Families of Constant-Depth Quantum Circuits for Rotations and Permutations

Simone Faro[1,†], Arianna Pavone[2,‡] and Caterina Viola[1,*,†]

[1]Università di Catania, viale A. Doria n.6, 95125, Catania, Italy
[3]Università di Palermo, via Archirafi n.34, 90123, Palermo, Italy

Abstract

Developing and expanding on the idea by Moore and Nilsson [1], we provide a detailed description of families of logspace uniform quantum circuits that implement cyclic shifts and permutations of qubits. This allows us to formally prove that such operations belong to QNC^0, the quantum analogue of the complexity class NC^0, which captures highly efficiently parallelizable classical computations.

Keywords

Quantum circuits, Cyclic shiftings, Permutations, Quantum complexity

1. Introduction

A quantum algorithm is designed to take full advantage of the computational power of quantum machines. In a quantum machine, the fundamental information units called *quantum bits* or *qubits*, are simulated by particles whose size is at and below the scale of atoms. The state of each of such particles is described by a wave function, which can be thought of as a probability distribution of the classical states in which the particle can be observed. A quantum computing machine works as predicted by the theory of quantum mechanics and therefore, in particular, a qubit can be in a *superposition* of the two basis states, and two or more qubits can be *entangled*, i.e., the probability of their states can be correlated. These peculiar properties of quantum bits allow for an increased amount of information encoded. However, when observing the qubits from the macroscopic world, this great amount of information loses its coherence and the qubits collapse in one of their classical probable states.

A *cyclic shifting operator* (or *rotation operator*) ROT_k applies a rightward (or leftward) shift of k positions to a register of n qubits, moving the element at position x to position $(x + k) \mod n$.

Cyclic shifts have various applications, including image and signal processing, where they can be employed to perform operations such as image rolling [2] or introducing a time delay in a signal. Cyclic shifts can also be used to design efficient data sorting algorithms [3]. Furthermore, sequences allowing cyclic shifts are relevant in various biological contexts, including viruses [4, 5] and bacteria [6]. Thus, the analysis of organisms with a cyclic structure can benefit from algorithms designed for strings that allow for cyclic shifts [7].

ICTCS'24: Italian Conference on Theoretical Computer Science, September 11–13, 2024, Torino, Italy
*Corresponding author.
†Supported by National Centre for HPC, Big Data and Quantum Computing, Project CN00000013, affiliated to Spoke 10, co-founded by the European Union - NextGenerationEU.
‡Supported by PNRR project ITSERR - Italian Strengthening of the ESFRI RI RESILIENCE
faro@dmi.unict.it (S. Faro); ariannamaria.pavone@unipa.it (A. Pavone); caterina.viola@unict.it (C. Viola)
0000-0001-5937-5796 (S. Faro); 0000-0002-8840-6157 (A. Pavone); 0000-0001-7042-3912 (C. Viola)

Algorithms for permutations serve as a filtering technique, commonly known as the counting filter, to expedite complex combinatorial search problems. The core concept is that in many approximation scenarios, a substring of the text that matches a given pattern, according to a specific distance function, is also a permutation of that pattern. The counting filter technique has been applied to solve approximate string-matching problems involving mismatches [8], differences [9], inversions [10], and translocations [11].

In a classical model of computation, permuting a multiple system of bits, particularly cyclically shifting it, exhibits linear time complexity because in the worst-case scenario, every element in the array must be shifted. This specialization of cyclic rotation becomes evident when dealing with bit strings on classical machines. While many concrete computers feature a built-in *shift* instruction that moves bits left or right, achieving a cyclic shift involves more effort than a simple one-sided shift, as it requires handling the wrap-around of the array. Employing a Boolean circuit model of computation, every permutation can be obtained very easily by just permuting the wires of a circuit, which can be done at no cost. It is therefore evident that permuting a multiple system of bits is not a problem susceptible to a computational speed-up due to the adoption of a quantum circuit model rather than a Boolean circuit model.

However, in quantum computing, permutations, especially cyclic shifts, of multiple quantum bits are essential subroutines within quantum circuits. They enable the creation of superpositions involving multiple permutations of quantum states, offering computational advantages in tasks such as database searching and optimization problems [12, 13, 14]. Therefore, finding optimal quantum-circuit implementing them is crucial to harness the synergies between circuit parallelism and quantum superposition parallelism. This is particularly relevant given that - as we already discussed - permutations are very relevant to problems arising in computational biology, which in turn are among the most likely problems to benefit from quantum superpowers[1] (cf. [16, 17, 15]).

While it is known that a quantum rotation operator can be implemented in $\mathcal{O}(\log(n))$-time through repeated parallel applications of elementary Swap operators [12], a systematic procedure for concretely constructing the quantum operator ROT for varying register sizes n and shift parameters k has been lacking. Recently, in [18] Pavone and Viola addressed this gap, providing a systematic implementation of the cyclic rotation operator in a quantum circuit model with depth $\mathcal{O}(\log(n))$ and size $\mathcal{O}(n\log(n))$. Furthermore, in [1] Moore and Nilsson pointed out that any permutation can be implemented by a constant-depth quantum circuit, since, in the dihedral group any rotation can be generated by the product of two reflections. However, the article does not contain details about such circuits and their construction.

In this paper, we present a detailed description of a family of quantum circuits implementing the cyclic shifting of the states of an arbitrary finite number of quantum bits; we generalize such a quantum circuit family to one implementing the permutation of the states of an arbitrary finite number of quantum bits, for any permutation defined on them. We provide the pseudocodes of the respective deterministic algorithms computing these quantum circuit families, and we show that, in fact, such families are logspace uniform. As a consequence, we formally prove that cyclic shiftings and, more in general, permutations are in the complexity class QNC^0, as already noted by Moore and Nilsson [1]. These authors observed that any permutation is in the class QNC, sketching the underlying idea; however, a formal proof of this observation never appeared in the literature, to the best of our knowledge.

[1] Citing [15], "quantum computers will offer large gains over current, classical computers in simulating quantum physics and chemistry" and "quantum computing will be in speeding up or gaining better control over molecular reactions".

