Solving the Subset Sum Problem via Quantum Walk Search

Giacomo Lancellotti, *Graduate Student Member, IEEE*, Simone Perriello, Alessandro Barenghi, and Gerardo Pelosi, *Member, IEEE*

***Abstract*—Quantum walk-based search algorithms have demonstrated an asymptotic quadratic speedup compared to classical search methods. Formulating a generic search problem as a (quantum) search over a graph makes the efficiency of the algorithm to be closely dependent on the properties of the graph itself. In this work, we present a complete implementation of a quantum walk search procedure on Johnson graphs, speeding up the solution of the Subset Sum Problem, a well-known computational problem with applications in resource allocation, scheduling, and cryptanalysis. We provide a detailed design of each sub-circuit, quantifying their costs in terms of gate count, circuit depth, and qubit width, and exhibit all figures of merit as a function of the problem parameters. Our approach includes two distinct implementations: one that minimizes qubit usage and another focused on optimizing circuit depth. We compare our solutions against the unstructured Grover based quantum search algorithm, demonstrating a reduction of the cost on terms of T-count and T-depth, for practically solvable instances. The proposed design serves as a foundational building block for the development of efficient quantum search algorithms that can be modeled on Johnson graphs, bridging the gap with existing theoretical complexity analyses.**

***Index Terms*—Grover search algorithm, Johnson graph, quantum computing, quantum walk search, subset sum problem.**

I. Introduction

RECENT advancements in qubit manufacturing, quantum error correction techniques, and both hardware and software control systems for quantum devices, make the development of practical quantum computers increasingly feasible [1], [2]. Quantum computers are attracting significant interest due to their potential to substantially improve computational efficiency. Problems that can be efficiently solved by a quantum computer (i.e., in polynomial time) fall within the *bounded-error quantum polynomial time* (**BQP**) computational complexity class. This class encompasses all problems solvable in the **P** class as well as some in **NP** and **co-NP** classes, including also the factoring and the discrete logarithm extraction problem. However, it is an open question whether the **NP-complete** class— which includes problems of significant interest— falls within BQP [3] or not.

While a polynomial-time quantum algorithm for NP-complete problems is not anticipated, Grover's proposal [4] offers a quantum algorithm to perform unstructured exhaustive searches quadratically faster than any classical search algorithms. This algorithm provides a robust framework to accelerate the solution retrieval of the search variants of NP-complete problems, which are equally difficult with respect to their decision counterparts.

An alternative approach to tackling search problems leverages *quantum walks*, the quantum analogue of classical random walks. In this framework, the search space, represented as a graph, is explored through a sequence of probabilistic steps until the target element is located. Compared to probabilistic methods that repeatedly sample elements of the search space independently, the quantum walk approach offers an advantage by allowing information generated in previous steps to be partially reused. Quantum walk-based searches have been proven to provide a quadratic speedup over classical random walks [5], [6], and they reduce the computational complexity of Grover-based quantum search algorithms by a polynomial-factor when applied to specific problems. For instance, Ambainis [7] introduced a quantum-walk search strategy on a Johnson graph for solving the *element distinctness problem*, outperforming any realization of the Grover's approach. A similar strategy was later used to solve also other computational problems, such as the *syndrome decoding* [8] and the *claw finding* [9] problems, both of particular relevance in the analysis of post-quantum cryptosystems.

In [10], the authors explore the theoretical advantages of using a quantum walk search algorithm to solve the *subset sum problem (SSP)*. By adapting a classical meet-in-the-middle approach to the quantum setting, their strategy significantly reduces the number of required operations compared to classical solvers. The asymptotic runtime of a quantum SSP solver was further improved in [11], [12] by exploiting different adaptations of the classical strategy. All these approaches

Received 30 October 2024; revised 11 August 2025; accepted 18 October 2025. Date of publication 24 October 2025; date of current version 11 December 2025. This work was supported in part by the Italian Centro Nazionale di Ricerca in HPC, Big Data e Quantum computing - SPOKE 10 under Grant D43C22001240001, in part by the Project POINTERID funded by the Italian MUR - PRIN 20222022M2JLF2, and in part by the Project SERICS (Italian NRRP MUR program - EU - NGEU) under Grant PE000014. Recommended for acceptance by J. Palsberg. *(Giacomo Lancellotti and Simone Perriello are co-first authors.) (Corresponding author: Giacomo Lancellotti.)*

The authors are with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, 20133 Milano, Italy (e-mail: giacomo.lancellotti@polimi.it; simone.perriello@polimi.it; alessandro.barenghi@polimi.it; gerardo.pelosi@polimi.it).

Digital Object Identifier 10.1109/TC.2025.3625044

rely on an exponential amount of *Quantum Random-Access Memory (QRAM)*, whose practical realization is still an open challenge [13].

To the best of our knowledge, the first comprehensive work detailing quantum circuits and providing finite-regime complexity metrics for quantum walks was presented in [14]. Focusing on the claw-finding problem, the authors describe the design of data structures for a quantum walk-based solver, modeled through a Johnson graph. However, their approach modifies the graph structure by incorporating self-loops at each vertex, leading to performance that falls short of theoretical expectations. Additionally, their approach requires an exponential number of qubits. Subsequently, in [15] the authors proposed an alternative quantum circuit for quantum walks, applying it to solve the subset-sum problem. Their approach eliminates the self-loops introduced in [14], and only relies on a polynomial amount of qubits. Additionally, the authors report trade-offs between the quantum walk and Grover-based approaches, noting that while the Grover-based method exhibits more favorable complexity metrics in terms of gate count for cryptographic-grade parameters, it requires a greater circuit depth for the same parameters.

Contributions. In this work, we apply a quantum walk strategy to solve the *Subset Sum Problem (SSP)*, a mathematical problem with applications in several fields such as cryptography [16], [17], job scheduling [18] and resource optimization [19]. Consistent with prior research, our analysis assumes a fault-tolerant quantum computing regime, deliberately omitting considerations of noise resilience and specific hardware implementations. Each quantum circuit is implemented using both a general gate set and the *Clifford+T* gate set, with the latter chosen for its strong potential in achieving fault tolerance. We derive closed-form expressions for key spatial and temporal complexity metrics, including quantum gate count, qubit count (also known as *width*), and circuit depth.

We present two distinct circuit architectures: one optimized for minimal qubit count (termed *low-width*) and the other for reduced circuit depth (termed *low-depth*). In both cases, our approach advances upon existing implementation in two distinct directions. Firstly, with respect to both theoretical and finite-regime analysis, it obviates the need for exponentially sized quantum random access memory (QRAM) and limits qubit requirements to polynomial growth with respect to the search space. Additionally, the low-depth approach improves upon [15] by reducing all complexity metrics, including circuit depth, gate count, and qubit usage. Meanwhile, the low-width approach introduces a novel insertion and deletion circuit for the support data structure, achieving the lowest qubit count compared to the state-of-the-art.

We additionally propose a novel strategy to encode a uniform superposition of a combinatorial number of integers into a quantum registers. The circuit exploits the Dicke state and a *Vertex Binary Encoder*, and improves upon a similar circuit proposed in [15] in terms of all complexity metrics.

Following the approach shown in [15], we compare our design with respect to a Grover-based search approach. Our comparison tackles a range of problem sizes, from practically solvable SSP instances to those large enough to be used in post-quantum cryptosystems, which are assumed to be unsolvable even with quantum computers (taking the intersection between NP-complete and BQP to be empty). Differently from [15], we show that the quantum walk approach is asymptotically better than the Grover approach in all cost metrics.

II. Background

In this section, we provide an overview of foundational concepts in quantum computation, with a particular emphasis on Grover's framework and the MNRS quantum walk framework, including the associated cost model for assessing computational complexity. We also present a formal mathematical formulation of the Subset Sum Problem (SSP) and illustrate how this problem can be transformed into a search problem built upon a Johnson graph.

A. Quantum Computing Basics

The fundamental unit of quantum information is the *qubit*, defined as a vector in the Hilbert space $\mathcal{H} = \mathbb{C}^2$. Using the standard bra-ket notation, the state of a qubit is expressed as a column vector that forms a linear combination, or *superposition*, of the basis states $|0\rangle$ and $|1\rangle$, i.e., $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ are the *amplitudes*. The state of an n-qubit system is represented as $|\psi_n\rangle = \sum_{i=1}^{2^n} \alpha_i |i\rangle$, a vector in the Hilbert space $\mathcal{H} = (\mathbb{C}^2)^{\otimes n} \cong \mathbb{C}^{2^n}$, where $\otimes$ denotes the tensor product. By convention, an n-qubit quantum state is labeled by an n-bit strings. For instance, the state of a two qubits system can be expressed as $|\psi_2\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$. When an n-qubit quantum state is measured, the state is said to *collapse* to one of its 2^n basis states, with a probability given by the square of the magnitude of the corresponding amplitude, e.g., $|\alpha_i|^2$ for $|i\rangle$.

The state of a n-qubit quantum system can be evolved by applying quantum *operators*, described through unitary matrices in $\mathbb{C}^{2^n \times 2^n}$. A matrix $\boldsymbol{U}$ is unitary iff $\boldsymbol{U}\boldsymbol{U}^\dagger = \boldsymbol{I}$, with $\boldsymbol{U}^\dagger$ being the conjugate transpose of $\boldsymbol{U}$. Starting from a generic quantum state $|\psi\rangle$, the application of the operator $\boldsymbol{U}$ leads to the new quantum state $\boldsymbol{U}|\psi\rangle$. If a sequence of k quantum operators is applied, they are executed in right precedence order as $\boldsymbol{U}_k, \ldots, \boldsymbol{U}_1|\psi\rangle$.

As an alternative to the algebraic formulation, a sequence of operators can be visualized using the *circuit model*, closely resembling a classical combinatorial circuit. An example of a quantum circuit is shown in Fig. 1(b). In this model, each qubit is represented by an horizontal wire; a logical grouping of qubits is called a *quantum register* $\underline{\boldsymbol{a}}$. We use classical array notation to index qubits within a register, e.g., $\underline{\boldsymbol{a}}_{|i}$ denotes the qubit at index i. The system is evolved by applying, from left to right, a sequence of *quantum gates*, corresponding to a circuit representation of the quantum operators used in the algebraic model. Fig. 1(a) shows an example of common quantum gates. When having a quantum gate corresponding to the operator $\boldsymbol{U}$, the circuit corresponding to $\boldsymbol{U}^\dagger$ is retrieved by reversing the order of the application of the gates used for $\boldsymbol{U}$, additionally replacing each gate with its adjoint. A generic quantum gate G is termed controlled when its application depends on the state

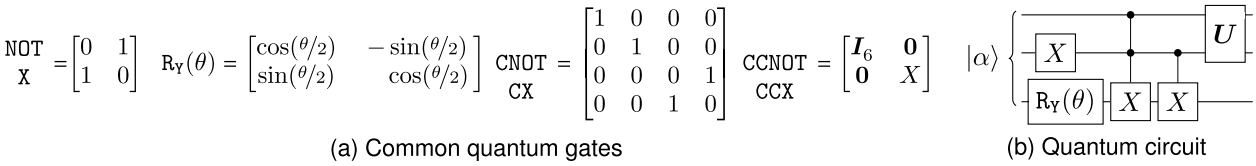

Fig. 1. (a) Unitary matrices associated to some common quantum gates. $\boldsymbol{I}_6$ denotes a 6×6 identity matrix. (b) An example of a quantum circuit, showing the application of gates (boxes) to qubits (horizontal wires). the $\boldsymbol{U}$ denotes a generic gate having an associated unitary matrix.

of m control qubits. We use the notation $\mathtt{C}^{\mathtt{m}}{-}\mathtt{G}$ to denote an arbitrary gate G in which the gate G is applied only if all the m control qubits are in state $|1\rangle$. When $m{=}1$, the exponent is typically omitted. Additionally, since the X gate is commonly interpreted as the quantum analogue of the classical NOT gate, it is standard to refer to $\mathtt{C}{-}\mathtt{X}$ and $\mathtt{C}^2{-}\mathtt{X}$ as CNOT and CCNOT, respectively. In the quantum circuit, the control qubits are represented through black circle on the corresponding horizontal wires, and they are connected to the gate by a vertical wire.

