Design of a Quantum Walk Circuit to Solve the Subset-sum Problem

Giacomo Lancellotti[*]
giacomo.lancellotti@polimi.it
Politecnico di Milano–DEIB
Milano, Italy

Simone Perriello[*]
simone.perriello@polimi.it
Politecnico di Milano–DEIB
Milano, Italy

Alessandro Barenghi
alessandro.barenghi@polimi.it
Politecnico di Milano–DEIB
Milano, Italy

Gerardo Pelosi
gerardo.pelosi@polimi.it
Politecnico di Milano–DEIB
Milano, Italy

ABSTRACT

Search algorithms based on quantum walks have emerged as a promising approach to solve computational problems across various domains, including combinatorial optimization, and cryptography. Stating a generic search problem in terms of a (quantum) search over a graph makes the efficiency of the algorithmic method depend on the structure of the graph itself. In this work, we propose a complete implementation of a quantum walk search on Johnson graphs, speeding up the solution of the subset-sum problem. We provide a detailed design of each sub-circuit, quantifying their cost in terms of gate number, depth, and width. We compare our solution against a Grover quantum search, showing a reduction of the T-count and T-depth for practically solvable problems. The proposed design provides a building block for the construction of efficient quantum search algorithms that can be modelled on Johnson graphs, filling the gap with the existing theoretical complexity analyses.

CCS CONCEPTS

• **Theory of computation** → **Design and analysis of algorithms**; • **Mathematics of computing** → *Graph algorithms*; • **Computer systems organization** → **Quantum computing**.

KEYWORDS

Quantum walks, Subset-sum, Johnson graph, Grover algorithm

ACM Reference Format:
Giacomo Lancellotti, Simone Perriello, Alessandro Barenghi, and Gerardo Pelosi. 2024. Design of a Quantum Walk Circuit to Solve the Subset-sum Problem. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3657337

1 INTRODUCTION

Random walks play a crucial role in the analysis of probabilistic approaches to search problems. Their importance was first recognized in [1], in which the authors addressed the shortest path problem by modeling the positions inside a maze as vertices of a graph, and then generating random sequences of directions. More recently, the quantum counterpart of random walks, referred to as *quantum walks*, were proven to provide a quadratic speedup in search algorithms [20, 21] compared to the classical paradigm of computation, reducing the computational complexity with respect to a Grover-based quantum search algorithm [9]. Inspired by [2], which proposes a quantum-walk search strategy on a Johnson graph to achieve an exponential-time algorithm surpassing Grover's approach for solving the *element distinctness* problem, researchers have adapted the same approach to address NP-complete problems like *information set decoding* [13] and *claw finding* [23].

In this study, we tackle the *subset-sum* problem, which involves finding a subset $\mathcal{I}\subseteq\mathcal{X}$ of a given set $\mathcal{X}$ of n integers such that the sum of its elements equals a given value $p>0$. The constrained version additionally requires that $|\mathcal{I}|=k$. To solve the unconstrained version, we can run its constrained version for all $k\in\{1,\ldots,\lceil n/2\rceil\}$. Both formulations belong to the class of NP-hard problems [7].

In [4], the authors explore the theoretical advantages of using a quantum walk search algorithm to solve the subset-sum problem. Adapting a classical meet-in-the-middle approach to the quantum setting, their strategy significantly reduces the number of required operations compared to classical algorithms. The asymptotic runtime was further improved in [10, 5] by exploiting a different adaptation of the classical strategy. All the works, however, rely on an exponential amount of memory with quantum random-access (QRAM), whose practical realization is still challenging [12].Notably, [12] shows the only partial design of a quantum walk-based search on the Johnson graph, focusing on the claw-finding problem. Nevertheless, their proposal modifies the graph structure by adding a self-loop to every vertex, introducing sub-optimal performance compared to theoretical expectations.

Contribution. We give a concrete design and implementation of a quantum walk search algorithm on the Johnson graph to solve the subset-sum problem. Our analysis assumes a fault-tolerant regime of quantum computation, avoiding any discussion related to noise correction and hardware architectures. We derive closed-form complexity metrics in terms of number of quantum gates, number of qubits (a.k.a., width) and depth of the quantum circuit. In contrast to the state-of-the-art theoretical approaches, our proposal does not depend on an exponentially-sized QRAM, but only on a polynomial amount of qubits. We additionally retrieve the complexity metrics of our circuits in terms of the Clifford+T gate set, considered the most promising one for fault-tolerant quantum computation [6].

We compare our implementation with a Grover-based search approach, showing improvements in terms of both the depth and depth×width measure, considering a problem size range from practically solvable subset sum instances, to problem sizes which are considered large enough to build post-quantum cryptosystems upon (and therefore assumed not solvable even with a quantum computer).

[*]Both authors contributed equally to this work

DAC '24, June 23–27, 2024, San Francisco, CA, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0601-1/24/06.
https://doi.org/10.1145/3649329.3657337

2 BACKGROUND

In this section, we introduce the notation and give an overview on Markov chains and the MNRS framework [15].

