SN Computer Science (2024) 5:830
https://doi.org/10.1007/s42979-024-03141-4

ORIGINAL RESEARCH

The Quantum Cyclic Rotation Gate

Arianna Pavone[1] · Caterina Viola[2]

Received: 11 May 2024 / Accepted: 13 July 2024

Abstract
A *circular shift operator* (or *cyclic rotation gate*) $\texttt{ROT}_k$ applies a rightward (or leftward) shift to an input register of n qubits o by as many positions as encoded by an additional input $k \in \mathbb{N}$. Specifically, the qubit at position x is moved to position $(x+k) \mod n$. While it is known that there exists a quantum rotation operator that can be implemented in $\mathcal{O}(\log(n))$-time, through the repeated parallel application of the elementary $\texttt{Swap}$ operators, there is no systematic procedure that concretely constructs the quantum operator $\texttt{ROT}$ for variable size n of the quantum register and a variable parameter k. We fill the gap, providing a systematic implementation of the cyclic rotation operator (denoted $\texttt{ROT}$) in a quantum circuit model of computation whose depth is $\mathcal{O}(\log(n))$. We show how the circular shift operator can be utilized in quantum approaches to text processing, focusing on the problem of getting all possible cyclic rotations of a string in $\mathcal{O}(\log^2(n))$ depth.

Keywords Quantum computing · Quantum gates · Combinatorics on words

Introduction

Quantum computing represents an avant-garde domain within the realm of computer science, where the intricate principles of quantum mechanics are harnessed to engineer formidable computing systems that manifest striking deviations from classical counterparts. In stark contrast to classical computers, which process information using discrete binary bits constrained to exclusively one of the states 0 and 1, quantum computing harnesses the power of *quantum bits*, or *qubits*, which effortlessly inhabit the *superposition* of multiple states. Moreover, the *entanglement* of two or more qubits bestows upon them the extraordinary ability to execute correlated operations.

In the recent few decades quantum computing has emerged as a transformative technology with the potential to revolutionize fields ranging from cryptography to materials science. This nascent field has witnessed significant milestones that underscore its potential to solve problems previously thought to be intractable for classical computers. Among the most notable achievements, quantum supremacy was demonstrated, showing that quantum computers can perform specific tasks faster than the world's most powerful supercomputers.

In the recent few decades quantum algorithms have made substantial progress in optimization and simulation tasks. For instance, Grover's [2] and Shor's [3] algorithms provided early examples of quantum advantage by offering quadratic speedup in database searching and exponential speedup in integer factorization, respectively. These theoretical advancements set the stage for practical applications in secure communications and complex problem solving. In addition to these developments, recent studies [4] have successfully applied quantum computation to tackle NP-complete problems, which are notoriously challenging for classical algorithms [5]. In natural language processing tasks as well, classifiers integrating quantum and classical computing techniques are anticipated to have

This article is an extended version of a result [1] initially presented in the proceedings of the ICTCS 2023 (24th Italian Conference on Theoretical Computer Science) conference, featuring expanded analyses and additional data to enhance the understanding of the findings.

✉ Arianna Pavone
ariannamaria.pavone@unipa.it

Caterina Viola
caterina.viola@unict.it

[1] Department of Mathematics and Computer Science, University of Palermo, Via Archirafi, 34, Palermo 90123, Italy

[2] Department of Mathematics and Computer Science, University of Catania, Viale A. Doria 6, Catania 95125, Italy

Published online: 27 August 2024

a substantial impact on key aspects, particularly in areas concerning classification [6].

As quantum hardware continues to evolve and algorithms become more sophisticated, the integration of quantum computing into broader scientific and industrial applications appears increasingly feasible. This progression promises not only to expand our computational capabilities but also to redefine problem-solving paradigms across numerous disciplines.

In this paper, we address the construction of a *quantum cyclic shift* (or *cyclic rotation*) operator, in the quantum circuit model of computation. Given a vector x of length n and an input parameter $s < n$, a cyclic rotation of a vector is a transformation that shifts the elements of the vector in circular positions, while maintaining their relative order. In other words, each vector entry is moved by s positions, and the last s elements are brought back to the first positions of the vector.

Cyclic rotations of vectors have various applications, including, for instance, image and signal processing, where they can be employed to perform cyclic shifts on images or signals, such as image rolling [7] or time delay of a signal. Cyclic rotations can be also used to design efficient algorithms sorting data [8]. In addition, sequences admitting cyclic rotations are also relevant in various biological contexts, including viruses [9, 10] and bacteria [11]. Thus, the analysis of organisms with a cyclic structure can benefit from algorithms designed for strings that allow for cyclic rotations [12].

Since a cyclic rotation of a vector of n elements of s positions to the right consists essentially of a permutation of the input vector in which each element of position i is moved to position $(i + s) \mod (n)$, it is easy to construct a classical procedure capable of achieving such a rotation in linear time. In fact, in a classical model of computation, the problem has $\Omega(n)$-time complexity, since in the worst-case scenario, every element in the array needs to be shifted. A specialization of the cyclic rotation problem arises when the vector is a string of bits (of classical machines). Here is interesting to observe that while many concrete computers have a built-in *shift* instruction that shifts the target bits to the left (or to the right), such an operation would still need to be executed on all the bits; furthermore, the cyclic desired manner of the shift slightly requires more effort than just shifting towards one side.

In the field of quantum computation, the cyclic rotation of the states of a given register of qubits has been effectively used in solutions for text processing [13–15], and specifically for exact and approximate string matching [16, 17]. The recent algorithm by Niroula and Nam [13] makes clever use of cyclic rotations of the registers encoding the input strings to achieve a superposition of all their possible alignments and to perform a parallel comparison against the pattern.

This idea was later generalized by Cantone et al. in [14] to get a quantum solution to the string matching problem allowing for swaps of adjacent characters, which is more time-efficient than the best known classical counterpart. In their paper, Niroula and Nam provide insight into the fact that a circuit performing a cyclic rotation of the states of a given register of qubits can be executed in time $\mathcal{O}(\log(n))$. The basic idea is that at each step of the algorithm that accomplishes the permutation, it is possible to place at least half of the qubits that still need to be moved to their final position. Since the number of qubits to be placed decreases by at least half at each iteration, $\mathcal{O}(\log(n))$ steps are needed to achieve the target permutation. However, they do not provide any procedure explaining how to construct this quantum circuit systematically, nor do they provide a more formal proof of its complexity.

In an attempt to fill the gap, this paper aims to provide a precise method for the construction of a circular rotation operator for a quantum register of dimension n as the parameter s, indicating the shift relative to the rotation, varies. As far as we know, this is the first work offering such a construction. A proof of the correctness of our procedure is also provided, along with an analysis of the time complexity of the resulting circuit.

We believe that this result may be of interest to the scientific community concerned with the design and simulation of quantum algorithms, especially in the area of text processing.

The paper is organized as follows. In Sect. Preliminaries and Definition of the Problem we recall some basic notions, introduce some useful notations adopted along the paper and give a more formal definition of the problem. In Sect. An Algorithm for $k = 2^m$ we present a solution for the specific case where $s = 2^p$ for some $p \in \mathbb{N}$, prove its correctness and discuss its complexity analysis. In Sect. The General Algorithm for $1 \le k \le n - 1$ we extend our solution to the general case. In Sect. The controlled circular shift gate for character strings, we exhibit an algorithm for the *controlled circular shift operator*, a quantum circuit operating any circular shift of an input quantum register encoding a character string $|q\rangle$ of any arbitrary size n. Finally, we draw our conclusions in Sect. Discussion and Conclusions.

