares results obtained using the l proposed in previous works ($l = \frac{4m}{N}$) with the ones obtained using the l proposed in this work ($l = \frac{2dm}{N}$), for a variety of cases in higher than two dimensions.

For all cases, $l = \frac{2dm}{N}$ generated success probabilities near to 1, while $l = \frac{4m}{N}$ generated inferior results that deteriorated further as d increased. For the cases presented in Table 5, not only experiments with those two fits of l were made, but no other adjustment surpassed the result of using $l = \frac{2dm}{N}$, indicating that it is the optimal value for d-dimensional grids with multiple solutions.

Therefore, this work becomes part of the community efforts that developed the LQW algorithm by adjusting the self-loop weight optimally for different scenarios, as summarized in Table 6. When there is a single solution in the search space, the optimal self-loop weight is $l = \frac{2}{N}$ for 1D grids [31] and $l = \frac{4}{N}$ for 2D grids [23]. For vertex-transitive graphs in general, $l = \frac{V}{N}$ is the optimal value, where V is the valency of the graph [28]. In contrast, the optimal

Table 5: Number of steps and final success probability for some d-dimensional grids using the value of l proposed in previous works, $l = \frac{4m}{N}$, and the value proposed in this work, $l = \frac{2dm}{N}$.

d	L	m	$l = \frac{4m}{N}$		$l = \frac{2dm}{N}$	
			T	Pr	T	Pr
3	32	4	187	0.959003	171	0.999531
3	64	8	381	0.959096	348	0.999736
4	16	2	364	0.888818	315	0.999912
4	30	3	1048	0.88888	907	0.99999
5	10	2	453	0.816318	377	0.999982
5	15	5	784	0.816322	658	0.999991
6	8	2	593	0.731387	600	0.999994
6	10	10	541	0.73811	525	0.999986
7	6	6	388	0.692785	354	0.99999
8	4	2	247	0.637346	295	0.999979

Table 6: Optimal self-loop weight proposed in different works that developed the LQW algorithm to search distinct scenarios. Basically, the graph structure defines each scenario. The scenarios are also characterized depending on the existence of a single solution or multiple solutions.

Work	Graph Structure	Solution	Self-Loop Weight
Giri and Korepin [31]	1D grid	Single	$l = \frac{2}{N}$
Wong [23]	2D grid	Single	$l = \frac{4}{N}$
Rhodes and Wong [28]	Vertex-transitive	Single	$l = \frac{V}{N}$
Souza et al. [38]	Hypercube	Multiple	$l = \frac{dm}{N}$
Saha et al. [32]	2D grid	Multiple	$l \approx \frac{4}{N(m+1)}$
Nahimovs [30]	2D grid	Multiple	$l = \frac{4(m-O(m))}{N}$
Nahimovs and Santos [34]	2D grid	Multiple	$l = \frac{Vm}{N}$
This work	dD grid	Multiple	$l = \frac{2dm}{N}$

value is $l = \frac{dm}{N}$ when searching for multiple solutions on d-dimensional hypercubes [38].

Regarding 2D grids again, the adjustment $l \approx \frac{4}{N(m+1)}$ is required if the multiple solutions are arranged as a block [32]. In contrast, $l = \frac{4(m-O(m))}{N}$ is the optimal value for arbitrary placements of the solutions [30]. Searching for multiple solutions on 2D grids with different valencies V requires the value $l = \frac{Vm}{N}$ to achieve optimal results [34]. This work, in turn, developed the LQW algorithm to a scenario that was not covered in previous works. The optimal fit of l proposed here, $l = \frac{2dm}{N}$, successfully enables the LQW algorithm to search for multiple solutions on d-dimensional grids.

5.3 Stopping Condition Revisited

As shown in Section 3, the more appropriate and natural choice of stopping condition is to monitor the probability evolution about the m marked vertices until this quantity achieves its maximum. The maximum is determined by finding a step whose success probability is smaller than the immediately previous one. That approach assumes a function that increases monotonically, achieves its maximum, and decreases monotonically after that maximum. However, it is not the case in some examples on grids with higher than two dimensions.

