Block encoding of sparse matrices via coherent permutation

Abhishek Setty [*]
Institute of Quantum Control (PGI-8), Forschungszentrum Jülich, D-52425 Jülich, Germany
and Institute for Theoretical Physics, University of Cologne, D-50937 Cologne, Germany

(Received 16 September 2025; accepted 19 August 2026; published 8 September 2026)

Block encoding of sparse matrices underpins quantum algorithms such as quantum singular value transformation, Hamiltonian simulation, and quantum linear system solvers, yet its efficient gate-level realization remains challenging, with index-mapping oracles constituting one important source of overhead. We introduce a block-encoding framework that focuses on the index-mapping component, where coherent permutation is used as the central mechanism to optimize shift, delete, and insert operations. This provides a unified treatment of index mapping and reduces local multicontrolled X control complexity through structured compression. We further connect coherent amplitude permutation to combinatorial optimization, enabling systematic assignment of control states under hardware connectivity constraints. The resulting construction supports entrywise block encoding for general sparse matrices, with efficiency gains in structured cases. We demonstrate the approach on representative examples and provide resource analysis using IBM superconducting backends, showing significant reductions in circuit depth.

DOI: 10.1103/1nts-v6y9

I. INTRODUCTION

Block encoding has emerged as a central primitive in modern quantum algorithms, providing a systematic way to embed a matrix into a unitary operator and thereby enabling polynomial transformation of operators via Quantum Singular Value Transformation (QSVT) [1,2]. The idea was first used implicitly in early breakthroughs such as the Harrow-Hassidim-Lloyd algorithm for solving linear systems of equations [3] and Hamiltonian simulation techniques [4–6], and was later formalized by Gilyén *et al.* [1]. Since then, block encoding has become indispensible in a wide range of domains, including quantum linear algebra, optimization, machine learning, and quantum chemistry [7–9]. Succinctly, block encoding is the process of embedding a given (possibly nonunitary) matrix A into a larger unitary operator U_A as

$$U_A = \begin{pmatrix} A/\alpha & * \\ * & * \end{pmatrix}, \tag{1}$$

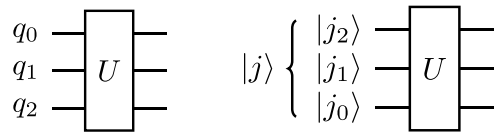
where α is a subnormalization factor ensuring $||A/\alpha||_2 \leqslant 1$, $*$ denotes inconsequential blocks, and $||\cdot||_2$ is the spectral norm. The factor α and the presence of $*$ blocks guarantee that a unitary U_A exists.

Understanding the importance of block encoding has motivated extensive efforts toward resource-efficient constructions for different matrix classes. For arbitrary dense matrices, exact block-encoding schemes based on state preparation and LCU (linear combination of unitaries) techniques have been studied extensively, together with their associated resource requirements [10,11]. More recently, approximate hardware-oriented constructions based on decompositions into single- and two-qubit gates have been proposed, including the FABLE framework [12,13] and subsequent improvements employing demultiplexor techniques [14]. For sparse matrices, block encodings have typically been formulated in terms of black-box oracles [1], but these works often omit explicit circuit-level implementations. More detailed realizations have been provided for structured sparsity [15].

The subnormalization factor α directly influences the complexity of QSVT-based algorithms, with the optimal choice being spectral norm $\alpha = ||A||_2$. Existing sparse block-encoding constructions [16,17] generally employ larger problem-dependent factors $\alpha > ||A||_2$. Whether one can construct block encodings with subnormalization factor $\alpha = ||A||_2$ remains an important open question.

Block encoding of sparse matrices consists predominantly of two components: data embedding via state preparation and data distribution via index-mapping operations. In this work, we focus primarily on the index-mapping component. We introduce coherent permutation operators as a central mechanism, enabling a unified treatment of shift, delete, and insert operations while reducing local multicontrolled X (MCX) control complexity through structured compression. The resulting framework supports entrywise construction to general sparse matrices, with efficiency gains expected in structured instances. We further establish a connection between coherent amplitude permutation and combinatorial optimization, enabling systematic assignment of control states and hardware-aware circuit constructions under connectivity constraints. This perspective is particularly relevant for

[*]Contact author: a.setty@fz-juelich.de



(a) (b)

FIG. 1. (a) Qubits of circuit U are numbered in increasing order from top to bottom. (b) The matrix column register $|j\rangle = |j_{n-1}\cdots j_0\rangle$. The binary representation of an integer state $|j\rangle$ is assigned to qubits in decreasing order from top to bottom.

architectures with limited qubit connectivity [18–20]. More broadly, the framework connects quantum circuit synthesis with classical optimization techniques, contributing to circuit compilation and routing strategies [21–24]. We demonstrate the approach on representative examples and provide hardware resource analysis on IBM superconducting backends, showing reductions in circuit depth. Finally, we demonstrate these ideas on representative sparse matrices with varying degrees of structure, illustrating how the framework translates abstract oracle constructions into executable circuits suitable for quantum algorithms.

II. NOTATIONS AND PREPROCESSING

Assuming familiarity with standard conventions in the quantum computing literature, we establish following notations and conventions:

(1) For an $N \times N$ matrix, the jth column is denoted by $|j\rangle$, where $j \in [0, N-1]$. Its binary representation is given by

$$j \mapsto j_{n-1} \times 2^{n-1} + \cdots + j_0 \times 2^0 \mapsto |j_{n-1}\cdots j_1 j_0\rangle, \quad (2)$$

where $j_k \in \{0,1\}, k \in [0, n-1]$ and n is the number of qubits.

(2) Qubits in a circuit diagram are ordered increasingly from top to bottom, as illustrated for the three-qubit circuit U in Fig. 1(a). The binary state $|j_{n-1}\cdots j_1 j_0\rangle$ is mapped to the quantum register $|q_0 q_1 \cdots q_{n-1}\rangle$, such that the highest-index qubit $|q_{n-1}\rangle$ corresponds to $|j_0\rangle$ and the lowest-index qubit $|q_0\rangle$ corresponds to $|j_{n-1}\rangle$ [Fig. 1(b)].

(3) The Hamming distance [25,26] between two binary strings $\vec{x} = (x_0, x_1, \ldots, x_{n-1})^T$ and $\vec{y} = (y_0, y_1, \ldots, y_{n-1})^T$, where $x_k, y_k \in \{0,1\}$, is defined as

$$D_H = |\vec{x} - \vec{y}| = \sum_{k=0}^{n-1} (x_k \oplus y_k), \quad (3)$$

where $\oplus$ denotes the XOR operation. For instance, $D_H(010, 001) = 2$.

(4) Multicontrolled NOT gates are denoted by $C^{|\text{state}\rangle}_{|\text{control}\rangle} X_{|\text{target}\rangle}$, where $|\text{control}\rangle$ denotes control qubits, $|\text{state}\rangle$ denotes control state [Eq. (2)], and $X_{|\text{target}\rangle}$ denotes NOT gate applied on $|\text{target}\rangle$ qubit. When convenient, we write

$$C^{|\text{state}_1\rangle}_{|\text{control}_1\rangle} C^{|\text{state}_2\rangle}_{|\text{control}_2\rangle} X_{|\text{target}\rangle}$$

to indicate a partition of the controls into two groups while still representing a single generalized MCX gate.

The main objective of block encoding is to seek a unitary U_A such that

$$(\langle 0|_{\text{anc}} \otimes \langle i|) U_A (|0\rangle_{\text{anc}} \otimes |j\rangle) = \frac{A_{ij}}{\alpha}, \quad (4)$$

where $|i\rangle$ and $|j\rangle$ denote row and column registers, respectively. Also, $|0\rangle_{\text{anc}}$ denotes ancilla qubits.

