Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Towards optimal preparation of triangular quantum cycle shifts

Asimakis Kydros

akydros@csd.auth.gr

Aristotle University of Thessaloniki

Konstantinos Prousalis

Aristotle University of Thessaloniki

Nikos Konofaos

Aristotle University of Thessaloniki

Research Article

Keywords: Quantum computing, Quantum image processing, Cycle shift, Quantum circuit optimization, Algorithm analysis, Non-adjacent form

Posted Date: October 27th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11493605/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Towards optimal preparation of triangular quantum cycle shifts

Asimakis Kydros [a,*], Konstantinos Prousalis [a], Nikos Konofaos [a]

[a] *Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece*

Abstract

An algorithm for the optimal preparation of triangular quantum cycle shifts is presented. Through theoretical analysis, it is shown that any optimal sequence of powers of two that sum to the fixed integer shift step must guarantee the exhaustive application of four local exchange rules, leading to the non-adjacent form of that integer step. Based on this, it is shown that the optimal form exists in the search sub-space formed by just considering the two closest powers of two from a given signed integer remainder, and that it can be reached via exhaustive depth-first search but also by a simpler approach, nicknamed "Domino". Experimental comparisons between Domino and previously studied preparators for triangular quantum cycle shifts verify that the former always identifies the optimal sequence. Notably, the proposed approach achieves the same benefit in quantum cost as the previously studied dynamic programming algorithm, at more than 8 times the speed. Experiments on noisy backends further imply that Domino becomes asymptotically superior in noise resistance to the other methods as the register width grows, retaining a roughly 9% fidelity lead. These results establish Domino as an appealing choice for maximum-performance triangular quantum cycle shift preparation.

Keywords: Quantum computing, Quantum image processing, Cycle shift, Quantum circuit optimization, Algorithm analysis, Non-adjacent form

1. Introduction

The ever-evolving paradigm of Quantum Computing (QC) highlights avenues of improvement for certain types of established classical operations, by employing the exotic properties of quantum mechanical systems, such as superposition and entanglement. Such an operation, of fundamental importance in image processing and cryptography, is the Cycle Shift Transform (CST). Defining a quantum version of this primitive holds significant utility, as it can allow the simultaneous parallel cyclic translation of a vast number of data encoded in a superposition state.

The literature has explored multiple approaches for implementing such a quantum operator. Notable definitions include techniques based on the `SWAP` quantum operator [1–3], quantum adder-based schemes [4–6], and, relevant to this work, the Toffoli-based, "triangular" construction [7–9]. Each of them inhabit a specific niche in terms of effect, time complexity and resource requirements. Research has also studied multiple areas of application, with prominent scopes currently being quantum image encryption [10–12], steganography [13], filtering [14–16], watermarking [17, 18], as well as feature extraction [19, 20], among others.

With regard to the triangular scheme, prior work by the authors has studied the operator's extension to multidimensional structures [9], as well as its more efficient construction, through greedy and Dynamic Programming (DP)-based algorithmic processes [21]. Triangular quantum CST construction constitutes an interesting mathematical problem, as a general integer shift step must be decomposed into a proper sequence of powers of two, where the gate count of each constituent grows as the implemented exponent drops. Ref. [21] had left unanswered the question of an optimal construction algorithm, which the present work aims to address. Concretely, this manuscript analytically identifies the essence of the optimal power sequence for a given integer shift step and then probes a procedure for generating it efficiently as a cascade of Multi-controlled-`NOT` (MCX) operators.

The rest of this work is structured as follows. Section 2 offers a summary of key basic concepts regarding QC and the triangular CST to ensure the manuscript's completeness. Section 3 then studies

*Corresponding author
Email address: `akydros@csd.auth.gr` (Asimakis Kydros)

the optimal sequence for any given integer theoretically by identifying all possible power substitutions, and lays out a pathway for computing it procedurally. Section 4 proposes such an algorithm, `Domino`, proves its correctness and optimality, and estimates its computational overhead. Afterwards, Section 5 validates `Domino` experimentally, comparing it with similar previously studied techniques. Section 6 concludes the text by summarizing the findings and proposing future axes of further improvement. Appendix A further enhances the completeness of the manuscript by including prior CST constructors as pseudocode schematics.

2. Preliminaries

2.1. Quantum computing

This section briefly overviews some basic concepts of QC such that this work remains self-contained. The interested reader can find a more thorough introduction in Ref. [22].

QC is a model of computation that manipulates information qualitatively through quantum mechanical phenomena, in order to achieve significant performance improvement in problem solving. Pertinent examples of quantum algorithms solving important computational problems vastly better than the classical alternative on paper include Shor's factorization algorithm [23] and Grover search [24, 25].

In QC, classical computational tools such as bits and logic gates are replaced with qubits and quantum operators that act upon quantum systems in-place. The theoretical description of the general state of a qubit, known as the *superposition*, is shown in Eq. (1).

$$|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,,\ \alpha,\beta \in \mathbb{C} : \alpha^*\alpha + \beta^*\beta = 1 \tag{1}$$

Contrary to traditional intuition, the basis states $|0\rangle, |1\rangle$ describe eigenvectors, and the qubit is *not* simultaneously in both of them; rather, in the general case, the qubit exists in an *intermediate* state, described fully by $|q\rangle$, that is projected onto $|0\rangle$ by α and onto $|1\rangle$ by β. These complex numbers are known as the *probability amplitudes* of the system. Multiple qubits can be grouped together to form a *register*, which, as a whole, is also a quantum system describable by a superposition, e.g.:

$$|reg\rangle = |q_2\rangle \otimes |q_1\rangle \otimes |q_0\rangle = \begin{bmatrix}\alpha_2\\ \beta_2\end{bmatrix} \otimes \begin{bmatrix}\alpha_1\\ \beta_1\end{bmatrix} \otimes \begin{bmatrix}\alpha_0\\ \beta_0\end{bmatrix} = \sum_{i=0}^{7} \gamma_i\,|i\rangle\,,\ \forall i\gamma_i \in \mathbb{C} : \sum_i \gamma_i^*\gamma_i = 1 \tag{2}$$

The grouping is described by the tensor product $\otimes$ between the tensor descriptions of the participant qubits. The number of participant qubits in a given register is known as that register's *width*. In Eq. (1), (2) and any other quantum system, the magnitude of the amplitudes must add to exactly 1.

Information manipulation in the quantum regime happens through quantum operators, also called *gates*. All valid gates are special unitary operators U, satisfying the conditions shown in Eq. (3).

$$U :\ U^\dagger U = UU^\dagger = I,\ \ U\left(\sum_i \gamma_i\,|i\rangle\right) = \sum_i \gamma_i U\,|i\rangle \tag{3}$$

The above conditions underline that QC is a kind of reversible computation; all valid gates are $1:1$ permutations of the input. Many gates exist; of fundamental importance to the field and this work is the Controlled-`NOT` (CNOT) gate, and its generalization, MCX. These gates affect multiple qubits, one being the *target* and the rest being the *controls*. The target qubit feels the effect of the quantum `NOT` gate, so long as the controls "find" a satisfactory state. More concretely, the controls lay out the template of the basis states that will receive the effect of `NOT`, and ignores the rest. Controls usually condition their qubits on state $|1\rangle$; such controls are symbolized by $\bullet$. The target gates are drawn as $\oplus$. Examples are given in Fig. 1. MCX gates are important and versatile for multiple reasons: they have a simple and relatively intuitive function; they can implement crucial logic functions such as `XOR` and `NAND`, enabling universal classical computation; and they effect *entanglement*.