Table 7 shows four exceptional cases that stopped at a considerably premature step by using

Table 7: Number of steps and final success probability for d-dimensional cases that prematurely stopped using both the value of l proposed in previous works, $l = \frac{4m}{N}$, and the value proposed in this work, $l = \frac{2dm}{N}$.

d	L	m	$l = \frac{4m}{N}$		$l = \frac{2dm}{N}$	
			T	Pr	T	Pr
5	10	2	24	0.009348	24	0.009374
5	15	3	24	0.001847	24	0.001848
5	15	5	14	0.001108	14	0.001108
7	6	6	6	0.000878	6	0.000878

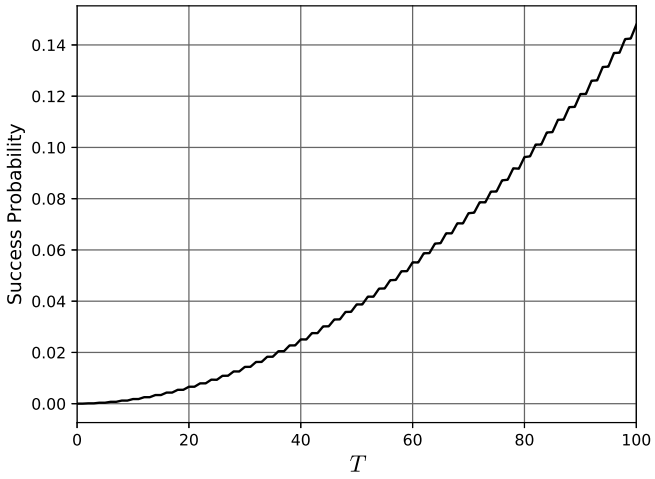


Figure 7: Success probability during the first 100 steps on the 5D grid with $L = 10$, $m = 2$, and the value of l proposed in this work, which is $l = \frac{2dm}{N}$.

as criterion finding a step whose success probability is smaller than the immediately previous one. We still considered the value of l proposed in previous works ($l = \frac{4m}{N}$), although it is not optimal for d-dimensional grids, as well as the optimal value of l found in this work ($l = \frac{2dm}{N}$) to assess that those premature stops are not related with the choice of the self-loop weight. As can be seen, regardless of the value of l, each case stopped at the same premature step and, consequently, with a highly unsatisfactory success probability.

To evaluate whether or not the system can evolve further, even though the stopping condition is satisfied too early, we stored the success probabilities during 100 steps of the LQW algorithm with $l = \frac{2dm}{N}$ for the first case presented in Table 7, which is the 5D grid with $L = 10$ and $m = 2$. Figure 7 shows such a system evolution. Qualitatively, the success probability improves continuously as more steps are performed, but there is a kind of fluctuation in the process. However, this fluctuation has a meaning.

Ambainis et al. [14] mathematically proved that two steps of a particular quantum walk search algorithm give precisely one step of Grover's algorithm. It turns out, in every two steps, the first is an intermediary step to the actual amplitude amplification generated by the second step of the quantum walk. That quantum walk occurred on a complete graph with a non-weighted self-loop for each vertex.

The result we showed in Figure 7 is supposed to be an experimental demonstration of that

two-to-one relation between quantum walks and Grover's algorithm steps. Interestingly, we used a quantum walk with weighted self-loops on grids with higher than two dimensions and not a quantum walk with non-weighted self-loops on a complete graph as used in [14]. Nevertheless, there are similarities because both quantum walk approaches apply a Grover diffusion coin and the flip-flop shift operator S_{ff} subsequently.

The practical implication in terms of the stopping condition is that the success probabilities of adjacent steps must not be compared anymore. As shown explicitly in Figure 7, considering two adjacent steps, one of them is an intermediary step subject to fluctuations. Instead of comparing with the immediately previous step, the solution is to compare with the penultimate step.

In this way, the simulation stops in the step whose success probability is smaller than the one of the penultimate step, and the success probability of that penultimate step is reported as the maximum found. This is enough to conceive a more robust stopping condition capable of escaping the premature stops reported in Table 7. To obtain those results reported in Table 5, the stopping condition needs this slight modification, especially for these exceptional cases.

6 Final Remarks

This research addressed the LQW search algorithm and its capabilities from an experimental point of view. We aimed to understand properties and existing limitations more clearly, in addition to contributing to a better quantum-walk-based solver of search problems.