We now discuss the data preprocessing needed for block encoding a sparse complex matrix $A \in \mathbb{C}^{2^n \times 2^n}$. It is important to note that this method embeds each data element only once in a row/column. Therefore, we collect unique data elements with respect to a fixed diagonal offset $d = i - j$. In the present work, two entries on the same diagonal are considered identical if they are exactly equal as complex numbers. For numerical implementations involving floating-point values, one may alternatively adopt a tolerance parameter $\varepsilon > 0$ and identify entries satisfying $|A_{ij} - A_{i'j'}| < \varepsilon$ as belonging to the same equivalence class. Such tolerance-based grouping introduces an approximation to the original matrix.

For a square matrix, we define $d > 0$ as lower diagonals and $d < 0$ as upper diagonals. For each diagonal offset $d \in \text{Diag} = \{-(2^n - 1), \ldots, (2^n - 1)\}$, let the nonzero entries be indexed by the local index $l = 1, \ldots, |S_d|$:

$$\begin{aligned} S_d &= \{A_{ij} \in \mathbb{C} \mid A_{ij} \neq 0, i - j = d\} \\ &= \{s_{d,l} \in \mathbb{C} \mid s_{d,l} \neq 0, l = 1, \ldots, |S_d|\}. \end{aligned} \quad (5)$$

The associated row set is denoted by $S_r(s_{d,l})$, and the corresponding column index is given by $j = S_r(s_{d,l}) - d$.

We introduce the following labeled contribution index set:

$$\begin{aligned} \mathcal{I} = \{(d, l, c) | \, d \in \text{Diag}, l = 1, \ldots, |S_d|, \\ c \in \{R, I\}, \gamma_c(s_{d,l}) \neq 0\}, \end{aligned} \quad (6)$$

where

$$\gamma_R(s_{d,l}) = \text{Re}(s_{d,l}), \quad \gamma_I(s_{d,l}) = \text{Im}(s_{d,l}).$$

Here, $c = R$ and $c = I$ denote the real and imaginary components, respectively. Importantly, the labels distinguish different contributions even when their numerical values are equal.

Choose a fixed ordering of the labeled index set $\mathcal{I}$, and denote its ordered enumeration by

$$(\iota_1, \ldots, \iota_N), \quad N = |\mathcal{I}|, \quad (7)$$

where the ordering is induced by fixed orderings of d, l, and c. For each $\iota_k = (d(k), l(k), c(k))$, $k = 1, \ldots, N$, define

$$m_k = \begin{cases} |\text{Re}(s_{d(k),l(k)})|, & c(k) = R, \\ |\text{Im}(s_{d(k),l(k)})|, & c(k) = I, \end{cases} \quad (8)$$

and

$$p_k = \begin{cases} \text{sgn}(\text{Re}(s_{d(k),l(k)})), & c(k) = R, \\ \text{sgn}(\text{Im}(s_{d(k),l(k)}))\text{i}, & c(k) = I, \end{cases} \quad (9)$$

where $\text{sgn}(a) = \frac{a}{|a|}$, $a \in \mathbb{R}\backslash\{0\}$. The data and sign vectors are then defined by

$$v_{\text{data}} = (m_1, \ldots, m_N), \quad (10)$$

$$v_{\text{sign}} = (p_1, \ldots, p_N). \quad (11)$$

TABLE I. Notations of data structures and quantum registers used in block encoding.

Data structure and registers	
Row register	$\|i\rangle$
Column register	$\|j\rangle$
Value of kth data element	$v_{\text{data},k}$
Sign of kth data element	$v_{\text{sign},k}$
Data register basis state	$\|k\rangle_{\text{data}}$
Diagonal shift	d_k
Row set	$S_r(k)$
Delete flag	$\|b\rangle_{\text{del}}$

By construction, all entries of v_{data} are non-negative real numbers obtained from the magnitudes of nonzero real or imaginary components. The dimension of the data vector is $\dim(v_{\text{data}}) = N$. Each kth element in the flattened vectors has the unique backmapping $k \mapsto \iota_k = (d(k), l(k), c(k))$, where $d(k)$ denotes the diagonal offset, $l(k)$ denotes the local position within $S_{d(k)}$, and $c(k)$ identifies the real or imaginary component. We therefore define $d_k = d(k)$, and $S_r(k) = S_r(s_{d(k),l(k)})$. The corresponding column index is then $j_k = S_r(k) - d_k$.

We summarize the notations of data structures and quantum registers as in Table I.

III. BLOCK ENCODING

In this section, we elaborate the PREP/UNPREP-based block encoding [16,17] and present a framework for constructing the corresponding quantum circuits. Primarily, block encoding consists of two oracles, namely, state preparation and index mapping.

Let the number of data qubits $m = \lceil \log_2 \dim(v_{\text{data}}) \rceil$ for the collected data v_{data} and signs v_{sign} [see Eqs. (10) and (11)]. Then, the state-preparation oracles PREP [Eq. (12)] and UNPREP [Eq. (13)] together embed $v_{\text{sign}} \circ v_{\text{data}}$ into the amplitudes of a quantum state. Here, $\circ$ denotes elementwise multiplication between two vectors.

Note that the state vectors are padded with zeros to fill the 2^m basis states required for state preparation. These additional basis states do not correspond to physical data elements but serve only as padding. Since the phase associated with a zero amplitude is physically irrelevant, we adopt the convention $\text{sgn}(0) = 0$ to obtain a canonical representation, which is convenient for the compression procedures introduced later.

$$\text{PREP}|0\rangle_{\text{data}}^{\otimes m} = \frac{1}{\sqrt{\sum_{k=0}^{\dim(v_{\text{data}})-1} v_{\text{data},k}}} \left(\sum_{k=0}^{\dim(v_{\text{data}})-1} v_{\text{sign},k} \sqrt{v_{\text{data},k}} |k\rangle_{\text{data}} + \sum_{k=\dim(v_{\text{data}})}^{2^m-1} 0|k\rangle_{\text{data}} \right), \tag{12}$$

$$\text{UNPREP}^\dagger|0\rangle_{\text{data}}^{\otimes m} = \frac{1}{\sqrt{\sum_{k=0}^{\dim(v_{\text{data}})-1} v_{\text{data},k}}} \left(\sum_{k=0}^{\dim(v_{\text{data}})-1} \sqrt{v_{\text{data},k}} |k\rangle_{\text{data}} + \sum_{k=\dim(v_{\text{data}})}^{2^m-1} 0|k\rangle_{\text{data}} \right), \tag{13}$$

where the left projection $\langle 0|_{\text{data}}^{\otimes m}\text{UNPREP}$ is determined by $\text{UNPREP}^\dagger|0\rangle_{\text{data}}^{\otimes m}$.

The index-mapping oracle is composed of two oracles such as shift O_{shift} and delete O_{del}. The shift oracle performs the injective mapping

$$f_k(j) : j \mapsto i = \text{mod}(j + d_k, 2^n), \tag{14}$$

thereby mapping column indices j to row indices i along the diagonal offset $d_k = i - j$. The oracle O_{del} performs deletion of elements from rows $f_k(j) \notin S_r(k)$.

With these oracles in place, the block-encoding scheme is formulated as described in Theorem 1.

Theorem 1. Let $A \in \mathbb{C}^{2^n \times 2^n}$ be a matrix that has collected data v_{data} and sign v_{sign}. If there exist shift oracle O_{shift} such that

$$\begin{aligned} O_{\text{shift}}|k\rangle_{\text{data}}|b\rangle_{\text{del}}|j\rangle &= |k\rangle_{\text{data}}|b\rangle_{\text{del}}|f_k(j)\rangle \\ &= |k\rangle_{\text{data}}|b\rangle_{\text{del}}|\text{mod}(j + d_k, 2^n)\rangle \end{aligned} \tag{15}$$

and delete oracle O_{del} such that

$$\begin{aligned} &O_{\text{del}}|k\rangle_{\text{data}}|b\rangle_{\text{del}}|f_k(j)\rangle \\ &\quad = |k\rangle_{\text{data}}|b \oplus \chi_k(f_k(j))\rangle_{\text{del}}|f_k(j)\rangle, \end{aligned} \tag{16}$$

where

$$\chi_k(f_k(j)) = \begin{cases} 0, & \text{if } f_k(j) \in S_r(k), \\ 1, & \text{else,} \end{cases}$$

and two state-preparation oracles PREP and UNPREP then, the unitary, $U_A = (\text{UNPREP} \otimes I_{2^{n+1}}) O_{\text{del}} O_{\text{shift}} (\text{PREP} \otimes I_{2^{n+1}})$, as shown in Fig. 2, can block encode A with the subnormalization $\alpha = \sum_k v_{\text{data},k}$.