Projection and Reflection Operators. A *projector operator* is defined as an operator that projects a state $|\psi\rangle$ onto a subspace spanned by orthonormal vectors. We denote the projector onto the subspace $|\phi\rangle$ as $\boldsymbol{U}_{\mathrm{prj}(\phi)} = |\phi\rangle\langle\phi|$. The *reflection operator* is defined as $\boldsymbol{U}_{\mathrm{ref}(\phi)} = \boldsymbol{I} - 2\boldsymbol{U}_{\mathrm{prj}(\phi)}$. This operator changes the sign of the component of $|\psi\rangle$ along the direction of $|\phi\rangle$ while leaving the orthogonal component unchanged. We also define $\boldsymbol{U}_{\mathrm{ref}^{\perp}(1)}$ as the reflection operator that inverts the amplitude of the quantum state labeled as the all-ones binary string, while leaving all other states unchanged. Similarly, $\boldsymbol{U}_{\mathrm{ref}^{\perp}(0)}$ is the operator inverting the amplitude of the quantum state labeled as the all-zeroes binary string, leaving all the other states unchanged. Both operators correspond, in the circuit model, to the application of a (multi-)controlled Z gate.

B. Unstructured Search via Grover's Algorithm

In its generic formulation, a search problem involves finding one or more elements from a domain set $\mathcal{X}$ that satisfy a certain condition expressible by a Boolean function $f\colon \mathcal{X}\mapsto\{0,1\}$. This function evaluates to 1 if the element meets the condition and 0 otherwise. All elements for which $f(x)$ evaluates to one are included in the *marked set*, defined as $\mathcal{M}{=}\{x^* \mid x^*{\in}\mathcal{X}, f(x^*){=}1\}$. We denote the fraction of marked elements as $\epsilon{=}|\mathcal{M}|/|\mathcal{X}|$.

To assess the computational costs of classical or quantum algorithms, we typically employ the query model, where the complexity of the procedure is estimated by counting the accesses (or *evaluations*) of an oracle that implements the function $f(x)$. For unstructured search problems, every deterministic classical algorithm will find a marked element in $\mathcal{O}(1/\epsilon)$ function evaluations.

In [4] Grover proposes a quantum unstructured-search algorithm that identifies a marked element in $\mathcal{O}(1/\sqrt{\epsilon})$ calls to a quantum oracle, achieving a quadratic speedup compared to its classical counterpart. The algorithm, whose circuit model is given in Fig. 2, consists of three phases: input preparation, oracle, and diffusion. By repeating the oracle and diffusion

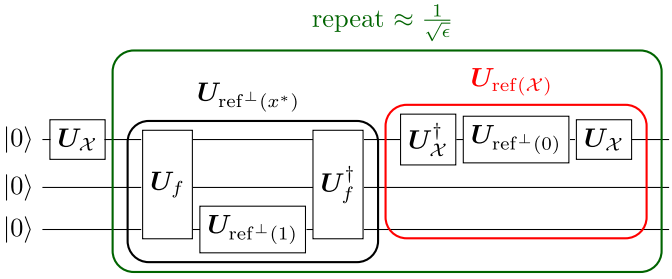

Fig. 2. High-level quantum circuit scheme of Grover's algorithm, showing the three main operators: the input preparation $\boldsymbol{U}_{\mathcal{X}}$, producing the uniform superposition of the bitstring belonging to the domain; the oracle $\boldsymbol{U}_{\mathrm{ref}^{\perp}(x^*)}$, performing a reflection around the state orthogonal to the solution state x^*; the diffusion operator $\boldsymbol{U}_{\mathrm{ref}(\mathcal{X})}$, performing a reflection around the uniform superposition produced by the input preparation circuit.

stages $\approx 1/\sqrt{\epsilon}$ times, the probability of observing the marked state is close to 1.

Input preparation $\boldsymbol{U}_{\mathcal{X}}$. This operator is used to create a uniform superposition of all the bitstrings corresponding to the elements of the domain $\mathcal{X}$, given by the expression $|\mathcal{X}\rangle = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}|x\rangle$. We denote the cost of this operator $\boldsymbol{U}_{\mathcal{X}}$ as the *setup* cost, represented by $\mathrm{T_S}$.

Oracle $\boldsymbol{U}_{\mathrm{ref}^{\perp}(x^*)}$. The oracle acts as a reflection that inverts the sign of the amplitude associated to the marked state $|x^*\rangle$. It can be logically decomposed into three sub-operators: $\boldsymbol{U}_f$, $\boldsymbol{U}_{\mathrm{ref}^{\perp}(1)}$, and $\boldsymbol{U}_f^{\dagger}$. In the circuit model, the operator $\boldsymbol{U}_f$ corresponds to the quantum implementation of the function f, and it stores the Boolean result into an ancillary qubit. Next, $\boldsymbol{U}_{\mathrm{ref}^{\perp}(1)}$ is applied to the output qubit of the function, followed by the operator $\boldsymbol{U}_f^{\dagger}$, used to restore the qubit labels to their initial values. The cost of the reflection is negligible, since it corresponds to a multi-controlled Z gate. The cost of the oracle step is equal to twice the cost of implementing $\boldsymbol{U}_f$. We denote such a cost as *check* cost, and represent it as $\mathrm{T_C}$.

Diffusion $\boldsymbol{U}_{\mathrm{ref}(\mathcal{X})}$. The diffusion operator is a reflection around the state corresponding to the initial superposition, and can be decomposed as $\boldsymbol{U}_{\mathrm{ref}(\mathcal{X})} = \boldsymbol{U}_{\mathcal{X}}\boldsymbol{U}_{\mathrm{ref}^{\perp}(0)}\boldsymbol{U}_{\mathcal{X}}^{\dagger}$. Since the cost of $\boldsymbol{U}_{\mathrm{ref}^{\perp}(0)}$ is negligible, the cost of the diffusion step is equal to two setup costs $2\,\mathrm{T_S}$.

The total cost of the Grover algorithm is

$$\mathrm{T_S} + (2\,\mathrm{T_S} + 2\,\mathrm{T_C})/\sqrt{\epsilon} \tag{1}$$

C. Probabilistic Search via Quantum Walks

Probabilistic search algorithms based on random walks are considered to be a promising alternative to exhaustive search.

In this context, when referring to random walks, we focus on discrete random walks.

A discrete random walk on a graph $G = (\mathcal{V}, \mathcal{E})$ consists in taking a sequence of steps moving from a vertex $v_i \in \mathcal{V}$ to one of its adjacent vertices $v_j \in \mathcal{V}$ accordingly a given probability distribution. In the special case where the probability of transitioning from one vertex to an adjacent vertex is independent of the sequence of previously visited vertices, i.e., $\mathbb{P}(V_{t+1} = v_{t+1} \mid V_t = v_t, \ldots, V_0 = v_0) = \mathbb{P}(V_{t+1} = v_{t+1} \mid V_t = v_t)$, the random walk is a *Markovian* process, which is equivalent to a *Markov chain*. From this point forward, we use the terms random walk and Markov chain interchangeably.

Given the set of all possible states of a Markov chain V, the state at any given time can be represented through a probability vector $\boldsymbol{v} \in \mathbb{R}^{|\mathcal{V}|}$, where $|\mathcal{V}|$ is the number of vertices in the graph. This vector encodes the probability distribution over the vertices, i.e., the probability of being in each specific vertex at a given time. A transition or stochastic matrix $\boldsymbol{P}$, of size $|\mathcal{V}|\times|\mathcal{V}|$, encodes the transition probabilities. In such a matrix, each element $\boldsymbol{P}_{ij}$ represents the probability of transitioning from state v_i to state v_j. Given an initial distribution $\boldsymbol{v}_0$, the probability distribution over the vertices after t steps is given by $\boldsymbol{v}_t = \boldsymbol{P}^t \boldsymbol{v}_0$. A Markov chain is said to be *irreducible* if the underlying graph is strongly connected, meaning every node is reachable from any other node. A state $\boldsymbol{v}$ has a period of k if returns to $\boldsymbol{v}$ occur only in multiples of k steps. If k=1, the state is aperiodic, and, if all states have a period of 1 the chain is said to be *aperiodic*. In the case of an irreducible chain, all states share the same period. A chain is considered *ergodic* if it is both irreducible and aperiodic. In our analysis, we focus only on stationary, ergodic Markov chains on finite graphs, meaning the graphs have a countable set of nodes (states), and the transition probabilities associated with each edge remain constant over time.

For stationary, ergodic states, the chain reaches a stable state $\boldsymbol{v}_\pi$ after a finite amount of steps, meaning that $\boldsymbol{P}^i \boldsymbol{v}_\pi = \boldsymbol{v}_\pi$ for each i. The state $\boldsymbol{v}_\pi$ corresponds to the eigenvector of $\boldsymbol{P}$ associated with the eigenvalue 1. For an ergodic chain, this eigenvalue is unique, while all other eigenvalues have magnitudes less than 1, as per the Perron-Frobenius theorem. We define the *spectral gap* δ of the chain as the absolute difference between the two largest eigenvalues of $\boldsymbol{P}$, i.e., $\delta = |\lambda_1| - |\lambda_2|$. For an ergodic chain, this simplifies to $\delta = 1 - |\lambda_2|$. The number of steps required for the chain to approach its stationary distribution is referred to as the *mixing time*, denoted by t_m, and is given by t_m=$1/\delta$.

Random walk search. To adapt the random walk framework to the search problem given in Sec. II-B, we model the domain space $\mathcal{X}$ as the nodes of the graph $G = (\mathcal{V}, \mathcal{E})$, where each vertex represents an element of $\mathcal{X}$, and an edge between two vertices encodes a relationship between the elements.