2.1 Quantum computing

In quantum computing, the *qubit* represents the basic unit of information, and its state is expressed as a linear combination of two column vectors, $|0\rangle$ and $|1\rangle$. Denoting a generic qubit as $|\psi\rangle$, its state can be written as $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$, where $a_0, a_1 \in \mathbb{C}$ are known as *amplitudes* of the basis states $|0\rangle$ and $|1\rangle$. A group of n qubits is expressed in terms of column vectors belonging $\mathbb{C}^{2^n}$, described by 2^n basis states labeled as length-n bitstrings, as in $|\psi\rangle = \sum_{i\in\{0,1\}^n} a_i|i\rangle$. We denote a subset of $m\leq n$ qubits as *quantum register*, and we use an underline to describe them, as in $\underline{a}$. To reference individual qubits within the register, we employ the standard 0-indexing notation commonly used for classical arrays (e.g., $\underline{a}[0]$ is the first qubit of $\underline{a}$). The system evolves through the application of quantum operators, described by a unitary matrix U, to quantum registers. In the *gate-based model* of quantum computation, each qubit is visualized as a wire of a circuit, and quantum operators are visualized as *quantum gates* acting on the wires. The application of a gate G to a quantum register modifies the *amplitudes* of all the basis states depending on the values of the *labels* of the qubits it acts on. In the rest of the work, we will employ the following quantum gates:

$$\mathrm{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \mathrm{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \mathrm{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{R_Y}(\theta) = \begin{bmatrix} \cos\theta/2 & -\sin\theta/2 \\ \sin\theta/2 & \cos\theta/2 \end{bmatrix}$$

The X gate, commonly known as NOT, switches the amplitude of the basis state $|0\rangle$ and $|1\rangle$, resembling the classical NOT gate. Given a generic gate G acting on m qubits, the gate $\mathrm{C}^c\mathrm{G}$ operates on m+c qubits, in which the additional c qubits serve as *controls*. The gate G is applied to the m qubits only if all the c control qubits are in state 1. If c=1 the superscript is omitted; moreover, CX and $\mathrm{C}^2\mathrm{X}$ are commonly known as CNOT and CCNOT respectively. In the circuit representation, $\mathrm{C}^c\mathrm{G}$ gate is depicted by a black circle on the control qubits, connected through a vertical line to the gate G.
Reflection operators. We define as $U_{\mathrm{r}(\alpha)}$ the operator performing a reflection of the quantum state through the hyperplane generated by $|\alpha^\perp\rangle$, where $|\alpha^\perp\rangle$ is any state orthogonal to $|\alpha\rangle$. Its action is such that $U_{\mathrm{r}(\alpha)}|\alpha\rangle = |\alpha\rangle$, while $U_{\mathrm{r}(\alpha)}|\alpha^\perp\rangle = -|\alpha\rangle$. We define $U_{\mathrm{r}(1^\perp)}$ as the reflection through the basis state labelled as the all-ones bitstring. Such reflection is realized in terms of a (multi-)controlled Z gate applied on all the qubits it acts upon. Dually, we denote as $U_{\mathrm{r}(0^\perp)}$ the reflection through the basis state labelled as the all-zeros bitstring. Its realization requires a (multi-)controlled Z gate, taking care of applying a NOT gate on all the involved qubits before and after its application. For a generic quantum state $|\alpha\rangle$, however, implementing $U_{\mathrm{r}(\alpha)}$ in terms of quantum gates may be difficult, and it is hence usually realized through the *Quantum Phase Estimation (QPE)* algorithm [16]. Indeed, when $|\alpha\rangle$ is the unique eigenvector of a unitary U with eigenvalue 1 (and eigenphase λ_0=0) — that is, $U|\alpha\rangle = e^{i\lambda_0}|\alpha\rangle = |\alpha\rangle$— the reflection $U_{\mathrm{r}(\alpha)}$ is approximated applying the QPE algorithm to U, storing the eigenphase on a set of auxiliary qubits, and then applying $U_{\mathrm{r}(0^\perp)}$ on them. The QPE, in its realization, requires the repeated application of the controlled version of U.

2.2 Probabilistic search through random walks over Johnson graphs

A random walk on a graph $G=(\mathcal{V},\mathcal{E})$ is a stochastic process modelled as a sequence of probabilistic transitions between a vertex $v_i\in\mathcal{V}$ and any of its adjacent vertexes $v_j\in\mathcal{V}$, with $(v_i, v_j)\in\mathcal{E}$ being an edge of $\mathcal{G}$. If the probability of taking a step to an adjacent vertex depends only on the current vertex, the random walk is equivalent to a *Markov chain* [17]. Its evolution is described in terms of a vector of probabilities s_i of size $|\mathcal{V}|$, denoting the occurrence of each vertex, and a stochastic matrix P of size $|\mathcal{V}|\times|\mathcal{V}|$, where the element in row i and column j represents the probability of transitioning from v_i to v_j. We consider Markov chains having a unique stationary distribution s_π. In such chains, starting from any initial distribution of the vertices, we reach s_π in a fixed number of steps, and any further step does not alter the probability distribution. Such number of steps, known as *mixing time*, is proportional to $1/\delta$, in which δ denotes the *spectral gap* of P — the difference between its two largest eigenvalues.
Search problem. A search problem can be described as the problem of finding one or more elements belonging to the domain $\mathcal{X}$ for which the Boolean function $f: \mathcal{X}\mapsto\{0,1\}$ is equal to 1. The set $\mathcal{M}=\{x \mid x\in\mathcal{X}, f(x)=1\}$ is termed the *marked set*, and we define the ratio of marked elements as $\varepsilon=|\mathcal{M}|/|\mathcal{X}|$. A naive probabilistic approach to the problem repeatedly samples an element $x\in\mathcal{X}$ according to the uniform distribution, stopping when $f(x)$=1. For each sample, the probability that $x\notin\mathcal{M}$ 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., $1/\varepsilon$. Denoting as T_s the cost to initialize the uniform distribution and taking a sample from it, and T_c the cost to check if an element belongs to $\mathcal{M}$, the expected cost of this approach is $\mathrm{T}_s + (\mathrm{T}_s+\mathrm{T}_c)/\varepsilon$. When the search space has a known structure, we can model it as the vertices of the transition graph $G=(\mathcal{V},\mathcal{E})$ associated to a Markov chain. This alternative probabilistic algorithm samples an element $x\in\mathcal{X}$ according to the stationary distribution s_π. After checking if $x\in\mathcal{M}$, it performs $\approx 1/\delta$ steps on the chain to return to the stationary distribution, restarting with a new, independent sample. Denoting T_u as the cost of taking a step on the Markov chain, this approach has an expected computational complexity of $\mathrm{T}_s + (\mathrm{T}_u/\delta+\mathrm{T}_c)/\varepsilon$, offering an advantage with respect to repeatedly sampling from the uniform distribution in all the cases in which $\mathrm{T}_u/\delta < \mathrm{T}_s$.
Johnson graphs. The Johnson graph $J(n,k)$ is defined as the undirected graph $G=(\mathcal{V},\mathcal{E})$, in which each vertex represents a size-k subset $\mathcal{S}$ of $\{1,\ldots,n\}$, and there exists an edge between two vertices $\mathcal{S}$ and $\mathcal{T}$ if $|\mathcal{S}\cap\mathcal{T}| = k-1$. From the previous definition, it follows that $|\mathcal{V}| = \binom{n}{k}$. Additionally, Johnson graphs have a fixed spectral gap $\delta = n/k(n-k)$, making them ideal candidates for quantum walks.