In this way, first, we demonstrated that different stopping conditions used in previous works are not interchangeable. Calculating the absolute value of the inner product $\langle\psi(t)|\psi(0)\rangle$ implies prematurely stops. Instead, the real value must be used. After choosing the stopping condition correctly, we demonstrated that the final success probability is inversely proportional to the density of solutions and directly proportional to the relative distance between solutions. However, those relations are guaranteed only for high values of the input parameters. We showed disturbed behaviors in a transition between small to high values of the input parameters from different perspectives.

Consolidating the work, we generalized the LQW algorithm to search for multiple solutions on grids of arbitrary dimensions, not only on the restricted 2D case. However, a new adjustment for the self-loop weight is necessary to obtain successful searches. The experiments we made allow concluding that $l = \frac{2dm}{N}$ is the generalized and optimal value of l for d-dimensional grids with multiple solutions. The fits proposed in previous works are only a specific case where d equals 2. The investigations on d-dimensional grids also clarified a two-to-one relation between the steps of the LQW and the ones of Grover's algorithm. An actual amplitude amplification occurs at every two steps, where the first is an intermediary step subject to numerical fluctuations. A fluctuation-tolerant stopping condition is obtained by comparing the success probabilities of the current step and the penultimate step, not between subsequent steps.

Future works should mathematically define upper and lower bounds considering the impacts of multiple solutions stated here. Those impacts of solution densities and relative distances should be studied for solutions randomly sampled from some probability distributions. Another possible direction is to investigate the symmetry breaking [46] that nonhomogeneous self-loop weights can cause on grids of arbitrary dimensions with multiple solutions. Inspired by the use of multiple quantum search agents to find optimal solutions for multiobjective optimization problems [47], one more future direction could be to combine the evolution of multiple lackadaisical quantum walkers in the grid. Mathematically or numerically estimated, the number of steps T to the maximum amplitude amplification should be defined *a priori* since it establishes the step where the measurement should occur when executed in quantum devices.

Then, theoretically, the LQW algorithm will be available to execute in quantum devices, ensuring high success probabilities on d-dimensional grids with multiple solutions. In practice, the LQW implementation will still need to deal with limitations in the existing quantum hardware. Inspired in [48], future works should implement the LQW algorithm on the available quantum computers. Finally, the LQW algorithm should be applied to solve search problems, like the optimization of artificial neural networks, where quantum meta-heuristics of search can be used to tune learning rates [49]. Moreover, the successful application of the LQW algorithm to transfer quantum states on complete bipartite graphs [50] encourages its application for quantum communication on grids.

Acknowledgments

This work was financially supported by the Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE), the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] J. Preskill, "Quantum computing and the entanglement frontier," *arXiv preprint arXiv:1203.5813*, 2012.

[2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge: Cambridge University Press, 2010.

[3] Z. Qu, Z. Zhang, and M. Zheng, "A quantum blockchain-enabled framework for secure private electronic medical records in internet of medical things," *Information Sciences*, vol. 612, pp. 942–958, 2022.

[4] W. Liu and J. Zhu, "A multistage decision-making method with quantum-guided expert state transition based on normal cloud models," *Information Sciences*, vol. 615, pp. 700–730, 2022.

[5] H. Situ, Z. He, Y. Wang, L. Li, and S. Zheng, "Quantum generative adversarial network for generating discrete distribution," *Information Sciences*, vol. 538, pp. 193–208, 2020.

[6] Y. Ruan, Z. Yuan, X. Xue, and Z. Liu, "Quantum approximate optimization for combinatorial problems with constraints," *Information Sciences*, vol. 619, pp. 98–125, 2023.

[7] G. Acampora and A. Vitiello, "Implementing evolutionary optimization on actual quantum processors," *Information Sciences*, vol. 575, pp. 542–562, 2021.

[8] W. Fang, J. Sun, H. Chen, and X. Wu, "A decentralized quantum-inspired particle swarm optimization algorithm with cellular structured population," *Information Sciences*, vol. 330, pp. 19–48, 2016.

[9] G. Li, W. Wang, W. Zhang, W. You, F. Wu, and H. Tu, "Handling multimodal multiobjective problems through self-organizing quantum-inspired particle swarm optimization," *Information Sciences*, vol. 577, pp. 510–540, 2021.

[10] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Physical review letters*, vol. 79, no. 2, p. 325, 1997.