Entanglement is a unique quantum effect with no classical analogue and the catalyst for QC's performative potential. Induced when manipulating multiple qubits with a single gate, the state space of those participant qubits is acted upon simultaneously, and they become strongly correlated. The state of one participant qubit is now no longer fully describable without the states of the rest, and a change in one affects the rest. This is especially significant for computation, since it allows for massive

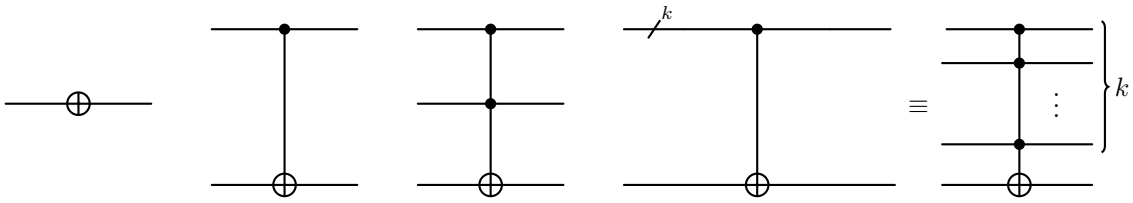


Figure 1: Example instances of MCX gates. Graphically, controls are drawn as • and targets as ⊕. Horizontal lines represent qubits. From left to right: the `NOT`, which generalizes into the MCX family as the member with no controls; the CNOT, the most widely used member; the controlled-controlled-`NOT`, usually called *Toffoli*, which is also a name for the entire MCX family; finally, a general MCX. When visualized, groupings of multiple qubits can be shown in a single line prepended with their number, here k. If a gate or control exists in such a qubit "bus", it is implied that the gate or control is repeated for each constituent qubit.

parallelism, conditional partition of the search space, direct and indirect manipulation of specific states, and other interesting and useful properties.

MCX gates are linked to a decomposition process. This is because, in general, quantum devices cannot consider every possible MCX iteration as a natively supported gate. Multiple such decompositions exist [22, 26–30]. This work attempts to unearth an analytical optimality that is largely independent of the decomposition, however the choice matters for the complete description of time and space complexities. Hereafter, the cost of a MCX gate that supports m controls is described by $\mathcal{M}(m)$. This function is expected to increase with m, as verified by all real-world decompositions. This cost refers to number of elementary quantum gates; their type depends on the decomposition model, although in general CNOT is assumed.

2.2. The triangular quantum cycle shift operation

In the quantum realm, the CST can be defined mathematically as the unitary operator acting on a register of width n (hereafter called the *host register*) and transforming its superposition in the way shown in Eq. (4).

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \gamma_i |i\rangle \xrightarrow{CST_s} |\psi'\rangle = \sum_{i=0}^{2^n-1} \gamma_i |(i+s) \bmod 2^n\rangle \tag{4}$$

The *shift step* s may be any integer, although by modulo rules all steps are effectively scaled down to the range $(-2^n, 2^n)$.

The literature has studied multiple ways of assuming this operation. Of central interest to this work is the approach of the *triangular* CST, a unitary operator consisting of only MCX gates, previously studied in Refs. [8, 9, 21]. The triangular method, informally named after the shape of its MCX implementation, is a simple architecture that includes a few useful characteristics:

- Apart from potential ancillas that depend on the MCX decomposition, the scheme requires no extra qubits.
- Its architecture trivially allows for any non-negative power of two step k to be implemented, by just starting the MCX cascade (the so-called *triangle*) k qubits later in the host.
- As a consequence of the above, higher powers of two lead to smaller triangles, which in turn means less gates; the quantum cost $w(\cdot)$ (number of elementary gates) of a triangle, therefore, decreases with the power k because it increases with its span $n - k$.
- Reflecting a triangle vertically, in other words reversing the order of the MCX sequence, flips the sign of the step, inducing no extra cost.

An example showcasing these qualities is offered in Fig. 2.

In order to define the triangular CST for integers other than signed powers of two, the given integer step must thus be decomposed into powers of two. The total cost of the step then would be the sum of the costs of each constituent power. The aforementioned qualities inspire the search for an efficient decomposition that uses as few powers of two as possible. Prior works have studied simple and more sophisticated techniques to reach such a sequence: the first procedure to appear decomposed the step according to its binary representation [8]. Due to the algorithm selecting the largest power of two

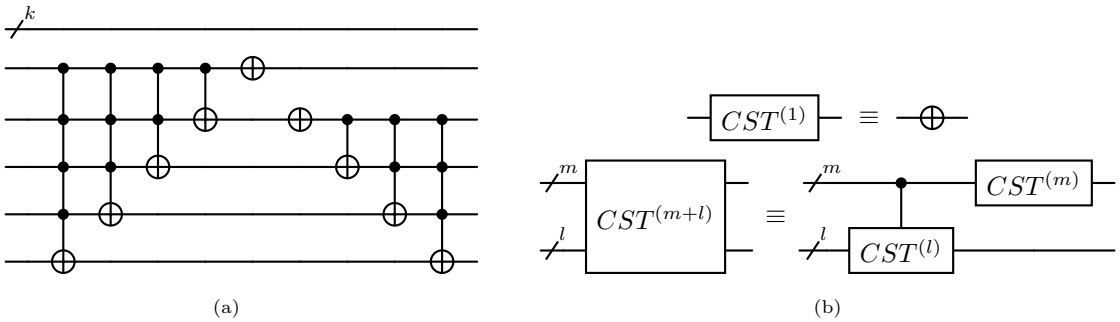


Figure 2: a An example implementation of a triangular CST. The circuit contains two triangles, therefore the total step is not a power of two. The first triangle starts k qubits later on the register, therefore it executes the sub-step $+2^k$. The order of its MCX gates signify positivity. The next triangle starts one qubit yet later, and the order is inverted; it executes the sub-step -2^{k+1}. Overall, the operation shifts the basis states by $-2^{k+1} + 2^k$ (mod the size of the host register's Hilbert space). b The inductive recipe of a CST triangle. Note that $m, l \geq 0$ and the superscript $^{(\cdot)}$ describes the qubit span of the triangle, not the step it performs. The triangle on a single qubit is just the `NOT` gate. Larger triangles group smaller ones via controls.

smaller than the intermediate steps, it is dubbed `MaxSmallest`. Recent work has introduced another two algorithms: a greedy process that selects the nearest power of two by absolute distance, named `Nearest`; and a DP approach that assumes the entire search space of nearest powers of two from a given step remainder and performs Depth-First Search (DFS) to find the cheapest combination, labeled `DP-DFS` [21]. Algorithmic schematics for these processes can be found in Appendix A. The latter two have been shown to be significantly better in gate cost than the former, although a provably optimal process has been missing.

3. Local exchange rules and the optimal sequence

In the context of triangular CST, given two powers, say $2^a, 2^b$, with no loss of generality respecting $a \geq b \geq 0$, the two unique local interactions between them are $\pm(2^a + 2^b)$ and $\pm(2^a - 2^b)$. Let us assume the positive case, since the negative one is reachable by simple sign flips. It is straightforward to show the general expansion of these two interactions:

$$2^a - 2^b = 2^b \left(2^{a-b} - 1\right) = 2^b \left(\sum_{i=0}^{a-b-1} 2^i \right) \implies 2^a - 2^b = \sum_{i=b}^{a-1} 2^i \tag{5}$$

$$(5) \implies -2^{a+1} + 2^a - 2^b = -2^{a+1} + \sum_{i=b}^{a-1} 2^i \implies 2^a + 2^b = 2^{a+1} - \sum_{i=b}^{a-1} 2^i \tag{6}$$

It follows that some instances of those equations describe substitutions that reduce the cost, by replacing a group of powers with another, more efficient one, that preserves the sum. The decreasing quality of the cost w on the power is the catalyst that allows one to reduce Eq. (5) and (6) into just a few local exchange rules, since useful rules must not increase the cost. The specific structure of the problem allows for an exhaustive check of all cases. They are listed below, slightly abusing notation by assuming that $\sum_{i=a}^{a-1} f(i) = 0$, for every $a \geq 0$ and for any transformation f.