2.3 Quantum walk search

The most advanced theoretical framework adapting the random walk search over a Johnson graph to the quantum computing paradigm is outlined in [15]. Termed *MNRS* from the initials of the authors' surnames, it searches for a subset of marked vertices $\mathcal{M}=\{\mathcal{S}\in\mathcal{V} \mid f(\mathcal{S})=1\}$ given the graph $G=(\mathcal{V},\mathcal{E})$ and a Boolean function f. The state of the system throughout the algorithm is described in terms of the quantum state $|\mathcal{S}\rangle|\mathcal{T}\rangle$, encoding the edge

$(\mathcal{S},\mathcal{T})\in\mathcal{E}$. We refer to the two registers as *left* and *right* respectively. After preparing the initial state $|\pi\rangle = 1/\sqrt{|\mathcal{E}|}\sum_{(\mathcal{S},\mathcal{T})\in\mathcal{E}}|\mathcal{S}\rangle|\mathcal{T}\rangle$ —that is, the uniform superposition of all the edges of the graph— the MNRS framework employs an amplitude amplification scheme [11] to increase the amplitude of all the states where $\mathcal{S}\in\mathcal{M}$. This scheme, which also serves as the basis for Grover's algorithm [9], relies on two reflection operators: *1)* $U_{\mathrm{r}(M^{\perp})}$, reflecting around the hyperplane generated by $|\mathcal{S}\rangle|\mathcal{T}\rangle$, for $\mathcal{S}\notin\mathcal{M}$; *2)* $U_{\mathrm{r}(\pi)}$, reflecting around the hyperplane generated by $|\pi\rangle$. Each reflection is repeated $1/\sqrt{\varepsilon}$ times to achieve a probability ≈ 1 of measuring a vertex $\in\mathcal{M}$. The authors then summarise the amplification procedure using only three operators, reminding of the classical steps of the random walk search: *a)* U_s, roughly corresponding to the *setup* circuit (cost T_s); *b)* U_u, denoting the update circuit (cost T_u); *c)* U_c, denoting the check circuit (cost T_c). In the following, we provide an overview on the MNRS framework relying on the previous three operators.

Generate $|\pi\rangle$. The initialization routine, used to create the state $|\pi\rangle$, relies on U_s and U_u. The operator U_s prepares a uniform superposition of all the vertices $\mathcal{S}\in\mathcal{V}$; that is, $|0\rangle|0\rangle \xrightarrow{U_s} 1/\sqrt{|\mathcal{V}|}\sum_{\mathcal{S}\in\mathcal{V}}|\mathcal{S}\rangle|0\rangle$. Given a generic state $|\mathcal{S}\rangle$, with $\mathcal{S}\in\mathcal{V}$, the operator U_u generates instead the uniform superposition of all its adjacent vertices; that is, $|\mathcal{S}\rangle|0\rangle \xrightarrow{U_u} |\mathcal{S}\rangle\sum_{(\mathcal{S},\mathcal{T})\in\mathcal{E}}|\mathcal{T}\rangle$. By applying the two operators one after the other, we obtain the uniform superposition over all the edges; that is, $|0\rangle|0\rangle \xrightarrow{U_sU_u} |\pi\rangle$.

Reflection $U_{\mathrm{r}(M^{\perp})}$. This operator is decomposed in terms of two auxiliary operators. The first one, U_c, computes f and sets an auxiliary qubit to $|1\rangle$ if $f(\mathcal{S})=1$. The second one is the reflection operator $U_{\mathrm{r}(1^{\perp})}$, which is applied on such auxiliary qubit. Finally, the application of $U_c^{\dagger}$ concludes the realization of $U_{\mathrm{r}(M^{\perp})}$.

Reflection $U_{\mathrm{r}(\pi)}$. The realization of $U_{\mathrm{r}(\pi)}$ in the MNRS framework is the main difference with respect to Grover's one. While the latter refers to this operator as *diffusion* and, to realize it, applies the sequence of operators $U_sU_{\mathrm{r}(0^{\perp})}U_s^{\dagger}$, the MNRS framework avoids U_s in this stage, since it can require a significant amount of resources to implement. To achieve the same effect of Grover's diffusion operator, the MNRS framework defines a new operator U_w, called walk operator, corresponding to a quantum realization of a single step taken on the Markov chain according to the stochastic matrix P. U_w is realized in terms of two distinct applications of the sequence of operators $U_uU_{\mathrm{r}(0^{\perp})}U_u^{\dagger}$, the first time having $U_{\mathrm{r}(0^{\perp})}$ applied on the right vertex, and the second time applied on the left vertex. The operator U_w is such that the quantum state $|\pi\rangle$ is the unique eigenvector having eigenphase 0. For this reason, to perform the reflection through $U_{\mathrm{r}(\pi)}$, the framework applies a QPE procedure as described in Sec. 2.1, applying the controlled version of U_w a number of times proportional to $1/\sqrt{\delta}$. The quadratic speedup on the mixing time $1/\sqrt{\delta}$ reflects the relation between the phase-gap of U_w and the spectral gap of the stochastic matrix P [15].