2. Quantum Preliminaries

Basic Notations. We denote by $\mathbb{N}$ and $\mathbb{Z}$, the sets of natural and integer numbers, respectively. For each $n \in \mathbb{N}$, and $i, j \in \mathbb{Z}$ we write $i = j \mod n$ iff $j = i + nq$ for some $q \in \mathbb{Z}$; furthermore, $\mathbb{Z}_n$ denotes the quotient set $\{i \in \mathbb{Z} \mid i = j \Leftrightarrow i = j \mod n\}$. We also recall that the floor operator $\lfloor \cdot \rfloor$ computes the maximum integer smaller or equal to its argument, while the ceiling operator $\lceil \cdot \rceil$ computes the minimum integer larger or equal to its argument.

Quantum Computational Systems. The fundamental unit in quantum computation is the *quantum bit*, also known as *qubit*. Formally, a single qubit is an element of the two-dimensional Hilbert space over the complex numbers, equipped with an inner product, $\mathcal{H}$, which is referred to as the *state space*. Therefore, the mathematical expression of a qubit, $|\psi\rangle$, is a linear combination of the two basis states, i.e., $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, where α and β, called *amplitudes*, are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. The squared magnitudes of α and β represent the probabilities of measuring the qubit in the state $|0\rangle$ or $|1\rangle$, respectively. A *quantum measurement* is the only operation through which we can gain some knowledge on the state of a qubit; more precisely, this operation causes the qubit to collapse to one of the two basis states, following the probability distribution described by the squares of its amplitudes before the measure. Multiple systems of qubits are referred to as *quantum registers*. A quantum register $|\psi\rangle = |q_0, q_1, \ldots, q_{n-1}\rangle$ of size n is an element from the tensor product of n state spaces, $\mathcal{H}^{\otimes n}$, and is thus expressed as a linear combination of the 2^n states in $\{0, 1\}^n$. Specifically, $|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle$, where the values α_k represent the probability amplitudes of the corresponding classical states $|k\rangle$ and satisfy $\sum_{k=0}^{2^n-1} |\alpha_k|^2 = 1$.

Quantum Operations. The allowed operations on (registers of) qubits are those permitted by quantum mechanics. Quantum measurements are operations that cause the collapse of a quantum system into a classical state and result in the loss of the probability amplitude wave. Quantum measurements are needed not only to have access to the elaboration of the information stored in qubits but also for their state preparation [19, Section 1.10].

The evolution of a quantum mechanical system is described by a *unitary transformation*. A unitary operator is a bounded linear operator U on a Hilbert space that satisfies $U^\dagger U = UU^\dagger = I$, where $U^\dagger$ is the conjugate transpose of U, and I is the identity operator. In other words, a unitary transformation is an automorphism U of the (tensor) state space that preserves the inner product, $\langle Ux, Uy\rangle = \langle x, y\rangle$, thus ensuring the conservation of the probability amplitudes. Formally, each unitary transformation on an n-qubit register can be described by a $2^n \times 2^n$ unitary matrix with complex entries. In particular, unitary transformations are reversible, meaning that no information is lost when performing them, and the composition of multiple unitary transformations is also a unitary transformation. Any unitary transformation on an n-qubit register can be implemented by a sequence of 2-qubit unitary transformations.

Quantum Circuits. A quantum algorithm on an n-dimensional input register consists of quantum unitary transformations and measurements.

Quantum algorithms can be formalized in various models, each with distinct advantages and challenges: the *adiabatic model* [20], the *topological model* [21], and the *measurement-based model* [22].

This paper adopts the *quantum circuits* model. Computational circuits are represented as directed acyclic graphs, where nodes denote gates operating on information carried by edges. Unlike classical

circuits, quantum circuits must be reversible (excluding measurement gates), ensuring the number of input edges matches the number of output edges. Consequently, quantum circuits cannot join (fan-in) multiple wires or copy/split (fan-out) information due to the No-Cloning Theorem [23]. Thus, the graph representation features parallel wires (qubits) passing through gates. To address the No-Cloning Theorem's limitations, *ancillæ* qubits are often included in quantum circuits.

Real quantum machines have a finite set of native gates, typically at-most-ternary gates. Therefore, constructing circuits with a minimal number of native gates is crucial not only for speed and efficiency but also to reduce *coherent quantum error* caused by gate miscalibration and to maximize qubit lifespan. Qubits have a limited *coherence time*, after which they lose information.

Circuit-Based Quantum Computational Classes. There are two major measures of computational complexity for circuit models. One is the total number of basis gates, referred to as the *size* of the circuit. The other is the *depth* of the directed acyclic graph that represents the circuit. Clearly, basis gates have a depth of $\Theta(1)$.

A complexity class can be thought of as a collection of computational problems that share common features regarding the computational resources needed to solve them. Perhaps the most well-known complexity classes in classical computation models are P and NP. These classes are defined in terms of the elementary operations that the best-performing Turing Machine needs to execute to solve the target problem. Boolean circuits, on the other hand, are non-uniform models of computation, meaning that inputs of different lengths are processed by different circuits, in contrast to uniform models such as Turing Machines. A similar distinction arises between Quantum Turing Machines [24] and Quantum Circuits.

Complexity classes defined in terms of Boolean circuits include NC, AC, and P/poly. The class NC captures computations that can be efficiently performed on highly parallel computers. A computational task has an efficient parallel algorithm if instances of size n can be solved using a parallel computer with $n^{\mathcal{O}(n)}$ processors in time $\log^{\mathcal{O}(1)}(n)$ (see [25] for details).

In the case of a family of quantum circuits, we can give a more concrete idea of what logspace uniform means. We do this in the style of [26, Definition 6.5] (see also [24]). To this end, we need to fix the representation of the quantum circuits.

Definition 2.1. A circuit family $\{C_n\}$ is a collection of circuits such that for each $n \in \mathbb{N}$ the circuit C_n has n inputs. A computational problem is solved by a circuit family $\{C_n\}$ if, for every $n \in \mathbb{N}$, the circuit C_n solves the instances of the problem of size n.

Definition 2.2. A circuit family $\{C_n\}$ is *logspace uniform* if there exists a logarithmic space deterministic Turing Machine mapping 1^n to the description of C_n, for all $n \in \mathbb{N}$.