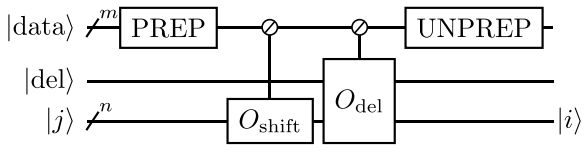


FIG. 2. Circuit structure for block encoding of sparse matrices. A single qubit is represented by a plain wire (del qubit), while multiple qubits are depicted as a wire marked with a short slash, labeled with the number of qubits it contains (m for data qubits and n for matrix qubits $|j\rangle$). The PREP and UNPREP blocks denote state-preparation operators. The oracles O_{shift} and O_{del} use multicontrolled gates denoted by circle with a slash. The control values are determined by the respective state in binary representation [Eq. (2)], following the qubit ordering convention shown in Fig. 1.

Proof. To recover the matrix from its block encoding, the ancilla qubits (i.e., the data and delete qubits) are initialized and postselected in the state $|0\rangle$. This is achieved by initializing the bottom register with $|j\rangle$ and postselecting (or measuring) the outcome $|i\rangle$, as follows:

$$\begin{aligned}
&{}_{\text{data}}^{\otimes m}\langle 0|_{\text{del}}\langle i|\ U_A\ |0\rangle_{\text{data}}^{\otimes m}|0\rangle_{\text{del}}|j\rangle \\
&= \frac{1}{\sqrt{\sum_{k'} v_{\text{data},k'} \sum_k v_{\text{data},k}}} \sum_{k',k} \langle k'|_{\text{data}}\langle 0|_{\text{del}}\langle i| \\
&\quad \times v_{\text{sign},k}\sqrt{v_{\text{data},k'} v_{\text{data},k}}\ O_{\text{del}} O_{\text{shift}}|k\rangle_{\text{data}}|0\rangle_{\text{del}}|j\rangle \\
&= \frac{1}{\sqrt{\sum_{k'} v_{\text{data},k'} \sum_k v_{\text{data},k}}} \sum_{k',k} \langle k'|_{\text{data}}\langle 0|_{\text{del}}\langle i| \\
&\quad \times v_{\text{sign},k}\sqrt{v_{\text{data},k'} v_{\text{data},k}}\ O_{\text{del}}|k\rangle_{\text{data}}|0\rangle_{\text{del}}|f_k(j)\rangle \\
&= \frac{1}{\sqrt{\sum_{k'} v_{\text{data},k'} \sum_k v_{\text{data},k}}} \sum_{k',k} \langle k'|_{\text{data}}\langle 0|_{\text{del}}\langle i| \\
&\quad \times v_{\text{sign},k}\sqrt{v_{\text{data},k'} v_{\text{data},k}}\ \mathbf{1}_{S_r(k)}(i)\ \delta_{k,k'}\ \delta_{i,f_k(j)} \\
&\quad \times |k\rangle_{\text{data}}|0\rangle_{\text{del}}|f_k(j)\rangle \\
&= \frac{1}{\alpha} \sum_{\substack{k:f_k(j)=i \\ i\in S_r(k)}} v_{\text{sign},k} v_{\text{data},k},
\end{aligned}$$

where δ denotes a Kronecker delta function $\delta_{ij} = 1$ if $i = j$, else 0 and indicator function $\mathbf{1}_{S_r(k)}(i) = 1$ if $i \in S_r(k)$, else 0.

In case of complex entries, we add its real and imaginary components within the block-encoded matrix. Consider kth and lth data elements in v_{data} and v_{sign} correspond to $\text{Re}(A_{ij})$ and $\text{Im}(A_{ij})$, respectively. Then, the above proof can block encode a complex entry as follows:

$$\begin{aligned}
\frac{A_{ij}}{\alpha} &= \frac{1}{\alpha}(\text{Re}(A_{ij}) + \text{Im}(A_{ij})\text{i}) \\
&= \frac{1}{\alpha}(v_{\text{sign},k} v_{\text{data},k} + v_{\text{sign},l} v_{\text{data},l}).
\end{aligned} \tag{17}$$

A. State-preparation oracle

In this section, we briefly discuss the existing methods of state-preparation oracles PREP/UNPREP, whose task is to embed data into the amplitudes of a quantum state. Möttönen *et al.* [27,28] introduced a state-preparation method based on uniformly controlled rotation gates, which can be decomposed into either multicontrolled rotations or sequences of single- and two-qubit gates. This construction leverages classical preprocessing—such as Gray code ordering—to structure the circuit efficiently, reducing the number of controlled rotations required. Later, Iten *et al.* [29] proposed a hardware-oriented approach that decomposes arbitrary isometries exactly into single-qubit and CNOT gates via a recursive synthesis procedure based on the cosine-sine decomposition. Both methods are exact and require no ancilla qubits, but generally scale exponentially in gate count and depth for arbitrary state preparation. More recent work has explored depth-optimized schemes [30], achieving circuit depths of $\mathcal{O}(m)$ for an m-qubit state or $\mathcal{O}(\log_2(ms))$ for s-sparse states. These improvements, however, often come at the cost of introducing additional ancilla qubits, which in some cases can scale exponentially requiring $\mathcal{O}(2^m)$ ancilla qubits.

B. Index-mapping oracle

Without the index-mapping oracles O_{shift} and O_{del} in Fig. 2, the block-encoded matrix reduces to a diagonal form (refer Theorem 1),

$$A/\alpha = \begin{bmatrix} x & & \\ & \ddots & \\ & & x \end{bmatrix}, \quad x = \frac{\sum_k v_{\text{sign},k} v_{\text{data},k}}{\sum_k v_{\text{data},k}}. \tag{18}$$

The oracle O_{shift} [15] may equivalently be interpreted as fixing the row index and updating the column register. Under the convention $d_k = i - j$, $j = i - d_k$, and therefore a main-diagonal entry (i, i) is mapped to

$$(i, i) \mapsto (i, (i - d_k) \bmod 2^n).$$

Specifically, we define the shift oracle O_{shift} as

$$O_{\text{shift}}(k, d) = \begin{cases} \prod_{b\in B(|d|)} L(k, b) & \text{if } d > 0, \\ I & \text{if } d = 0, \\ \prod_{b\in B(|d|)} R(k, b) & \text{if } d < 0, \end{cases} \tag{19}$$

where we define absolute value of diagonal $|d|$ in binary form:

$$|d| = \sum_{l=0}^{n-1} b_l 2^l, \quad b_l \in \{0, 1\}. \tag{20}$$

For an integer $|d|$ in binary $(b_{n-1} \cdots b_0)_2$, we define a set of 1-bit positions as $B(|d|) = \{l | b_l = 1\}$. In other words, a shift $|d|$ is decomposed into a sequence of conditional shifts by powers of 2. Then, for $d > 0$, we define a left shift oracle for kth data element $v_{\text{data},k}$ shifting left by 2^b columns as

$$L(k, b) = \prod_{l=b}^{n-1} C_{|\text{data}\rangle}^{|k\rangle} C_{|j_{l-1}\cdots j_b\rangle}^{|1\rangle^{\otimes(l-b)}} X_{|j_l\rangle}. \tag{21}$$

Note that the order of operators are written from left to right and the rightmost operator is applied first in the circuit. Similarly, for $d < 0$, we define a right shift oracle for kth data element $v_{\text{data},k}$ shifting right by 2^b columns as

$$R(k, b) = \prod_{l=b}^{n-1} C_{|\text{data}\rangle}^{|k\rangle} C_{|j_{l-1}\cdots j_b\rangle}^{|0\rangle^{\otimes(l-b)}} X_{|j_l\rangle}. \tag{22}$$

To visualize the circuit, we consider three matrix qubits $|j\rangle$ and represent left shift ($d = 1$) of kth data element by one column $L(k, 0)$ [see Eq. (21)] in Fig. 3(a). Similarly, the right shift ($d = -1$) of kth data element by one column $R(k, 0)$ [see Eq. (22)] is shown in Fig. 3(b). Furthermore, the representation of shifting left the kth and lth data elements by two and four columns, respectively, is shown in Fig. 3(c).