Overall. The total cost of the MNRS search is equal to $(\mathrm{T}_s+\mathrm{T}_u) + (\mathrm{T}_u/\sqrt{\delta}+\mathrm{T}_c)/\sqrt{\varepsilon}$, in which the first term corresponds to the initialization cost (i.e., the complexity of U_s and U_u), while the second one corresponds to the iterative application of the update and check routines (i.e., U_u and U_c). Since, for all relevant problems, the quantity $1/\sqrt{\varepsilon}$ is large, the quantum walk results in a quadratic speedup compared to the classical probabilistic approach.

Table 1: Main quantum registers (QRegs) used in our quantum circuit implementation of the MNRS quantum walk.

QRegs	Semantics of the content
$\underline{\mathcal{S}}\,\underline{\mathcal{T}}$	Edge of the Johnson graph $J(n,k)$, with $\mathcal{S},\mathcal{T}\subset\{1,\ldots,n\}$, $\lvert\mathcal{S}\rvert=\lvert\mathcal{T}\rvert=k$, $\lvert\mathcal{S}\cap\mathcal{T}\rvert=k-1$
$\underline{\mathcal{S}'}\,\underline{\mathcal{T}'}$	Complement, i.e. $\mathcal{S}'=\{1,\ldots,n\}\setminus\mathcal{S}$ and $\mathcal{T}'=\{1,\ldots,n\}\setminus\mathcal{T}$
$\underline{m}$	Stores the sum of k integers
$\underline{\sigma}$	Stores the Dicke state $\lvert D_n^k\rangle$
$\underline{\omega},\underline{\omega'}$	Stores the Dicke states $\lvert D_k^1\rangle$ and $\lvert D_{n-k}^1\rangle$ respectively

3 IMPLEMENTATION

In this section, we present our realization of the MNRS framework components for the Johnson graph $J(n,k)$ to solve the subset-sum problem. Tab. 1 gives an overview of the main registers used. Henceforth, we use the shorthand m to represent $\log_2(n)$.

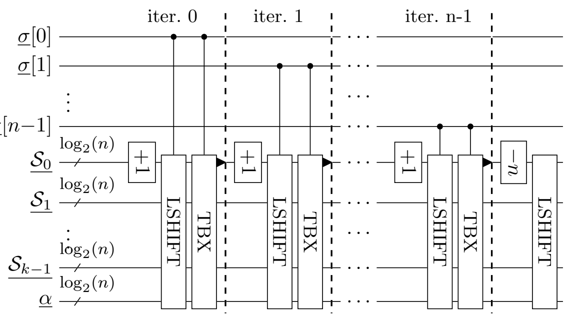
3.1 Setup U_s

Our design for this operator combines the Dicke state generation circuit, a novel circuit named *Bitstring Index eXtractor* (BIX), and a circuit computing the sum of k integer values. The total number of resources used to implement this operator is reported in Tab.2.

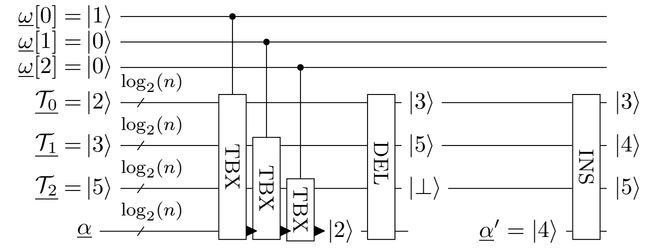
A. Dicke state generation. Presented in [3], this circuit generates on the register $\underline{\sigma}$ the Dicke state $|D_k^n\rangle=\sqrt{\binom{n}{k}}\sum_d|d\rangle$, with d being a length-n bitstring having Hamming weight k.

B. Binary Index eXtractor (BIX). We interpret the indices of the qubits of $\underline{\sigma}$ with values 1 (resp., 0) as the integers of a vertex $\mathcal{S}$ (resp., $\mathcal{S}'$) to be encoded in $\underline{\mathcal{S}}$ (resp., $\underline{\mathcal{S}'}$). To obtain them, we describe the BIX quantum circuit, marking the first-ever solution to this problem in our knowledge. Such a circuit stores the elements of $\mathcal{S}$ and $\mathcal{S}'$ in increasing order in fixed location, avoiding therefore the dependence on any random-access memory location.