Remark 2.3. Let N be the size of C_n. Since C_n is an acyclic graph, we can assume to represent it by its $N \times N$ adjacency matrix and by an array of size N that encodes the labels of each gate (i.e. a label that identifies each vertex of the acyclic graph as an input qubit or one of the basis gate to be applied). Assuming such a representation, a family of quantum circuits $\{C_n\}$ is logspace uniform if, and only if, the following functions can be computed in $\mathcal{O}(\log(n))$ space: (1) the function $\textsc{size}\,(n)$, returning the size of the circuit C_n; (2) the function $\textsc{label}\,(n, i)$, returning the label of the ith gate of C_n (corresponding to the ith entry of the labels array); and (3) the function $\textsc{edge}\,(n, i, j)$ returning 1 whenever there is a direct wire (edge) from the ith gate to the jth one, and returning 0 otherwise.

Definition 2.4 ([1])**.** A computational problem is in QNC^i if there exists a constant $c > 0$ such that it can be solved by a logspace uniform family of quantum circuits (whose basis gate are at-most-binary) $\{C_n\}$, where C_n has size $\mathcal{O}(n^c)$ and depth $\mathcal{O}(\log^i(n))$. We set $\mathrm{QNC}= \bigcup_{i\geq 0}\mathrm{QNC}^i$.

We conclude this section by highlighting the relationship between space-bounded Turing Machines and depth-bounded Boolean circuits in classical computation. Space-bounded computations can be efficiently simulated by bounded-depth circuits, and vice versa, through depth-first exploration (see [27, 28] for details). However, such a relationship is not established for quantum computations. While space-bounded computations on Quantum Turing Machines can be efficiently simulated by bounded-depth quantum circuits, the reverse is not known and is considered highly unlikely. Beyond theoretical considerations (see [24]), practical quantum computations are constrained by short coherence times, making highly parallelized quantum circuits more feasible than Quantum Turing Machines.

3. Quantum Circuits implementing Rotations and Permutations

In this section, we show how to construct a logspace uniform family of constant-depth and linear-size quantum circuits for rotations and, more generally, for permutations. Before going into the details of our solution, some knowledge about rotations and permutations is necessary.

3.1. Rotations and Permutations of Qubits

Formally, a *rightward rotation* (or rightward cyclic shift) is a quantum operator ROT_k^+ that applies a rightward shift of k positions to a register of size n so that the element at position x is moved to position $(x+k) \mod n$. In other words, the elements whose position exceeds the size n of the register are moved, in a circular fashion, to the first positions of the register. Formally, the operator ROT_k^+ applies the following permutation

$$|q_0, q_1, \ldots, q_{n-1}\rangle \longmapsto |q_{n-k}, q_{n-k+1}, \ldots, q_{n-1}, q_0, q_1, \ldots, q_{n-k-1}\rangle .$$

We call the parameter k the *magnitude* of the rotation.

A *leftward rotation* (or leftward cyclic shift) ROT_k^- applies a leftward shift of k positions to a register of size n so that the element at position x is moved to position $(x-k) \mod n$. Formally, the operator ROT_k^- applies the following permutation

$$|q_0, q_1, \ldots, q_{n-1}\rangle \longmapsto |q_k, q_{k+1}, \ldots, q_{n-1}, q_0, q_1, \ldots, q_{k-1}\rangle .$$

Example 3.1. Let $|\psi\rangle = |q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\rangle$ be a quantum register of 8 qubits. Right rotating $|\psi\rangle$ of 5 positions results in the quantum register $\text{ROT}_5^+ |\psi\rangle = |q_3, q_4, q_5, q_6, q_7, q_0, q_1, q_2\rangle$, while left rotating $|\psi\rangle$ of 5 positions results in the quantum register $\text{ROT}_5^- |\psi\rangle = |q_5, q_6, q_7, q_0, q_1, q_2, q_3, q_4\rangle$

Unless we explicitly declare that this is not the case, in this paper we use the term *rotation* and the symbol ROT_k by referring to the *rightward* version of the quantum operator. However, it is immediate to verify that

$$\text{ROT}_k^- |q\rangle = \text{ROT}_{n-k}^+ |q\rangle ,$$

for every register of size n and every $0 \leq k < n$. Thus, all the results stated for the rightward rotations hold true for the leftward rotations, too, and the needed modifications in the algorithmic techniques are trivial.

Remark 3.2. Let us consider the n-qubit register $|q\rangle = \bigoplus_{i=0}^{n-1} |q_i\rangle = \sum_{j=0}^{2^n-1} k_j |j\rangle$. Applying a rotation of magnitude k, we obtain $= \text{ROT}_k |q\rangle = \bigoplus_{i=n-k}^{n-k-1} |q_i\rangle = \sum_{h=0}^{2^n-1} \ell_h |h\rangle$, where $\ell_h = k_j$ for each $h, j \in \{0, \ldots, 2^n - 1\}$ such that the qubit representation of h is a rotation of the qubit representation of j by k positions.

For every $n \in \mathbb{N}$, a *permutation* π of n elements is a bijection from a set S of cardinality $|S| = n$ into itself. The set of all permutations of a set with n elements is denoted by S_n and forms a group (the symmetric group) with the composition operation. The composition of two permutations σ and τ is the permutation $\pi = \sigma\tau$ such that $\pi(i) = \sigma(\tau(i))$. In general, the composition of permutations is not commutative.

There are several ways to represent a permutation, the one that we use in this paper - and perhaps the most common - is the *cycle representation*, which we describe in the following definition.

Definition 3.3 (Cyclic permutation)**.** For a set S of cardinality $|S| = n$ and for $m \leq n$, a *cyclic permutation* of length m (or an m*-cycle*) $\sigma \in S_n$ is a permutation for which there exists a sequence $(n_0, n_1, \ldots, n_{m-1})$ of pairwise distinct elements of S such that

- $\sigma(n_0) = n_1, \sigma(n_1) = n_2, \ldots, \sigma(n_{m-1}) = n_0$, i.e. it cyclically permutes all the elements from the sequence;
- $\sigma(s) = s$, for all $s \in S \setminus \{n_0, n_1, \ldots, n_{m-1}\}$, i.e. it fixes the elements that are not in the sequence.