A simple random walk search procedure involves sampling an initial vertex v_0 from the graph, following any probability distribution, which need not necessarily be uniform. We then check if the vertex is marked as v_i^*, i.e., if the element associated with the vertex satisfies the search condition. If the solution is found, we output it. Otherwise, we sample a new vertex among the adjacent ones according to the current probability distribution. The probability that $v_i \notin \mathcal{M}$ is equal to $1-\varepsilon$, and the expected number of trials required to get a success is equal to the mean of the geometric distribution, i.e., $1/\varepsilon$. Since we want to avoid biasing the search toward any particular group of nodes, we want every node to have the same probability of being sampled. Therefore, we sample every time from the uniform distribution, which, for d-regular graphs (the case of our interest), corresponds to the stationary distribution. To reach the uniform distribution before a new sample, we need to perform a number of steps proportional to the mixing time t_m. To quantify the complexity of this type of search procedure, we adapt the cost notation used in Sec. II-B. Let $\mathrm{T_S}$ denote the cost of preparing the initial distribution and generating the first sample, $\mathrm{T_U}$ the cost of performing a step over the graph, and $\mathrm{T_C}$ the cost of verifying whether a node is marked. The overall complexity of this algorithm is expressed as:

$$\mathrm{T_S} + (\mathrm{T_U}/\delta + \mathrm{T_C})/\epsilon \tag{2}$$

Quantum walk. A quantum walk aims to evolve the state of a quantum system, through the application of a walk operator $\boldsymbol{U}_W$, in a way that mirrors the dynamics of a random walk on a graph [20]. Several models of quantum walks have been proposed in the literature, to construct a walk operator, with one of the earliest being the so-called coin model introduced by [5]. In this model, a *coin* state $|c\rangle$ is associated with each vertex of the graph. The coin role is to store the direction that will be followed in the next step of the walk, and a coin operator is used to initialize the coin at each step. Once a direction is selected, the walk proceeds through a *shift* operator that acts on the vertex space $|v\rangle$, performing the actual step. This operator can be seen as a permutation of the labels used to encode the vertices in the quantum state. Thus, the walk operator is a sequence of coin and shift operators.

While the coin model offers a simple way to define a quantum walk, it is not easily generalizable to arbitrary graph topologies. This is because assigning a unique labeling to each node and coin state using a finite-dimensional coin is not always possible [20, Chap. 7]. In this paper, we utilize the Szegedy, or double reflection, model, initially proposed by Ambainis to solve the *Element Distinctness Problem* [7] and later formalized by Szegedy in [6]. A single step in the Szegedy model consists of transitioning from an edge to a superposition of all edges originating from the endpoint of the initial edge, and then repeating this process.

MNRS search algorithm. In [21], the authors developed a search algorithm to find a marked vertex based on the Szegedy model. Their algorithm, known in the literature as the MNRS framework (named after the initials of the authors' surnames), consists of a similar amplitude amplification scheme to the one proposed by Grover, coupled with a Quantum Phase Estimation (QPE) routine to approximate the diffusion step, i.e., the reflection around the state corresponding to the initial superposition. Before presenting the main phases of the MNRS algorithm, we introduce the operators needed for the subsequent discussion. The operator $\boldsymbol{U}_\mathcal{V}$ creates a uniform superposition of all the quantum states corresponding to the vertices $v_i \in \mathcal{V}$ of the

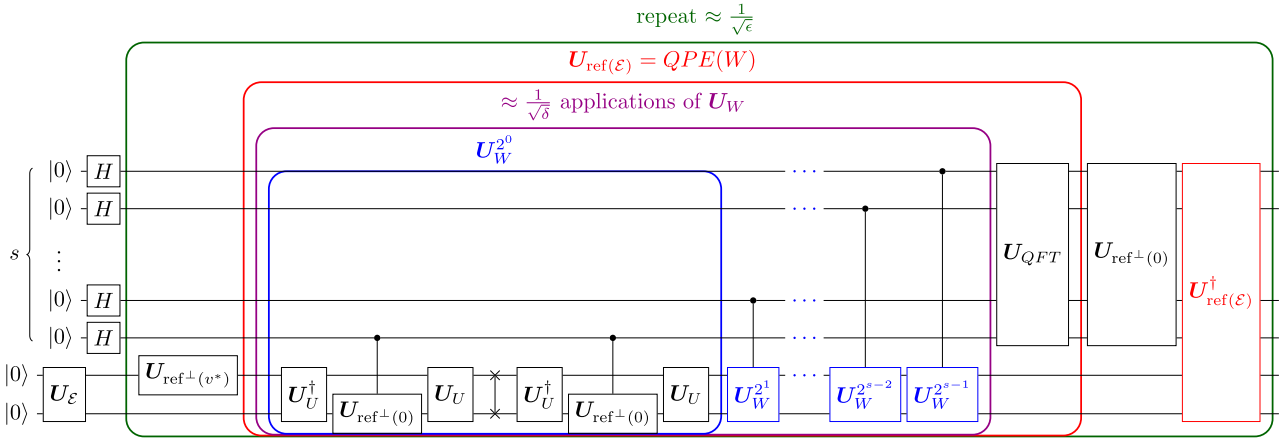

Fig. 3. Quantum circuit representation of the MNRS quantum walk search algorithm, consisting of three main steps: the input preparation $\boldsymbol{U}_{\mathcal{E}}$, producing the uniform superposition of al the edges of the graph; the oracle $\boldsymbol{U}_{\mathrm{ref}^{\perp}(v^*)}$, performing a reflection around the state orthogonal to the marked vertex v^*; the approximate reflection $\boldsymbol{U}_{\mathrm{ref}(\mathcal{E})}$, implemented through a Quantum Phase Estimation (QPE) circuit on the walk operator (W).

original graph. This is represented as $|\mathcal{V}\rangle{=}\frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}}|v\rangle$. This superposition will serve as the building block to generate the superposition over the edges of the double cover. We associate the cost of implementing this operator with $\mathrm{T_S}$. The update operator $\boldsymbol{U}_U$ generates the superposition of the adjacent vertices given the current state vertex i.e., $|a_v\rangle{=}\sum_{v\sim v_j}|v_j\rangle$. The cost of implementing $\boldsymbol{U}_U|v\rangle{=}|v\rangle|a_v\rangle$, is denoted by the update cost $\mathrm{T_U}$. The algorithm can be dived in three main stages: input preparation, oracle and the approximate diffusion.

Input preparation $\boldsymbol{U}_{\mathcal{E}}$. This operator creates a uniform superposition of all the quantum states corresponding to the edges $e_i{\in}\mathcal{E}$ of the double cover graph, represented as by $|\mathcal{E}\rangle=\frac{1}{|\mathcal{E}|}\sum_{v_i\in\mathcal{V}_1}|v_i\rangle|a_{v_i}\rangle$. Since the edge superposition can be achieved through the application of $\boldsymbol{U}_{\mathcal{V}}$ and $\boldsymbol{U}_U$ the cost of this step is $\mathrm{T_S}$ + $\mathrm{T_U}$.

Oracle $\boldsymbol{U}_{\mathrm{ref}^{\perp}(v^*)}$. Similar to the Grover algorithm, this step inverts the sign of the marked vertex, which contains the solution to the problem, i.e., $\boldsymbol{U}_{\mathrm{ref}^{\perp}(v^*)}|v^*\rangle=-|v^*\rangle$, leaving all the other vertices unchanged. As per Grover's algorithm, it is implemented through the sequence of gates $\boldsymbol{U}_f^{\dagger}\boldsymbol{U}_{\mathrm{ref}^{\perp}(1)}\boldsymbol{U}_f$. We denote the cost of $\boldsymbol{U}_f$ as $\mathrm{T_C}$, while the cost to implement $\boldsymbol{U}_{\mathrm{ref}^{\perp}(1)}$ is negligible; the total cost of the oracle is equal to $2\,\mathrm{T_C}$.

Approximate Reflection $\boldsymbol{U}_{\mathrm{ref}(\mathcal{E})}$. The walk operator $\boldsymbol{U}_W$ is used to take a step in the double cover graph, simulating the dynamics of a Markov chain on the edges. It is constructed using a double reflection approach: $\boldsymbol{U}_W=\boldsymbol{U}_{\mathrm{ref}(a_{v_j})}\boldsymbol{U}_{\mathrm{ref}(a_{v_i})}=\boldsymbol{U}^{\dagger}{}_U\boldsymbol{U}_{\mathrm{ref}^{\perp}(0)}\boldsymbol{U}_U\boldsymbol{U}^{\dagger}{}_U\boldsymbol{U}_{\mathrm{ref}^{\perp}(0)}\boldsymbol{U}_U$ which comes at a cost of $4\,\mathrm{T_U}$.

To understand how Quantum Phase Estimation (QPE) can be used to realize an approximate reflection over edge states, we need to consider the relationship between the eigenvalues of the stochastic matrix $\boldsymbol{P}$ of the original chain and those of the walk operator $\boldsymbol{U}_W$. The eigenvalues λ_i of $\boldsymbol{P}$ can be expressed as $|\lambda_i|{=}\cos(\theta_i)$, while the eigenvalues of $\boldsymbol{U}_W$ are $\eta_i{=}e^{\pm 2i\theta_i}$. Since $|\mathcal{E}\rangle$ corresponds to the stationary distribution, which is unique (as a Markov chain has a unique eigenvalue equal to 1 corresponding to the stationary distribution), $|\mathcal{E}\rangle$ is an eigenvector of $\boldsymbol{U}_W$ with eigenvalue 1, implying its phase is $\theta{=}0$. Therefore, QPE can estimate the phase of the eigenvalue corresponding to $|\mathcal{E}\rangle$, thus realizing the approximate reflection $\boldsymbol{U}_{\mathrm{ref}(\mathcal{E})}$. By applying the controlled version of the walk operator for approximately $\mathcal{O}(1/\sqrt{\delta})$ iterations [21], we achieve sufficient precision to estimate the correct phase.

Following an amplitude amplification scheme, this process requires applying the two reflections, $\boldsymbol{U}_{\mathrm{ref}^{\perp}(v^*)}$ and $\boldsymbol{U}_{\mathrm{ref}(\mathcal{E})}$, for $\mathcal{O}(1/\sqrt{\epsilon})$ iterations to achieve a probability close to 1 of measuring the marked vertex.

The total cost of the MNRS search algorithm is:

$$\mathrm{T_S}+(4\,\mathrm{T_U}/\sqrt{\delta}+2\,\mathrm{T_C})/\sqrt{\epsilon} \tag{3}$$

This cost has a quadratic advantage with respect to the classical random walk cost (compare Eq. (2)), and it outperforms a Grover-based search if $4\,\mathrm{T_U}/\sqrt{\delta}<2\,\mathrm{T_S}$ (compare Eq. 1).

A high-level view of the quantum circuit implementing the MNRS walk is illustrated in Fig. 3.

D. Subset Sum Problem as a Walk on a Johnson Graph

Given a set of n integers $\mathcal{X}$ the Subset Sum Problem is defined as finding a subset $\mathcal{S}\subset\mathcal{X}$ such that the sum of its elements equals a given value $p\geq 0$ i.e., $\sum_{x\in\mathcal{S}}x=p$. A common variation of the SSP is the *constrained subset-sum problem* (CSSP), which imposes an additional constraint on the size of the subset. Specifically, this version requires that the subset $\mathcal{S}$ must contain exactly k elements. Both formulations belong to the class of NP-Complete problems [22].