Fig. 1a shows the abstract gates required for the indices to be stored in $\underline{\mathcal{S}}$. In the picture, and for the rest of this section, we denote as $\underline{\mathcal{S}_i}$ the element at position i of the quantum register $\underline{\mathcal{S}}$, encoding the ordered set $\mathcal{S}$. The quantum circuit applies the same sequence of operations over n iterations on two main elements of $\underline{\mathcal{S}}$, namely $\underline{\mathcal{S}_0}$, serving as an accumulator, and $\underline{\alpha}$, an auxiliary register encoding the all-zeros bitstring. At the beginning of iteration i, $\underline{\alpha}$ contains the last index found up to that point; that is, $\underline{\alpha}=j$ such that $\underline{\sigma}[j]=1$ for $j<i$. $\underline{\mathcal{S}_0}$, on the other hand, accumulates the number of iterations before the next index is found, starting from the value of $\underline{\alpha}$. To keep the two elements updated, the algorithm performs, at each iteration, three operations. First, the +1 gate increase the integer value stored in $\underline{\mathcal{S}_0}$ by 1. Then, if the value of $\underline{\sigma}[i]=1$, it first performs a cyclic left shift of the elements of $\underline{\mathcal{S}}$ (gate LSHIFT), and then XOR the value of $\underline{\alpha}$ to $\underline{\mathcal{S}_0}$, storing the result in $\underline{\mathcal{S}_0}$ (gate TBX, shorthand for Topmost to Bottommost XOR). Note that the cyclic shift always moves into $\underline{\mathcal{S}_0}$ a group of qubits in state 0, and therefore the following XOR operation stores the value of $\underline{\alpha}$ into $\underline{\mathcal{S}_0}$. At the end of the last iteration, given the unconditioned +1 gate applied at each iteration, the front register $\underline{\mathcal{S}_0}$ contains the value n. For this reason, the last step of the algorithm cleans this difference by performing a fixed, constant subtraction described by the gate −n. The final LSHIFT moves the clean qubits to $\underline{\alpha}$, additionally ensuring



(a)



$\underline{\omega}[0] = |1\rangle$
$\underline{\omega}[1] = |0\rangle$
$\underline{\omega}[2] = |0\rangle$
$\underline{\mathcal{T}_0} = |2\rangle$
$\underline{\mathcal{T}_1} = |3\rangle$
$\underline{\mathcal{T}_2} = |5\rangle$
$\underline{\alpha}$
$\log_2(n)$
TBX
$|2\rangle$
DEL
$|3\rangle$
$|5\rangle$
$|\perp\rangle$
$\underline{\alpha}' = |4\rangle$
INS
$|3\rangle$
$|4\rangle$
$|5\rangle$

(b)

Figure 1: (a) The Bitstring Index eXtractor (BIX) encodes in $\underline{\mathcal{S}}$ the indices of the k qubits $\underline{\sigma}$ in state 1. LSHIFT performs a cyclic left shift on $\underline{\mathcal{S}}$; TBX XORes the topmost $\log_2(n)$ qubits to the bottommost $\log_2(n)$ qubits it acts upon, putting the result on the qubits marked by the black triangle. (b) Example showing the action of U_u on $\underline{\mathcal{T}}$ for a fixed state $\underline{\omega}$=$|100\rangle$. Since the unique element of $\underline{\omega}$ having state 1 is located at the first index, DEL deletes the sample $\underline{\alpha}$=2 from $\underline{\mathcal{T}}$. The sample $\underline{\alpha}'$=4, on the other hand, is assumed to be generated by the dual subcircuit sampling from $\underline{\mathcal{T}}'$ based on $\underline{\omega}'$. The INS gate inserts its value into the $\underline{\mathcal{T}}$ register, generating therefore a valid adjacent vertex of the initial state.

the correct ordering of the values of $\underline{\mathcal{S}}$. The sequence of operations described is analogous for $\underline{\mathcal{S}}'$, taking care of negating each qubit of $\underline{\sigma}$ before applying the conditional gates of the iteration.

The LSHIFT gate requires $(k-1)m$ CSWAP gates when acting on $\underline{\mathcal{S}}$. Since the control qubit is shared among all CSWAP, all the gates are applied sequentially. A TBX gate can be implemented using m CNOT gates, each having as control qubit the one stored in $\underline{\alpha}$ and as target the corresponding one in $\underline{\mathcal{S}_i}$. The controlled version of this gate requires therefore m CCNOT gates, all sharing the same additional control qubit of $\underline{\sigma}$. The gates +1 is implemented using the quantum adder design of [22], which employs $5m-5$ CNOT gates, and $2m-1$ CCNOT gates and has a depth equal to $5m-3$. Additionally, to store the constant value of 1, we use an auxiliary quantum register of size m, not shown in the picture, initialized to the binary value 1 using one X gate. The subtraction circuit −n is implemented using the same adder circuit, taking care of initializing the auxiliary qubits to the binary complement of n. Since $n>0$, its complement requires at most $m-1$ X gates to be encoded. Given the sequential structure enforced by the shared control qubit inside the iteration itself, and since all the qubits involved in the TBX circuit are shared across distinct iterations, the circuit is not prone to parallelization. Nonetheless, it is trivial to parallelize the execution between the gates involving $\underline{\mathcal{S}}$ and the ones involving $\underline{\mathcal{S}}'$.

C. Subset sum. The last step of the setup circuit stores the sum of the k integers of $\underline{\mathcal{S}}$ in the register $\underline{m}$ using a series of k distinct adders, each taking as first input one element of $\underline{\mathcal{S}}$ at a time and the $\underline{m}$ circuit as the second addend. Each adder is implemented using the proposal presented in [22].

3.2 Check U_c

The operator U_c checks if the value encoded in $\underline{m}$ is equal to p. To implement it, we perform, on the qubits of $\underline{m}$, a bitwise XOR with the binary complement of p using X gates. If the equality holds, all the qubits $\underline{m}$ will be in state 1, and they can be used as controls in a C^{m}X gate having as target an auxiliary qubit.

3.3 Update U_u

We detail in this section all the subcircuits composing U_u, reporting all their costs in Tab.2.