The sequence $(n_0, n_1, \ldots, n_{m-1}) = \big(n_0, \sigma(n_0), \sigma^2(n_0), \ldots, \sigma^{m-1}(n_0)\big)$ is the cycle representation of the m-cycle σ, and is unique up to cyclic shifting. The identity is the trivial cyclic permutation which fixes all the elements of S. The composition of two disjoint cyclic permutations is commutative.

Example 3.4. Let $S = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}, \mathsf{g}, \mathsf{h}\}$ be a set of elements of cardinality $|S| = 8$. The permutation $\sigma = \{\mathsf{a}, \mathsf{g}, \mathsf{b}, \mathsf{c}, \mathsf{e}, \mathsf{f}, \mathsf{d}, \mathsf{h}\}$ is a cyclic permutation of length 4, since the sequence $(\mathsf{c}, \mathsf{b}, \mathsf{g}, \mathsf{d})$ is a sequence of pairwise distinct elements of S such that $\sigma(\mathsf{c}) = \mathsf{b}$, $\sigma(\mathsf{b}) = \mathsf{g}$, $\sigma(\mathsf{g}) = \mathsf{d}$ and $\sigma(\mathsf{d}) = \mathsf{c}$. In addition $\sigma(\mathsf{a}) = \mathsf{a}$, $\sigma(\mathsf{e}) = \mathsf{e}$, $\sigma(\mathsf{f}) = \mathsf{f}$ and $\sigma(\mathsf{h}) = \mathsf{h}$.

We now present a foundational result in permutation theory, often referred to as a folklore theorem

Theorem 3.5 (Folklore)**.** *Every permutation of finitely many elements can be written as the composition of two or more disjoint cyclic permutations. Such a decomposition is unique up to different orders of the cycles.*

Remark 3.6. Given a different representation of a permutation σ, it is possible to find its decomposition into cycles in polynomial time [29, 30].

Let $\pi \in S_n$ be a permutation of n elements and let $\pi = \sigma_1 \cdots \sigma_\ell$ be the decomposition of σ into disjoint cycles. then the *cycle representation* of π is the array $(t_1, \ldots, t_\ell)$, where t_i is the sequence representing σ_i for every $i \in \{1, \ldots, \ell\}$.

Example 3.7. Let $S = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}, \mathsf{g}, \mathsf{h}\}$ be a set of elements of cardinality $|S| = 8$. The permutation $\sigma = \{\mathsf{d}, \mathsf{f}, \mathsf{h}, \mathsf{a}, \mathsf{g}, \mathsf{e}, \mathsf{b}, \mathsf{c}\}$ can be represented by the following decomposition in disjoint cycles $((\mathsf{a}, \mathsf{d}), (\mathsf{b}, \mathsf{f}, \mathsf{e}, \mathsf{g}), (\mathsf{c}, \mathsf{h}))$, which is the cycle representation of σ.

Example 3.8. Let $S = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}, \mathsf{g}, \mathsf{h}\}$ be a set of elements of cardinality $|S| = 8$. The permutation $\sigma = \{\mathsf{h}, \mathsf{c}, \mathsf{a}, \mathsf{b}, \mathsf{e}, \mathsf{f}, \mathsf{d}, \mathsf{g}\}$ can be represented by the following decomposition in disjoint cycles $((\mathsf{a}, \mathsf{h}, \mathsf{g}, \mathsf{d}, \mathsf{b}, \mathsf{c}), (\mathsf{e}), (\mathsf{f}))$, which is the cycle representation of σ.

Remark 3.9. Any rotation, and more generally, any permutation of the states of the qubits within a register, can be realized through the application of a unitary transformation.[2]. This implies that the reordering of qubit states, whether it be a simple rotation or a more complex permutation, can be effectuated by an appropriate unitary matrix, ensuring the transformation adheres to the principles of quantum mechanics.

[2]We recall that a permutation can be formalized through a unitary matrix with entries in $\{0, 1\}$

3.2. Quantum Circuits implementing Rotations

In classical circuital models of computation, the wires carrying the information on the state of a bit can be permuted as desired without any computational cost; the same does not hold true in quantum circuits. In [1], Moore and Nilsson observed that any permutation of the states of a finite set of qubits can be achieved with a constant-depth quantum circuit.

In the remainder of the current section, we will develop their idea and give a constructive proof of this fact, providing both the detailed description of such a circuit and the pseudocode of a logarithmic-space and polynomial-time algorithm constructing such a circuit for any cardinality n of the set of qubits and any permutation of n elements. As a consequence of this, we will observe that permuting the states of the constituent qubits from a quantum register is in the computational class QNC^0.[3]

We start by focusing on the implementation of the rotation of the states of a quantum register by a given magnitude.

Let us consider a n-qubit register $|q\rangle = \bigoplus_{i=0}^{n-1} |q_i\rangle$ to be rotated by k positions. Let us arrange the constituent qubits as the vertices of a regular polygon with n sides from an Euclidean two-dimensional space. Enumerate such vertices by the indices of the corresponding qubits, i.e., $0, 1, \ldots, n-1$ or, equivalently, $i \in \mathbb{Z}_n$. In such a formulation, a rotation of $0, 1, \ldots, n-1$ by k positions (to the right) corresponds to a rotation of the polygon about its centre of symmetry, C.

It is well-known that every rotation of a regular polygon about its centre of symmetry by an angle θ can be obtained as the composition of two reflections across two axes such that they intersect in the centre of the polygon and form an angle equal to $\frac{\theta}{2}$. We remark that even if the observation above is behind our construction, it is not necessary to know it to see the correctness of our algorithm, which will be proved directly.