[11] P. Benioff, "Space searches with a quantum robot," in *Quantum Computation and Information (Washington, DC, 2000)* (S. J. Lomonaco Jr. and H. E. Brandt, eds.), vol. 305 of *Contemporary Mathematics*, pp. 1–12, Providence, RI, USA: American Mathematical Society, 2002.

[12] S. Aaronson and A. Ambainis, "Quantum search of spatial regions," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, pp. 200–209, IEEE, 2003.

[13] A. M. Childs and J. Goldstone, "Spatial search by quantum walk," *Physical Review A*, vol. 70, no. 2, p. 022314, 2004.

[14] A. Ambainis, J. Kempe, and A. Rivosh, "Coins make quantum walks faster," in *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, (USA), pp. 1099–1108, Society for Industrial and Applied Mathematics, 2005.

[15] A. M. Childs and J. Goldstone, "Spatial search and the dirac equation," *Physical Review A*, vol. 70, no. 4, p. 042312, 2004.

[16] D. A. Meyer and T. G. Wong, "Connectivity is a poor indicator of fast quantum search," *Physical review letters*, vol. 114, no. 11, p. 110503, 2015.

[17] N. Shenvi, J. Kempe, and K. B. Whaley, "Quantum random-walk search algorithm," *Physical Review A*, vol. 67, no. 5, p. 052307, 2003.

[18] T. G. Wong, "Grover search with lackadaisical quantum walks," *Journal of Physics A: Mathematical and Theoretical*, vol. 48, no. 43, p. 435304, 2015.

[19] H. Tanaka, M. Sabri, and R. Portugal, "Spatial search on johnson graphs by continuous-time quantum walk," *Quantum Information Processing*, vol. 21, no. 74, pp. 1–13, 2022.

[20] H. Tanaka, M. Sabri, and R. Portugal, "Spatial search on johnson graphs by discrete-time quantum walk," *Journal of Physics A: Mathematical and Theoretical*, vol. 55, no. 255304, pp. 1–16, 2022.

[21] D. Qu, S. Marsh, K. Wang, L. Xiao, J. Wang, and P. Xue, "Deterministic search on star graphs via quantum walks," *Physical Review Letters*, vol. 128, no. 050501, pp. 1–6, 2022.

[22] D. Qu, L. Xiao, K. Wang, X. Zhan, and P. Xue, "Experimental investigation of equivalent laplacian and adjacency quantum walks on irregular graphs," *Physical Review A*, vol. 105, no. 062448, pp. 1–7, 2022.

[23] T. G. Wong, "Faster search by lackadaisical quantum walk," *Quantum Information Processing*, vol. 17, no. 3, p. 68, 2018.

[24] N. Inui, N. Konno, and E. Segawa, "One-dimensional three-state quantum walk," *Physical Review E*, vol. 72, no. 056112, pp. 1–7, 2005.

[25] P. R. N. Falcão, A. R. C. Buarque, W. S. Dias, G. M. A. Almeida, and M. L. Lyra, "Universal dynamical scaling laws in three-state quantum walks," *Physical Review E*, vol. 104, no. 054106, pp. 1–6, 2021.

[26] L. T. Tude and M. C. de Oliveira, "Temperature and entanglement of the three-state quantum walk," *Quantum Science and Technology*, vol. 7, no. 035009, pp. 1–12, 2022.

[27] L. T. Tude and M. C. de Oliveira, "Decoherence in the three-state quantum walk," *Physica A: Statistical Mechanics and its Applications*, vol. 605, no. 128012, pp. 1–11, 2022.

[28] M. L. Rhodes and T. G. Wong, "Search on vertex-transitive graphs by lackadaisical quantum walk," *Quantum Information Processing*, vol. 19, no. 9, p. 334, 2020.

[29] P. Høyer and Z. Yu, "Analysis of lackadaisical quantum walks," *arXiv preprint arXiv:2002.11234*, 2020.

[30] N. Nahimovs, "Lackadaisical quantum walks with multiple marked vertices," in *SOFSEM 2019: Theory and Practice of Computer Science* (B. Catania, R. Královič, J. Nawrocki, and G. Pighizzini, eds.), vol. 11376 of *Lecture Notes in Computer Science*, pp. 368–378, Springer, 2019.

[31] P. R. Giri and V. Korepin, "Lackadaisical quantum walk for spatial search," *Modern Physics Letters A*, vol. 35, no. 08, p. 2050043, 2020.