D. XOR $\underline{\mathcal{S}}$ into $\underline{\mathcal{T}}$. This subcircuit applies the TBX gate to the quantum registers $\underline{\mathcal{S}}$ and $\underline{\mathcal{T}}$, using the latter as a target. The operation requires km CNOT gates, each one having as control a qubit of $\underline{\mathcal{S}}$ and as target a qubit of $\underline{\mathcal{T}}$. Since $\underline{\mathcal{T}}$, at this stage, encodes the all-zeros bitstring, the operation stores the values encoded in $\underline{\mathcal{S}}$ on $\underline{\mathcal{T}}$. An identical operation is applied on $\underline{\mathcal{S}}'$ and $\underline{\mathcal{T}}'$.

E. Sample from $\underline{\mathcal{T}}$ and $\underline{\mathcal{T}}'$. This subcircuit selects an element from $\underline{\mathcal{T}}$ and another one from $\underline{\mathcal{T}}'$ uniformly at random. To sample one element out of the k (resp., $n-k$) elements of $\underline{\mathcal{T}}$ (resp., $\underline{\mathcal{T}}'$), we reuse the Dicke generation circuit shown in Sec. 3.1 to generate the state $|D^1_k\rangle$ (resp., $|D^1_{n-k}\rangle$) on the auxiliary register $\underline{\omega}$ (resp., $\underline{\omega}'$). The unique qubit $\underline{\omega}[i]$ (resp., $\underline{\omega}'[i]$) equal to 1 is then used to apply a TBX gate between the element $\underline{\mathcal{T}_i}$ (resp., $\underline{\mathcal{T}_i}'$) and the auxiliary register $\underline{\alpha}$ (resp., $\underline{\alpha}'$), using the qubits of the register $\underline{\alpha}$ (resp., $\underline{\alpha}'$), initially labelled as the all-zeros bitstring, as targets.

F. Generate adjacent vertex; Sum. The overall idea of this stage is to delete, from $\underline{\mathcal{T}}$ (resp., $\underline{\mathcal{T}}'$) the element having the same value of the one stored in $\underline{\alpha}$ (resp., $\underline{\alpha}'$). Then, we insert into $\underline{\mathcal{T}}$ (resp., $\underline{\mathcal{T}}'$), the element having the same value of the one stored in $\underline{\alpha}'$ (resp., $\underline{\alpha}$). The effect of this sequence is the generation, in superposition, of all the neighbours of each vertex stored in $\underline{\mathcal{S}}$.

The main challenge of this step is to keep the values of $\underline{\mathcal{T}}$ (resp., $\underline{\mathcal{T}}'$) ordered, since the quantum state representing the set $\underline{\mathcal{T}}$ (resp., $\underline{\mathcal{T}}'$) must have a unique representation. Fig. 1b shows an example of such a strategy, adapted from the one shown in [12]. The INS starts with a sorted array $\underline{\mathcal{T}}$, having one free spot in its last position, and an element $\underline{\alpha}'$ to be inserted; it relies on two auxiliary arrays $\underline{\mathcal{U}}$ and $\underline{\mathcal{V}}$ having the same size of $\underline{\mathcal{T}}$ and initialized to 0. The first stage of the insertion XORs the value of $\underline{\alpha}'$ into all the spots of $\underline{\mathcal{U}}$, and then stores the element-wise comparison between $\underline{\mathcal{T}}$ and $\underline{\mathcal{U}}$ on $\underline{\mathcal{V}}$. A series of conditional swaps controlled by $\underline{\mathcal{V}}$ is then performed between the elements of $\underline{\mathcal{T}}$ and $\underline{\mathcal{U}}$ to move $\underline{\alpha}'$ in the right position. A final sequence of operations is used to restore the

Table 2: Cost metrics for U_s and U_u, detailed in each of their components. The metrics are expressed as a function of the Johnson graph $J(n,k)$ parameters; m is a shorthand notation for $\log_2(n)$.

Cost	Setup U_s			Update U_u		
metrics	A. Dicke generation	B. BIX	C. Subset sum	D. $\underline{\mathcal{S}} \oplus \underline{\mathcal{T}}$	E. Sample	F. Generate adjacent vertex; Sum
X	k	$n+m$	0	0	2	0
$\mathrm{R_y}$	$4nk-4k^2-2n+1$	0	0	0	$2n-2$	0
CNOT	$5nk-5k^2-2n$	$10nm-10n+10m-10$	$5km-5k$	nm	$2n-2$	$28nm-20n+10m-10$
CCNOT	0	$6nm-2n+4m-2$	$2km-k$	0	nm	$8nm-4n+4m-4$
CSWAP	0	n^2m-2nm	0	0	0	$4nm$
Depth	$\frac{27nk-12n-27k^2+3}{k-2}$	$n^2m+5nm-3n-nkm+km+10m-6$	$5km-3k$	1	$nm-km+4n-4k-4$	$30m+4\log_2(n-k)+4\log_2(m)-14$
Qubits	n	$nm+m$	$\log_2((2nk-k^2+k)/2)$	nm	$n+2m$	$2nm$

auxiliary arrays to their original state. The DEL applies the same operations in reverse order. To keep the subset sum updated, the INS and DEL procedures are followed by the subtraction of the integer value stored in $\underline{\alpha}$ from the one in register $\underline{m}$, followed by the addition of the one stored in $\underline{\alpha}'$. These steps require adder gates, which we realize using [22].

4 PERFORMANCE

All the components of our design have been extensively tested using the open-source Qiskit simulator. To assess the soundness of the quantum-walk search strategy, we compare its computational complexity to a Grover-based search strategy. Although both strategies rely on an amplitude amplification scheme, they substantially differ in their implementations of the reflection operator $U_{\mathrm{r}(\pi)}$. Grover's implementation of $U_{\mathrm{r}(\pi)}$ is expressed in terms of the sequence of operators $U_{s'} U_{\mathrm{r}(0^\perp)} U_{s'}^\dagger$, in which the $U_{s'}$ describes a modified version of the U_s used in our implementation. Indeed, since Grover's approach does not need all the auxiliary data structure to perform a walk on the Johnson graph, the $U_{s'}$ has a lower cost compared to U_s. Specifically, the operator generates a superposition of all the size-k sets of integers in the range $\{1,\dots,n\}$ through 1) the Dicke state generation circuit; 2) a reduced version of the BIX circuit acting only on register $\underline{\mathcal{S}}$, and not on $\underline{\mathcal{S}}'$. The generated integers are then added together using a sequence of k adders, storing the result on the register $\underline{m}$.