For every $i \in \mathbb{Z}_n$, we have that the angle formed by points i, C, and $i+1$ equals $\frac{2\pi}{n}$ radians; therefore, a rotation of the qubits by k positions corresponds to a rotation of the polygon by $k\frac{2\pi}{n}$ radians about C. We have therefore to perform two reflections across two axes ℓ_1 and ℓ_2 such that they pass through C and form an angle of $k\frac{\pi}{n}$ radians. We choose ℓ_1 and ℓ_2 based on n and k. Let us first assume that n is even. Then ℓ_1 is chosen as the line passing through 0 and C, which contains the vertex $\frac{n}{2}$. A reflection across ℓ_1 has the effect of swapping the pairs $(i, n-i)$ for $i = 1, \ldots, \frac{n}{2} - 1$, while the vertices 0 and $\frac{n}{2}$ stay fixed. Let us call q_i^0 the initial state of the ith qubit q_i, and let q_i^1 and q_i^2 the states of the ith qubit after the first and after the second reflection, respectively.

Then, we have that

$$\begin{aligned} &q_i^0 = q_{n-i}^1 && \text{for } i = 1, \ldots, \frac{n}{2} - 1, \frac{n}{2} + 1, \ldots, n-1, \\ &q_0^0 = q_0^1, && \text{and} \\ &q_{\frac{n}{2}}^0 = q_{\frac{n}{2}}^1. \end{aligned}$$

The definition of ℓ_2 depends on whether k is even or odd. If k is even, then ℓ_2 is the line through $\frac{k}{2}$ and C, which contains $\frac{n}{2} + \frac{k}{2}$; else, if k is odd, ℓ_2 is the line through the centre C and the midpoint of $\frac{k-1}{2}$ and $\frac{k+1}{2}$, which happens to contain the midpoint of $\frac{n}{2} + \frac{k+1}{2}$. The reflection across ℓ_2 has the effect of swapping the pairs $(\frac{k}{2} - j, \frac{k}{2} + j)$ for $j = 1, \ldots, \frac{n}{2} - 1$ whenever k is even, and has the effect of swapping the pairs $(\frac{k+1}{2} - j, \frac{k-1}{2} + j)$ for $j = 1, \ldots, \frac{n}{2} - 1$ whenever k is odd. In either case, we have that after this second reflection

$$q_i^1 = q_{k-i}^2 \text{ for } i = 0, \ldots, n-1$$

[3] The authors of [1] pointed out that permuting the states of finitely many qubits is in QNC.

ALGORITHM 1: Algorithm constructing a quantum circuit for the rotation of a register of size n by k positions.

Input: n, k

1 Initialize n input wires (qubits) $|q_0\rangle, \ldots, |q_{n-1}\rangle$;
2 **for** $i = 1$ *to* $\lceil \frac{n}{2} \rceil - 1$ **do**
3 Swap $|q_i, q_{n-i}\rangle$
4 **for** $j = 1$ *to* $\lceil \frac{n}{2} \rceil - 1$ **do**
5 Swap $\left|q_{\lceil \frac{k}{2} \rceil - j}, q_{\lfloor \frac{k}{2} \rfloor + j}\right\rangle$

(observe that for even k, $q^1_{\frac{k}{2}}$ and $q^1_{\frac{n}{2}+\frac{k}{2}}$ stay fixed).

Let us now assume that n is odd. Then ℓ_1 is defined as the line passing through 0 and C, which contains the midpoint between $\frac{n-1}{2}$ and $\frac{n+1}{2}$. A reflection across ℓ_1 has the effect of swapping the pairs $(i, n-i)$ for $i = 1, \ldots, n-1$. The result of this first rotation is

$$q^0_i = q^1_{n-i} \qquad \text{for } i = 1, \ldots, \frac{n-1}{2}, \frac{n+1}{2}, \ldots, n-1, \text{ and}$$
$$q^0_0 = q^1_0.$$

To define ℓ_2 for n odd, we have, once more, to take into account the parity of k. If k is even, then ℓ_2 is the line through $\frac{k}{2}$ and C, which contains the midpoint between $\frac{n-1}{2} + \frac{k}{2}$ and $\frac{n+1}{2} + \frac{k}{2}$; else, if k is odd, ℓ_2 is the line through the centre C and the midpoint of $\frac{k-1}{2}$ and $\frac{k+1}{2}$, which contains $\frac{n}{2} + \frac{k}{2}$. The reflection across ℓ_2 has the effect of swapping the pairs $(\frac{k}{2} - j, \frac{k}{2} + j)$ for $j = 1, \ldots, \frac{n-1}{2}$ whenever k is even, and has the effect of swapping the pairs $(\frac{k+1}{2} - j, \frac{k-1}{2} + j)$ for $j = 1, \ldots, \frac{n-1}{2}$ whenever k is odd. In either case, we have that after this second reflection

$$q^1_i = q^2_{k-i} \text{ for } i = 0, \ldots, n-1$$

(observe that for even k, the state of $q_{\frac{k}{2}}$ stays fixed, while, for odd k, the state of $q_{\frac{k}{2}+\frac{n}{2}}$ stays fixed).

Algorithm 1, which runs in linear time, takes as input a pair (n, k) of nonnegative integers and constructs a constant-depth and linear-size quantum circuit that performs the rotation of any n-qubit register by k positions.

It is immediate to see that Algorithm 1 has time-complexity $\mathcal{O}(n)$, that is, it has linear-time complexity.

On input (n, k), Algorithm 1 constructs a circuit with n wires (qubits). In lines 2-3, are defined $\lceil \frac{n}{2} \rceil$ swaps that involve disjoint pairs of qubits and can, therefore, be applied in parallel, we will refer to the swaps defined in lines 2-3 as the *first layer of swaps of the circuit*; similarly, we will refer to the swaps defined in lines 2-3 as the *second layer of swaps of the circuit*. Figures 1 and 2 illustrate a few examples of circuits built by Algorithm 1. Observe that the first layer depends only on the first input parameter n.

Remark 3.10. Algorithm 1 works for $n \geq 3$. Clearly, if the register has only 2 qubits then any rotation either does nothing or simply swaps the states of the two qubits, in which case it is enough to employ an elementary swap gate. In the remainder, we assume to apply Algorithm 1 to registers with at least 3 qubits.