[32] A. Saha, R. Majumdar, D. Saha, A. Chakrabarti, and S. Sur-Kolay, "Faster search of clustered marked states with lackadaisical quantum walks," *Quantum Information Processing*, vol. 21, no. 275, pp. 1–13, 2022.

[33] J. H. A. de Carvalho, L. S. de Souza, F. M. de Paula Neto, and T. A. E. Ferreira, "Impacts of multiple solutions on the lackadaisical quantum walk search algorithm," in *Intelligent Systems* (R. Cerri and R. C. Prati, eds.), vol. 12319 of *Lecture Notes in Computer Science*, pp. 122–135, Springer, 2020.

[34] N. Nahimovs and R. A. M. Santos, "Lackadaisical quantum walks on 2d grids with multiple marked vertices," *Journal of Physics A: Mathematical and Theoretical*, vol. 54, no. 415301, pp. 1–12, 2021.

[35] L. S. de Souza, J. H. A. de Carvalho, and T. A. E. Ferreira, "Quantum walk to train a classical artificial neural network," in *8th Brazilian Conference on Intelligent Systems (BRACIS 2019)*, pp. 836–841, IEEE, 2019.

[36] L. S. de Souza, J. H. A. de Carvalho, and T. A. E. Ferreira, "Classical artificial neural network training using quantum walks as a search procedure," *IEEE Transactions on Computers*, vol. 71, no. 2, pp. 378–389, 2022.

[37] X. Zhan, H. Qin, Z.-h. Bian, J. Li, and P. Xue, "Perfect state transfer and efficient quantum routing: A discrete-time quantum-walk approach," *Physical Review A*, vol. 90, no. 012331, pp. 1–5, 2014.

[38] L. S. de Souza, J. H. A. de Carvalho, and T. A. E. Ferreira, "Lackadaisical quantum walk in the hypercube to search for multiple marked vertices," in *Intelligent Systems* (A. Britto and K. V. Delgado, eds.), vol. 13073 of *Lecture Notes in Computer Science*, pp. 249–263, Springer, 2021.

[39] N. S. Yanofsky and M. A. Mannucci, *Quantum Computing for Computer Scientists*. Cambridge: Cambridge University Press, 2008.

[40] R. Portugal, *Quantum walks and search algorithms*. New York, NY, USA: Springer, 2013.

[41] H. Zhang, H. Zhao, Q. Liu, T. Xu, E. Chen, and X. Huang, "Finding potential lenders in p2p lending: A hybrid random walk approach," *Information Sciences*, vol. 432, pp. 376–391, 2018.

[42] T. G. Wong, "Unstructured search by random and quantum walk," *arXiv preprint arXiv:2011.14533*, 2020.

[43] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," in *Quantum Computation and Information (Washington, DC, 2000)* (S. J. Lomonaco Jr. and H. E. Brandt, eds.), vol. 305 of *Contemporary Mathematics*, pp. 53–74, Providence, RI, USA: American Mathematical Society, 2002.

[44] T. G. Wong, "Coined quantum walks on weighted graphs," *Journal of Physics A: Mathematical and Theoretical*, vol. 50, no. 47, p. 475301, 2017.

[45] R. Portugal and T. D. Fernandes, "Quantum search on the two-dimensional lattice using the staggered model with hamiltonians," *Physical Review A*, vol. 95, no. 4, p. 042341, 2017.

[46] J. Rapoza and T. G. Wong, "Search by lackadaisical quantum walk with symmetry breaking," *Physical Review A*, vol. 104, no. 062211, pp. 1–15, 2021.

[47] P. Singh, "Fqtsfm: A fuzzy-quantum time series forecasting model," *Information Sciences*, vol. 566, pp. 57–79, 2021.

[48] F. Acasiete, F. P. Agostini, J. K. Moqadam, and R. Portugal, "Implementation of quantum walks on ibm quantum computers," *Quantum Information Processing*, vol. 19, no. 12, pp. 1–20, 2020.

[49] G. Liu and W. Ma, "A quantum artificial neural network for stock closing price prediction," *Information Sciences*, vol. 598, pp. 75–85, 2022.

[50] R. A. M. Santos, "Quantum state transfer on the complete bipartite graph," *Journal of Physics A: Mathematical and Theoretical*, vol. 55, no. 125301, pp. 1–17, 2022.