Case $a = b \overset{(5)}{\implies} 2^a - 2^a = \sum_{i=a}^{a-1} 2^i \implies \boxed{2^a - 2^a = 0}$

The Left-Hand Side (LHS) has total cost $w(a) + w(a)$, while the Right-Hand Side (RHS) has cost $w(0) = 0$. The RHS is cheaper therefore this case constitutes an exchange rule (**R1**).

Case $a = b \overset{(6)}{\implies} 2^a + 2^a = 2^{a+1} - \sum_{i=a}^{a-1} 2^i \implies \boxed{2^a + 2^a = 2^{a+1}}$

The LHS has total cost $w(a) + w(a)$, while the RHS has cost $w(a+1)$. Since $a + 1 > a$, it holds that $w(a+1) < w(a) < 2w(a)$, therefore the RHS is cheaper. As such, this case constitutes an exchange rule (**R2**).

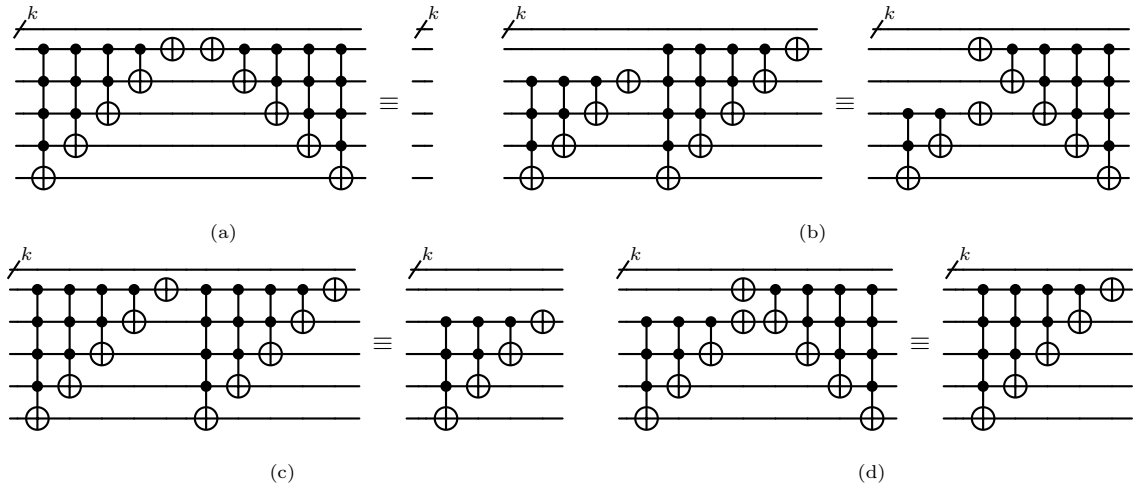


Figure 3: Local exchange rules expressed as triangular quantum cycle shift circuit instances. From a to d, it goes **R1**, **R4**, **R2**, **R3**, ordered as such to minimize figure space. Notice how each replacement strictly drops the cost on a gate level.

Case $a = b + 1 \overset{(5)}{\implies} 2^{b+1} - 2^b = \sum_{i=b}^{b} 2^i \implies \boxed{2^{b+1} - 2^b = 2^b}$

The LHS has total cost $w(b) + w(b+1)$, while the RHS has cost $w(b)$. The RHS is cheaper therefore this case constitutes an exchange rule (**R3**).

Case $a = b + 1 \overset{(6)}{\implies} 2^{b+1} + 2^b = 2^{b+2} - \sum_{i=b}^{b} 2^i \implies \boxed{2^{b+1} + 2^b = 2^{b+2} - 2^b}$

The LHS has total cost $w(b) + w(b+1)$, while the RHS has total cost $w(b) + w(b+2)$. Since $b+2 > b+1$, it holds that $w(b+2) < w(b+1) \implies w(b) + w(b+2) < w(b) + w(b+1)$, therefore the RHS is cheaper. As such, this case constitutes an exchange rule (**R4**).

Case $a > b + 1 \overset{(5)}{\implies} 2^a - 2^b = 2^b + \cdots + 2^{a-1}$

The LHS has total cost $w(a) + w(b)$. The RHS has *at least* cost $w(b) + w(a-1)$. Since $a > a-1$, it holds that $w(a) < w(a-1) \implies w(b) + w(a) < w(b) + w(a-1)$, therefore the RHS is more expensive even in the best case scenario. Thus, this instance does not constitute an exchange rule.

Case $a > b + 1 \overset{(6)}{\implies} 2^a + 2^b = 2^{a+1} - 2^b - \cdots - 2^{a-1}$

Once again, the LHS has total cost $w(a) + w(b)$, while the RHS has at least cost $w(b) + w(a-1) + w(a+1)$. Following similar reasoning as in the above rule, the RHS is always more expensive, therefore this case does not constitute an exchange rule.

Once again, all above cases are symmetric for the negative cases ($-2^a - 2^b$, $-2^a + 2^b$). This is easily verifiable so it is skipped here. The four exchange rules are visualized as circuit cases in Fig. 3.

The above rules correspond exactly with the ones shown in [31]. This means that their continuous application into any power of two sequence will drive the representation towards the Non-Adjacent Form (NAF) of the described integer. As it turns out, in the context of triangular CST, the NAF constitutes the optimal sequence in terms of sub-shift triangles. The above are proven in Theorem 1.

Theorem 1. For any step $s \in \mathbb{Z}$, expunging all invocations of rules **R1**-**R4** in its power-of-two representation turns it into its NAF. In the context of triangular CSTs, the NAF is the optimal sequence of s.

Proof. Rules **R1** and **R2** eliminate duplicate powers, ensuring that all powers of two appear at most once. Rules **R3** and **R4** target additions and subtractions of adjacent powers: **R3** completely eliminates the larger neighbor, while **R4** increments it, breaking the adjacency. All rules maintain the total sum. Therefore, the power sequence that does not invoke any of them is a sequence that contains each power of two at most once, sums to s (under modulus) and exhibits at least one non-utilized power between any utilized pair. Therefore, this sequence is a NAF of s. This description is CST-optimal

in terms of amount of sub-shift triangles and triangle spans because the NAF of any integer is unique and any non-NAF sequence invokes at least one of the aforementioned substitution rules, therefore it would not be optimal [31]. □

This significant detail can also be leveraged to show that the previously studied algorithm, `DP-DFS`, reaches the globally optimal sequence. As studied in Ref. [21], `DP-DFS` performs DFS on the search space formed by every combination of applying $hi(r)$, $lo(r)$ from a given shift remainder r, minimizing for total CST cost. It is noted that $hi(r)$ indicates the smallest power of two larger than r, while $lo(r)$ the largest one at most equal to it. The optimality of its outcome is proven in Theorem 2.

Lemma 1. The NAF of a shift step $s \in (-2^n, 2^n)$ to be performed on a host register of width n describes a hi, lo combination that starts from s and adds up to it.