To visualize the block encoded matrix after shifting, we represent an example as follows. Consider the data vector $v_{\text{data}} = [\psi_0, \psi_1, \psi_2, \psi_3, \psi_4]^T$, sign vector $v_{\text{sign}} = [1, 1, -\text{i}, -1, \text{i}]^T$, and block-encoding matrix to be 8×8 requiring three matrix qubits. If we apply the operator $L(4, 2)L(3, 1)L(3, 0)L(2, 1)L(1, 1)L(0, 0)$ [see Eqs. (19) and (21), and Fig. 3(a)], then the block encoded matrix will be

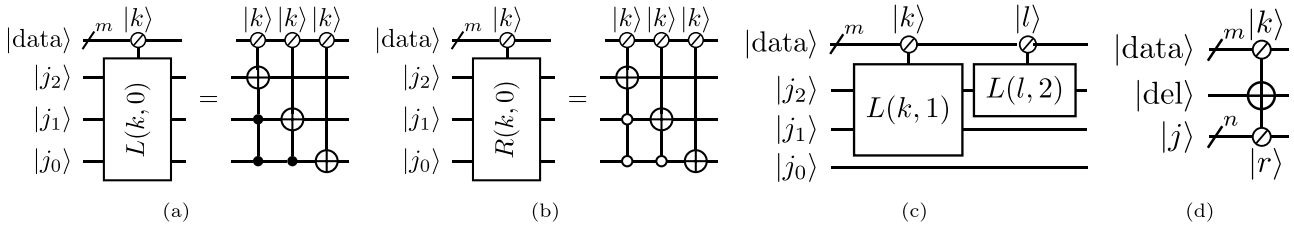


FIG. 3. (a) Circuit illustration of left shift oracle $L(k,0)$ [refer Eq. (21)] with three matrix qubits $|j\rangle$ and m data qubits. Here, the control state is given by $|k\rangle$ and denoted by circle with a slash. (b) Similarly, circuit illustration of right shift oracle $R(k,0)$ [refer Eq. (22)]. (c) Left shift oracle of kth and lth data elements by two and four columns, respectively. (d) Delete oracle O_{del} illustrating deletion of kth data element from rth row [refer Eq. (25)].

given as

$$A/\alpha = \frac{1}{\psi_0+\psi_1+\psi_2+\psi_3+\psi_4} \times \begin{bmatrix} 0 & 0 & 0 & 0 & \psi_4\mathrm{i} & -\psi_3 & \psi_1-\psi_2\mathrm{i} & \psi_0 \\ \psi_0 & 0 & 0 & 0 & 0 & \psi_4\mathrm{i} & -\psi_3 & \psi_1-\psi_2\mathrm{i} \\ \psi_1-\psi_2\mathrm{i} & \psi_0 & 0 & 0 & 0 & 0 & \psi_4\mathrm{i} & -\psi_3 \\ -\psi_3 & \psi_1-\psi_2\mathrm{i} & \psi_0 & 0 & 0 & 0 & 0 & \psi_4\mathrm{i} \\ \psi_4\mathrm{i} & -\psi_3 & \psi_1-\psi_2\mathrm{i} & \psi_0 & 0 & 0 & 0 & 0 \\ 0 & \psi_4\mathrm{i} & -\psi_3 & \psi_1-\psi_2\mathrm{i} & \psi_0 & 0 & 0 & 0 \\ 0 & 0 & \psi_4\mathrm{i} & -\psi_3 & \psi_1-\psi_2\mathrm{i} & \psi_0 & 0 & 0 \\ 0 & 0 & 0 & \psi_4\mathrm{i} & -\psi_3 & \psi_1-\psi_2\mathrm{i} & \psi_0 & 0 \end{bmatrix}. \tag{23}$$

Note that complex entries are block encoded by adding its nonzero real and imaginary components [see Eq. (17)]. Similarly, if we apply the operator $R(4,2)R(3,1)R(3,0)R(2,1)R(1,1)R(0,0)$ [see Eqs. (19) and (22), and Fig. 3(b)], then the block encoded matrix will be given as

$$A/\alpha = \frac{1}{\psi_0+\psi_1+\psi_2+\psi_3+\psi_4} \times \begin{bmatrix} 0 & \psi_0 & \psi_1-\psi_2\mathrm{i} & -\psi_3 & \psi_4\mathrm{i} & 0 & 0 & 0 \\ 0 & 0 & \psi_0 & \psi_1-\psi_2\mathrm{i} & -\psi_3 & \psi_4\mathrm{i} & 0 & 0 \\ 0 & 0 & 0 & \psi_0 & \psi_1-\psi_2\mathrm{i} & -\psi_3 & \psi_4\mathrm{i} & 0 \\ 0 & 0 & 0 & 0 & \psi_0 & \psi_1-\psi_2\mathrm{i} & -\psi_3 & \psi_4\mathrm{i} \\ \psi_4\mathrm{i} & 0 & 0 & 0 & 0 & \psi_0 & \psi_1-\psi_2\mathrm{i} & -\psi_3 \\ -\psi_3 & \psi_4\mathrm{i} & 0 & 0 & 0 & 0 & \psi_0 & \psi_1-\psi_2\mathrm{i} \\ \psi_1-\psi_2\mathrm{i} & -\psi_3 & \psi_4\mathrm{i} & 0 & 0 & 0 & 0 & \psi_0 \\ \psi_0 & \psi_1-\psi_2\mathrm{i} & -\psi_3 & \psi_4\mathrm{i} & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{24}$$

The delete oracle [16] for removing the kth data element from the rth row is defined as

$$O_{\text{del}}(k,r) = D_{|r\rangle}^{|k\rangle} = C_{|\text{data}\rangle}^{|k\rangle} C_{|j\rangle}^{|r\rangle} X_{|\text{del}\rangle}. \tag{25}$$

The corresponding gate is illustrated in Fig. 3(d).

To insert a single data element into one or a few rows, one could in principle delete the data element from all remaining rows. However, this approach may require an exponential number of MCX gates in general. To avoid this overhead, we instead introduce an insert operator $I_{|r\rangle}^{|k\rangle}$, which directly places the kth data element into the rth row without requiring exponential MCX gates. The construction of this operator is detailed later in Sec. VII C.

IV. COMPOSITION OF MULTICONTROLLED X GATES

In this section, we analyze the composition and simplification of MCX gates that arise in index-mapping oracles.