Preliminaries and Definition of the Problem

The fundamental unit in quantum computation is the *qubit*. A qubit is a coherent superposition of the two orthonormal computational basis states, which are denoted by $|0\rangle$ and $|1\rangle$, using the conventional *bra-ket* notation. Formally, a single qubit is an element from the *state space* $\mathcal{H}$, that is the two-dimensional Hilbert space on the complex numbers equipped with the inner product; 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 the values α and β are called *amplitudes*, are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$, representing the probability of measuring the qubit in the state $|0\rangle$ or $|1\rangle$, respectively. A *quantum measurement* is the only operation giving access to the information on the state of a qubit; however, this operation causes the qubit to collapse to one of the two basis states.

Multiple qubits taken together are referred to as *quantum registers*. A quantum register $|\psi\rangle = |q_0, q_1, .., q_{n-1}\rangle$ of size n is an element from the tensor product of n state spaces, $\mathcal{H}^{\otimes n}$, and thus it is expressed as a linear combination of the 2^n states in $\{0,1\}^n$, that is $|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k|k\rangle$, where the values α_k represent the probability of measuring the register in the state $|k\rangle$, and such that with $\sum_{k=0}^{2^n-1} |\alpha_k|^2 = 1$.

Let k be an integer value that can be represented by a binary string of length n. The symbol $|k\rangle$ denotes the register of size n such that $|k\rangle = \bigotimes_{i=0}^{n-1} |k_i\rangle$, where $|k_i\rangle$ takes the value of the i-th least significant binary digit of k. For example, the quantum register $|9\rangle$ with 4 qubits is given by $|9\rangle = |1\rangle \otimes |0\rangle \otimes |0\rangle \otimes |1\rangle$.

We use $|q\rangle^{\oplus n}$ to denote a quantum register of size n such that each of its constituent qubits is in the state $|q\rangle$.

The Quantum Circuit Model

The model of computation we adopt in this paper is that of *quantum circuits*. In fact, there are several ways to model quantum computation, each with its own advantages and challenges. These include the *adiabatic model* [18], based on the adiabatic theorem of quantum mechanics, the *topological model* [19], based on the principles of topological quantum field theory, and the *measurement-based model* [20], where the computation is performed by making measurements on an entangled resource state known as a cluster state.

Perhaps, the the most common and widely used model in actual quantum programming is the *circuit model* [21].

In general, a computational circuit can be represented as a direct acyclic graph whose nodes are to be interpreted as the gates that operate on the information carried by the edges. While circuit models are also used to formalize classical computation, e.g. Boolean circuits and arithmetic circuits, there are a few requirements on quantum circuits that depend on the principle of quantum mechanics. Indeed, a quantum circuit needs to be reversible, this means, in particular, that for each gate the number of input edges equals that of output edges. Furthermore, because of the No-Cloning Theorem [22], we cannot either copy or split (fan-out) the information carried by an edge. As a consequence, we can see the graph representation of a circuit as a sequence of parallel wires

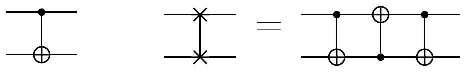
Fig. 1 The representation of the CNOT and Swap gates. The Swap gate corresponds to three CNOT gates

(each corresponding to a qubit) passing through certain gates that operate on them.

There are two major measures of the computational time complexity in the circuit model. One of them is the total number of basic gates, namely the *size* of the circuit, and the other is the *depth* of the direct acyclic graph that represents the circuit. The latter is usually the measure of election for the complexity of quantum circuits. The reason is that in a quantum system it is often possible to run two or more gates in parallel whenever they operate on disjoint sets of qubits. Therefore, the depth of a quantum circuit coincides with the number of time steps performed before the output, that is, with its computational time. The space complexity of a quantum circuit is the number of qubits that it involves, which can also be pictured as the number of parallel wires of the circuit representation.

The popularity of the quantum circuit model is due also to the existence of fundamental quantum circuit libraries provided by almost every concrete quantum programming language (e.g., IBM Qiskit, Google Cirq, Microsoft Q#), which allow for high-level programming of real quantum machines and quantum simulators. This contributes to making the design and simulation of efficient quantum circuits capable of solving specific tasks, including through the use of artificial intelligence models [23], a particularly active area of research in recent years.

Elementary quantum gates are those performing elementary operations that are implemented in constant time by real quantum machines and, by definition, have depth $\Theta(1)$ in the quantum circuit model. There is a great variety of elementary quantum operators that can be combined to obtain non-elementary gates that perform more complex quantum operations. We only list the two elementary gates that will be used in this paper. To define such basic operations, we use the linearity of quantum maps.

- The *controlled* NOT *gate* (or CNOT) operates on a register of size two qubits $q_0, q_1\rangle$. If the control qubit $|q_0\rangle$ is in the state $|1\rangle$, CNOT complements the target qubit $|q_1\rangle$ otherwise, all qubits stay the same. Formally, it maps $|q_0, q_1\rangle$ to $|q_0, q_0 \oplus q_1\rangle$.
- The *Swap gate* is a two-qubit operator that swaps the state of the two qubits $|q_0, q_1\rangle$ involved in the operation, mapping them to $|q_1, q_0\rangle$. Interestingly, the swap gate can be achieved by the application of three CNOT operators.

Figure 1 shows the representation of the CNOT and Swap gates.

We finally note that, in the definition of a circuit, it is often necessary to include *ancillæ* qubits, which are needed to achieve some specific tasks in computation that otherwise could not be achieved. Such ancillæare needed to fan the information out, which would otherwise be impossible because of the No-Cloning Theorem.

The Circular Shift Operator

A *rightward circular shift operator* (or rightward cyclic rotation operator) $\mathtt{ROT}^+_k$ 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 $\mathtt{ROT}^+_k$ applies the following permutation

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

We call the parameter k the *magnitude* of the circular shift.

A *leftward circular shift operator* (or leftward cyclic rotation operator) $\mathtt{ROT}^-_k$ 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 $\mathtt{ROT}^-_k$ applies the following permutation

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

All the results presented in this article refer to the *rightward* circular shift operator unless we explicitly declare that this is not the case. However, it is immediate to verify that

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

for every register of size n and every $0 \le k < n$. Thus, all the results stated for the rightward circular shift operator hold true for the leftward circular shift operator, too, and the needed modifications in the algorithmic techniques are trivial. For this reason, often we simply write *circular shift operator* and $\mathtt{ROT}_k$ referring to its rightward version.

In the next sections, we present an algorithm building a quantum circuit that systematically performs $\mathtt{ROT}_k(n)$ for a quantum register encoding a character string of arbitrary size n and for arbitrary magnitude k. We present our algorithm and discuss its correctness in two steps. We first exhibit and discuss the construction of quantum gates performing circular shift with a magnitude of the form $k = \frac{n}{2^\ell}$, for $0 < \ell \le \log(n)$. Secondly, we slightly modify such construction to make it work in the general case, that is for any k such that $1 \le k \le n-1$.