Proof. Starting from the binary representation of s, continuously applying **R1**-**R4** reaches the NAF of s as per Theorem 1. In the NAF of s thus the largest power is at most one over the bit length of s due to **R4**. By definition this is the $hi(s)$, and if it does not exist in the NAF of s then the largest power must be $lo(s)$. Popping this largest power leaves a reduced NAF of a smaller number, and since the NAF of any integer is unique [31], the same logic applies. This recursively continues until the reduced integer becomes a power of two itself, where at that point it and its lo meet by definition. Thus all the popped powers form a hi, lo chain that starts from s and adds up to it. □

Theorem 2. The algorithm `DP-DFS` (Alg. 4 in [21]), minimizing the function $f(\mathcal{S}) = \sum_{k \in \mathcal{S}} n - k$ for each participant power k in sequence $\mathcal{S}$, produces the globally optimal sequence for any $s \in (-2^n, 2^n)$ to be performed on host register of width n.

Proof. It directly follows from Lemma 1 that the NAF of s belongs in the search space explored by `DP-DFS`. By Theorem 1, the NAF uniquely minimizes the total triangle span, which is also the objective $f(\mathcal{S})$. Therefore, $\min f$ over the defined search space is attained at the NAF, and as such `DP-DFS` will return it. □

4. An algorithm for the optimal sequence of triangles

As explained in the previous section, `DP-DFS` appears to generate the optimal group of triangles for a desired triangular quantum cycle shift. That being said, it thoroughly explores a significantly sized search space, and utilizes increased memory space to do so. Based on existing strategies for reaching the NAF of an integer, a better process is now within reach. In Ref. [31], the transducer automaton is presented, describing a process for generating the NAF by reading the binary representation right-to-left and essentially applying **R4** at every applicable case met. Inspired by this formulation, a construction algorithm for any triangular CST using optimally many subshift triangles is given in Alg. 1. Given that **R4** elimination introduces a higher power than those participating in the pair it considers, a power "carry" is generated after each reduction. This carry has to then be propagated to later bits, potentially toppling adjacent ace chains in a domino-like effect. Due to this effect, the algorithm is dubbed `Domino`.

Alg. 1 constitutes a valid process that always terminates yielding a correct solution. This fact is proven in the following theorem.

Theorem 3. Let $n \geq 0$ be the width of the host register. Alg. 1 always terminates, for every $s \in \mathbb{Z}$, and always produces a sequence of signed powers of two that add to exactly $\text{sign}(s)(s \bmod 2^n)$.

Proof. This proof considers the case $s' : |s'| < 2^n$, since every other case gets scaled down into this range at line 1. The algorithm will always terminate: the loop ends once $|s'|$ reaches zero. At every iteration, this magnitude gets right shifted once, so at every iteration, a bit is discarded. Line 11 adds to the magnitude, however this operation will increase the overall bit length of $|s'|$ by at most one. This is because, for the bit length to increase, the carry needs to domino all the way to the most significant bit; this can only happen if an ace chain connecting the current bit and the leading bit exists. Once this chain is eliminated by the propagation of the carry, it cannot occur naturally again. Therefore, zero will be reached after at most $\texttt{bitlen}(|s'|) + 1$ iterations. The produced sequence always sums up to s' because the application of **R4** does not alter the overall sum and the original bit sequence of s' describes a sequence that clearly sums to s'. □

Algorithm 1: `Domino`

Data: The host register $|\psi\rangle$, its width n, the desired step $s \in \mathbb{Z}$.
Result: The host register shifted by s modulo 2^n.

1 $s' \leftarrow |s| \bmod 2^n$
2 $\sigma \leftarrow \text{sign}(s)$
3 $\beta \leftarrow 0$
4 **while** $s' \neq 0$ **do**
5 $\sigma' \leftarrow \sigma$
6 $LSB \leftarrow s' \wedge 1$
7 Shift s' once to the right
8 **if** $LSB = 1$ **then**
9 **if** $s' \wedge 1 = 1$ **then**
 `// If true, R4 is invoked`
10 $\sigma' \leftarrow -\sigma'$
11 $s' \leftarrow s' + 1$ `// Propagates the higher power carry to later bits`
12 **end**
13 Perform a cycle shift of step $\sigma 2^\beta$ on $|\psi\rangle$;
14 **end**
15 $\beta \leftarrow \beta + 1$
16 **end**

The strategy of Alg. 1 naturally expunges all invocations of rules **R1**-**R4**, which, by Theorem 1, means that it returns the optimal solution. The following theorem proves this optimality.

Theorem 4. Alg. 1 always returns the optimal sequence for any given $s \in \mathbb{Z}$.

Proof. This proof considers the case $s' : |s'| < 2^n$, since every other case gets scaled down into this range at line 1. In the bit representation of s', every power participates at most once, and all of them share the global sign. This fact immediately disqualifies rules **R1**-**R3** from arising naturally. The iteration explicitly corrects any **R4** instances. Because this correction happens in ascending power order, once a power's turn has passed, it will never be altered again. From a given power position, later bits can be altered via the domino effect. In fact, given an ace chain of length 3 or more, correcting the rightmost bit invokes **R2**. Line 11 solves this predicament by adding 1 starting from the next bit position from the current power. This operation will eliminate any adjacent ace chains, ensuring that **R2** is satisfied. After the loop, there are thus no **R1**-**R4** invocations, which by Theorem 1 means that the sequence is in its optimal form. □

`Domino` avoids examining the entire search space and instead probes the optimal sequence directly, significantly decreasing the runtime. Experiments in Sect. 5 help visualize this benefit. The time complexity remains on par with previously studied algorithms for the given problem [8, 9, 21]. However, `Domino` scores a complexity victory in terms of ancillary bit space, dropping the required overhead by an order of magnitude compared to `DP-DFS`. These two points are proven below.

Theorem 5. For a given shift step $s \in \mathbb{Z}$, hosted on a quantum register of width $n \in \mathbb{Z}_+^*$, Alg. 1 requires $\mathcal{O}(\log(s')n\mathcal{M}(n-1))$ time steps and $\mathcal{O}(\log(s'))$ ancillary bit space, where $s' = |s| \bmod 2^n$ (worst-case $\mathcal{O}(n^2 M(n-1))$, $\mathcal{O}(n)$ respectively).

Proof. In Alg. 1, all lines excluding line 13 run for at most $\mathcal{O}(\log(s'))$ time steps. The loop will also perform $\mathcal{O}(\log(s'))$ iterations, as explained previously. Line 13 is known to cost at worst $\mathcal{O}(nM(n-1))$ [9]. Given that s' is capped by 2^n, the algorithm has expected time complexity $\mathcal{O}(\log(s')nM(n-1))$ and worst-case $\mathcal{O}(n^2 M(n-1))$. In terms of memory, every new variable inside the loop is a boolean and lives only for a single iteration. Line 2 similarly costs a single bit, and s' and β are capped in bits by $\mathcal{O}(\log(s'))$, therefore the total space cost is expected $\mathcal{O}(\log(s'))$ and worst case $\mathcal{O}(n)$. □

5. Experimental validation of Domino

In this section, an experimental probing of Alg. 1's performance is performed, compared to similar previously studied algorithms, those being `Max Smallest`, `Nearest` and `DP-DFS` [8, 19, 21]. For the

purposes of replicability, it is noted that all following experiments were performed on an AMD Ryzen 5 3600 6-core processor, running Windows 10 and utilizing Python 3.12.10. The available RAM was 16 GB. All scripts are offered freely and openly under the MIT license, see Declarations for the repository link.