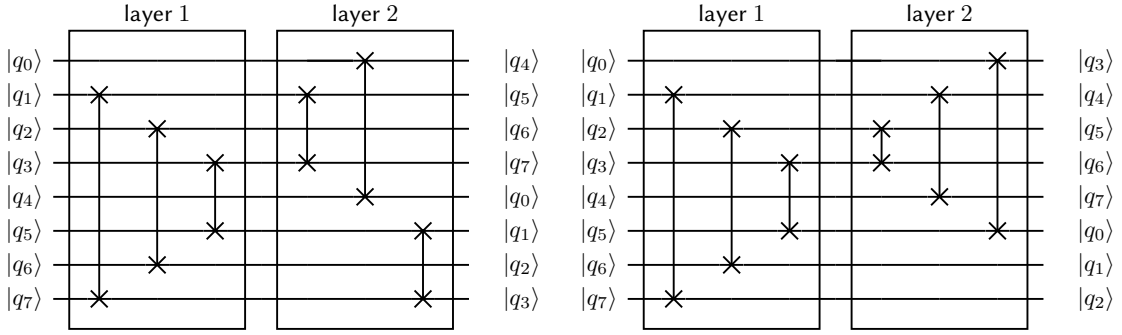

Figure 1: The circuits taking in input a register of 8 qubits and operating a rotation of 4 and 5 positions, respectively.

The following Lemma 3.11 represents a significant result regarding the depth and size characteristics of the circuit constructed by Algorithm 1.

Lemma 3.11. *The circuit constructed by Algorithm 1 has constant depth (more precisely, the depth equals* 6*) and linear size. If we run the circuit on a quantum machine so that the CNOT gate is a native one, we get that the depth equals* 6 *and the size is bounded by* $3(n-1)$.

Proof. The swap gates built in lines 2-3 involve disjoint pairs of qubits, therefore they can be applied in parallel, and form the first layer of parallel swap gates of the circuit. Similarly, the swap gates built in lines 4-5 form the second layer of parallel swap gates. Each layer of the circuit has at most $\frac{n-1}{2}$ gates, therefore we have size bounded by $n-1$, that is $\mathcal{O}(n)$. Observe that a swap can be implemented by 3 CNOTs (see Section 2). Therefore, if the CNOT elementary gate is a native gate for our quantum machine we get a depth equal to 6 and a size bounded by $3(n-1)$. □

The following Proposition 3.12 instead demonstrates the correctness of Algorithm 1.

Proposition 3.12. *Algorithm 1 on input* (n,k) *constructs a quantum circuit such that, when applied to a register* $|q\rangle = \bigotimes_{i=0}^{n-1} |q_i\rangle$*, returns* $\bigotimes_{i=0,}^{n-1} |q_{i-k \mod n}\rangle = \textit{ROT}_k |q\rangle$*. That is, Algorithm 1 on input* (n,k) *constructs a quantum gate performing the rotation by* k *positions,* $\textit{ROT}_k$.

Proof. The first layer of swaps of the circuit, i.e. that constructed in lines 2 and 3 of Algorithm 1, has the effect of swapping the states of q_i and q_{n-i} for every $i = 1, \ldots, \lceil \frac{n}{2} \rceil - 1$. Therefore, if $\left|q_i^0\right\rangle$ is the ith qubit of $|q\rangle$, and $\left|q_j^1\right\rangle$ is the jth qubit of $|q\rangle$ after the application of the first layer, we have that $q_i^0 = q_{n-i \mod n}^1$ for $i = 0, \ldots, n-1$. The second layer of swaps of the circuit, i.e. that constructed in lines 4 and 5 of Algorithm 1, has the effect of swapping the states of q_i and q_{k-i} for $i \in \{0, \ldots, n-1\}$ in all cases but that of even k and that of odd k and odd n, in which cases $q_{\frac{k}{2}}$ and $q_{\frac{k}{2}+\frac{n}{2}}$, respectively are fixed. However, since $k - \frac{k}{2} = \frac{k}{2}$, we have that $q_j^1 = q_{k-j \mod n}^2$, where $\left|q_h^2\right\rangle$ denotes the hth qubit of $|q\rangle$ after the application of the second and last layer of swaps of the circuit. Summing up, we have the circuit built by Algorithm 1 acting on $\left|q_i^0\right\rangle$, for $i = 0, \ldots, n-1$ as follows

$$q_i^0 = q_{n-i}^1 = q_{k-n+i \mod n}^2 = q_{i+k \mod n}^2,$$

that is the circuit operates a rotation by k positions. □

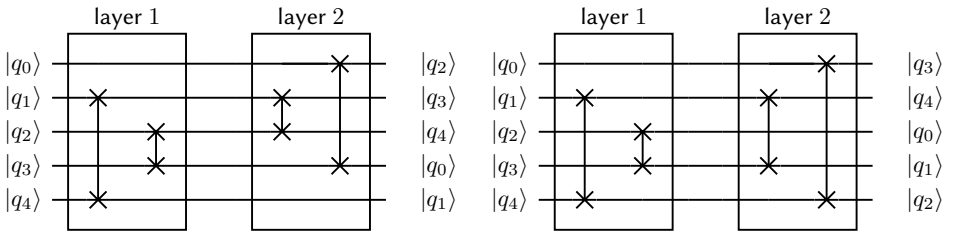

Figure 2: The circuits taking in input a register of 5 qubits and operating a rotation of 3 and 4 positions, respectively.

We conclude this section with the main result anticipated in the introduction, which establishes the complexity class of the cyclic shift problem for quantum registers.

Corollary 3.13. *The problem of cyclically shifting the states of a quantum register by any amount of positions is in QNC^0.*

Proof. Since Proposition 3.12 holds, to prove the corollary it is enough to observe that the functions $\text{SIZE}(n)$, $\text{LABEL}(n, i)$, $\text{EDGE}(n, i, j)$ - corresponding to the circuits constructed by Algorithm 1 - can be computed in logarithmic space (refer to Remark 2.3), by a slight modification of Algorithm 1. □

3.3. Quantum Circuits implementing Permutations

In this Section, we extend our previous result on cyclic shifts to the more general case of arbitrary permutations of a quantum register. This generalization involves defining an algorithm to construct quantum circuits that can perform any given permutation of the states of a quantum register. The main result of this section is encapsulated in the following Proposition 3.14.