Finally, we exhibit a quantum circuit that, given the quantum register $|q\rangle$ encoding a character string of length n over an alphabet of fixed cardinality, operates on this *quantum string* circularly shifting $|q\rangle$ by a number of characters depending on an input value k ranging from 0 to $n-1$. We call such a circuit the *controlled circular shift operator*. Throughout the document we assume that the size n of the quantum register to be rotated is of the form 2^p for some $p \in \mathbb{N}$. We can make this assumption without loss of generalization because given a character string of length n we can always add to the string a suitable number of copies of a special character outside of the alphabet so as to get a string of length 2^p for some $p \in \mathbb{N}$.

An Algorithm for $k = 2^m$

In this section, we describe an algorithm that cyclically rotates a quantum register *ket q* of size n by k positions, with $0 \le k < n$. We recall that we assume $n = 2^p$ for some $p \in \mathbb{N}$.

We first consider the case that the magnitude of the rotation is of the form $k = 2^m$, with $m < p$. Actually, it is enough to assume $k = \frac{n}{2^h}$ for some $h : 1 \le h \le p-1$. The pseudocode of the quantum procedure performing the cyclic rotation is presented in Algorithm 2.

Our procedure performs a permutation of the qubits contained in the input quantum register. This is done through a sequence of swap operations applied to pairs of qubits from the register. During the execution of the algorithm, we distinguish qubits having reached their final position, which we indicate by the term *placed qubits*, from qubits having not yet been placed correctly, which are indicated by the term *out-of-place qubits*.

In brief, the algorithm works as follows. At the beginning of the procedure, the register $|q\rangle$ only contains n not-in-place qubits. Then, an iterative cycle starts: at each iteration, the algorithm selects half of the remaining not-in-place qubits and swaps them to their final positions. Therefore, the procedure stops in at most $\log(n)$ steps.

The worst case is obtained when each swap operation moves only one of the two involved qubits to its final destination, terminating in $\log(n)$ steps. The best case occurs when each swap happening in the first iteration succeeds in placing both qubits in their final positions, in this case $(k = \frac{n}{2})$, the algorithm terminates in constant time.

We now deepen into the design of Algorithm 1, which is based on a main iterative loop (line 1) that runs $\log(n) - \log(k)$ times.[1]

[1] We'll discuss why the iterative loop on line 1 runs $\log(n) - \log(k)$ times later.

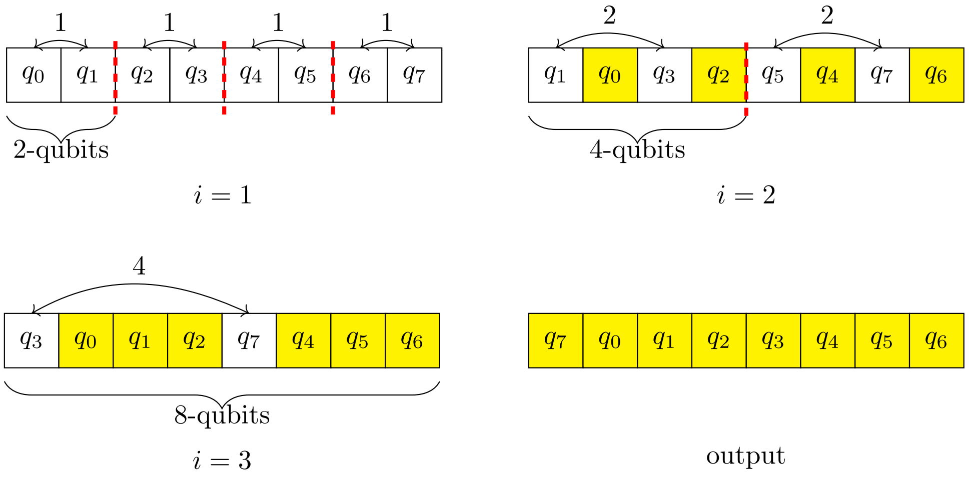
1
1
1
1
q_0 q_1 q_2 q_3 q_4 q_5 q_6 q_7
2-qubits
$i = 1$
2
2
q_1 q_0 q_3 q_2 q_5 q_4 q_7 q_6
4-qubits
$i = 2$
4
q_3 q_0 q_1 q_2 q_7 q_4 q_5 q_6
8-qubits
$i = 3$
q_7 q_0 q_1 q_2 q_3 q_4 q_5 q_6
output

Fig. 2 Illustration of the iterations of the algorithm implementing the circular shift operator for a register of 8 qubits in which a rotation of 1 position is performed. The coloured qubits are *placed qubits*

Algorithm 1 Algorithm for $k = 2^m$

Input: a n-qubit register $q := \bigotimes_{x=0}^{n-1} |q_x\rangle$, and $k := 2^m$.
Output: a n-qubit register $q' := \bigotimes_{x=0}^{n-1} \left|q_{x+k \mod (n)}\right\rangle$.

```
1 for $i = 1, \ldots, \log(n) - \log(k)$; $i++$ do
2     for $j = 0, \ldots, \frac{n}{2^i k} - 1$; $j++$ do
3         for $x = jk2^i, \ldots, (j2^i + 1)k - 1$; $x++$ do
4             Swap $|q_x, q_{x+2^{i-1}k}\rangle$
```

In the first iteration (namely for $i = 1$), the algorithm decomposes the register $|q\rangle$ into $\frac{n}{2k}$ intervals, each of size $2k$ (line 2). Let I_j, for $0 < j < \frac{n}{2k}$, be the j-th interval into which the register $|q\rangle$ has been divided. The algorithm divides each interval I_j into two halves of size k. In this context, let I_j^ℓ be the left half of the interval I_j and let I_j^r be the right half of the same interval. The algorithm operates by swapping the qubits in I_j^ℓ with the corresponding qubits in I_j^r (line 3). Specifically it applies a Swap to the pair of qubits (q_x, q_{x+k}) for every x corresponding to a position in I_j^ℓ, that is $x \in \{2jk, \ldots, (2j+1)k - 1\}$. We stress the fact that the algorithm performs these swaps in parallel for each $j \in \{0, \ldots, \frac{n}{2k} - 1\}$.

Since this operation shifts the qubits in I_j^ℓ by exactly k positions to the right, after the first iteration, half of the qubits are correctly placed. It is immediate to see, indeed, that the algorithm correctly places the qubits that have been moved to positions $x + k$, for $x \in \{2jk, \ldots, (2j+1)k - 1\}$ and for $j \in \{0, \ldots, \frac{n}{2k} - 1\}$.

In Sect. Correctness of Algorithm 1 we prove that if $k = \frac{n}{2}$, that is if $\log(n) - \log(k) = 1$, we do not need further iterations as also the qubits at positions q_x, for $x \in \{2jk, \ldots, (2j+1)k - 1\}$ and for $j \in \{0, \ldots, \frac{n}{2k} - 1\}$, are correctly placed and the algorithm correctly terminates. Otherwise, the algorithm starts a new iteration.

The interval decomposition that we described in the first iteration can be generalized in subsequent iterations of the main for loop as follows. In the ith step, being $1 < i \le \log(n) - \log(k))$, the algorithm decomposes the register $|q\rangle$ into $\frac{n}{2^i k}$ intervals, each of size $2^i k$ (line 2).

Now observe that the first k elements of the left interval I_j^ℓ are in that position as a result of a swap of the previous iteration. We are, therefore, dealing with not-in-place qubits. The same is true for the first k qubits of the interval I_j^r. Thus, the algorithm swaps the qubits in the first k positions of I_j^ℓ with the qubits in the corresponding first k positions of I_j^r (line 3). This is done to forbid swaps of qubits that are already correctly placed.