Initial testing aimed to measure the output similarity between Alg. 1 and `DP-DFS`. Under the contest `domino_vs_dp_similarity.py`, the sequences of both algorithms were compared for absolute correspondence. For a given step on a given register width, a similarity "hit" was recorded if both sequences, sorted and expressed as triangle spans, contained exactly the same elements. Experimentation was scaled up to and including width $n = 50$. For n cases where the cardinality of the step domain did not exceed $cap = 10,000$, all possible steps were considered exhaustively. For larger cases, steps were sampled uniformly at random from each corresponding domain, capping the sampling number at the arbitrarily set limit of

$$\#samples = cap\left(1 + \frac{n - \lceil \log_2 cap \rceil}{2}\right) \tag{7}$$

Results of this comparison reveal that the hit percentage for each n is 100%. This means that the two algorithms are in perfect output agreement. This result supports Theorems 2 and 4, given that even a single deviation would serve to dismantle these arguments.

The next experiment tackled the runtime performance of `Domino`, specifically whether or not it actually is faster than `DP-DFS` or not and by how much. The comparison includes `Nearest` and `Max Smallest` as controls. It should be noted here that a version of `DP-DFS` was used that did not use a cost function f. Ref. [21] shows by experiment that such an f is not necessary, and its removal enhances fairness by shaving off redundant function calls. The experiment is conducted using the script `domino_vs_all_runtime_comp.py`. Given that runtimes for $0 \leq n \leq 50$ are on the order of microseconds for all considered algorithms, noticeable deviation was identified in the results between whole experiment runs. To combat this effect, the results reported here have been averaged over multiple experiment repeats[1]. Further, reports are offered with the garbage collector both enabled and disabled throughout. The outcome is visualized in Fig. 4.

From Fig. 4, it is clear that `Domino` is marginally slower compared to the simple methods `Nearest` and `Max Smallest`, but significantly faster than `DP-DFS`. In fact, it would appear that an 8 to 9 times speedup is approached. This outcome underlines the effectiveness of `Domino`; contrary to the former two methods, it achieves optimality, and contrary to the latter it maintains performance.

Another interesting experimental direction is capturing the quantum cost relative improvement between `Domino` and the simple methods. The script `domino_vs_simples_comp.py` drives this investigation. Using the same sampling method as previously, for all widths $0 \leq n \leq 50$, the total cost $\mathcal{C}^s_\alpha$ per algorithm α for a given step s was calculated by summing the generated sequence of triangle spans. Hits and strict hits were then recorded if the cost of `Domino` was at most equal and strictly less than the cost of the other algorithm, respectively. Relative improvement was captured as a percentage, following the formula $RI_s = 1 - \mathcal{C}^s_{\texttt{Domino}}/\mathcal{C}^s_{\alpha'}$, where α' is one of the simple algorithms. Figure 5 visualizes the improvement, as a graph of the median RI_s per width surrounded by the 25th-75th percentile range.

Relative improvement results paint an optimistic picture. "Soft" winrate (hit percentage) for both test cases yield a mean 100%, which indicates that the winrate for all widths is in fact 100%. Similarly to the first experiment, this fact points to the optimality of the proposed algorithm. Its efficiency is further attested by the mean strict winrates, where it becomes clear that `Max Smallest` is all but completely out-performed by `Domino`, marking significant and growth-trending improvement. The effect is similar, if less pronounced, for `Nearest`, where `Domino` manages to score an even lower cost more than 58% of the time, albeit by a modest amount that trends to consolidate between 0.5% and 1% relative improvement as n grows. In fact, both of these results, as well as the median curves and the percentile clouds, almost exactly reflect those reported in Ref. [21], when comparing `Nearest` to `Max Smallest` and `DP-DFS` to an ensemble of the simples, respectively. This upshot is another strong indication that the proposed approach Alg. 1 and `DP-DFS` share qualitative performance.

A similar comparison can be performed over the relative improvement to the depth of the circuits. Since `Domino` optimally reduces the required powers, and by extension the needed amount

[1]Concretely, 10 repeats were performed. Each repeat recorded the median runtime per register width for each algorithm. Then, the mean of that for each algorithm was graphed as the final result.

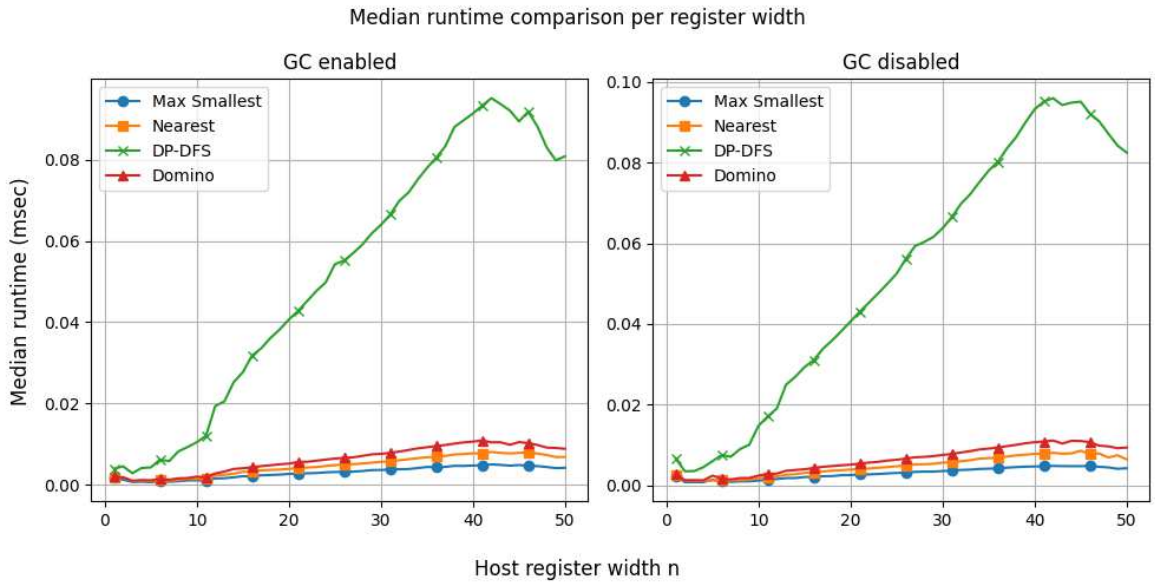


Figure 4: Runtime comparison between `Domino`, `DP-DFS`, `Nearest` and `Max Smallest` (averaged across iterations). Unsurprisingly, the latter two score the fastest, given their bare-bones implementation and functionality. It is similarly not unexpected to see `DP-DFS` be so slow compared to the rest, given the extensive search it performs. `Domino` sits between these two extremes, nearing a $8-9\times$ speedup compared to the DFS approach while being remarkably close to the controls. This result is encouraging, because `Domino` is indeed a slightly heavier process compared to `Nearest` and `Max Smallest`, but achieves optimality with a significant performance benefit than an exhaustive search. The activity of the garbage collector does not seem to alter the runtime in any noticeable manner. This would suggest that, barring unavoidable processor "noise" effects, the result shown is accurate.