At a high level, the classical algorithm to retrieve a solution to the CSSP iterates through all the $\binom{n}{k}$ subsets $\mathcal{S}$ having size k. Its runtime is $\in\mathcal{O}\left(\binom{n}{k}\right)$ To solve a version of the SSP having subsets up to a fixed k, we can cycle through all the $\binom{n}{i}$ subsets of size $i\leq k$, and, for every one of them, check if the subset elements adds up to p. This approach would require $\sum_{i=1}^{k}\mathcal{O}(\binom{n}{i})\in\mathcal{O}(\binom{n}{k})$ operations, having the same asymptotic complexity. On the other hand, to solve the SSP, we can

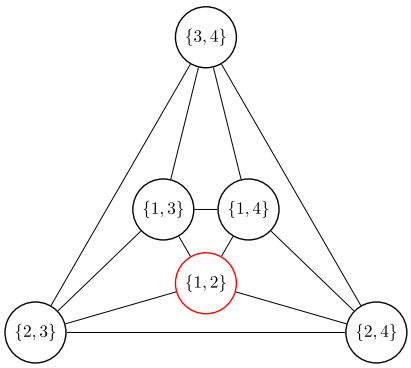

Fig. 4. An example of a Johnson graph $J(n=4,k=2)$ having as initial set $\mathcal{X}=\{1,2,3,4\}$, used to solve the CSSP with target value $p=3$. the vertex highlighted in red, which corresponds to the subset $\mathcal{S}=\{1,2\}$ represents the solution to the problem.

run its constrained version for all $i \in \{1,\ldots,\lceil n/2 \rceil\}$, since that, for values of $i > \lceil n/2 \rceil$, the problem can be tackled by finding the subset $|\mathcal{S}| = n - i$ whose elements add up to $t - p$, with t being the sum of all the integers of $\mathcal{X}$.

To tackle the CSSP using a quantum walk approach, we need to model the search space using a Markov chain associated to a graph suited for our needs. In this work, we use a Johnson graph $J(n,k)$, defined as an undirected graph $G=(\mathcal{V},\mathcal{E})$, where each vertex represents a size-k subset $\mathcal{S}$ of $\{1,\ldots,n\}$. There exists an edge between two vertices $\mathcal{S}_i$ and $\mathcal{S}_j$ if $|\mathcal{S}_i \cap \mathcal{S}_j| = k-1$, meaning that two vertices are adjacent if their corresponding sets differ by exactly one element. By definition, this graph has $|\mathcal{V}| = \binom{n}{k}$ vertices and $|\mathcal{E}| = \frac{1}{2}k(n-k)\binom{n}{k}$ edges. The property of Johnson graphs that make them interesting for quantum walks is that their spectral gap has a known value equal to be $\delta = n/k(n-k)$, that does not require the materialization of the stochastic matrix associated to the graph, and is big enough to retain the quadratic speedup offered by quantum walks. Using this model, solving the CSSP means finding a node in the Johnson graph such that the sum of the elements of its corresponding subset equals the target value. Fig. 4 shows an example of a Johnson graph $J(4,2)$, used to solve the CSSP for $p=3$.

III. Circuit Design

In this section, we present our construction of the MNRS quantum walk search algorithm over the Johnson graph to solve the CSSP. We outline the key implementation choices for the main components of the quantum circuit. In the following, we denote by m the number of bits or qubits required to represent the maximum value present in the initial set $\mathcal{X}$; that is, $m = \log_2(\max(\mathcal{X}))$.

All the main support registers used in our implementation are reported in Table I. As common in quantum computing, we assume that all the qubits are in state $|0\rangle$ at the beginning of the computation.

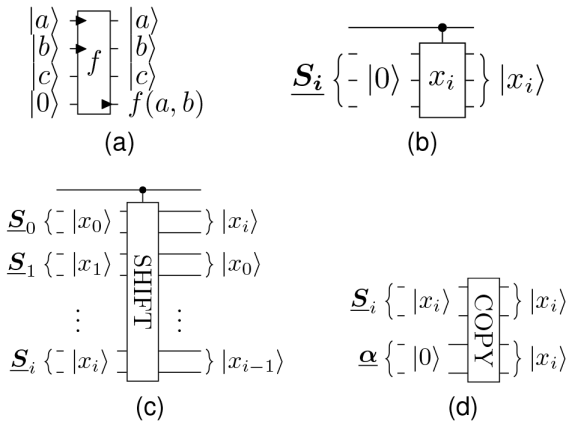

Fig. 5. The main abstract gates used in our quantum circuits. (a) a quantum gate implementing a reversible function f, for which the left black triangle(s) indicate which qubit(s) is taken as input, while the right black triangle denotes on which qubits the output of the computation is stored. (b) The $\texttt{C}-\texttt{x}_\texttt{i}$ gate, implemented through a series of sequential CNOT gates to write the binary encoding of x_i on the target qubits. (c) The SHIFT gate is a linear cyclic shift on a quantum register of size k. It corresponds to a sequence of CSWAP gates. (d) The COPY gate performs a bitwise copy of the value in a cell onto a new cell with a sequence of parallel CNOT.

TABLE I
Primary Quantum Registers Utilized in Our MNRS Quantum Walk Search Algorithm on the Johnson Graph

Quantum Register	Semantics of the content
$\underline{\boldsymbol{\mathcal{S}}}$, $\underline{\boldsymbol{\mathcal{T}}}$	Edge of the Johnson graph $J(n,k)$, with $\mathcal{S},\mathcal{T} \subset \{1,\ldots,n\}$, $\lvert\mathcal{S}\rvert = \lvert\mathcal{T}\rvert = k$, $\lvert\mathcal{S}\cap\mathcal{T}\rvert = k-1$
$\underline{\boldsymbol{\mathcal{S}'}}$, $\underline{\boldsymbol{\mathcal{T}'}}$	Complement sets, i.e. $\mathcal{S}' = \{1,\ldots,n\}\setminus\mathcal{S}$ and $\mathcal{T}' = \{1,\ldots,n\}\setminus\mathcal{T}$
$\underline{\boldsymbol{m}}$	Stores the sum of k values
$\underline{\boldsymbol{\alpha}}$, $\underline{\boldsymbol{\alpha'}}$	Stores the element to be inserted or deleted during the updates
$\underline{\boldsymbol{\sigma}}$	Stores the Dicke state $\lvert D_n^k\rangle$
$\underline{\boldsymbol{\omega}}$, $\underline{\boldsymbol{\omega'}}$	Stores the Dicke states $\lvert D_k^1\rangle$ and $\lvert D_{n-k}^1\rangle$, corresponding to W-states $\lvert W_k\rangle$ and $\lvert W_{n-k}\rangle$ respectively

A. Data Structures for Johnson Vertices

As discussed in Sec. II-D, the Johnson graph's favorable spectral gap and regularity makes it highly suitable for quantum walk applications. However, implementing it with elementary quantum gates necessitates defining data structures satisfying specific properties.

The data structure associated to the Johnson graph should encode a finite subset of elements, denoted by $\mathcal{S}$, without relying on QRAM, while efficiently support sampling, addition, and deletion of elements. Such a data structure should be implemented in a reversible way to comply with the quantum paradigm of computation. Furthermore, to permit predictable, non-random accesses, the internal arrangements of the elements must remain identical regardless of the specific sequence of insertion and deletions, a property known as *history independence*

[23]. Such property guarantees that each subset $\mathcal{S}_i$ has a unique quantum state representation $|\mathcal{S}_i\rangle$.

Several approaches to building history-independent data structures for sets are discussed in the literature, including combinations of hash tables and skip-lists [7], radix trees [10], and sorted arrays [14]. In this work, we adopt the sorted array approach and optimize its design for greater efficiency.

B. Operator $\boldsymbol{U_{\mathcal{V}}}$

The operator $\boldsymbol{U_{\mathcal{V}}}$ is used inside the input preparation circuit $\boldsymbol{U_{\mathcal{E}}}$. It generates the uniform superposition of all vertices belonging to the Johnson graph $J(n,k)$, and it corresponds to the cost $\mathrm{T_S}$ in Eq. (3).

Dicke state. Firstly, we use a circuit to generate the so-called Dicke state $|D_k^n\rangle$, which corresponds to a uniform superposition of the n-length bitstrings d having a Hamming weight equal to k, i.e., $|D_k^n\rangle = \binom{n}{k}^{-1/2} \sum_d |d\rangle$ with $Hw(|d\rangle) = k$. We generate the Dicke state on the register $\underline{\boldsymbol{\sigma}}$ using the circuit construction proposed in [24].

Vertex Binary Encoding (VBE). This circuit is used to store the elements of a subset $\mathcal{S}$ into k distinct cells of the quantum register $\underline{\boldsymbol{S}}$ in increasing order. Additionally, it stores the elements of the complement set $\mathcal{S}' = \mathcal{X} \setminus \mathcal{S}$ into the quantum register $\underline{\boldsymbol{S'}}$. In [15], the authors utilize the BIX procedure to generate a uniform superposition of the k-length subsets corresponding to the vertices of the Johnson graph, using the indices of the labels of the states contained in $\underline{\boldsymbol{\sigma}}$ as controls. Although their procedure correctly generates all the subsets corresponding to the vertices of the Johnson graph, it is limited to encoding only elements $\in \{1,\ldots,n\}$.

We improve with respect to their proposal to allow for the presence of both arbitrary values and duplicates. While the inclusion of arbitrary values does not pose relevant problems when used with our data structures, the existence of duplicates, though permissible in the CSSP formulation, prevents the use of the Johnson graph to directly encode the data. Knowing that the elements of $\mathcal{X}$ presents duplicates — it can be easily checked in time $\mathcal{O}(n)$ before generating the quantum circuit — we can use the elements of the set $\mathcal{X} = \{1,\ldots,n\}$ to label the nodes of the Johnson graph, while using each element of such a set to access an additional, fixed set of qubits storing the original data. Since the encoding procedure is the same for subsets with or without duplicates, we will present the design for the latter case, accounting for an additional lookup circuit when using the subsets as indices.

Fig. 6(a) illustrates the abstract gates used in our implementation for the vertex encoding procedure. We use a register $\underline{\boldsymbol{S}}$ of km qubits to encode the k elements contained in any given subset $\mathcal{S}_i$. Note that, while the elements contained in the set are known at circuit generation time, the superposition of edges require the Dicke state generation circuit, and hence available only at runtime. For this reason, after the Dicke state generation circuit, we perform i iterations, with $i \in \{0,\ldots,n-1\}$, and if the value of $\underline{\boldsymbol{\sigma}}_{|i} = 1$, we insert the value of $\mathcal{X}_i$ into $\underline{\boldsymbol{S_0}}$ using the gate $\mathtt{C{-}x_i}$. We then perform a cyclic left shift operation (gate $\mathtt{C{-}SHIFT}$) on $\underline{\boldsymbol{S}}$, which shifts the position of the value in $\underline{\boldsymbol{S}}$ by one position in a circular fashion. In this way, at the start of the next iteration, we have a blank slot ready for writing in $\underline{\boldsymbol{S}}$. Since all the values of $\mathcal{X}$ are known at circuit generation time, they can be inserted using a sequence of $\mathtt{CNOT}$ gates controlled by $\underline{\boldsymbol{\sigma}}_{|i}$, with the targets being the m qubits of $\underline{\boldsymbol{S_0}}$, used to hold the binary encoding of the value. If we want to encode only indices in the set $\{1,\ldots,n\}$, a $+1$ increment is applied at each iteration.