More formally, the algorithm applies a swap to the pair of qubits $(q_x, q_{x+2^{i-1}k})$ for every x corresponding to a position in the first half of I_j, that is $x \in \{2^i jk, \ldots, (2^i j + 1)k - 1\}$. Also in this case the algorithm performs these swaps in parallel for each $j \in \{0, \ldots, \frac{n}{2^i k} - 1\}$. Figure 2 illustrates the iteration of the algorithm implementing the circular shift operator for a register of 8 qubits in which a cyclic rotation by 1 position is performed. Figure 3 illustrates the application of the circular shift operator as prescribed by Algorithm 2, for a register of 8 qubits in which a rightward circular shift of magnitude 1, 2, and 4 is performed. In the representation of each operator, the time steps, within which the swaps are executed in parallel, have been framed. Each time-step is associated with a label $\mathtt{ROT}_k^{[i]}$, where k represents the shift amount and i enumerates the iterative steps.

To analyze the computational complexity, let us observe that in a quantum circuit model of computation the **for**-loops at lines 2 and 3 of Algorithm 1 can be executed in parallel. To see this it is enough to check that the aim of line 2 is to partition the register into disjoint intervals, while the aim of lines 3-4 is to swap disjoint qubits in such intervals. More

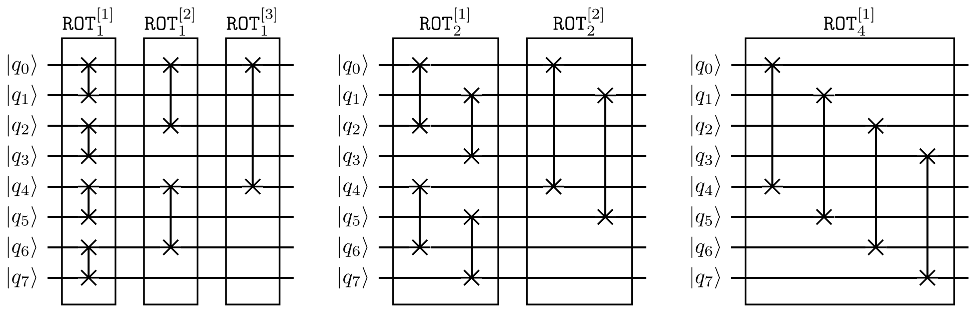

Fig. 3 The application of the circular shift operator for a register of 8 qubits in which a rotation of 1, 2, and 4 positions is performed, respectively

precisely, each iteration of line 3 refers to a specific interval among those individuated in line 2; because such intervals are disjoint, the whole computation described in lines 2-4 happens in parallel in a one-time step.

It follows that the running time of Algorithm 1 is $\log(n) - \log(k)$, which is $\mathcal{O}(\log(n))$.

Correctness of Algorithm 1

In this section, we prove the correctness of Algorithm 1. We start by making the following considerations.

1. At the ith iteration, with $1 < i \leq \log(n) - \log(k)$, a qubit is not involved in any Swap if and only if it has already been correctly shifted in a previous iteration, i.e. if and only if it is a placed qubit.
2. After $\log(n) - \log(k)$ iterations of Algorithm 1, every qubit is correctly cyclically shifted by k positions rightwardly, i.e. the jth qubit has been placed at position $j + k \mod (n)$.

For $i \in \{1, \dots, \log(n) - \log(k)\}$, the symbol q_x^i denotes the xth qubit of the register $|q\rangle$ at the end of the ith iteration of the algorithm; also, we set $q_x^0 := q_x$, and $|q^i\rangle = \bigotimes_{x=0}^{n} q_x^i$, accordingly.

The correctness of Algorithm 1 immediately follows from the next lemma.

Lemma 1 *If* $1 \leq i \leq \log(n) - \log(k)$, *at the end of the ith iteration of Algorithm* 1, *for every* $j \in \{0, \dots, \frac{n}{2^i k} - 1\}$ *it holds that*

- $q_x^i = q_{x-k}^0$, for $x \in \{(2^i j + 1)k, \dots, 2^i(j+1)k - 1\}$, and
- $q_y^i = q_{y-(1-2^i)k}^0$, for $y \in \{2^i jk, \dots, (2^i j + 1)k - 1\}$.

In particular, for $i = \ell = \log(n) - \log(k)$ it holds $q_z^\ell = q_{z-k}^0$, for $z \in \{0, \dots, n-1\}$, that is, after ℓ iterations of Algorithm 1 every qubit is correctly placed.

Proof We prove the lemma by induction on the number of iterations i.

For $i = 1$, it is immediate to see that for every $j \in \{0, \dots, \frac{n}{2^i k} - 1\}$, every $y \in \{2^i jk, \dots, (2^i j + 1)k - 1\}$, and every $x = y + 2^{1-1}k = y + k$, the execution of $\mathtt{Swap}\left|q_x^0, q_y^0\right\rangle$ yields $\left|q_x^1, q_y^1\right\rangle$ where $q_y^1 = q_x^0 = q_{y+k}^0 = q_{y-(1-2^i)k}^0$ and $q_x^1 = q_y^0 = q_{x-k}^0$.Let $1 \leq i < \log(n) - \log(k)$, assume the claim true for $|q^0\rangle, \dots, |q^i\rangle$, and let us prove it is true for $|q^{i+1}\rangle$. By inductive hypothesis, for $j \in \{0, \dots, \frac{n}{2^i k} - 1\}$, it holds

$q_x^i = q_{x-k}^0$, for $x \in \{(2^i j + 1)k, \dots, 2^i(j+1)k - 1\}$, and
[0.2cm] $q_y^i = q_{y-(1-2^i)k}^0$, for $y \in \{2^i jk, \dots, (2^i j + 1)k - 1\}$.

It does not require much effort to check that

$$\{2^{i+1}jk, \dots, (2^{i+1}j + 1)k - 1 \mid 0 \leq j \leq \frac{n}{2^{i+1}k} - 1\}$$
$$= \{2^i jk, \dots, (2^i j + 1)k - 1 \mid 0 \leq j \leq \frac{n}{2^i k} - 2 \text{ such that } j \text{ is even}\},$$

and that

$$\{(2^{i+1}j + 2^i)k, \dots, (2^{i+1}j + 1 + 2^i)k - 1 \mid 0 \leq j \leq \frac{n}{2^{i+1}k} - 1\}$$
$$= \{2^i jk, \dots, (2^i j + 1)k - 1 \mid 0 \leq j \leq \frac{n}{2^i k} - 1 \text{ such that } j \text{ is odd}\}.$$

It follows that all the qubits involved in a swap during the $(i+1)$st iteration of the algorithm are of the form $q_y^i = q_{y-(1-2^i)k}^0$. Therefore, $q_z^{i+1} = q_z^i$ for $z \in \{(2^i j + 1)k, \dots, 2^i(j+1)k - 1\}$ and $j \in \{0, \dots, \frac{n}{2^i k} - 1\}$. Moreover, during the $(i+1)$th iteration, the algorithm executes the operation $\mathtt{Swap}\left|q_x^i, q_y^i\right\rangle$ for $x \in \{2^{i+1}jk, \dots, (2^{i+1}j + 1)k - 1\}$ and $y = x + 2^i k$, where $j \in \{0, \dots, \frac{n}{2^{i+1}k} - 1\}$; each of these swaps results in $\left|q_x^{i+1}, q_y^{i+1}\right\rangle$
$q_x^{i+1} = q_y^i = q_{x+2^i k}^i = q_{x-(1-2^i)k+2^i k}^0 = q_{x-(1-2^{i+1})k}^0$, and
$q_y^{i+1} = q_{x+2^i k}^{i+1} = q_x^i = q_{x-(1-2^i)k}^0 = q_{x+2^i k-k}^0 = q_{y-k}^0$.