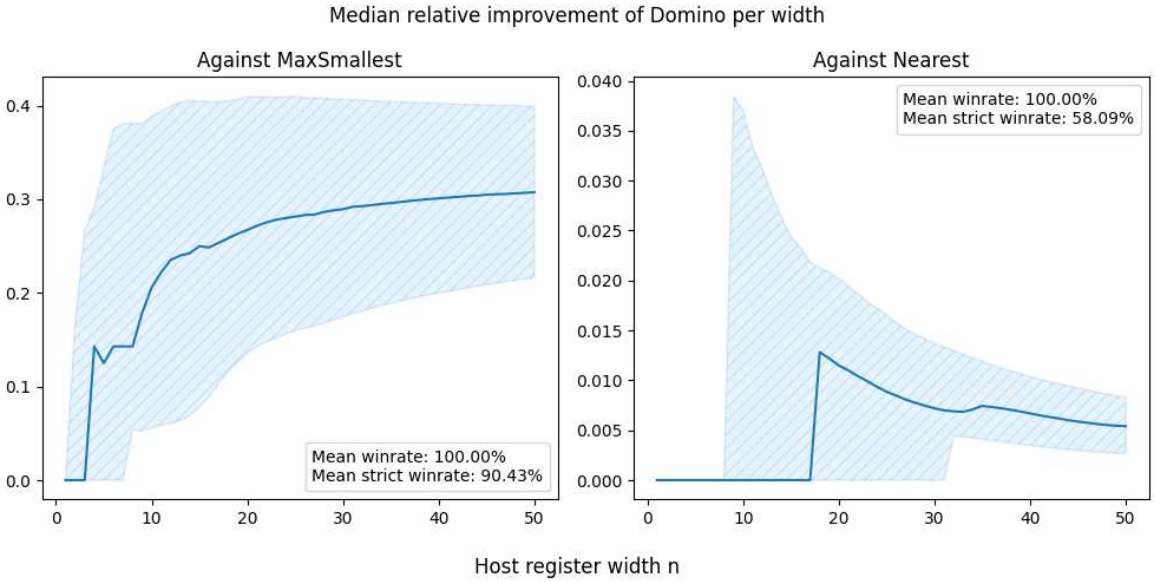


Figure 5: Relative improvement of `Domino` compared to `Max Smallest` and `Nearest`, respectively. The mean "soft" winrate again verifies claims of optimality, since it signifies that Alg. 1 is always at least as good as the other two methods. The mean strict winrate further enhances this point, as the overwhelming majority of tests scored better cost than `Max Smallest`, and more than 58% did so than `Nearest`. The shape and growth of the median curves and the percentile clouds almost exactly mirror the relative improvement results reported in Ref. [21]. This is yet another verification that Alg. 1 and `DP-DFS` are direct analogues, and a positive sign that they are both optimal.

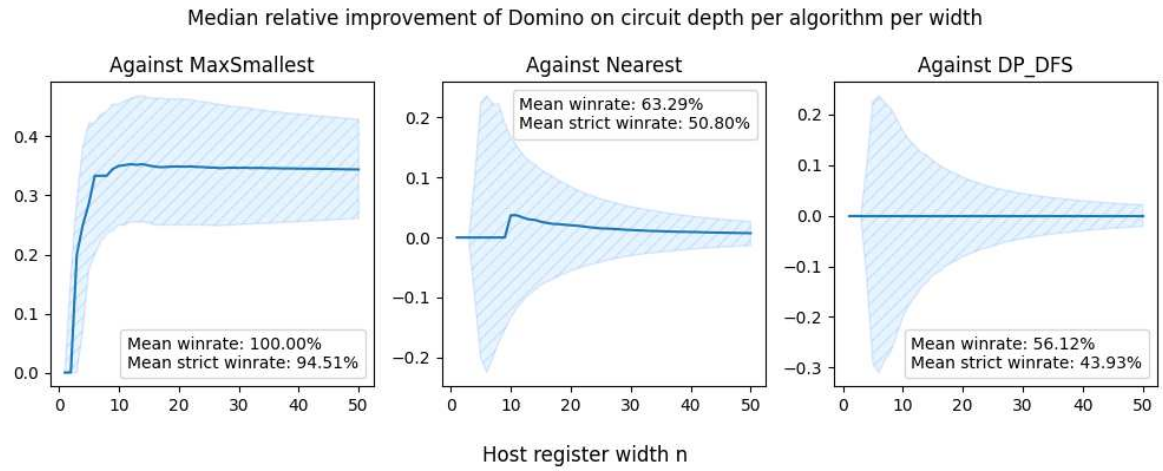


Figure 6: Relative improvement of `Domino` compared to the other algorithms over the circuit depth. Results clearly show that `Max Smallest` is completely out-performed by `Domino`, with the latter always matching the former's depth and improving upon it further by more than 94% on average. The median relative improvement quickly stagnates at around 35%, a significant gain. The effect does not translate to `Nearest` and `DP-DFS`. `Domino` creates shallower circuits than the former just a bit more than 50% of the time on average, while only 43% for the latter. Both show a median relative improvement of zero for most widths, with percentile clouds roughly symmetrical over the x-axis. This implies that both algorithms improve upon `Domino` around as much on half the cases as it does over them on the rest.

of CST triangles, the overall circuit depth across methods will differ in the general case. It should not be expected that `Domino` reigns supreme in this comparison axis too; the optimization principles do not take the depth into consideration at all. Still, some depth benefit can be expected to appear. The median relative improvement against the previous three methods is performed by the script `domino_vs_all_depth_comp.py` and the results are visualized in Fig. 6. One can notice that, against `Max Smallest`, `Domino` does show considerable superiority, dropping the depth by around 35% almost always and from early widths onwards. The same does not hold true for the other two methods, however. `Domino` manages to keep up more than 55% of the time on average in both cases, yielding improvement just barely above half the time against `Nearest` and $\sim 43\%$ of the time against `DP-DFS`. Interestingly, the median curves sit overwhelmingly at zero for most widths and the percentile clouds appear roughly symmetric over the x-axis in both comparisons. This implies that the relative improvement is bi-directional; `Nearest` and `DP-DFS` improve upon the depth yielded by `Domino` in half the cases as much as it does upon them in the rest of the cases.

Finally, the above results positively imply the enhancement of CST's robustness on noisy environments, when created by `Domino` compared to the rest. Both gate count and circuit depth are directly linked to noise accumulation in real quantum devices [22]. While `Domino` should not be expected to lead to high accuracies on a per device basis, there is sufficient reason to expect that it maintains higher accuracies in comparison to the other algorithms, over the same circuits and device-agnostically. To test this hypothesis, the experiment `fidelity_comp_2.py` leverages the Qiskit SDK in conjunction with its popular complementary backend Aer [32] to simulate noisy backends and compare the fidelity evolution of each algorithm.[2] Metadata for the 10 available Quantum Processing Unit (QPU) offered by IBM where accessed from Ref. [33], shown in Table 1. Three representative backends, one with weaker noise intensity, one with moderate and another with strong intensity, were selected from this collection: this was performed by segregating the listings into three groups by two-qubit gate median error and randomly sampling one member from each. Two-qubit gate error dominates on noise accumulation, and as such this metric was deemed an adequate separator. The sampling procedure for this reported instance decided upon an imitation of `IBM Pittsburg` for the weak case, `IBM Berlin` for the moderate case and `IBM Miami` for the strong case.

The experiment runs the same sampled CST steps, created by the different algorithms, on the selected simulated backends and computes the fidelity of the result. This is achieved by first performing a control execution on an ideal simulator and then comparing the density matrices produced by each.

[2]The experiment used Qiskit version 1.3 and Qiskit-Aer version 0.17.2.

Table 1: Selected QPU metadata among those collected from Ref. [33]. From left to right the columns read: device name, 1-qubit gate error, 2-qubit gate error, readout error, T1 thermal relaxation factor, T2 thermal relaxation factor, 1-qubit gate length, 2-qubit gate length. All metrics are median averages across the qubits of each real device.