Consider two successive shift operators $L(k,0)L(l,0)$ [see Eq. (21) and Fig. 3(a)] acting on three matrix qubits as shown in Fig. 4. Expanding both operators, we obtain

$$L(k,0)L(l,0) = \left[\left(C_{|\text{data}\rangle}^{|k\rangle} X_{|j_0\rangle}\right)\left(C_{|\text{data}\rangle}^{|k\rangle} C_{|j_0\rangle}^{|1\rangle} X_{|j_1\rangle}\right) \times \left(C_{|\text{data}\rangle}^{|k\rangle} C_{|j_1 j_0\rangle}^{|1\rangle^{\otimes 2}} X_{|j_2\rangle}\right)\right]$$

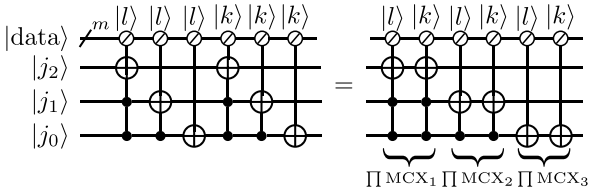


FIG. 4. Consecutive left shift oracles $L(k,0)L(l,0)$ on three matrix qubits $|j\rangle$. The MCX gates can be grouped into three compositions $\prod \text{MCX}_i$ [see Eq. (26)].

$$
\begin{aligned}
&\times \left[\left(C^{|l\rangle}_{|\text{data}\rangle} X_{|j_0\rangle}\right)\left(C^{|l\rangle}_{|\text{data}\rangle} C^{|1\rangle}_{|j_0\rangle} X_{|j_1\rangle}\right)\right.\\
&\left.\times \left(C^{|l\rangle}_{|\text{data}\rangle} C^{|1\rangle^{\otimes 2}}_{|j_1 j_0\rangle} X_{|j_2\rangle}\right)\right]\\
=&\left[\left(C^{|k\rangle}_{|\text{data}\rangle} X_{|j_0\rangle}\right)\left(C^{|l\rangle}_{|\text{data}\rangle} X_{|j_0\rangle}\right)\right]\\
&\times \left[\left(C^{|k\rangle}_{|\text{data}\rangle} C^{|1\rangle}_{|j_0\rangle} X_{|j_1\rangle}\right)\left(C^{|l\rangle}_{|\text{data}\rangle} C^{|1\rangle}_{|j_0\rangle} X_{|j_1\rangle}\right)\right]\\
&\times \left[\left(C^{|k\rangle}_{|\text{data}\rangle} C^{|1\rangle^{\otimes 2}}_{|j_1 j_0\rangle} X_{|j_2\rangle}\right)\left(C^{|l\rangle}_{|\text{data}\rangle} C^{|1\rangle^{\otimes 2}}_{|j_1 j_0\rangle} X_{|j_2\rangle}\right)\right]\\
=&\left[\prod \text{MCX}_1\right]\left[\prod \text{MCX}_2\right]\left[\prod \text{MCX}_3\right]. \qquad (26)
\end{aligned}
$$

Since the two operators act on the same qubits, we can reorder the factors by grouping terms acting on identical control and target qubits. This reordering is valid because the controlled operations corresponding to distinct control states $|k\rangle$ and $|l\rangle$ act on orthogonal subspaces, i.e., $|k\rangle\langle k||l\rangle\langle l| = 0$, which implies that the corresponding MCX gates commute. Therefore, each group forms a commuting composition of MCX gates acting on the same qubits, which we denote as $[\prod \text{MCX}_1][\prod \text{MCX}_2][\prod \text{MCX}_3]$ (see Fig. 4). A similar structure arises in deletion operations across multiple rows [see Fig. 3(d)]. In both cases, we obtain compositions of MCX gates acting on identical sets of control and target qubits. Such compositions can be simplified under suitable conditions.

Let $G = \{0, \ldots, P-1\}$ denote the index set of P qubits [see Eq. (2) and Fig. 1]. Let

$$S_1 = \{a^j\}_{j=0}^{2^P-1}$$

be the set of all P-bit binary strings, where

$$a^j = \left(a_i^j\right)_{i\in G}, \quad a_i^j \in \{0, 1\}.$$

Consider a composition of 2^n MCX gates ($1 \leqslant n < P$):

$$\prod_{j=0}^{2^n-1} C_G^{|a^j\rangle} X_{|t\rangle},$$

where $|t\rangle$ is the target qubit and the controls act on G qubits. The simplification of such compositions is possible under certain conditions as described in Theorem 2.

Theorem 2. Let $S_2 = \{a^j\}_{j=0}^{2^n-1} \subset S_1 = \{0, 1\}^P$. Suppose there exists a subset of indices $F \subset G$ with $|F| = P - n$ such that

$$a_i^j = a_i^k \quad \forall i \in F,\ \forall j, k,$$

and the substrings on the remaining indices satisfy

$$\left\{\left(a_i^j\right)_{i\in G\setminus F}\right\}_{j=0}^{2^n-1} = \{0, 1\}^n.$$

Then,

$$\prod_{j=0}^{2^n-1} C_G^{|a^j\rangle} X_{|t\rangle} = C_F^{|\tilde{a}\rangle} X_{|t\rangle},$$

where $|\tilde{a}\rangle = \bigotimes_{i\in F} |a_i\rangle$.

Proof. Consider each MCX gate is written as

$$C_G^{|a^j\rangle} X_{|t\rangle} = |a^j\rangle\langle a^j| \otimes X_{|t\rangle} + (I - |a^j\rangle\langle a^j|) \otimes I_{|t\rangle},$$

where the identity operator I has dimension corresponding to the number of qubits it acts on. Let $F \subset G$ denote the set of fixed indices, where $a_i^j = a_i^k\ \forall i \in F, \forall j, k \in \{0, \ldots, 2^n - 1\}$, and let $G \setminus F$ denote the varying indices. We define

$$|\tilde{a}\rangle = \bigotimes_{i\in F} |a_i\rangle, \quad |a'^j\rangle = \bigotimes_{i\in G\setminus F} \left|a_i^j\right\rangle,$$

where the substrings $\{a'^j\}_{j=0}^{2^n-1} = \{0, 1\}^n$ enumerate all 2^n binary strings.

Without loss of generality, we assume that the qubits are ordered such that the indices in F precede those in $G \setminus F$. This does not affect the final operation and allows us to rewrite the control state as

$$|a^j\rangle\langle a^j| = |\tilde{a}\rangle\langle\tilde{a}| \otimes |a'^j\rangle\langle a'^j|.$$

Then, the product of MCX gates can be written as

$$
\begin{aligned}
\prod_{j=0}^{2^n-1} C_G^{|a^j\rangle} X_{|t\rangle} = \prod_{j=0}^{2^n-1} [&|\tilde{a}\rangle\langle\tilde{a}| \otimes |a'^j\rangle\langle a'^j| \otimes X_{|t\rangle}\\
&+ (I - (|\tilde{a}\rangle\langle\tilde{a}| \otimes |a'^j\rangle\langle a'^j|)) \otimes I_{|t\rangle}].
\end{aligned}
$$

In this product expansion, due to orthogonality $\langle a'^j|a'^k\rangle = \delta_{jk}$, the cross terms get eliminated resulting in the summation as

$$
\begin{aligned}
\prod_{j=0}^{2^n-1} C_G^{|a^j\rangle} X_{|t\rangle} = \sum_{j=0}^{2^n-1} &(|\tilde{a}\rangle\langle\tilde{a}| \otimes |a'^j\rangle\langle a'^j| \otimes X_{|t\rangle}\\
&+ (I - |\tilde{a}\rangle\langle\tilde{a}|) \otimes |a'^j\rangle\langle a'^j| \otimes I_{|t\rangle}),
\end{aligned}
$$

where $\sum_{j=0}^{2^n-1} |a'^j\rangle\langle a'^j| = I$, since the summation of projectors over complete computational basis forms the identity operator. Therefore, the above expression can be further simplified to

$$
\begin{aligned}
\prod_{j=0}^{2^n-1} C_G^{|a^j\rangle} X_{|t\rangle} &= |\tilde{a}\rangle\langle\tilde{a}| \otimes I \otimes X_{|t\rangle} + (I - |\tilde{a}\rangle\langle\tilde{a}|) \otimes I \otimes I_{|t\rangle}\\
&= C_F^{|\tilde{a}\rangle} X_{|t\rangle}.
\end{aligned}
$$

In the special case $n = P$, the control set spans the entire computational basis, and hence

$$\prod_{j=0}^{2^P-1} C_G^{|a^j\rangle} X_{|t\rangle} = X_{|t\rangle}. \qquad (27)$$

V. COMBINATORIAL-OPTIMIZATION-BASED MAPPING OF BASIS STATES

In the previous section (see Sec. IV), we showed that a composition of MCX gates can be compressed when the control set $S_2 = \{a^j\}_{j=0}^{2^n-1}$ exhibits a fixed set of indices $F \subset G$, with $|F| = P - n$, such that

$$a_i^j = a_i^k, \quad \forall i \in F,\ \forall j, k \in \{0, \ldots, 2^n - 1\},$$

and the substrings over the remaining indices enumerate all binary strings of length n.