To complete the proof it is enough to observe that

$$\{(2^i j + 1)k, \dots, 2^i(j+1)k - 1 \mid 0 \leq j \leq \frac{n}{2^i k} - 1\}$$
$$\cup\{(2^{i+1}j + 2^i)k, \dots, (2^{i+1}j + 1 + 2^i)k - 1 \mid 0 \leq j \leq \frac{n}{2^{i+1}k} - 1\}$$
$$=\{(2^{i+1}j + 1)k, \dots, 2^{i+1}(j+1)k - 1 \mid 0 \leq j \leq \frac{n}{2^{i+1}k} - 1\}.$$

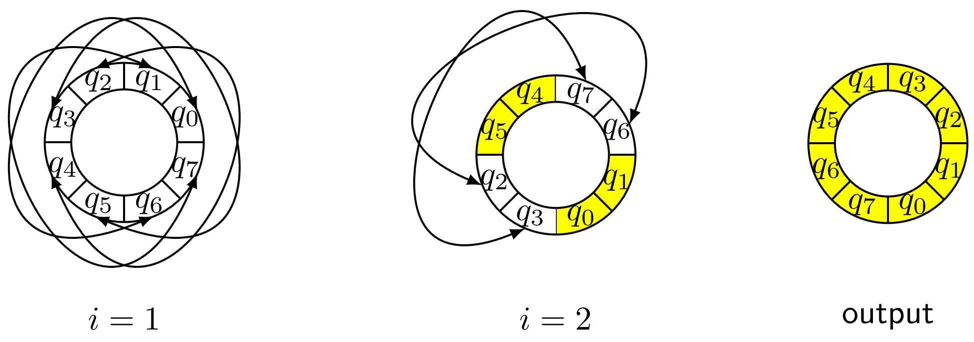
q_2 q_1 q_0 q_7 q_6 q_5 q_4 q_3
$i = 1$
q_4 q_7 q_6 q_1 q_0 q_3 q_2 q_5
$i = 2$
q_4 q_3 q_2 q_1 q_0 q_7 q_6 q_5
output

Fig. 4 Illustration of the iterations of the algorithm implementing the circular shift operator for a register of 8 qubits in which a rotation of 6 position is performed. The coloured qubits are *placed qubits*. In this circular representation of the register is evident that all swapped pairs have the same distance

Indeed, the previous equality implies that

$$q_x^{i+1} = q_{x-k}^0,\ \text{for } x \in \{(2^{i+1}j+1)k, \ldots, 2^{i+1}(j+1)k-1\},\ \text{and } j \in \left\{0, \ldots, \frac{n}{2^{i+1}k}-1\right\}.$$

Finally, observe that for $i = \ell = \log(n) - \log(k)$, the variable j assumes only the value 0; furthermore, for $x \in \{0, \ldots, k-1\}$ it holds

$$q_x^\ell = q_{x-(1-2^\ell)k}^0 = q_{x-k}^0.$$

This completes our proof.

□

To prove the correctness of Algorithm 1, it is enough to observe that, by Lemma 1, after $\ell = \log(n) - \log(k)$ iterations of Algorithm 1 every qubit is correctly placed, that is, rightward cyclically shifted by k positions.

Remark 1 From the proof of correctness (cf. Sect. Correctness of Algorithm 1), it is easy to verify that Algorithm 1 outputs a quantum circuit performing the circular rotation of a quantum register $|q\rangle$ of size n by k positions in the more general case in which the quotient $\frac{n}{k}$ of n and k is a power of 2.

The General Algorithm for $1 \le k \le n-1$

In this section, we present the algorithm implementing the circular shift operator on the input register of k positions to the right, for the general case in which $1 \le k \le n-1$.

The idea behind the general algorithm is very similar to the idea underlying the algorithm for $k = 2^m$. The pseudocode of such a general procedure is depicted in Algorithm 2.

Specifically, a new parameter ℓ is defined, by setting

$$\ell := \min\{1 \le i \le \log(n) \mid 2^i k = 0 \mod (n)\}.$$

To understand the choice of ℓ, imagine the qubits from $|q\rangle$ arranged circularly. Observe that, by its definition, ℓ is the smallest positive integer such that an interval of length $2^\ell k$ starting at the qubit q_0 finishes at the qubit q_{n-1} when wrapped around $|q\rangle$.

Algorithm 2: Algorithm for a generic $k \in \{1, \ldots, n-1\}$

```
  Input: a $n$-qubit register $q := \bigotimes_{x=0}^{n-1} |q_x\rangle$, and $1 \le k \le n-1$.
  Output: a $n$-qubit register $q' := \bigotimes_{x=0}^{n-1} |q_{x+k \mod (n)}\rangle$.
1 $\ell := \min\{1 \le i \le \log(n) \mid 2^i k = 0 \mod (n)\}$;
   for $i = 1, \ldots, \ell$; $i++$ do
2     for $j = 0, \ldots, 2^{\ell-i} - 1$; $j++$ do
3         for $x = j\frac{n}{2^\ell}2^i, \ldots, (j2^i + 1)\frac{n}{2^\ell} - 1$; $x++$ do
4             Swap($q_x, q_{x+2^{i-1}k \mod (n)}$)
```

Informally, the main difficulty encountered when trying to extend the approach from Algorithm 1 to the general case consists in the fact that it is not possible to decompose the n-qubits register $|q\rangle$ in disjoint intervals of length $2k$, 2^2k, etc., in general. However, once again, we can imagine that the qubits from $|q\rangle$ are arranged along a circle to which we wrap the decomposition in intervals of length 2^ik around. Therefore, we can adapt Algorithm 1 to the general case of an arbitrary k, by reasoning in the n-modular arithmetic. Indeed, by comparing the pseudocodes of the two algorithms, it is immediate to see that in Algorithm 2 we replaced $\log(n) - \log(k)$ by ℓ, and k by $\frac{n}{2^\ell}$.

Figure 4 illustrates the iterations of the algorithm implementing the circular shift operator for a register of 8 qubits in which a rotation of 6 position is performed. Figure 5 and Fig. 6 provide an illustration of the application of the circular shift operator as prescribed by Algorithm 2, for a register of 8 qubits in which a rightward circular shift of magnitude 3, 5, and 6 is performed.

To analyze the computational complexity, we observe that - as in the case of Algorithm 1 - in a quantum circuit model of computation the **for**-loops at lines 3 and 4 are executed in parallel. This means that, once ℓ is known, Algorithm 2 executes $\ell \le \log(n)$ time-steps. Furthermore, to compute the number ℓ of iterations needed, the algorithm has to perform at most $\log(n)$ multiplications. Therefore, the overall time-complexity of Algorithm 2 is $\mathcal{O}(\log(n))$.