To compare the performance of the two quantum algorithms, we translated the abstract quantum gates used in the previous section into the Clifford+T, considered to be the most promising one for fault-tolerant quantum computation [6]. Although the translation between different gate sets results in at most a polynomial overhead [16], the T requires extensively more resources to be implemented in a fault-tolerant way [8] with respect to the Clifford gates. Therefore, the literature adopts the T-count and the T-depth as the relevant computational complexity metrics. Using the Clifford+T gate sets, the only admissible two-qubit gate is the CNOT one, and all the other abstract gates used in Sec. 3 require a decomposition. The results shown in [18, Tab. III] summarize the key translations required for our proposal. The CSWAP and CCNOT both requires 7 T gates and a T-depth of 3. The $\mathrm{R_y}$, on the other hand, has a T-depth and T-count of 149. Finally, adopting a strategy similar to [19], the translation of the multi-controlled Z gates required to realize the reflection operators $U_{\mathrm{r}(0^\perp)}$ inside U_u and $U_{\mathrm{r}(\pi)}$ is obtained by first translating them in terms of CCNOT and Z gates, and then translating the CCNOT gates using the same strategy used before. When using a controlled gate having c control qubits, this strategy requires $c-2$ auxiliary qubits, $2c-4$ CCNOT plus one CNOT, and has a depth equal to $\log_2(c+1)+1$. The values of the T-count and T-depth for the two different realizations of the $U_{\mathrm{r}(\pi)}$ used in the Grover approach and a quantum-walk approach based on our proposal is shown in Tab.3.

We compare the efficiency of the proposed quantum walk strategy with a Grover search approach in Figure 2, where the ratio between the two approaches in terms of T-count, T-depth and T-depth×width, omitting the width ratio as it stays constant. We chose a range of subset sum problem starting from values of n and k low enough to be tackled classically, and reaching values which are large enough to be deemed unpractical even with a quantum computer, and therefore useful when designing post-quantum cryptosystems [14].

Our quantum walk approach outperforms Grover search for all practical parameters ($n \leq 2^{11}$), showing a T-count reduction of $\approx 33\%$ for all ratios k/n, and a similar reduction in T-depth for the hardest instances (t=n). When considering parameter sizes large enough to be employed in a cryptosystem ($n \geq 2^{12}$), our approach yields a (up to 5×) shorter quantum circuit in terms of T-depth than Grover's. Finally, considering the T-depth×width (Fig. 2, right), which captures the effective use of a quantum computer (since applying quantum gates to some qubits requires the remaining ones to be idle), our approach improves on cryptosystem-building grade parameters, pointing to the concrete re-evaluation of such parameters as an interesting research direction for cryptanalysts.

5 CONCLUDING REMARKS

We presented a complete design of a quantum-walk based search algorithm over a Johnson graph to accelerate the solution of the subset-sum problem. Our proposal shows a novel technique to generate a uniform superposition of all the integers in the set of all the size-k subsets of a size-n set of integers, which may be of independent interest. We assessed the performances of our proposal, considering the T-count, T-depth and T-depth×width metrics, and showed that the advantage of quantum walks persist up to a very large parameter regime, namely the one of interest for cryptanalytic purposes. As a potential future direction, it would be beneficial to assess the computational complexity of our circuit design when

Table 3: Comparison between the $U_{r(\pi)}$ realization in Grover's approach with respect to our quantum-walk approach, in terms of T-count and T-depth. m is a shorthand for $\log_2(n)$.

	MNRS $U_{r(\pi)}$	Grover $U_{r(\pi)}$
T count	$\frac{1}{\sqrt{\delta}}(364nm + 1080n + 28nk + 112m - 1360) + O(\log_2(1/\sqrt{\delta}))$	$28nm + 582n + 14nkm + 1192nk - 1192k^2 + 28km - 14k + 28m + 277 + 14\log_2(2nk - k^2 + k)$
T depth	$\frac{1}{\sqrt{\delta}}(12nm + 1216n - 12km - 1192k + 96m + 6\log_2(km+1) - 1204) + O(\log_2(1/\sqrt{\delta}))$	$(1788n + 3576nk - 3576k^2 + 894)/(k-2) + 12nm - 4n + 6nkm + 121km - 6k + 12m - 6 + O(\log_2(m))$

graphs different from the Johnson's graphs are used to model the search space.

ACKNOWLEDGMENTS

The activity was partially funded by the Italian *Centro Nazionale di Ricerca in HPC, Big Data e Quantum computing - SPOKE 10* and the Italian MUR (PRIN 2022 project POst quantum Identification and eNcryption primiTives: dEsign and Realization (POINTER) ID-2022M2JLF2).