If S_2 does not satisfy these structural conditions, then the objective is to find a structured set S_3 together with a bijection $\phi : S_2 \mapsto S_3$, which satisfies the conditions required for compression (see Theorem 2).

For a given set of fixed indices $F \subset G$, with $|F| = P - n$, we define a structured set $S_3 = \{b^j\}_{j=0}^{2^n-1} \subset \{0,1\}^P$ such that

$$b_i^j = b_i^k, \quad \forall i \in F, \ \forall j,k,$$

and the substrings over the remaining indices exhaust all configurations,

$$\left\{\left(b_i^j\right)_{i\in G\setminus F}\right\}_{j=0}^{2^n-1} = \{0,1\}^n.$$

This construction ensures that $|S_3| = 2^n$ and that S_3 satisfies the structural conditions required for MCX compression. We then formalize the mapping of basis states as a combinatorial optimization problem as follows.

Theorem 3. Given an arbitrary set of binary strings $S_2 \subset \{0,1\}^P$ with $|S_2| = 2^n$ and a target set S_3 constructed according to the mode-based rule, the bijection $\phi : S_2 \to S_3$ that minimizes

$$\sum_{x\in S_2} D_H(x, \phi(x))$$

can be obtained by solving a linear assignment problem over the unmatched subsets $S_2 \setminus S_3$ and $S_3 \setminus S_2$, while fixing identity mapping on $S_2 \cap S_3$.

Proof. Let $S_2 = \{a^j\}_{j=0}^{2^n-1}$. The mode-based rule first constructs the target set $S_3 = \{b^j\}_{j=0}^{2^n-1}$ by selecting the most frequent fixed-bit pattern

$$\tilde{a} = \text{mode}\left(\left\{a_F^j\right\}_{j=0}^{2^n-1}\right), \tag{28}$$

where a_F^j denotes the restriction of a^j to the fixed index set F. The target set is then defined by fixing

$$b_i^j = \tilde{a}_i, \quad i \in F,$$

while allowing the remaining bits to enumerate all binary strings over $G \setminus F$. Consequently, $|S_3| = 2^n$.

Define

$$S_2' = S_2 \setminus S_3, \quad S_3' = S_3 \setminus S_2.$$

Since $|S_2| = |S_3|$, it follows that $|S_2'| = |S_3'|$. Elements in $S_2 \cap S_3$ are assigned via identity mapping and do not contribute to the optimization.

Construct the cost matrix

$$C_{jk} = D_H(a^j, b^k), \quad a^j \in S_2', \quad b^k \in S_3',$$

and solve the assignment problem

$$\begin{aligned} \min_{x_{jk}} \quad & \sum_j \sum_k C_{jk} x_{jk}, \\ \text{s.t.} \quad & \sum_k x_{jk} = 1, \quad \forall j, \\ & \sum_j x_{jk} = 1, \quad \forall k, \\ & x_{jk} \in \{0,1\}, \quad \forall j,k. \end{aligned} \tag{29}$$

Here, $x_{jk} = 1$ if and only if $\phi(a^j) = b^k$.

Equation (29) is the classical linear assignment problem, which admits an optimal solution computable in $O(|S_2'|^3)$ time using the Hungarian algorithm [31–34]. Therefore, for the constructed target set S_3, the resulting bijection ϕ minimizes the total Hamming distance $\sum_{x\in S_2} D_H(x, \phi(x))$.

Theorem 3 establishes the optimal assignment for the constructed target set S_3. The mode-based selection of the fixed pattern $\tilde{a}$ serves as a heuristic for constructing S_3 by maximizing the overlap $|S_2 \cap S_3|$, thereby reducing the number of basis states requiring permutation and the size of the subsequent assignment problem.

A further (computationally more expensive) extension is to additionally search over the choice of the fixed pattern $\tilde{a}$. This leads to the discrete optimization problem

$$\min_{\tilde{a}} \left(\min_{\phi: S_2 \to S_3(\tilde{a})} \sum_{x\in S_2} D_H(x, \phi(x)) \right), \tag{30}$$

where for each fixed $\tilde{a}$ the inner problem is a linear assignment problem.

Since $|S_3(\tilde{a})| = 2^n$ for all admissible $\tilde{a}$, this induces a finite family of assignment problems indexed by $\tilde{a}$. In this work, we adopt a mode-based heuristic to select $\tilde{a}$, followed by solving the corresponding optimal assignment via the Hungarian algorithm. If multiple optimal bijections exist, a canonical choice may be obtained via deterministic tie-breaking rules or by a small perturbation $C_{jk} \mapsto C_{jk} + \epsilon r_{jk}$ with $\epsilon > 0$ arbitrarily small and distinct perturbation r_{jk}, which preserves optimality while ensuring uniqueness of the minimizer [33,35,36].

VI. COHERENT PERMUTATION USING MULTICONTROLLED X GATES

In this section, we introduce a coherent permutation of amplitudes among basis states using MCX gates. Here, *coherent* refers to a unitary (reversible) transformation that preserves superposition and relative phases, without measurement or state collapse.

Consider a P-qubit quantum state $|\psi\rangle = \sum_{i=0}^{2^P-1} \psi_i |i\rangle$, where the computational basis is indexed by binary strings over the index set $G = \{0, \ldots, P-1\}$ [see Eq. (2) and Fig. 1]. Any basis state $|a\rangle$ is represented by a binary string $a = (a_i)_{i\in G}$. We begin by defining a primitive operation that swaps amplitudes between two basis states differing in a single qubit.

Definition 1 A_SWAP(a,b). Let $a, b \in \{0,1\}^P$ be binary strings such that they differ only at qubit $t \in G$, i.e.,

$$a_i = b_i \ \ \forall i \in G \setminus \{t\}, \quad a_t \oplus b_t = 1.$$

Define the common control string $c = (c_i)_{i\in G\setminus\{t\}}$, where $c_i = a_i = b_i$. Then, the amplitudes corresponding to $|a\rangle$ and $|b\rangle$ are swapped via

$$\psi_a|a\rangle + \psi_b|b\rangle \ \longmapsto \ \psi_b|a\rangle + \psi_a|b\rangle,$$

by applying the gate

$$C_{G\setminus\{t\}}^{|c\rangle} X_{|t\rangle}.$$

Note that A_SWAP permutes amplitudes without affecting amplitudes in other basis states.

Definition 2 (Coherent permutation). Let $S_2, S_3 \subset S_1 = \{0,1\}^P$ with $|S_2| = |S_3|$, and let $\phi : S_2 \to S_3$ be a bijection satisfying

$$\phi(x) = x, \quad x \in S_2 \cap S_3.$$

Define

$$S_2' = S_2 \setminus S_3, \quad S_3' = S_3 \setminus S_2.$$

Then, ϕ restricts to a bijection $\phi : S_2' \to S_3'$. For each $a \in S_2'$, choose a shortest Hamming path

$$\mathcal{P}_a : \quad a = a^{(0)} \to \cdots \to a^{(m_a)} = \phi(a),$$

where $m_a = D_H(a, \phi(a))$, such that

$$D_H(a^{(k)}, a^{(k+1)}) = 1,$$

and

$$a^{(k)} \notin S_2 \cap S_3, \quad k = 0, \ldots, m_a - 1.$$

The corresponding local walk operator is defined by

$$\mathcal{W}_a = \prod_{k=0}^{m_a-1} \text{A\_SWAP}\big(a^{(k)}, a^{(k+1)}\big),$$

where the product is ordered along the path. Each $\mathcal{W}_a$ implements a cyclic permutation of the basis states along the ordered vertex set

$$(a^{(0)}, a^{(1)}, \ldots, a^{(m_a)}),$$

transporting the amplitude initially at $a^{(0)}$ to $a^{(m_a)} = \phi(a)$.