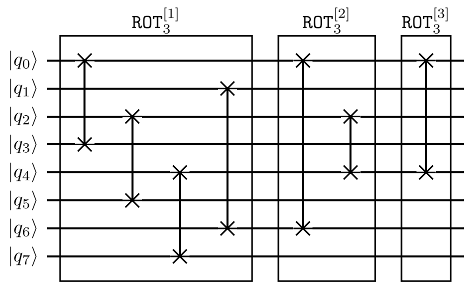

Fig. 5 The application of the circular shift operator for a register of 8 qubits in which a rotation of 3 positions is performed. Observe that all the pairs of qubits swapped in the same iteration have the same distance. For instance, in $\mathtt{ROT}_3^{[1]}$, the pair q_6 and q_1 have distance $1-6=3 \mod (8)$, which is the same distance between the other swapped pairs of qubits. Similarly, all the swapped pairs of qubits in $\mathtt{ROT}_3^{[2]}$ have distance $6 \mod (8)$

The next Theorem states the correctness of the general algorithm.

Theorem 1 *Algorithm* 2 *correctly outputs the rightward circular shifting of an input n-qubits register q by k positions.*

To prove Theorem 1, we employ the following Lemma 2.

Lemma 2 *If* $1 \le i \le \ell$, *at the end of the ith iteration of Algorithm* 2, *for every* $j \in \{0,\dots,2^{\ell-i}-1\}$ *it holds that*

- $q_x^i = q^0_{x-k \mod n}$, for $x \in \{(2^ij+1)k,\dots,2^i(j+1)k-1\}$, and
- $q_y^i = q^0_{y-(1-2^i)k \mod n}$, for $y \in \{2^ijk,\dots,(2^ij+1)k-1\}$.

In particular, for $i=\ell$ it holds $q_z^\ell = q^0_{z-k \mod n}$, for $z \in \{0,\dots,n-1\}$, that is, after ℓ iterations of Algorithm 1 every qubit is correctly placed.

Lemma 2 and its proof are very similar to Lemma 1 and its proof, respectively. Nevertheless, we report the proof of Lemma 2 for clarity and completeness.

Proof We prove the lemma by induction on the number of iterations i. For $i=1$, it is immediate to see that for every $j \in \{0,\dots,\frac{n}{2^ik}-1\}$, every $y \in \{2^ijk,\dots,(2^ij+1)k-1\}$, and every $x = y+2^{1-1}k = y+k \mod n$, the execution of $\mathtt{Swap}\left|q_x^0,q_y^0\right\rangle$ yields $\left|q_x^1,q_y^1\right\rangle$ where $q_y^1 = q_x^0 = q^0_{y+k} = q^0_{y-(1-2^i)k \mod n}$ and $q_x^1 = q_y^0 = q^0_{x-k \mod n}$. Let $1 \le i < \ell$, assume the claim true for $|q^0\rangle,\dots,|q^i\rangle$, and let us prove it is true for $|q^{i+1}\rangle$. By inductive hypothesis, for $j \in \{0,\dots,\frac{n}{2^ik}-1\}$, it holds

$$q_x^i = q^0_{x-k \mod n}, \quad \text{for } x \in \{(2^ij+1)k,\dots,2^i(j+1)k-1\}, \text{ and}$$
$$[0.2cm] \quad q_y^i = q^0_{y-(1-2^i)k \mod n}, \quad \text{for } y \in \{2^ijk,\dots,(2^ij+1)k-1\}.$$

It does not require much effort to check that

$$\{2^{i+1}jk,\dots,(2^{i+1}j+1)k-1 \mid 0 \le j \le \frac{n}{2^{i+1}k}-1\}$$
$$= \{2^ijk,\dots,(2^ij+1)k-1 \mid 0 \le j \le \frac{n}{2^ik}-2 \text{ such that } j \text{ is even}\},$$

and that

$$\{(2^{i+1}j+2^i)k,\dots,(2^{i+1}j+1+2^i)k-1 \mid 0 \le j \le \frac{n}{2^{i+1}k}-1\}$$
$$= \{2^ijk,\dots,(2^ij+1)k-1 \mid 0 \le j \le \frac{n}{2^ik}-1 \text{ such that } j \text{ is odd}\}.$$

It follows that all the qubits involved in a swap during the $(i+1)$st iteration of the algorithm are of the form $q_y^i = q^0_{y-(1-2^i)k \mod n}$. Therefore, $q_z^{i+1} = q_z^i$ for $z \in \{(2^ij+1)k,\dots,2^i(j+1)k-1\}$ and $j \in \{0,\dots,\frac{n}{2^ik}-1\}$. Moreover, during the $(i+1)$th iteration, the algorithm executes the operation $\mathtt{Swap}\left|q_x^i,q_y^i\right\rangle$ for $x \in \{2^{i+1}jk,\dots,(2^{i+1}j+1)k-1\}$ and $y = x+2^ik \mod n$, where $j \in \{0,\dots,\frac{n}{2^{i+1}k}-1\}$; each of these swaps results in $\left|q_x^{i+1},q_y^{i+1}\right\rangle$

$$q_x^{i+1} = q_y^i = q^i_{x+2^ik \mod n} = q^0_{x-(1-2^i)k+2^ik \mod n} = q^0_{x-(1-2^{i+1})k \mod n}, \text{ and}$$
$$q_y^{i+1} = q^{i+1}_{x+2^ik \mod n} = q_x^i = q^0_{x-(1-2^i)k \mod n} = q^0_{x+2^ik-k \mod n} = q^0_{y-k \mod n}.$$

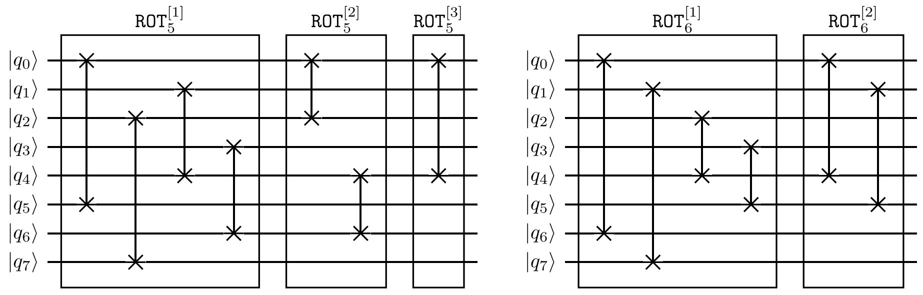

Fig. 6 The application of the circular shift operator for a register of 8 qubits in which a rotation of 5 and 6 positions is performed, respectively

To complete the proof it is enough to observe that

$$\begin{aligned}&\{(2^ij+1)k,\dots,2^i(j+1)k-1 \mid 0\le j\le \frac{n}{2^ik}-1\}\\ \cup&\{(2^{i+1}j+2^i)k,\dots,(2^{i+1}j+1+2^i)k-1 \mid 0\le j\le \frac{n}{2^{i+1}k}-1\}\\ =&\{(2^{i+1}j+1)k,\dots,2^{i+1}(j+1)k-1 \mid 0\le j\le \frac{n}{2^{i+1}k}-1\}.\end{aligned}$$

Indeed, the previous equality implies that

$$q^{i+1}_x = q^0_{x-k \mod n}, \text{ for } x\in\{(2^{i+1}j+1)k, \dots, 2^{i+1}(j+1)k-1\}, \text{ and } j\in\{0,\dots,\frac{n}{2^{i+1}k}-1\}.$$

Finally, observe that for $i=\ell$, the variable j assumes only the value 0; furthermore, for $x\in\{0,\dots,k-1\}$ it holds

$$q^\ell_x = q^0_{x-(1-2^\ell)k \mod n} = q^0_{x-k \mod n}.$$

This completes our proof. □

Proof of Theorem 1 By Lemma 2, after ℓ iterations of Algorithm 2 every qubit is correctly placed, that is, rightward cyclically shifted by k positions. □

The Controlled Circular Shift Gate for Character Strings

In this section, we define a *controlled cyclic shift gate* that rotates a quantum register encoding a string of n characters by a certain number of characters depending on an input value k, which can range from 0 to $n-1$, so as to obtain any rotation of the (encoding of the) string. The algorithm constructing such a circuit takes as only input the length of the string and the cardinality of the alphabet to whom the characters belong. Before deepening into the construction of the circuit, let us set some encoding details.