Proposition 3.14. *Any permutation π of the states of n qubits can be performed by a quantum circuit $C_{n,\pi}$ having constant depth and linear size. Furthermore, there exists a polynomial-time algorithm that, on input (n, π)[4], constructs $C_{n,\pi}$ in polynomial time. Thus, the problem of permuting the states of finitely many qubits is in QNC^0.*

Proof. It is immediate to see that the application of an m-cycle $\sigma \in S_n$, for $m > 2$, to a quantum register $\bigotimes_{j=0}^{n-1} |q_i\rangle$ is equivalent to the rotation of magnitude 1 of the states of the qubits $q_{\sigma[0]}, q_{\sigma[1]}, \ldots, q_{\sigma[m-1]}$, where $(\sigma[0], \sigma[1], \ldots, \sigma[m-1])$ is the array corresponding to the cycle representation of σ. Therefore, the application of σ can be implemented by applying the quantum circuit for rotating m qubits by 1 position (constructed by algorithm 1 on input $(m, 1)$) to the qubits $q_{\sigma[0]}, q_{\sigma[1]}, \ldots, q_{\sigma[m-1]}$ (rather than $q_0, q_1, \ldots, q_m$). Clearly, such a circuit has a constant depth and a linear size. Specifically, the depth equals 6 and the size equals $6(m + 1)$, being the gate CNOT a native one. For $m = 2$, an m-cycle is simply a state transposition achieved through a swap gate.

Now, let π be a permutation of n elements that can be decomposed into cycles as $\pi = \sigma_1 \cdots \sigma_\ell$. Since a constant-depth and linear-size quantum circuit can implement each cycle σ_h and since the cycles are disjoint, we can run them in parallel getting a larger constant-depth and linear-size quantum circuit $C_{n,\pi}$. More precisely, by denoting m_h the length of the cycle σ_h, the circuit $C_{n,\pi}$ has still depth equal to 6 and size bounded by $6((m_1 + 1) + \cdots + (m_\ell + 1)) = 6(n + \ell) = \mathcal{O}(n)$.

[4]The permutation π is represented by its cycle-representation array.

ALGORITHM 2: The algorithm constructing a quantum circuit for applying a permutation $\pi \in S_n$ to a register of size n by k.

Input: $n, \pi = [\pi[1,0], \ldots, \pi[1, m_1 - 1], \pi[2,0], \ldots, \pi[\ell, 0], \ldots, \pi[\ell, m_\ell - 1]]$

Initialize n input wires (qubits) $|q_0\rangle, \ldots, |q_{n-1}\rangle$;
for $h = 1$ *to* ℓ **do**
  if $m_h = 2$ **then**
    Swap $\left|q_{\pi[h,0]}, q_{\pi[h,1]}\right\rangle$
  else
    for $i = 1$ *to* $\lceil \frac{m_h}{2} \rceil - 1$ **do**
      Swap $\left|q_{\pi[h,i]}, q_{\pi[h,m_h - i]}\right\rangle$
    for $j = 1$ *to* $\lceil \frac{m_h}{2} \rceil - 1$ **do**
      Swap $\left|q_{\pi[h,m_h+1-j]}, q_{\pi[h,j]}\right\rangle$

To prove the second part of the statement, it is enough to see that Algorithm 2 constructs the circuit $C_{n,\pi}$ described in the paragraph above. We remark that a gate of the form Swap $|q_i, q_i\rangle$ is interpreted as a gate that swaps the state of a qubit with its own, and therefore this does not change the state of system [5].

The algorithm consists of a for loop that is nested with two independent for loops and it takes at most $\sum_{h=1}^{\ell} 2\left(\lceil \frac{m_h}{2} \rceil - 1\right) \leq n = \mathcal{O}(n)$ steps. To complete the proof, it is enough to observe that the functions SIZE (n), LABEL (n, i), EDGE (n, i, j) - corresponding to the circuits constructed by Algorithm 2 - can be computed in logarithmic space (refer to Remark 2.3), by a slight modification of Algorithm 2. □

4. Conclusion and Future Works

In this paper, we presented a comprehensive framework for constructing logspace uniform families of constant-depth and linear-size quantum circuits capable of performing rotations and permutations of qubit states. We have formally demonstrated that these operations are within the complexity class QNC^0, ensuring their efficient parallelizability on quantum computers.

Our work builds upon foundational concepts in quantum circuit design, providing explicit algorithms and detailed constructions that ensure practical implementability. By extending the principles of cyclic shifts to general permutations, we have shown that even complex reordering operations can be achieved with minimal computational overhead, thus reinforcing the versatility and robustness of the quantum circuit model for various computational tasks.

The systematic construction of these circuits, independent of specific rotations or permutations, offers a significant advantage. It allows these circuits to be hardwired to a control quantum register, enabling the implementation of superpositions of rotations or permutations. This capability can be leveraged in advanced quantum algorithms, enhancing their efficiency and broadening their applicability in fields such as quantum cryptography, error correction, and complex system simulations.

There are several promising directions for future research stemming from our results. One avenue is the exploration of optimized implementations of these quantum circuits on current and near-term quantum hardware, focusing on minimizing gate errors and decoherence. Additionally, investigating the integration of these circuits with quantum algorithms for specific applications, such as large-scale data sorting and molecular simulations, could yield further insights into their practical utility.

[5] It is possible to have a refined version of the algorithm that recognizes such applications of the swap gate to the same qubit and does nothing.

References

[1] C. Moore, M. Nilsson, Parallel quantum computation and quantum codes, SIAM Journal on Computing 31 (2001) 799–815. URL: https://doi.org/10.1137/S0097539799355053. doi:10.1137/S0097539799355053.

[2] Y. Lao, O. Ait-Aider, Rolling shutter homography and its applications, IEEE Trans. Pattern Anal. Mach. Intell. 43 (2021) 2780–2793. URL: https://doi.org/10.1109/TPAMI.2020.2977644. doi:10.1109/TPAMI.2020.2977644.

[3] D. R. Musser, Introspective sorting and selection algorithms, Softw. Pract. Exp. 27 (1997) 983–993.

[4] R. Weil, J. Vinograd, The cyclic helix and cyclic coil forms of polyoma viral dna, Proceedings of the National Academy of Sciences 50 (1963) 730–738. URL: https://www.pnas.org/doi/abs/10.1073/pnas.50.4.730. doi:10.1073/pnas.50.4.730. arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.50.4.730.