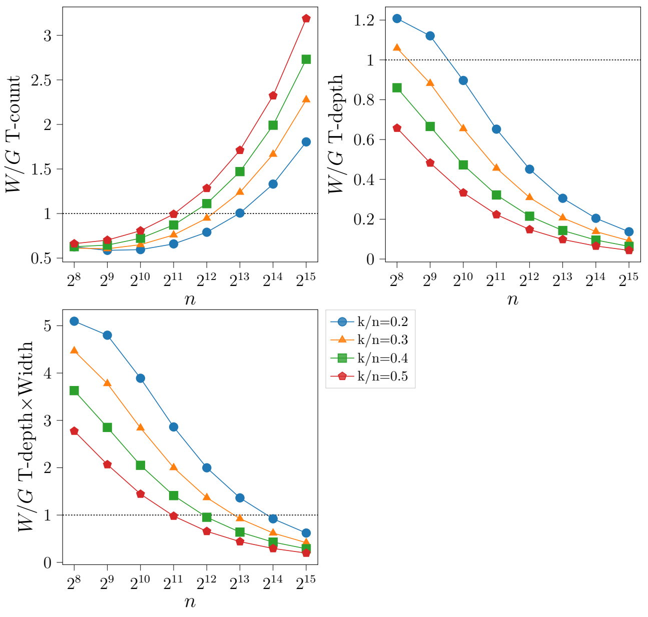


Figure 2: T-count, T-depth and T-depth×width ratios between our quantum-walk approach (W) and Grover's approach (G), as a function of the parameters n and k/n of the constrained subset-sum problem.

REFERENCES

[1] Romas Aleliunas, Richard M. Karp, Richard J. Lipton, László Lovász, and Charles Rackoff. 1979. Random walks, universal traversal sequences, and the complexity of maze problems. In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*. IEEE Computer Society.

[2] Andris Ambainis. 2004. Quantum walk algorithm for element distinctness. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*. IEEE Computer Society.

[3] Andreas Bärtschi and Stephan J. Eidenbenz. 2019. Deterministic preparation of dicke states. In *Fundamentals of Computation Theory - 22nd International Symposium, FCT 2019, Copenhagen, Denmark, August 12-14, 2019, Proceedings* (LNCS). Leszek Antoni Gasieniec, Jesper Jansson, and Christos Levcopoulos, (Eds.) Vol. 11651. Springer.

[4] Daniel J. Bernstein, Stacey Jeffery, Tanja Lange, and Alexander Meurer. 2013. Quantum algorithms for the subset-sum problem. In *Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings* (LNCS). Philippe Gaborit, (Ed.) Vol. 7932. Springer.

[5] Xavier Bonnetain, Rémi Bricout, André Schrottenloher, and Yixin Shen. 2020. Improved classical and quantum algorithms for subset-sum. In *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II* (LNCS). Shiho Moriai and Huaxiong Wang, (Eds.) Vol. 12492. Springer.

[6] P. Oscar Boykin, Tal Mor, Matthew Pulver, Vwani P. Roychowdhury, and Farrokh Vatan. 2000. A new universal and fault-tolerant quantum basis. *Inf. Process. Lett.*, 75.

[7] M. R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman.

[8] Vlad Gheorghiu, Michele Mosca, and Priyanka Mukhopadhyay. 2022. T-count and t-depth of any multi-qubit unitary. *npj Quantum Information*, 8.

[9] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual {ACM} Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996.* Gary L. Miller, (Ed.) ACM.

[10] Alexander Helm and Alexander May. 2018. Subset sum quantumly in 1.17^n. In *13th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2018, July 16-18, 2018, Sydney, Australia* (LIPIcs). Stacey Jeffery, (Ed.) Vol. 111. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[11] Peter Høyer, Michele Mosca, and Ronald de Wolf. 2003. Quantum search on bounded-error inputs. In *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings* (LNCS). Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, (Eds.) Vol. 2719. Springer.

[12] Samuel Jaques and John M. Schanck. 2019. Quantum cryptanalysis in the RAM model: claw-finding attacks on SIKE. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I* (LNCS). Alexandra Boldyreva and Daniele Micciancio, (Eds.) Vol. 11692. Springer.

[13] Ghazal Kachigar and Jean-Pierre Tillich. 2017. Quantum information set decoding algorithms. In *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings* (LNCS). Tanja Lange and Tsuyoshi Takagi, (Eds.) Vol. 10346. Springer.

[14] Vadim Lyubashevsky, Adriana Palacio, and Gil Segev. 2010. Public-key cryptographic primitives provably as secure as subset sum. In *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings* (LNCS). Daniele Micciancio, (Ed.) Vol. 5978. Springer.

[15] Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. 2011. Search via quantum walk. *SIAM J. Comput.*, 40.

[16] Michael A Nielsen and Isaac L Chuang. 2010. *Quantum Computation and Quantum Information.* Cambridge University Press.

[17] J. R. Norris. 1997. *Markov Chains. Cambridge Series in Statistical and Probabilistic Mathematics.* Cambridge University Press.

[18] Simone Perriello, Alessandro Barenghi, and Gerardo Pelosi. 2021. A complete quantum circuit to solve the information set decoding problem. In *IEEE International Conference on Quantum Computing and Engineering, QCE 2021, Broomfield, CO, USA, October 17-22, 2021.* Hausi A. Müller, Greg Byrd, Candace Culhane, and Travis Humble, (Eds.) IEEE.

[19] Simone Perriello, Alessandro Barenghi, and Gerardo Pelosi. 2023. Improving the efficiency of quantum circuits for information set decoding. *ACM Transactions on Quantum Computing.*

[20] Neil Shenvi, Julia Kempe, and K. Birgitta Whaley. 2003. Quantum random-walk search algorithm. *Phys. Rev. A*, 67, 11 pages, 5.

[21] Mario Szegedy. 2004. Quantum speed-up of markov chain based algorithms. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings.* IEEE Computer Society.

[22] Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. 2010. Quantum addition circuits and unbounded fan-out. *Quantum Inf. & Comp.*, 10.

[23] Seiichiro Tani. 2009. Claw finding algorithms using quantum walk. *Theoretical Compututer Science*, 410.