Encoding and Initialization

Let q be a string of n characters over an alphabet Σ. We can make the length of q equaling power of 2, by picking p as $\min\{z\in\mathbb{N} : n\le 2^p\}$, extend Σ with a fresh new symbol \$, and chaining the last character of q to 2^p-n copies of \$. Abusing the notation, we rename q the new string, Σ the extended alphabet, and let σ be the cardinality of such a new Σ.

Each character of q is encoded by $\lceil\log\sigma\rceil$ qubits. We denote $|q_i\rangle = \bigotimes_{s=0}^{\lceil\log\sigma\rceil-1}\left|q^s_i\right\rangle$ the quantum register encoding the ith character of q; in such a notation, $\left|q^s_i\right\rangle$ represents the sth digit of the binary encoding of the ith character of q. The overall string q is then encoded by the quantum register

$$|q\rangle = \bigotimes_{i=0}^{n-1}|q_i\rangle = \bigotimes_{i=0}^{n-1}\bigotimes_{s=0}^{\lceil\log\sigma\rceil-1}\left|q^s_i\right\rangle.$$

Besides of the register $|q\rangle$, the algorithm constructing the quantum circuit, initializes the register $|k\rangle$ which will encode the magnitude of the circular rotation. Specifically, we set $|k\rangle = \bigotimes_{j=0}^{\log(n)-1}\left|k_j\right\rangle$, and each qubit k_j will store the value of the jth least significant digit of the binary encoding of k. The circuit also needs $\frac{n\log\sigma}{2}$ ancillæqubits all of them to be initialized on the state zero.

The Circuit

We denote the controlled cyclic shift gate by CROT, and its action on the register $|q\rangle$, controlled by the register $|k\rangle$ is formalized by

$$\begin{aligned}&\mathtt{CROT}\,(|k\rangle\otimes\left|q_0,q_1,\cdots,q_{n-1}\right\rangle)\\ &=|k\rangle\otimes\left|q_{n-k},q_{n-k+1}\cdots,q_{n-k-1}\right\rangle\\ &=|k\rangle\otimes\mathtt{ROT}_{\,k\cdot\log\sigma}|q\rangle.\end{aligned}$$

The circuit operates by applying to the register $|q\rangle$ the rotation operator $\mathtt{ROT}_{\,2^j\cdot\log\sigma}$ controlled by the qubit $\left|k_j\right\rangle$, for any value of j, such that $0\le j<\log(n)$. Since n is a power of 2, the quotient of $n\log\sigma$ and $2^j\log\sigma$ is a power of 2, for every $0\le j<\log(n)$; therefore, by Remark 1, this can be obtained by a quantum gate for $\mathtt{ROT}_{\,2^j\log\sigma}$ - as explained is Sect. An Algorithm for $k=2^m$—controlled by the qubit $\left|k_j\right\rangle$. Note that the rotation operator $\mathtt{ROT}_{\,2^j\log\sigma}$ consists of a sequence of at most $\log(n)$ time-layers, each containing at most, $\frac{n\log\sigma}{2}$ parallel gates, controlled by the same qubit. Thus, to keep the parallelism of the cyclic shift operator in its controlled version, we employ $\log(n)\frac{n\log\sigma}{2}$ ancillæ qubits for the application of all parallel operators controlled by the same qubit, that is, for each $0\le j<\log(n)$, the controlled circular shift by 2^j positions we need $\frac{n\log\sigma}{2}$ ancillæqubit. We use the same ancillæto fan-in the control in each controlled rotation $\mathtt{ROT}_{\,2^j\log\sigma}$. In fact, the controlled rotation $\mathtt{ROT}_{\,2^j\log\sigma}$ are applied one after the other, by making j range from 0 to $\log(n)-1$; therefore, after the application of the controlled $\mathtt{ROT}_{\,2^j\log\sigma}$ operator, we clean-up the ancillæ, setting them ready for the upcoming controlled $\mathtt{ROT}_{\,2^{j+1}\log\sigma}$ operator. The controlled cyclic shift by 2^j positions requires $n\log\sigma = \mathcal{O}(\log(n))$ time-steps to fan the information on the control qubit out to the ancillæplus $\mathcal{O}(\log(n))$ to perform the cyclic rotation. Therefore, the overall controlled circular shift operator CROT over a register of size n and dependent on the value of an input quantum register of size $\log(n)$ can be implemented by a quantum circuit with depth equal to $\mathcal{O}(\log^2(n))$. Figure 7 contains an illustration of such a circuit.

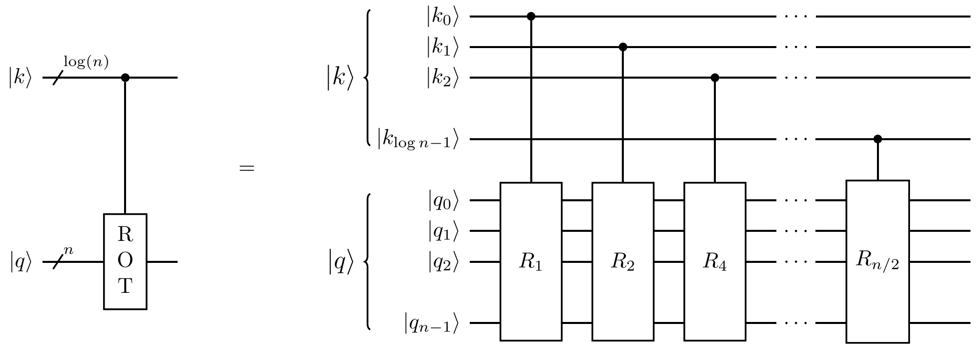

Fig. 7 A circular shift operator on a $n \log \sigma$-qubit register $|q\rangle$ controlled by a $\log(n)$-qubit register $|k\rangle$. The circuit makes use of $\log(n \log \sigma)$ ancillæ qubits which are not shown in this graphical representation. On the left of the figure, we show the succinct graphical representation used below in this paper

Discussion and Conclusions

We have presented a quantum algorithm that performs the rightward circular shift of a quantum register of size $n = 2^p$ by k positions. As we already discussed, the circular shift operator is a staple ingredient of many quantum recipes. For example, in the framework of quantum text processing, it is employed to get all possible portions of a certain fixed length of a text in the [13, 14]. Whereas it was already known that such a gate can be implemented in at most $\log(n)$ steps, a systematic way to build it was missed, and this motivated our work.

We have also presented an application of the quantum algorithm for obtaining the circular rotations of a quantum encoding of a character string by any number of characters. Such a quantum gate has several potential applications to string matching e text processing problems.