Similarly, we encode in $\underline{\boldsymbol{S'}}$ the complement subset $\mathcal{S}'=\{1,\ldots,n\}\setminus\mathcal{S}$; the only difference is that all the controlled operations are managed using the indices equal to zero on $\underline{\boldsymbol{\sigma}}$. The $\mathtt{C{-}SHIFT}$ gate requires $(k-1)m$ $\mathtt{CSWAP}$ gates. Since the control qubit is shared among all $\mathtt{CSWAP}$ gates, they are applied sequentially. The $\mathtt{C{-}x_i}$ gate requires the sequential application of m $\mathtt{CNOT}$ gates.

The total depth and number of gates used in the vertex encoding procedure is $\mathcal{O}(n)$, which is linear in the size of the initial set. The closed formulas for the complexity are reported in Table II.

C. Operator $\boldsymbol{U_f}$

The operator $\boldsymbol{U_f}$, used inside the oracle operator $\boldsymbol{U}_{\mathrm{ref}^{\perp}(v^*)}$, inverts the sign of the marked vertex that contains the solution to the problem by verifying if the sum of its elements equals the target value p. To implement this, we sum the k values from the register $\underline{\boldsymbol{S}}$ using a support register $\underline{\boldsymbol{m}}$ and a series of k distinct adders, each taking one element from $\underline{\boldsymbol{S}}$ and using $\underline{\boldsymbol{m}}$ as the second addend. The sum of k integers can be performed in various ways, such as summing in parallel and merging the partial results, or using a sequence of additions while saving qubits. Once the sum is computed in $\underline{\boldsymbol{m}}$, we can perform the quantum analogue of a logical bitwise XOR between the result and the target value p using a parallel chain of $\mathtt{CNOT}$ gates. For the adder construction, we utilize the circuit proposed in [25], which requires $5m{-}5$ $\mathtt{CNOT}$ gates and $2m{-}2$ $\mathtt{CCNOT}$ gates, with a depth equal to $5m{-}3$.

D. Operator $\boldsymbol{U_U}$

The update circuit for the operator $\boldsymbol{U_U}$, used inside the approximate reflection $\boldsymbol{U}_{\mathrm{ref}(\mathcal{E})}$, serves to realize the uniform superposition of the adjacent vertices of the vertex we are currently visiting, with the additional property that two vertices are adjacent if they differ by only one element (see Sec. II-D). Since we want to generate the new subset leveraging the information that we have already computed for the current vertex, the idea is to swap an element $x_i \in \mathcal{S}$ with another element $x_j \in \mathcal{X}\setminus\mathcal{S}$ for each of the $k(n-k)$ adjacent vertices. We use a second register $\underline{\boldsymbol{T}}$ to encode the superposition of the adjacent vertex subsets along with their complement sets on the register $\underline{\boldsymbol{T'}}$.

The high level overview of the update circuit is shown in Fig. 6(b). In the picture, we do not represent the copy step performed as a first operation.

Copy $\underline{\boldsymbol{S}}$ into $\underline{\boldsymbol{T}}$. The update circuit expects to find the registers $\underline{\boldsymbol{T}}$ and $\underline{\boldsymbol{T'}}$ to be in the all-zero state. As a first step, we then copy the contents of $\underline{\boldsymbol{S}}$ into $\underline{\boldsymbol{T}}$, and similarly, we copy $\underline{\boldsymbol{S'}}$ into

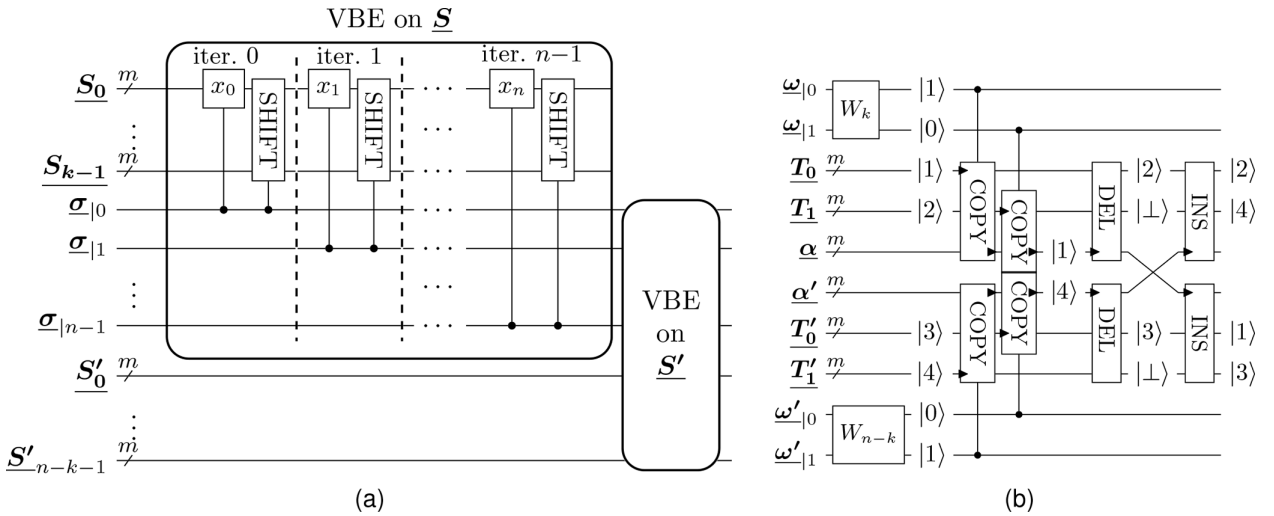

Fig. 6. (a) The Vertex Binary Encoding (VBE) circuit, used to store the elements of $\mathcal{S}$ in $\underline{S}$. each gate at iteration i is controlled by the i-th qubit of $\underline{\sigma}$ being in state $|1\rangle$. the C−x_i gate inserts the value x_i, available at circuit generation time, into the first cell of $\underline{S}$, while the C−SHIFT gate performs a cyclic left shift on all the cells of $\underline{S}$. the circuit is analogously applied to $\underline{S'}$, with the only difference that the controlled gates at iteration i are applied only if the qubit i of $\underline{\sigma}$ is in state $|0\rangle$. (b) the quantum circuit corresponding to the operator U_U, in which the first stage, corresponding to a copy of $\underline{S}$ ($\underline{S'}$) on $\underline{T}$ ($\underline{T'}$) is omitted. To explain its functioning, we provide the state of the quantum registers after each gate application for a specific example of a johnson graph $J(4,2)$ having initial set $\mathcal{X}$=$\{1,2,3,4\}$. the quantum register $\underline{T}$ is assumed to encode the subset $\{1,2\}$, while $\underline{T'}$ encodes the complement set $\{3,4\}$. while the W states generate a superposition state, the example assumes to work with only one state of the superposition, specifically $|10\rangle$ for $\underline{w}$ and $|01\rangle$ for $\underline{w'}$. the i-th qubit of $\underline{w}$ is used to control the sampling of the element of $\underline{T}$ placed in the i-th cell, and copy it into $\underline{\alpha}$ (the C−COPY gate); the same applies to $\underline{w'}$. then, the element stored in $\underline{\alpha}$ is deleted from $\underline{T}$ (DEL gate) and inserted in $\underline{T'}$ (INS gate). analogously, the element stored in $\underline{\alpha'}$ is deleted from $\underline{T'}$ and inserted in $\underline{T}$.

TABLE II
COST METRICS IN TERMS OF NUMBER OF GATES, DEPTH, AND NUMBER OF QUBITS USED IN THE QUANTUM CIRCUIT DESIGN, DIVIDED BY GATE TYPE, FOR THE MAIN SUBCIRCUIT COMPONENT DESCRIBED IN SEC. III ADAPTED TO A QUANTUM WALK ON THE JOHNSON GRAPH $J(n,k)$. THE OPERATOR $U_\mathcal{V}$ GENERATES A UNIFORM SUPERPOSITION OF JOHNSON GRAPH VERTICES; IT CORRESPONDS TO COST $\mathrm{T_S}$ IN EQ. (3). THE U_f IMPLEMENTS THE FUNCTION f; IT CORRESPONDS TO COST $\mathrm{T_C}$ IN EQ. (3). THE OPERATOR U_U GENERATES THE SUPERPOSITION OF ALL VERTICES ADJACENT TO THE CURRENTLY VISITED VERTEX AND CORRESPONDS TO COST $\mathrm{T_U}$ IN EQ. (3). FOR THIS OPERATOR, WE REPORT BOTH A LOW-WIDTH AND LOW-DEPTH IMPLEMENTATION

Costs	$U_\mathcal{V}$		U_f	U_U	
	Dicke	**VBE**		**Low-Width**	**Low-Depth**
X	k	$2n$	0	2	$64nm-46n$
$\mathrm{R_y}$	$4nk-4k^2-2n+1$	0	0	$n-2$	$n-2$
CNOT	$5nk-5k^2-5n$	$2nm$	$5km-5k$	$17nm-11n-28m+22$	$17nm-11n-2$
CCNOT	0	0	$2km-2k$	$9nm-4n-16m+12$	$25nm-20n$
CSWAP	0	n^2m-2nm	0	$2nm-4m$	$4nm$
Depth	$\frac{27nk-12n-27k^2+3}{k-2}$	$n^2m-nkm+2n$	$5km-3k$	$10nm-10km+n-k+12n-12k-6m+2\log_2(n-k)-7$	$n-k+6\log_2(n-k)+56\log_2(m)+11$
Qubit	n	nm	$\log_2(\frac{2nk-k^2+k}{2})$	$nm+n+2m+4$	$7nm+n+2m-3$

$\underline{T'}$. This operation can be realized through CNOT gates having as control qubits the one in $\underline{S}$ ($\underline{S'}$), and as target qubits the ones in $\underline{T}$ ($\underline{T'}$).

Uniform sample from $\underline{T}$ and $\underline{T'}$. This stage selects a uniformly random element from $\underline{T}$. To do so, we generate the Dicke states $|D^1\rangle_k$ and $|D^1\rangle_{(n-k)}$ on $\underline{w}$ and $\underline{w'}$, respectively. These states represent a superposition of all the bitstrings of size k and $n-k$ respectively, each one having a single qubit set to $|1\rangle$. Such special cases of Dicke states are also known as W-states, and for this reason we denote them as $|W_k\rangle$ and $|W_{n-k}\rangle$ respectively. As reported in [26], the circuit generating such a state has a logarithmic-depth and a linear amount of gates, and does not require ancillary qubits.

The unique qubit of $\underline{w}$ ($\underline{w'}$) having value $|1\rangle$ is used as a control qubit to sample a unique element from $\underline{T}$ ($\underline{T'}$) and copy it to the quantum register $\underline{\alpha}$ ($\underline{\alpha'}$). This sample and copy procedure, denote as C−COPY, can be realized through a sequence of CCNOT gates. While all the gates of a single C−COPY all depend on the same control qubit of $\underline{w}$ ($\underline{w'}$), we can interleave the CCNOT gates belonging to different C−COPY gates, and therefore achieve a depth of $n-k$.