Device	e_{1q}^G	e_{2q}^G	e^R	T1 (sec)	T2 (sec)	L_{1q} (sec)	L_{2q} (sec)
Pittsburg	1.891e-4	1.59e-3	4.272e-3	270.98e-6	306.27e-6	32e-9	88e-9
Berlin	1.644e-4	2.09e-3	2.106e-2	327.10e-6	251.24e-6	32e-9	68e-9
Miami	3.676e-4	3.07e-3	2.466e-2	319.50e-6	217.29e-6	32e-9	128e-9

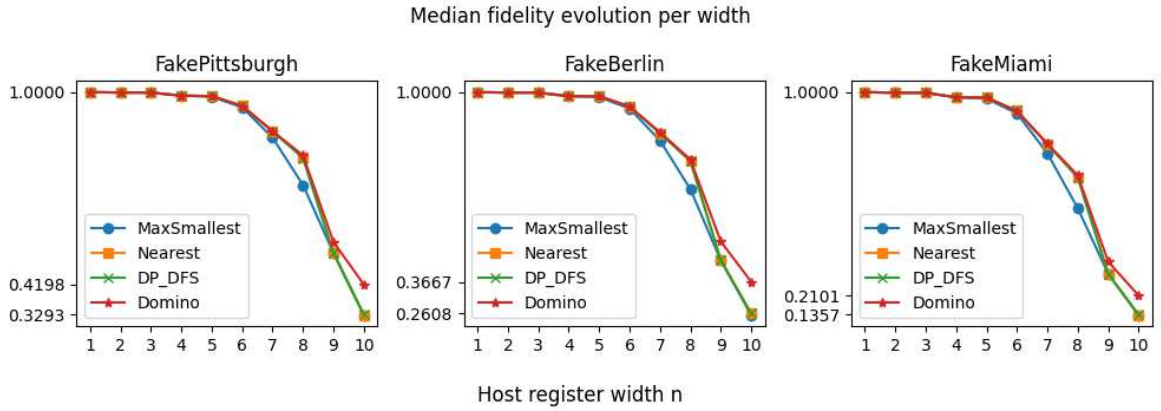


Figure 7: Median fidelity evolution per width, per considered algorithm, on each sampled noisy backend. All y-axes highlight the minimum values of `Domino` and the next highest performant. All graphs start at around 100% and gradually plummet, an overall effect expected for near-term hardware given the significant complexity of the executed circuits. As the host register grows to 10 qubits in size, `Domino` marks a roughly 9% increase in accuracy on `FakePittsburg`, 11% on `FakeBerlin` and 7% on `FakeMiami`.

Due to significant runtimes imposed by Qiskit, the test considers weights up until and including $n = 10$. For the same reason, the step sampler used is similar to Eq. (7) but sets the cap to 10 as well. The comparison results are shown in Fig. 7. It becomes clear that `Domino` becomes superior for higher widths, maintaining higher median fidelities as the register grows across all three backends. While all methods perform roughly identical for initial, tiny circuits, `Domino` maintains a mean fidelity lead as the register grows, with the gap approximating 9% for the low-intensity noise case, 11% for the moderate case and 7% for the higher noisy scenario. These values are compared against the maximum scored by the other methods for the given width. These results positively verify the ameliorative effect `Domino` has on the noise robustness of the CST operation, by sheer "brute force", as the improvement stems from the method's analytical optimality over the gate count and not over a targeted optimization of noisy operations for a given QPU.

6. Conclusions

This work has made three notable advancements. First, it has unearthed the nature of the optimal solution for the problem of preparing triangular quantum cycle shifts. This nature is the NAF description of the integer step number and it can arise from the exhaustive application of four local exchange rules. Secondly, armed with this knowledge, it has proven the optimality of the output received by the previously studied DFS method, and has laid out a strategy for conceiving similarly optimal algorithms. Finally, it has proposed a novel strategy, the `Domino` preparator, that performantly reaches the optimal solution. Experimental testing has verified that `Domino` and the DFS method always mirror each other in output, and that the former outperforms other previously studied methods: in speed, when compared to the DFS approach, reducing the runtime by more than a factor of 8, and in quantum cost, when compared to simpler methods. Experiments on noisy backends further show that the proposed procedure tends to retain significantly higher fidelity than the other approaches, as the width of the host register grows. These results establish `Domino` as an appealing and highly performant solution for the problem of triangular quantum CST preparation.

The analysis has also highlighted an additional avenue for improvement. The ordering of the optimal sequence has been, as of yet, absent from the discussion, and the fidelity differences between the DFS method and `Domino` indicate that the noise robustness of the preparation is sensitive to it. Future work will therefore examine methods that shuffle the participant sub-shifts around, such that the total depth of the circuit is minimized, and/or its fidelity potential is maximized.

Acknowledgments

It is acknowledged that the LaTeX package Quantikz was used for graphically rendering quantum circuits [34].

Declarations

- **Funding**
  No funding was received for conducting this study.
- **Competing interests**
  The authors of this manuscript declare no competing interests.
- **Ethics approval and consent to participate**
  Not applicable.
- **Consent for publication**
  All authors have read and consent to the publication of this version of the manuscript.
- **Data availability**
  No new data were produced while conducting this study.
- **Materials availability**
  Not applicable.
- **Code availability**
  Code produced to aid in conducting this study is offered freely and openly in:
  https://github.com/asimakiskydros/Quantum-Hypercube.
- **Author contribution**
  A.K. conceived the idea for the study, performed the theoretical analysis, produced the relevant software, conducted the experimental testing and wrote the manuscript. K.P. and N.K. critically revised the work.

References

[1] Arianna Pavone and Caterina Viola. The quantum cyclic rotation gate. *SN Computer Science*, 5 (7), 2024. doi: 10.1007/s42979-024-03141-4.

[2] Zigang Chen, Yi Yan, Ji Pan, and HaiHua Zhu. Cyclic shift-based mqir image encryption scheme. *Quantum Information Processing*, 21(5), May 9 2022. doi: 10.1007/s11128-022-03510-z.

[3] Ping Fan, Ri-Gui Zhou, Naihuan Jing, and Hai-Sheng Li. Geometric transformations of multi-dimensional color images based on nass. *Information Sciences*, 340:191–208, May 1 2016. doi: 10.1016/j.ins.2015.12.024.

[4] Jian Wang, Nan Jiang, and Luo Wang. Quantum image translation. *Quantum Information Processing*, 14(5, Si):1589–1604, May 2015. doi: 10.1007/s11128-014-0843-6.

[5] Ping Fan, Mengjuan Hou, Aihan Yin, and Hai-Sheng Li. The linear cyclic translation and two-point swapping transformations for quantum images. *Quantum Information Processing*, 20(3), Mar 10 2021. doi: 10.1007/s11128-021-03044-w.

[6] Ri-Gui Zhou, Canyun Tan, and Hou Ian. Global and local translation designs of quantum image based on frqi. *International Journal Of Theoretical Physics*, 56(4):1382–1398, April 2017. doi: 10.1007/s10773-017-3279-9.

[7] Phuc Q Le, Abdullahi M Iliyasu, Fangyan Dong, and Kaoru Hirota. Fast geometric transformations on quantum images. *IAENG Int. J. Appl. Math*, 40(3):113–123, 2010.

[8] Phuc Q. Le, Abdullahi M. Iliyasu, Fangyan Dong, and Kaoru Hirota. Strategies for designing geometric transformations on quantum images. *Theoretical Computer Science*, 412(15):1406–1418, Mar 25 2011. doi: 10.1016/j.tcs.2010.11.029.

[9] Konstantinos Prousalis, Asimakis Kydros, and Nikos Konofaos. A quantum computing method for multidimensional cycle shift transformations for quantum image processing. *Physica Scripta*, 100(11), Nov 1 2025. doi: 10.1088/1402-4896/ae1d2b.