[5] R. Dulbecco, M. Vogt, Evidence for a ring structure of polyoma virus dna, Proceedings of the National Academy of Sciences 50 (1963) 236–243. URL: https://www.pnas.org/doi/abs/10.1073/pnas.50.2.236. doi:10.1073/pnas.50.2.236. arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.50.2.236.

[6] M. Thanbichler, S. Wang, L. Shapiro, The bacterial nucleoid: A highly organized and dynamic structure, Journal of cellular biochemistry 96 (2005) 506–21. doi:10.1002/jcb.20519.

[7] F. Lisacek, Algorithms on strings, trees and sequences: Dan gusfield., Comput. Chem. 24 (2000) 135–137. URL: http://dblp.uni-trier.de/db/journals/candc/candc24.html#Lisacek00.

[8] R. Grossi, F. Luccio, Simple and efficient string matching with k mismatches, Information Processing Letters 33 (1989) 113–120. URL: https://www.sciencedirect.com/science/article/pii/0020019089901889. doi:https://doi.org/10.1016/0020-0190(89)90188-9.

[9] P. Jokinen, J. Tarhio, E. Ukkonen, A comparison of approximate string matching algorithms, Software: Practice and Experience 26 (1996). URL: https://doi.org/10.1002/(SICI)1097-024X(199612)26:12<1439::AID-SPE71>3.0.CO;2-1.

[10] D. Cantone, S. Cristofaro, S. Faro, Efficient matching of biological sequences allowing for non-overlapping inversions, in: R. Giancarlo, G. Manzini (Eds.), Combinatorial Pattern Matching, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 364–375.

[11] S. Grabowski, S. Faro, E. Giaquinta, String matching with inversions and translocations in linear average time (most of the time), Information Processing Letters 111 (2011) 516–520. URL: https://www.sciencedirect.com/science/article/pii/S002001901100055X. doi:https://doi.org/10.1016/j.ipl.2011.02.015.

[12] P. Niroula, Y. Nam, A quantum algorithm for string matching, npj Quantum Information 7 (2021) 37. doi:10.1038/s41534-021-00369-3.

[13] S. Faro, A. Pavone, C. Viola, Quantum path parallelism: A circuit-based approach to text searching, in: X. Chen, B. Li (Eds.), Theory and Applications of Models of Computation - 18th Annual Conference, TAMC 2024, Hong Kong, China, May 13-15, 2024, Proceedings, volume 14637 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 247–259. URL: https://doi.org/10.1007/978-981-97-2340-9_21. doi:10.1007/978-981-97-2340-9_21.

[14] D. Cantone, S. Faro, A. Pavone, C. Viola, Longest common substring and longest palindromic substring in $\tilde{o}(\sqrt{n})$ time, CoRR abs/2309.01250 (2023). URL: https://doi.org/10.48550/arXiv.2309.01250. doi:10.48550/ARXIV.2309.01250. arXiv:2309.01250.

[15] M. Brooks, Quantum computers: what are they good for?, Nature Spotlight 617, S1-S3 (2023). URL: https://doi.org/10.1038/d41586-023-01692-9.

[16] H. Liu, G. H. Low, D. S. Steiger, T. Häner, M. Reiher, M. Troyer, Prospects of quantum computing

for molecular sciences, Materials Theory 6 (2021) 1–17. URL: https://api.semanticscholar.org/CorpusID:231979408.
[17] F. Bova, A. Goldfarb, R. G. Melko, Commercial applications of quantum computing, Epj Quantum Technology 8 (2021). URL: https://doi.org/10.1140/epjqt/s40507-021-00091-1.
[18] A. Pavone, C. Viola, The quantum cyclic rotation gate, in: G. Castiglione, M. Sciortino (Eds.), Proceedings of the 24th Italian Conference on Theoretical Computer Science (ITCTS 2023), volume Vol-3587 of *Italian Chapter of the European Association for Theoretical Computer Science*, IC-EATCS, University of Palermo, Italy, 2023, pp. 206–218.
[19] N. D. Mermin, Quantum Computer Science: An Introduction, Cambridge University Press, USA, 2007.
[20] E. Farhi, J. Goldstone, S. Gutmann, M. Sipser, Quantum computation by adiabatic evolution, 2000. arXiv:quant-ph/0001106.
[21] E. Witten, Topological quantum field theory, Communications in Mathematical Physics 117 (1988) 353 – 386.
[22] R. Jozsa, An introduction to measurement based quantum computation, 2005. arXiv:quant-ph/0508124.
[23] W. K. Wootters, W. K. Wootters, W. H. Zurek, A single quantum cannot be cloned, Nature 299 (1982) 802–803. URL: https://api.semanticscholar.org/CorpusID:4339227.
[24] J. Watrous, Quantum Computational Complexity, Springer New York, New York, NY, 2009, pp. 7174–7201. URL: https://doi.org/10.1007/978-0-387-30440-3_428. doi:10.1007/978-0-387-30440-3_428.
[25] F. T. Leighton, Introduction to parallel algorithms and architectures: array, trees, hypercubes, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991.
[26] S. Arora, B. Barak, Computational Complexity: A Modern Approach, Cambridge University Press, 2009.
[27] A. Borodin, On relating time and space to size and depth, SIAM Journal on Computing 6 (1977) 733–744. URL: https://doi.org/10.1137/0206054. doi:10.1137/0206054.
[28] A. Borodin, S. Cook, N. Pippenger, Parallel computation for well-endowed rings and space-bounded probabilistic machines, Information and Control 58 (1983) 113–136. doi:https://doi.org/10.1016/S0019-9958(83)80060-6.
[29] C. C. Sims, Computational methods in the study of permutation groups, in: J. Leech (Ed.), Computational Problems in Abstract Algebra, Pergamon, 1970, pp. 169–183. doi:https://doi.org/10.1016/B978-0-08-012975-4.50020-5.
[30] D. E. Knuth, Efficient representation of perm groups, Combinatorica 11 (1991) 33–43. URL: https://api.semanticscholar.org/CorpusID:18263902.