Insertion and deletion. For the insertion and deletion procedure, we describe two different strategies: the one proposed in [14], which has a lower depth, and a novel procedure that minimize number of qubits. We describe the two procedures using the quantum register $\underline{T}$, since the circuit for

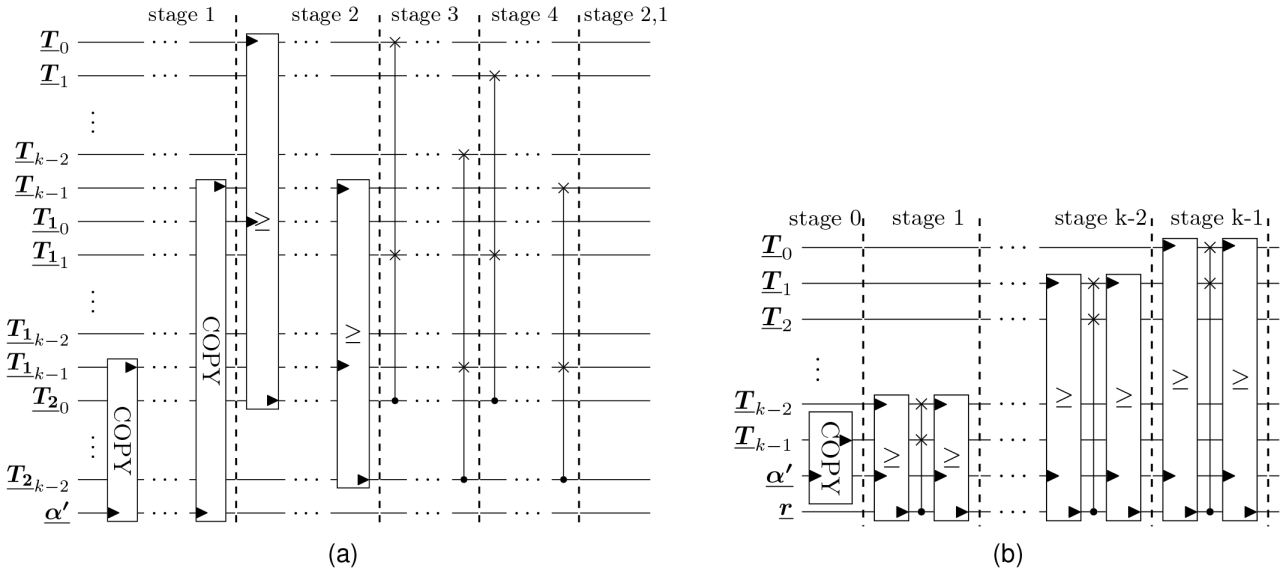

Fig. 7. (a) Low depth insertion circuit for vector $\underline{\boldsymbol{T}}$ from [14]. It is structured in six stages. The circuit uses two additional registers of size k, $\underline{\boldsymbol{T_1}}$ and $\underline{\boldsymbol{T_2}}$. first, the sample value from $\underline{\boldsymbol{\alpha}}$ is copied into $\underline{\boldsymbol{T_1}}$ and the last cell of $\underline{\boldsymbol{T}}$. In the second iteration, the greater or equal result of the comparison between the i-th cell of $\underline{\boldsymbol{T}}$ and $\underline{\boldsymbol{T_1}}$ is written to $\underline{\boldsymbol{T_2}}$. the next two iterations use conditional swap gates to move the sampled element into its correct position within $\underline{\boldsymbol{T}}$. Finally the last two iteration are used to uncompute the additional ancilla qubits. (b) Low width circuit has a sequential design that uses only one additional ancilla qubit, $\underline{\boldsymbol{r}}$, to control the insertion operations. First, the value in $\underline{\boldsymbol{\alpha}}$ is copied into the last cell of $\underline{\boldsymbol{T}}$. At each iteration i, the value of cell $\underline{\boldsymbol{T}}_{|i}$ is swapped with the value in $\underline{\boldsymbol{T}}_{|i-1}$ if the value in the current cell is greater than the value in the previous cell. A second comparison is needed to reset $\underline{\boldsymbol{r}}$ to 0 after each iteration.

$\underline{\boldsymbol{T'}}$ is identical. Additionally, the deletion circuit (DEL) is the conjugate transpose of the insertion circuit INS, and therefore its description is omitted. In both cases, the insertion circuit assumes to find a blank space, denoted as $|\perp\rangle$, in the last cell of the register in which we perform the insertion, i.e., $\underline{\boldsymbol{T}}_{|k-1} = |\perp\rangle$. Such a blank space is due to a previous deletion. Both implementations ensure that at the end of an insertion/deletion procedure, the resulting set is ordered. This is essential for the history-independent property, which guarantees the correct interference between computational paths.

The low-depth strategy, shown in Fig. 7(a) and first described in [14], is composed of six stages. It uses two additional registers, $\underline{\boldsymbol{T_1}}$ and $\underline{\boldsymbol{T_2}}$, having the same size as $\underline{\boldsymbol{T}}$. First, the value contained in $\underline{\boldsymbol{\alpha'}}$ is copied into all the cells of $\underline{\boldsymbol{T_1}}$ and into the last cell of $\underline{\boldsymbol{T}}$. This can be done using km CNOT gates with a depth of $\log k$ using a fan-out procedure.

In the second stage, each cell of $\underline{\boldsymbol{T}}$ is compared with the one at the same positions in $\underline{\boldsymbol{T_1}}$ using k parallel comparators, storing an all-ones string in the corresponding qubits of $\underline{\boldsymbol{T_2}}$ if the value contained in $\underline{\boldsymbol{T}}$ is greater or equal than the corresponding value in $\underline{\boldsymbol{T_1}}$. For such comparison, we implement and optimize the logarithmic-depth comparator circuit outlined in [27], which compares two m-bit bitstrings at a cost of $6m-5$ CCNOT gates, $3m-3$ CNOT gates, and $16m-12$ X gates. This comparator requires $4m-4$ ancilla qubits and has a depth of $14\log(m)+3$.

Based on the values of $\underline{\boldsymbol{T_2}}$ the third stage apply k conditional swaps between the cell $\underline{\boldsymbol{T}}_{|i}$ and $\underline{\boldsymbol{T}}_{|i+1}$ if the value of $\underline{\boldsymbol{T_2}}_{|i}$ is equal to the all-ones bitstring. All the k conditional swaps can be performed in parallel using m CSWAP gates for each cell. A second layer of conditional swaps, controlled again by $\underline{\boldsymbol{T_2}}_{|i}$, is then applied between $\underline{\boldsymbol{T}}_{|i+1}$ and $\underline{\boldsymbol{T_1}}_{|i+1}$. This also costs km CSWAP gates, and has a depth of 1.

At the end of the first four stages, the original value to be inserted, stored in $\underline{\boldsymbol{\alpha'}}$, is placed in its correct position within $\underline{\boldsymbol{T}}$, ensuring that all values in $\underline{\boldsymbol{T}}$ greater than the value to be inserted are moved one position forward. The last two stages resets the cells of $\underline{\boldsymbol{T_2}}$ and $\underline{\boldsymbol{T_1}}$ to zero by performing the second stage and first stage again.

The low-width version for the update utilizes only three additional ancillae qubits to insert a new element, at the cost of increasing the depth of the procedure. The strategy, whose full design is reported in Fig. 7(b), first copies the value stored in $\underline{\boldsymbol{\alpha'}}$ into the last cell of $\underline{\boldsymbol{T}}$ using m parallel CNOT gates, and then performs $k-1$ identical stages. Each stage i, with $1 \leq i < k$, compare the value contained in $\underline{\boldsymbol{T}}_{|k-i-1}$ with the value contained in $|\alpha\rangle$ and XOR the state $|1\rangle$ in the single-qubit register $\underline{\boldsymbol{r}}$ if the comparison holds true. Since the register $\underline{\boldsymbol{T}}$ starts with all elements in increasing order, and without duplicates, the result of the $\underline{\boldsymbol{T_{k-i}}}$ contains the new element to be inserted, originally stored in $\underline{\boldsymbol{\alpha'}}$, and such element is not in the correct position. For this reason, using the value in $\underline{\boldsymbol{r}}$ as control, we perform a controlled swap operation between $\underline{\boldsymbol{T}}_{|k-i}$ and $\underline{\boldsymbol{T}}_{|k-i-1}$, thus shifting the lowest value in the lower-index cell. To reuse the same ancillary qubit $\underline{\boldsymbol{r}}$ in the next stage, we perform the comparison $\underline{\boldsymbol{T}}_{|k-i-1} = \underline{\boldsymbol{\alpha'}}$. This comparison holds true only if the previous comparison was positive, and hence, by XORing into $\underline{\boldsymbol{r}}$ the value $|1\rangle$, it brings back its state to $|0\rangle$ Since all the swaps are controlled by the same qubit in $\underline{\boldsymbol{r}}$, all operations must be performed sequentially for each stage.

Since this variation aims to reduce the number of ancillary qubits, all the comparators are implemented using the high-bit-only design proposed in [28], which uses only 1 ancillary qubits as input, $2m-1$ CCNOT, $4m-3$ CNOT, and has a linear depth of $2m+3$. Specifically, while the approach uses the input qubits for intermediate computations, it restores all of them to their original value.

IV. Performance Analysis and Validation

In order to evaluate the computational complexity of our quantum walk circuit implementation, we compare it to a Grover-based search strategy. This comparison aims to assess whether the theoretical advantages of a structured search are materialized when considering the implementation associated costs, including those related to auxiliary data structures. The source code associated to our implementation will be made publicly available upon publication.

Both algorithms rely on an amplitude amplification scheme, which, after the initial superposition preparation, requires $\mathcal{O}(1/\sqrt{\epsilon})$ iterations of the oracle and the diffusion operator for Grover's algorithm or the approximate reflection operator for the quantum walk. Since the initialization only occurs once for both procedures, and the oracle construction (involving a sequence of addition operations) is shared between both approaches, the major cost differences arise from the diffusion operator $\boldsymbol{U}_{\mathrm{ref}(\mathcal{X})}$ in Grover and the approximate reflection operator $\boldsymbol{U}_{\mathrm{ref}(\mathcal{E})}$ in the walk. Consequently, our analysis focuses on these operator differences.

To ensure a consistent comparison between Grover's approach and the quantum walk approach, we reuse the same circuit construction whenever possible. In Grover's algorithm, for instance, we use the vertex encoding procedure from Sec. III-B to initialize the input superposition. Since the goal in the CSSP is to generate quantum states corresponding to a superposition of sets of size k, we reuse the vertex encoding circuit to produce the superposition over valid subsets $\mathcal{S}_i$ (excluding their complement sets $\mathcal{S'}_i$, which are only involved in the update step of the walk). Thus, for the diffusion step, we account for the cost of applying the vertex encoding twice, once to uncompute the initial superposition and once to recompute it, as well as the cost to construct a reflection through the all-zero state.

For the walk, we account for the costs associated with the QPE routine, which comprises $\mathcal{O}(1/\sqrt{\delta})$ iterations of the walk operator $\boldsymbol{U}_W$, plus the cost of implementing the Quantum Fourier Transform on the qubit used to estimate the phase of the walk operator, we add also and the cost of uncomputing the QPE routine. Additionally, we include the cost of applying a reflection through the all-zero state on the qubit used to estimate the phase.