The coherent permutation operator is

$$\text{A\_PERMUTE} = \prod_{a \in S_2'} \mathcal{W}_a,$$

where the ordering of the local walk operators is specified by an admissible routing schedule (Definition 3).

Definition 3 (Admissible routing schedule). Let

$$\mathcal{W}_{a_1}, \mathcal{W}_{a_2}, \ldots, \mathcal{W}_{a_r}, \quad r = |S_2'|,$$

be an ordering of the local walk operators. The ordering is called *admissible* if it satisfies the following:

(1) Each local walk operator is executed completely before the next local walk begins.

(2) A source basis state in S_2' may be reused as an intermediate vertex only after its assigned amplitude has reached its destination.

(3) A target basis state in S_3' may be used as an intermediate vertex only before its assigned amplitude arrives.

(4) Intermediate basis states outside $S_2 \cup S_3$ may be reused arbitrarily.

(5) Distinct routing paths may intersect and share intermediate basis states. No path-disjointness condition is required, provided conditions (1)–(4) are satisfied.

Theorem 4 (Global permutation induced by coherent routing). Under the coherent permutation construction of Definition 2, suppose that the selected Hamming paths admit an admissible routing schedule according to Definition 3. Then, A_PERMUTE is a permutation unitary inducing a permutation $\sigma : S_1 \mapsto S_1$ such that

$$\sigma(a) = \phi(a), \quad \forall a \in S_2.$$

Consequently,

$$\sigma(S_2) = S_3, \quad \sigma(S_1 \setminus S_2) = S_1 \setminus S_3.$$

In general, the restriction of σ to $S_1 \setminus S_2$ is a nontrivial complementary permutation induced by the same A_SWAP network.

Proof. Each A_SWAP is a transposition of computational basis states, and hence A_PERMUTE, being a finite product of such operations, is a permutation unitary. Therefore, there exists a unique permutation $\sigma : S_1 \mapsto S_1$ such that

$$\text{A\_PERMUTE}|x\rangle = |\sigma(x)\rangle, \quad \forall x \in S_1.$$

It remains to show that $\sigma|_{S_2} = \phi$.

Let $(a_1, \ldots, a_r)$ be the ordering of the elements of S_2' specified by the admissible routing schedule, where $r = |S_2'|$ and $\{a_1, \ldots, a_r\} = S_2'$. Then, define

$$U_0 = I, \quad U_q = \mathcal{W}_{a_q} \cdots \mathcal{W}_{a_1}, \quad q = 1, \ldots, r. \tag{31}$$

We prove by induction on q that

$$U_q|a_\ell\rangle = |\phi(a_\ell)\rangle, \quad \ell = 1, \ldots, q, \tag{32}$$

and

$$U_q|x\rangle = |x\rangle, \quad x \in S_2 \cap S_3. \tag{33}$$

For $q = 0$, both statements are immediate. Assume Eqs. (32) and (33) hold for $q - 1$. Since a_q is an unprocessed source, condition (2) of Definition 3 prevents a_q from being used as an intermediate vertex in any of the preceding local walks. Hence,

$$U_{q-1}|a_q\rangle = |a_q\rangle.$$

By the definition of the local walk,

$$\mathcal{W}_{a_q}|a_q\rangle = |\phi(a_q)\rangle, \tag{34}$$

and therefore

$$U_q|a_q\rangle = \mathcal{W}_{a_q} U_{q-1}|a_q\rangle = |\phi(a_q)\rangle. \tag{35}$$

For every $\ell < q$, the induction hypothesis gives

$$U_{q-1}|a_\ell\rangle = |\phi(a_\ell)\rangle.$$

Since the assigned target $\phi(a_\ell)$ has already been reached, admissibility prevents it from being used in any subsequent local walk. Thus,

$$\mathcal{W}_{a_q}|\phi(a_\ell)\rangle = |\phi(a_\ell)\rangle, \quad \ell < q,$$

and hence

$$U_q|a_\ell\rangle = \mathcal{W}_{a_q} U_{q-1}|a_\ell\rangle = |\phi(a_\ell)\rangle, \quad \ell < q. \tag{36}$$

Finally, no state in $S_2 \cap S_3$ occurs on any routing path. Therefore,

$$\mathcal{W}_{a_q}|x\rangle = |x\rangle, \quad x \in S_2 \cap S_3,$$

and, using the induction hypothesis,

$$U_q|x\rangle = \mathcal{W}_{a_q} U_{q-1}|x\rangle = |x\rangle. \tag{37}$$

Equations (35)–(37) establish Eqs. (32) and (33) for q, completing the induction.

Setting $q = r$, we obtain

$$\sigma(a) = \phi(a), \quad a \in S_2', \tag{38}$$

and

$$\sigma(x) = x, \quad x \in S_2 \cap S_3. \tag{39}$$

Since

$$S_2 = S_2' \cup (S_2 \cap S_3)$$

and $\phi(x) = x$ on $S_2 \cap S_3$, it follows from Eqs. (38) and (39) that

$$\sigma(a) = \phi(a), \quad a \in S_2.$$

Therefore,

$$\sigma(S_2) = \phi(S_2) = S_3.$$

Since σ is a permutation of S_1,

$$\sigma(S_1 \backslash S_2) = S_1 \backslash S_3.$$

The action on $S_1 \backslash S_2$ is therefore the complementary permutation induced by the same A_SWAP network and is not, in general, the identity.

For the proposed construction, the selected shortest Hamming paths are required to admit an admissible routing schedule. If no admissible schedule can be identified, the assignment ϕ, the target set S_3, or the selected shortest paths may be reconsidered until an admissible routing configuration is obtained. Accordingly, Theorem 4 guarantees the correctness of the resulting coherent permutation whenever such a configuration exists.

We now consider a small example to illustrate the coherent permutation.

Example 1. Consider three qubits as $|j_2 j_1 j_0\rangle$. Let the bijection

$$\begin{aligned}\phi:\ S_2 &= \{000, 001, 100, 111\}\\ &\mapsto S_3 = \{000, 001, 010, 011\},\end{aligned}$$

where $S_2' = \{100, 111\}$, $S_3' = \{010, 011\}$. We can choose the following shortest Hammings paths for the elements of S_2':

$$\mathcal{P}_{100} = 100 \mapsto 110 \mapsto 010, \quad \mathcal{P}_{111} = 111 \mapsto 011.$$

These paths form an admissible routing schedule according to Definition 3, since the common states 000 and 001 are not used as intermediate vertices and the two routing operations do not interfere with previously assigned destinations. The corresponding A_PERMUTE operator (see Theorem 4) is

$$\begin{aligned}&\text{A\_PERMUTE}|x\rangle\\ &= C^{|11\rangle}_{|j_1 j_0\rangle} X_{|j_2\rangle} \times C^{|10\rangle}_{|j_1 j_0\rangle} X_{|j_2\rangle} \times C^{|10\rangle}_{|j_2 j_0\rangle} X_{|j_1\rangle} |x\rangle\\ &= C^{|1\rangle}_{|j_1\rangle} X_{|j_2\rangle} \times C^{|10\rangle}_{|j_2 j_0\rangle} X_{|j_1\rangle} |x\rangle. \qquad (40)\end{aligned}$$

Note that the two MCX gates in Eq. (40) are compressed according to Theorem 2. Generalizing this example, we now consider the inverse construction: expressing a single MCX gate as a composition of MCX gates.