[10] Xingbin Liu. Quantum image encryption based on baker map and dna circular shift operation. *Physica Scripta*, 98(11), 2023. doi: 10.1088/1402-4896/ad0099.

[11] Wentao Hao, Tianshuo Zhang, Xianyi Chen, and Xiaoyi Zhou. A hybrid neqr image encryption cryptosystem using two-dimensional quantum walks and quantum coding. *Signal Processing*, 205, April 2023. doi: 10.1016/j.sigpro.2022.108890.

[12] Jing-Yi Dai, Yan Ma, and Nan-Run Zhou. Quantum multi-image compression-encryption scheme based on quantum discrete cosine transform and 4d hyper-chaotic henon map. *Quantum Information Processing*, 20(7), July 2021. doi: 10.1007/s11128-021-03187-w.

[13] Ri-Gui Zhou, Jia Luo, XingAo Liu, Changming Zhu, Lai Wei, and Xiafen Zhang. A novel quantum image steganography scheme based on lsb. *International Journal Of Theoretical Physics*, 57(6): 1848–1863, June 2018. doi: 10.1007/s10773-018-3710-x.

[14] Panchi Li, Xiande Liu, and Hong Xiao. Quantum image median filtering in the spatial domain. *Quantum Information Processing*, 17(3), March 2018. doi: 10.1007/s11128-018-1826-9.

[15] Abdalla Essam Ali, Hala Abdel-Galil, and Soha Mohamed. Quantum image mid-point filter. *Quantum Information Processing*, 19(8), Jul 9 2020. doi: 10.1007/s11128-020-02738-x.

[16] Rui Zhang, Jian Wang, and Nan Jiang. Boundary extension methods to quantum signal mean filtering. *Quantum Information Processing*, 21(2), February 2022. doi: 10.1007/s11128-021-03386-5.

[17] Zhiguo Qu, Zhenwen Cheng, Mingxing Luo, and Wenjie Liu. A robust quantum watermark algorithm based on quantum log-polar images. *International Journal Of Theoretical Physics*, 56 (11):3460–3476, November 2017. doi: 10.1007/s10773-017-3512-6.

[18] Zhanhong Wei, Huijuan Yin, Mengdi Han, and Dandan Ai. Geometric attacks on a color quantum image watermarking protocol based on the lsb algorithm. 2024. doi: 10.1109/oncon62778.2024.10931626.

[19] Konstantinos Prousalis, Asimakis Kydros, and Nikos Konofaos. A quantum string-matching algorithm. *Advanced Quantum Technologies*, 8(3), October 2024. ISSN 2511-9044. doi: 10.1002/qute.202400250.

[20] Yi Zhang, Kai Lu, Kai Xu, Yinghui Gao, and Richard Wilson. Local feature point extraction for quantum images. *Quantum Information Processing*, 14(5):1573–1588, September 2014. ISSN 1573-1332. doi: 10.1007/s11128-014-0842-7.

[21] Asimakis Kydros, Vasilis Skarlatos, Konstantinos Prousalis, and Nikos Konofaos. Preparation algorithms for cheaper toffoli-based quantum cycle shifts. *Advanced Quantum Technologies*, 9(9): e70445, 2026. doi: https://doi.org/10.1002/qute.70445.

[22] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, Cambridge [u.a.], 10th anniversary ed., repr. edition, 2012. ISBN 9781107002173.

[23] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[24] Lov K. Grover. A fast quantum mechanical algorithm for database search, 1996.

[25] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4-5):493–505, 1998. doi: https://doi.org/10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P.

[26] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, November 1995. ISSN 1094-1622. doi: 10.1103/physreva.52.3457.

[27] Ben Zindorf and Sougato Bose. Efficient implementation of multicontrolled quantum gates. *Physical Review Applied*, 24(4), October 2025. ISSN 2331-7019. doi: 10.1103/8blx-nfcr.

[28] Jonathan M. Baker, Casey Duckering, Alexander Hoover, and Frederic T. Chong. Decomposing quantum generalized Toffoli with an arbitrary number of ancilla, 2019.

[29] Baptiste Claudon, Julien Zylberman, César Feniou, Fabrice Debbasch, Alberto Peruzzo, and Jean-Philip Piquemal. Polylogarithmic-depth controlled-NOT gates without ancilla qubits. *Nature Communications*, 15(1), July 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-50065-x.

[30] D. Michael Miller, Robert Wille, and Zahra Sasanian. Elementary quantum gate realizations for multiple-control Toffoli gates. In *2011 41st IEEE International Symposium on Multiple-Valued Logic*, pages 288–293. Ieee, May 2011. doi: 10.1109/ismvl.2011.54.

[31] Helmut Prodinger. On binary representations of integers with digits- 1, 0, 1. *Integers*, pages A8–14, 2000.

[32] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit, 2024.

[33] IBM. Compute resources. IBM Quantum Platform. URL https://quantum.cloud.ibm.com/computers. Accessed: 2026-07-07.

[34] Alastair Kay. Tutorial on the quantikz package. September 2018. doi: 10.48550/arxiv.1809.03842.

Appendix A. Pseudocodes for prior triangular cycle shift preparators

In this section, algorithmic schematics for `MaxSmallest`, `Nearest` and `DP-DFS` are presented, such that this work remains self-contained. Note that this does not serve as an extensive description; further details can be found in Refs. [8, 9, 21].

Algorithm 2: `MaxSmallest` (adapted from Ref. [8])

Data: The host register $|\psi\rangle$, its width n, the desired step $s \in \mathbb{Z}$.
Result: The host register shifted by s modulo 2^n.

$s' \leftarrow |s| \bmod 2^n$
$\sigma \leftarrow \mathrm{sign}(s)$
while $s' \neq 0$ **do**
  $p \leftarrow \lfloor \log_2(s') \rfloor$
  $s' \leftarrow s' - 2^p$
  Perform a cycle shift of step $\sigma 2^p$ on $|\psi\rangle$;
end

Algorithm 3: `Nearest`

Data: The host register $|\psi\rangle$, its width n, the desired step $s \in \mathbb{Z}$.
Result: The host register shifted by s modulo 2^n.

$s' \leftarrow |s| \bmod 2^n$
$\sigma \leftarrow \mathrm{sign}(s)$
while $s' \neq 0$ **do**
  $lo \leftarrow \lfloor \log_2(s') \rfloor$
  $hi \leftarrow lo + 1$
  $p \leftarrow lo$ **if** $|s' - 2^{lo}| < |s' - 2^{hi}|$ **else** hi
  Perform a cycle shift of step $\sigma 2^p$ on $|\psi\rangle$
  $\sigma \leftarrow -\sigma$ **if** $p = hi$ **else** σ
  $s' \leftarrow |s' - 2^p|$
end

Algorithm 4: `DP-DFS`

Data: The host register $|\psi\rangle$, its width n, the desired step $s \in \mathbb{Z}$.
Result: The host register shifted by s modulo 2^n.

$s' \leftarrow |s| \bmod 2^n$
$\sigma \leftarrow \mathrm{sign}(s)$
Perform DFS on the binary tree $\mathbf{T}$ rooted at s' where the step remainder r on each node is reduced by the largest power of two at most equal to r via one edge and by the smallest power greater than r via the other edge, ending on leaf nodes that contain powers of two. Perform memoization.
$P \leftarrow$ the edge path in $\mathbf{T}$ that minimizes $\sum_{p:2^p \in P} n - p$
foreach $p \in P$ **do**
  Perform a cycle shift of step $\sigma 2^p$ on $|\psi\rangle$
  $\sigma \leftarrow -\sigma$ **if** $2^p > s'$ **else** σ
  $s' \leftarrow |s' - 2^p|$
end