To compare the two procedures beyond their asymptotic costs, we first need a set of basic operations to represent all the abstract gates discussed in Sec. III. For this purpose, we utilize the Clifford+T gate set. Indeed, for two-dimensional topological codes, such as surface codes, which are among the most promising error-correcting codes [29], all Clifford gates can be implemented transversally by applying the same gate to all physical qubits associated with a logical qubit. However, the T gate requires different techniques for its implementation, including magic state distillation, code switching, and lattice surgery, which necessitate extensive resource consumption. Consequently, performance of quantum circuits is often characterized in terms of T-count and T-depth. We do not claim that the decompositions into Clifford+T employed in this work are optimal, as many alternatives exist with polynomial overhead [30], targeting various optimization objectives such as depth or width.

With respect to one-qubit gates, the only gate employed not belonging to the Clifford+T set is the $\mathtt{R_y}(\theta)$ gate. The authors of [31] propose a synthesis algorithm for approximate unitaries up to a precision ξ, utilizing only discrete gates. Fixing an arbitrary precision ξ, it requires $3.067\log_2(1/\xi)-4.322$ T gates to achieve the desired approximation. The authors additionally report that a value $\xi=10^{-15}$ is sufficient for most applications of practical interest; as reported in [32], this value results in a T-count and T-depth of approximately 149 for the $\mathtt{R_y}(\theta)$ gate. For the multi-controlled gate, we decompose the CCNOT gates using two different strategies: for the low-width implementation, we use the decomposition strategy from [30], which requires 4 T gates with a T-depth of 3, without ancillary qubits; for the low-depth implementation, we use the decomposition strategy of [33], having a T-depth of 1 at the expense of 1 additional qubit. The CSWAP can be decomposed into one CCNOT and two CNOT gates, and hence only requires the decomposition of the CCNOT gate as before. For quantum gates with $\mathtt{C}^n$ controls, we utilize the decomposition strategy presented in [34], which requires $2n-4$ CCNOT gates and a depth of $\log_2(n+1)+1$, plus $n-2$ ancilla qubits.

In Fig. 8, we present the comparison between the efficiency of the proposed quantum walk strategy, low-width construction, and the Grover search approach. The ratio between the cost of the approximate reflection of the walk operator (using the low-width update implementation) and the diffusion operator (W/G) is presented in terms of T-count, T-depth, and T-depth multiplied by width. We focus on the low-width implementation because it exhibits an asymptotically lower T-depth compared to Grover's algorithm. Additionally, the low-depth approach demonstrates even lower T-depth, while the difference in width is not substantial enough to affect the asymptotic behavior of T-depth$\times$width. We chose a range for the CSSP that includes values of n and k low enough to be tackled classically, as well as values that are sufficiently large to be deemed impractical even for a quantum computer. This range is particularly useful for designing post-quantum cryptosystems [16]. We report the results for four values of the ratio k/n: 0.2, 0.3, 0.4, and 0.5.

Our quantum walk approach outperforms Grover's search in terms of asymptotic behavior, further improving the design proposed in [15]. Specifically, we observe that our approach begins to show a lower T-count for all values of k/n starting from $n\approx 2^5$, and a lower T-depth from $n\approx 2^{13}$. When considering parameter sizes large enough to be used in cryptosystems, i.e., $n\geq 2^{13}$, our approach yields quantum circuits that are up to 50% shorter in terms of T-depth compared to Grover's. Finally, when evaluating T-depth$\times$width, which captures the effective

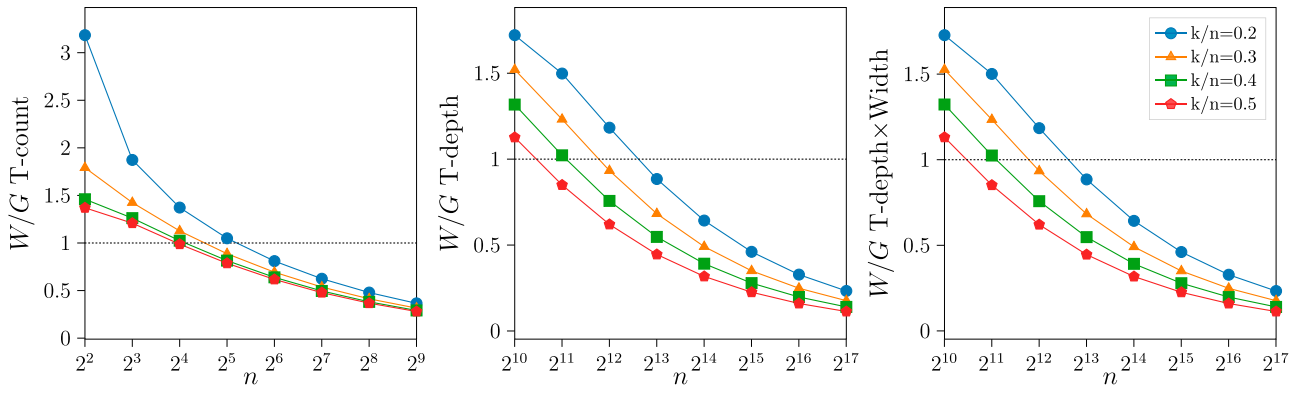

Fig. 8. T-Count, T-depth and T-depth×width ratios between the low-depth implementation of the quantum-walk approximate reflection operator $\boldsymbol{U}_{\mathrm{ref}(\mathcal{E})}$ (W), and Grover's diffusion $\boldsymbol{U}_{\mathrm{ref}(\mathcal{X})}$ (G), as a function of the parameters n and k/n of the constrained subset-sum problem. It can be noticed how the quantum walk approach outperforms Grover's search in terms of asymptotic behavior. The low-depth implementation, not shown, exhibits an even better performance of the quantum-walk approach when compared to the Grover-based one.

TABLE III
NUMBER OF QUBITS, CIRCUIT DEPTH, AND NUMBER OF GATES FOR SMALL CSSP INSTANCES, ASSUMING FULL QUBIT CONNECTIVITY AND QISKIT'S DEFAULT SIMULATION BASIS GATE SET, CONSISTING OF THE UNIVERSAL SINGLE-QUBIT ARBITRARY-ROTATIONU GATE AND THE TWO-QUBIT CNOT GATE

n	k	m	Qubits	U	CNOT	Depth
4	2	2	38	4107	3453	2220
5	2	3	57	8910	7440	5711
7	3	3	75	14915	12420	8837
10	5	4	126	32408	26916	16095

use of a quantum computer, as quantum gates on some qubits require others to remain idle—our approach demonstrates significant improvements for cryptosystem-relevant parameters.

A distinguishing feature of our quantum-walk-based approach is the explicit encoding of Johnson graph indices. This encoding incurs an overhead of $n\log_2(n)$ additional qubits compared to the $\mathcal{O}(n)$ required by Grover-based implementations. Although this overhead has negligible impact on overall circuit complexity for practical problem sizes—as shown by the minimal difference between the depth and depth×width metrics in Fig. 8—it significantly limits the feasibility of classical simulation. In particular, classical simulation on standard hardware (e.g., systems with 64 GiB of RAM) is typically limited to circuits of about 32 qubits[1]. However, as reported in Table III, resource estimates obtained with the Qiskit simulator[2] for small problem instances ($n \in \{4,\ldots,10\}, k = \lfloor n/2 \rfloor$) show that even the smallest non-trivial case ($n = 4, k = 2$) already requires 38 qubits, exceeding the practical limits of classical simulation.

For this reason, to ensure the correctness of our design, we validated and tested all key components of the proposed quantum circuit architecture. To facilitate reproducibility and enable further experimentation, we provide open-source implementations in both the Qiskit framework[3] and the Atos QLM environment[4].

V. DISCUSSION AND CONCLUSION

We presented a complete design of a quantum walk-based search algorithm over a Johnson graph to accelerate the solution of the constrained subset-sum problem. Our algorithm is formulated in terms of logical qubits and noiseless quantum gates, intentionally abstracting away hardware constraints. Like other algorithmic families that rely on similar assumptions, such as Grover's and Shor's algorithms [35], it is designed for *fault-tolerant quantum computing* (FTQC) platforms, which are assumed to exhibit quantum error correction capabilities that enable a reliable execution of deep circuits as well as high-fidelity gate operations.

Since our analysis targets the fault-tolerant regime, the feasibility of implementing our approach on near-term *Noisy Intermediate-Scale Quantum* (NISQ) devices [36] remains limited. Current hardware typically supports around 100 qubits, with constrained connectivity and relatively high error rates[5] (in the order of 10^{-3}, with respect to the 10^{-15} required by FTQC algorithms [32]), which prevent the execution of our quantum-walk-based approach. Exploring hybrid or hardware-efficient variants of our method, while further reducing both circuit depth and width, could be a promising direction for future research.

Our proposal improves upon the current state-of-the-art techniques for generating a uniform superposition of all subsets of size k from an n-sized set of integers. We detailed the quantum circuit construction for each component of our algorithm and provided closed-form complexity expressions for their respective costs. Additionally, we compared the performance of our

[1] See, for example, https://fiqci.fi/publications/2025-04-01-LUMI-quantum-simulations-qiskit-aer

[2] Freely available at https://github.com/Qiskit/

[3] https://github.com/mantixpolimi/SubsetSum_QWSearch.git

[4] https://github.com/paper-codes/2025-TC

[5] For example IBM reports https://newsroom.ibm.com/2025-06-23-ibm-and-riken-unveil-first-ibm-quantum-system-two-outside-of-the-u-s, for its Heron quantum computer, error gates in the order of $\approx 10^{-3}$

solution against a Grover-based search, using metrics such as T-count, T-depth, and T-depth×width. Our results show improvements across all metrics, demonstrating that the advantages of quantum walks persist even for very large parameter regimes—particularly those relevant to cryptanalytic applications. These results demonstrate that while the asymptotic cost suggests that Grover's and the quantum walk framework both achieves the same quadratic speedup with respect to classical solvers, practical implementations may result in polynomial advantages of one with respect to the other.

Our algorithm, which solves the *constrained subset-sum problem (CSSP)* over n items with a constant size k of the subset, can be naturally extended to address the unconstrained subset-sum problem by running the solver for the constrained version for all subset sizes $k \in \{1, \ldots, \lfloor n/2 \rfloor\}$. For $k > \lfloor n/2 \rfloor$, the problem can be formulated to check whether a complementary subset of size $n - k$ satisfies the corresponding complementary condition on the total sum of the input. Since the number of combinations grows with k, the worst-case of the execution time complexity of the unconstrained solver is dominated by the case $k = \lfloor n/2 \rfloor$.

The design of the CSSP solver can be naturally extended to address the 0-1 Knapsack problem, where each item is associated with both a weight and a value, and the objective is to maximize the total value of the selected items while ensuring that their total weight does not exceed a fixed limit W^*. In this setting, the CSSP oracle circuit is adapted to simultaneously enforce two constraints: (i) the total weight of the selected subset must not exceed W^*, and (ii) the total value must be at least a given threshold V^*. For a fixed subset size k, a binary search is performed over V^* to identify the maximum achievable value under the weight constraint. At each iteration, only the threshold value V^* within the oracle circuit is updated; the remaining CSSP subcircuits remain unchanged. This procedure is repeated for all $k \in \{1, \ldots, \lfloor n/2 \rfloor\}$, and the best feasible solution across all subset sizes is selected. The overhead introduced by this strategy, relative to the original SSP solver, is logarithmic in the total sum of item values, and thus negligible in practice.