The following link, https://colab.research.google.com/drive/1bbRjsYl7UCVT6P4gNwJulARfd64L1RCT?usp=sharing#scrollTo=be1xNHxtjFqp, leads to a public Google Colab tutorial on implementing the quantum algorithm for the cyclic string matching problem, as presented in [24]. The tutorial includes an implementation of the Controlled Cyclic Rotation Operator, which serves as a subroutine.

We assumed to work with *registers of qubits*, that is, the register is made of 2-dimensional quantum systems. In the quantum computation landscape, this is however not always the case. For example, *qutrits* are quantum systems of dimension 3 that have attracted some interest in quantum cryptography [25, 26].

We point out that the implementation of our circuits for cyclic rotations on current or near-future quantum hardware would significantly suffer from the effects of noise. The logarithmic depth of our circuits results in a computation time that, even for short instances, exceeds the coherence time of present and near-future quantum machines. Additionally, the linear size of these circuits amplifies the error due to the current and near-term low gate fidelity. Therefore, future research on quantum implementations of cyclic rotations should focus on designing uniform families of quantum circuits with depths smaller than logarithmic and sizes smaller than linear. Achieving such a result would not only have practical applications but also be theoretically interesting and expected.

Acknowledgements We would like to express our gratitude to Simone Faro for participating in constructive discussions on the problem and for valuable advice. We would also like to thank Domenico Cantone for suggestions regarding the proof of some lemmas presented in this paper.

Funding Open access funding provided by Università degli Studi di Palermo within the CRUI-CARE Agreement. Arianna Pavone is supported by PNRR project ITSERR—Italian Strengthening of the ESFRI RI RESILIENCE. Caterina Viola is supported by the National Centre for HPC, Big Data and Quantum Computing, Project CN00000013, affiliated to Spoke 10, co-founded by the European Union—NextGenerationEU.

Declarations

Conflict of Interest The authors certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

1. Pavone A, Viola C. The quantum cyclic rotation gate. In: Castiglione, G., Sciortino, M. (eds.) Proceedings of the 24th Italian Conference on Theoretical Computer Science, Palermo, Italy, September 13-15, 2023. CEUR Workshop Proceedings, vol. 3587, pp.

206–218. CEUR-WS, 2023. https://ceur-ws.org/Vol-3587/4071.pdf

2. Grover LK. A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. STOC '96, pp. 212–219. ACM, New York, NY, USA 1996. https://doi.org/10.1145/237814.237866 .
3. Shor PW. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J Comp. 1997;26(5):1484–509. https://doi.org/10.1137/s0097539795293172.
4. Campbell ET, Khurana A, Montanaro A. Applying quantum algorithms to constraint satisfaction problems. Quantum 2018.
5. Viola C, Živný S. The Combined Basic LP and Affine IP Relaxation for Promise VCSPs on Infinite Domains. ACM Trans Algorithms. 2021;17(3):21–12123. https://doi.org/10.1145/3458041.
6. Buonaiuto G, Guarasci R, Minutolo A, Pietro GD, Esposito M. Quantum transfer learning for acceptability judgements. Quantum Mach Intell. 2024;6(1):13. https://doi.org/10.1007/S42484-024-00141-8.
7. Lao Y, Ait-Aider O. Rolling shutter homography and its applications. IEEE Trans Pattern Anal Mach Intell. 2021;43(8):2780–93. https://doi.org/10.1109/TPAMI.2020.2977644.
8. Musser DR. Introspective sorting and selection algorithms. Softw Pract Exp. 1997;27(8):983–93.
9. Weil R, Vinograd J. The cyclic helix and cyclic coil forms of polyoma viral dna. Proceedings of the National Academy of Sciences. 1963;50(4):730–8. https://doi.org/10.1073/pnas.50.4.730. https://www.pnas.org/doi/pdf/10.1073/pnas.50.4.730
10. Dulbecco R, Vogt M. Evidence for a ring structure of polyoma virus dna. Proceedings of the National Academy of Sciences. 1963;50(2):236–43. https://doi.org/10.1073/pnas.50.2.236. https://www.pnas.org/doi/pdf/10.1073/pnas.50.2.236
11. Thanbichler M, Wang S, Shapiro L. The bacterial nucleoid: A highly organized and dynamic structure. Journal of cellular biochemistry 2005;96:506–21. https://doi.org/10.1002/jcb.20519
12. Lisacek F. Algorithms on strings, trees and sequences: Dan gusfield. Comput Chem. 2000;24(1):135–7.
13. Niroula P, Nam Y. A quantum algorithm for string matching. npj Quantum Information 7, 2021:37 https://doi.org/10.1038/s41534-021-00369-3
14. Cantone D, Faro S, Pavone A. Quantum string matching unfolded and extended. In: Kutrib M, Meyer U, editors. Reversible Computation. Cham: Springer; 2023. p. 117–33.
15. Faro S, Pavone A, Viola C. Quantum path parallelism: A circuit-based approach to text searching. In: Chen, X., Li, B. (eds.) Theory and Applications of Models of Computation - 18th Annual Conference, TAMC 2024, Hong Kong, China, May 13-15, 2024, Proceedings. Lecture Notes in Computer Science, vol. 14637, pp. 247–259. Springer, 2024. https://doi.org/10.1007/978-981-97-2340-9_21 .
16. Faro S, Lecroq T. The exact online string matching problem: A review of the most recent results. ACM Comput Surv. 2013;45(2):13–11342. https://doi.org/10.1145/2431211.2431212.
17. Faro S, Marino FP, Pavone A. Efficient online string matching based on characters distance text sampling. Algorithmica. 2020;82(11):3390–412. https://doi.org/10.1007/S00453-020-00732-4.
18. Farhi E, Goldstone J, Gutmann S, Sipser M. Quantum Computation by Adiabatic Evolution 2000.
19. Witten E. Topological quantum field theory. Communications in Mathematical Physics. 1988;117(3):353–86.
20. Jozsa R. An introduction to measurement based quantum computation 2005.
21. Nielsen MA, Chuang, IL. Quantum Computation and Quantum Information (10th Anniversary Edition). Cambridge University Press, 2016. https://www.cambridge.org/de/academic/subjects/physics/quantum-physics-quantum-information-and-quantum-computation/quantum-computation-and-quantum-information-10th-anniversary-edition?format=HB
22. Wootters WK, Wootters WK, Zurek WH. A single quantum cannot be cloned. Nature. 1982;299:802–3.
23. Zulehner A, Wille R. In: Ulidowski, I., Lanese, I., Schultz, U.P., Ferreira, C. (eds.) Simulation and Design of Quantum Circuits, pp. 60–82. Springer, Cham 2020. https://doi.org/10.1007/978-3-030-47361-7_3 .
24. Pavone A, Viola C. A quantum circuit for the cyclic string matching problem. In: Holub, J., Zdárek, J. (eds.) Proceedings of the Prague Stringology Conference 2024. Czech Technical University in Prague, Czech Republic, ??? 2024. to appear
25. Lanyon BP, Weinhold TJ, Langford NK, O'Brien JL, Resch KJ, Gilchrist A, White AG. Manipulating biphotonic qutrits. Phys Rev Lett 2008;100:060504. https://doi.org/10.1103/PhysRevLett.100.060504
26. Smania M, Elhassan AM, Tavakoli A, Bourennane M. Experimental quantum multiparty communication protocols. npj Quantum Information 2016;2(1):16010. https://doi.org/10.1038/npjqi.2016.10