We emphasize that our standalone circuit constructions, such as the Vertex Binary Encoding to generate a uniform superposition of vertices of a graph, the uniform sample of adjacent vertices of a node of a graph, and the insertion circuit into an ordered data structure, may hold independent interest.

We leave as future works the adaptation of our design to other domain-specific problems (e.g., the syndrome decoding problem for code-based cryptography [32], [34], [37], [38]), and the analysis of the computational complexity when other kind of graphs, both regular and irregular, are used to model the search space.

REFERENCES

[1] F. Arute et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 501–510, 2019.

[2] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, "High-threshold and low-overhead fault-tolerant quantum memory," *Nature*, vol. 627, no. 8005, pp. 778–782, 2024, doi: 10.1038/s41586-024-07107-7.

[3] S. Aaronson, "BQP and the polynomial hierarchy," in *Proc. 42nd ACM Symp. Theory Comput. (STOC)*, L. J. Schulman, Ed. Cambridge, MA, USA: ACM, Jun. 2010, pp. 141–150, doi: 10.1145/1806689.1806711.

[4] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput.*, G. L. Miller, Ed. Philadelphia, PA, USA: ACM, May 22–24, 1996, pp. 212–219, doi: 10.1145/237814.237866.

[5] N. Shenvi, J. Kempe, and K. B. Whaley, "Quantum random-walk search algorithm," *Phys. Rev. A*, vol. 67, May 2003, Art. no. 052307. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.67.052307

[6] M. Szegedy, "Quantum speed-up of Markov chain based algorithms," in *Proc. 45th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, Washington, DC, USA: IEEE Computer Society, 2004, pp. 32–41, doi: 10.1109/FOCS.2004.53.

[7] A. Ambainis, "Quantum walk algorithm for element distinctness," *SIAM J. Comput.*, vol. 37, no. 1, pp. 210–239, 2007.

[8] G. Kachigar and J. Tillich, "Quantum information set decoding algorithms," in *Proc. 8th Int. Workshop, Post-Quantum Cryptography (PQCrypto)*, T. Lange and T. Takagi, Eds. Utrecht, The Netherlands. New York, NY, USA: Springer-Verlag, Jun. 26–28, 2017, vol. 10346, pp. 69–89, doi: 10.1007/978-3-319-59879-6_5.

[9] S. Tani, "Claw finding algorithms using quantum walk," *Theor. Comput. Sci.*, vol. 410, no. 50, pp. 5285–5297, 2009, doi: 10.1016/j.tcs.2009.08.030.

[10] D. J. Bernstein, S. Jeffery, T. Lange, and A. Meurer, "Quantum algorithms for the subset-sum problem," in *Proc. 5th Int. Workshop Post-Quantum Cryptography (PQCrypto)*, P. Gaborit, Ed. Limoges, France. New York, NY, USA: Springer-Verlag, 2013, vol. 7932, pp. 16–33, doi: 10.1007/978-3-642-38616-9_2.

[11] A. Helm and A. May, "Subset sum quantumly in 1.17^n," in *Proc. 13th Conf. Theory Quantum Comput., Commun. Cryptography (TQC)*, S. Jeffery, Ed., Schloss Dagstuhl - Leibniz-Zentrum Für Informatik, 2018, vol. 111, pp. 5:1–5:15, doi: 10.4230/LIPIcs.TQC.2018.5.

[12] X. Bonnetain, R. Bricout, A. Schrottenloher, and Y. Shen, "Improved classical and quantum algorithms for subset-sum," in *Proc. Adv. Cryptology - ASIACRYPT 2020 - 26th Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Daejeon, South Korea, S. Moriai and H. Wang, Eds. New York, NY, USA: Springer-Verlag, vol. 12492, 2020, pp. 633–666, doi: 10.1007/978-3-030-64834-3_22.

[13] S. Jaques and A. G. Rattew, "QRAM: A survey and critique," 2023. [Online]. Available: https://arxiv.org/abs/2305.10310

[14] S. Jaques and J. M. Schanck, "Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE," in *Proc. 39th Annu. Int. Cryptology Conf. Adv. Cryptology (CRYPTO)*, Santa Barbara, CA, USA, A. Boldyreva and D. Micciancio, Eds. New York, NY, USA: Springer-Verlag, 2019, vol. 11692, pp. 32–61. [Online]. Available: https://doi.org/10.1007/978-3-030-26948-7_2

[15] G. Lancellotti, S. Perriello, A. Barenghi, and G. Pelosi, "Design of a quantum walk circuit to solve the subset-sum problem," in *Proc. 61st ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, V. De, Ed. New York, NY, USA: ACM, 2024, pp. 1–298, doi: 10.1145/3649329.3657337.

[16] V. Lyubashevsky, A. Palacio, and G. Segev, "Public-key cryptographic primitives provably as secure as subset sum," in *Proc. 7th Theory Cryptography Conf. Theory of Cryptography (TCC)*, Zurich, Switzerland, D. Micciancio, Ed. New York, NY, USA: Springer-Verlag, 2010, vol. 5978, pp. 382–400, doi: 10.1007/978-3-642-11799-2_23.

[17] J. C. Lagarias and A. M. Odlyzko, "Solving low-density subset sum problems," in *Proc. 24th Annu. Symp. Found. Comput. Sci.*, Tucson, AZ, USA. Washington, DC, USA: IEEE Computer Society, Nov. 1983, pp. 1–10, doi: 10.1109/SFCS.1983.70.

[18] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. New York, NY, USA: Springer-Verlag, 2012.

[19] D. Biesner, T. Gerlach, C. Bauckhage, B. Kliem, and R. Sifa, "Solving subset sum problems using quantum inspired optimization algorithms with applications in auditing and financial data analysis," 2022. [Online]. Available: https://arxiv.org/abs/2211.02653

[20] R. Portugal, *Quantum Walks and Search Algorithms*. New York, NY, USA: Springer-Verlag, 2013.

[21] F. Magniez, A. Nayak, J. Roland, and M. Santha, "Search via quantum walk," in *Proc. 39th Annu. ACM Symp. Theory Comput.*, San Diego, CA, USA, Jun. 11–13, 2007, pp. 575–584.

[22] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: W. H. Freeman, 1979.

[23] M. Naor and V. Teague, "Anti-persistence: History independent data structures," in *Proc. IACR Cryptol.*, 2001, p. 36. [Online]. Available: http://eprint.iacr.org/2001/036

[24] A. Bärtschi and S. J. Eidenbenz, "Deterministic preparation of Dicke states," in *Proc. 22nd Int. Symp. Fundamentals Comput. Theory*

(FCT), L. A. Gasieniec, J. Jansson, and C. Levcopoulos, Eds. New York, NY, USA: Springer-Verlag, 2019, vol. 11651, pp. 126–139, doi: 10.1007/978-3-030-25027-0_9.
[25] Y. Takahashi, S. Tani, and N. Kunihiro, "Quantum addition circuits and unbounded fan-out," *Quantum Inf. Comput.*, vol. 10, no. 9&10, pp. 872–890, 2010, doi: 10.26421/QIC10.9-10-12.
[26] D. Cruz et al., "Efficient quantum algorithms for GHZ and W states, and implementation on the IBM quantum computer," *Adv. Quantum Technol.*, vol. 2, nos. 5–6, 2019, Art. no. 1900015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/qute.201900015
[27] H. Thapliyal, N. Ranganathan, and R. Ferreira, "Design of a comparator tree based on reversible logic," in *Proc. 10th IEEE Int. Conf. Nanotechnol.*, 2010, pp. 1113–1116.
[28] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," 2004. [Online]. Available: https://arxiv.org/abs/quant-ph/0410184
[29] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, vol. 86, Sep. 2012, Art. no. 032324.
[30] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information.* Cambridge, U.K.: Cambridge Univ. Press, 2016.
[31] V. Kliuchnikov, D. Maslov, and M. Mosca, "Practical approximation of single-qubit Unitaries by single-qubit quantum Clifford and T circuits," *IEEE Trans. Comput.*, vol. 65, no. 1, pp. 161–172, Jan. 2016.
[32] S. Perriello, A. Barenghi, and G. Pelosi, "A complete quantum circuit to solve the information set decoding problem," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Broomfield, CO, USA, H. A. Müller, G. Byrd, C. Culhane, and T. S. Humble, Eds. Piscataway, NJ, USA: IEEE Press, 2021, pp. 366–377, doi: 10.1109/QCE52317.2021.00056.
[33] S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, "Implementing Grover Oracles for quantum key search on AES and LowMC," in *Proc. 39th Annu. Int. Conf. Theory Appl. Cryptographic Techn. Adv. Cryptology (EUROCRYPT),* Zagreb, Croatia. New York, NY, USA: Springer-Verlag, 2020, vol. 12106, pp. 280–310.
[34] S. Perriello, A. Barenghi, and G. Pelosi, "Improving the efficiency of quantum circuits for information set decoding," *ACM Trans. Quantum Comput.*, vol. 4, no. 4, pp. 1–40, 2023, doi: 10.1145/3607256.
[35] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, Santa Fe, NM, USA. Washington, DC, USA: IEEE Computer Society, 1994, pp. 124–134.
[36] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018. doi: 10.22331/q-2018-08-06-79.
[37] S. Perriello, A. Barenghi, and G. Pelosi, "A quantum circuit to speed-up the cryptanalysis of code-based cryptosystems," in *Proc. 17th EAI Int. Conf., SecureComm Secur. Privacy Commun. Netw.*, J. García-Alfaro, S. Li, R. Poovendran, H. Debar, and M. Yung, Eds. 2021, vol. 399. New York, NY, USA: Springer-Verlag, pp. 458–474, doi: 10.1007/978-3-030-90022-9_25.
[38] S. Perriello, A. Barenghi, and G. Pelosi, "Quantum circuit design for the Lee-Brickell based information set decoding," in *Proc. Appl. Cryptography Netw. Secur. Workshops (ACNS) Satellite Workshops*, Abu Dhabi, United Arab Emirates, M. Andreoni, Ed., New York, NY, USA: Springer-Verlag, 2024, vol. 14587, pp. 8–28, doi: 10.1007/978-3-031-61489-7_2.

Giacomo Lancellotti (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science and engineering from Politecnico di Milano, Italy. He is currently working toward the Ph.D. degree in information technology with Politecnico di Milano. His research interests include quantum algorithms for post-quantum cryptography applications and combinatorial optimization. He is also investigating quantum compilation and design automation techniques for quantum circuit synthesis.

Simone Perriello is a Postdoctoral Researcher with the Politecnico di Milano, Italy. His research interests include quantum cryptanalysis of both symmetric and asymmetric cryptosystems, as well as the development of quantum algorithms.

Alessandro Barenghi is an Associate Professor with Politecnico di Milano, Italy. His interests include computer and network security, formal languages and compilers and he has published more than 100 papers in international peer reviewed venues.

Gerardo Pelosi (Member, IEEE) is an Associate Professor with Politecnico di Milano, Italy. His research interests include computer security, cryptography, security in hardware and in the area of data security and privacy. He has published more than 100 papers in international peer reviewed journals and conference proceedings and is co-inventor of 10 patents concerning the design of cryptographic systems.