Theorem 5. Let $|\tilde{a}\rangle$ be a control state defined on indices $F \subset G$, and $|G \setminus F| = n$. Then, there exists a set of control states $S_2 = \{a^j\}_{j=0}^{2^n-1}$ such that

$$a_i^j = \tilde{a}_i \quad \forall i \in F, \quad \left\{a^j_{G\backslash F}\right\}_{j=0}^{2^n-1} = \{0,1\}^n,$$

and

$$C_F^{|\tilde{a}\rangle} X_{|t\rangle} = \prod_{j=0}^{2^n-1} C_G^{|a^j\rangle} X_{|t\rangle}.$$

Proof. Let $H = G \setminus F$, $|H| = n$. Since the free subsystem spans all binary strings of length n, there exists an orthonormal basis $\{|a'^j\rangle\}_{j=0}^{2^n-1} = \{|0,1\rangle^{\otimes n}\}$, satisfying

$$\sum_{j=0}^{2^n-1} |a'^j\rangle\langle a'^j| = I_H.$$

Then,

$$C_F^{|\tilde{a}\rangle} X_{|t\rangle} = |\tilde{a}\rangle\langle\tilde{a}| \otimes I_H \otimes X_{|t\rangle} + (I - |\tilde{a}\rangle\langle\tilde{a}|) \otimes I_H \otimes I_{|t\rangle}.$$

Substituting $I_H = \sum_{j=0}^{2^n-1} |a'^j\rangle\langle a'^j|$ gives

$$\begin{aligned}C_F^{|\tilde{a}\rangle} X_{|t\rangle} = \sum_{j=0}^{2^n-1} &(|\tilde{a}\rangle\langle\tilde{a}| \otimes |a'^j\rangle\langle a'^j| \otimes X_{|t\rangle}\\ &+ (I - |\tilde{a}\rangle\langle\tilde{a}|) \otimes |a'^j\rangle\langle a'^j| \otimes I_{|t\rangle}).\end{aligned}$$

Now, define $|a^j\rangle = |\tilde{a}\rangle \otimes |a'^j\rangle$. By construction, $a_i^j = \tilde{a}_i$, $i \in F$, and $\{a^j_{G\backslash F}\}_{j=0}^{2^n-1} = \{0,1\}^n$. Using the orthogonality relation $\langle a'^j | a'^k\rangle = \delta_{jk}$, the summation above is precisely the product

$$\begin{aligned}C_F^{|\tilde{a}\rangle} X_{|t\rangle} = \prod_{j=0}^{2^n-1} &[|\tilde{a}\rangle\langle\tilde{a}| \otimes |a'^j\rangle\langle a'^j| \otimes X_{|t\rangle}\\ &+ (I - (|\tilde{a}\rangle\langle\tilde{a}| \otimes |a'^j\rangle\langle a'^j|)) \otimes I_{|t\rangle}].\end{aligned}$$

Hence,

$$\begin{aligned}C_F^{|\tilde{a}\rangle} X_{|t\rangle} &= \prod_{j=0}^{2^n-1} [|a^j\rangle\langle a^j| \otimes X_{|t\rangle} + (I - |a^j\rangle\langle a^j|) \otimes I_{|t\rangle}]\\ &= \prod_{j=0}^{2^n-1} C_G^{|a^j\rangle} X_{|t\rangle}.\end{aligned}$$

Operationally, this implies that a single MCX gate can be interpreted as simultaneously performing A_SWAP operations across all pairs of basis states whose control patterns belong to S_2. Note that Theorem 5 becomes highly relevant in further reducing the implementation cost of A_PERMUTE operator as discussed in Sec. VIII B. To demonstrate this process, consider an example of three qubits (j_2, j_1, j_0) and a quantum state vector $[\psi_i]_{i=0}^7$. The CONTROLLED-NOT gate

$$C^{|1\rangle}_{|j_1\rangle} X_{|j_0\rangle} = C^{|01\rangle}_{|j_2 j_1\rangle} X_{|j_0\rangle} \times C^{|11\rangle}_{|j_2 j_1\rangle} X_{|j_0\rangle},$$

permutes the amplitudes as

$$\begin{bmatrix}\psi_0\\ \psi_1\\ \psi_2\\ \psi_3\\ \psi_4\\ \psi_5\\ \psi_6\\ \psi_7\end{bmatrix} \longmapsto \begin{bmatrix}\psi_0\\ \psi_1\\ \psi_3\\ \psi_2\\ \psi_4\\ \psi_5\\ \psi_7\\ \psi_6\end{bmatrix}.$$

VII. OPTIMIZED INDEX-MAPPING ORACLE

We have seen that index-mapping oracles O_{shift} and O_{del} (Sec. III B) can shift and delete the data elements in the block-encoded matrix. We can compress the composition of 2^n MCX gates when the control set S_2 exhibits a fixed set of indices $F \subset G$ and satisfies the structural constraints in

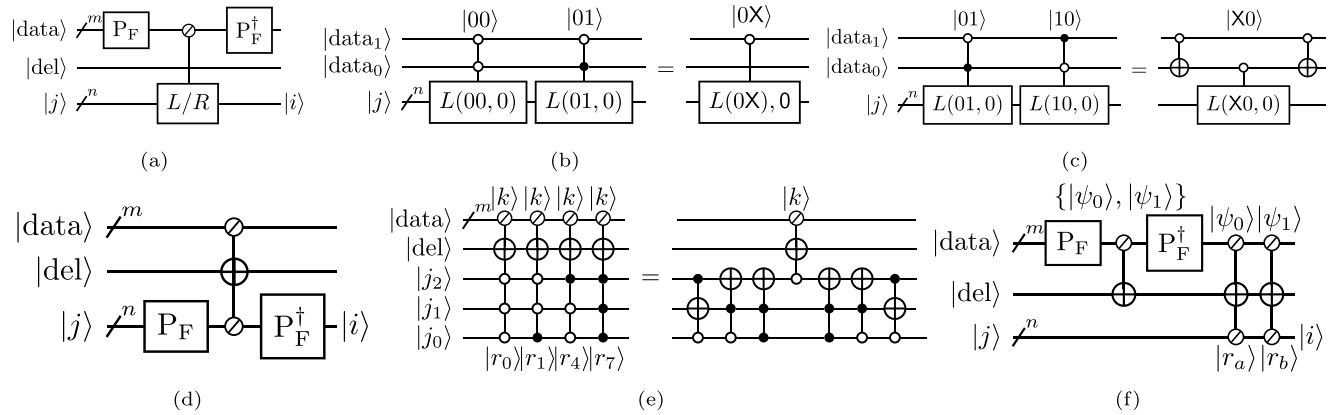


FIG. 5. (a) Circuit representation of combined shift operation with permutation of amplitudes A_PERMUTE$_F$ (represented by P_F). (b) Circuit for Example 2, showing the combined left shift of data elements $\{\psi_0, \psi_1\}$ in basis states $\{|00\rangle, |01\rangle\}$ by one column. (c) Circuit for Example 3, illustrating permutation of amplitudes and left shift of data elements $\{\psi_1, \psi_2\}$ by one column. (d) Circuit representation of combined deletion with permutation of rows P_F. (e) Circuit for Example 5, showing deletion of data element $|k\rangle$ in rows $\{|r_0\rangle, |r_1\rangle, |r_4\rangle, |r_7\rangle\}$. Note that the MCX gates here within P_F can also be compressed as in Eq. (40), but retained to show the full walk operator (Example 1). (f) Circuit for Example 6, illustrating the insertion of data elements $\{|\psi_0\rangle, |\psi_1\rangle\}$ in rows $|r_a\rangle, |r_b\rangle$, respectively, along with P_F.

Sec. IV. Otherwise, we can determine an arbitrary F and find a set S_3 such that $\phi : S_2 \mapsto S_3$ (Sec. V). The choice of fixed indices F determines the qubits on which the MCX gates are applied. Therefore, we can use this as an advantage in superconducting quantum hardware and apply the MCX gates on nearest-neighbor qubits reducing the control complexity.

In this section, we illustrate the optimized index-mapping operations through several examples arising in sparse-matrix block encoding, covering a range of representative applications. We further present hardware-level resource analyses for the presented examples to assess the practical impact of the proposed constructions. The analysis is performed for IBM superconducting quantum processor with heavy-hex and square lattice topologies. The circuits are transpiled to IBM superconducting hardware topologies and evaluated using two-qubit depth as the primary resource metric. Two-qubit depth is defined as the depth of native two-qubit operations in the transpiled circuit after accounting for device connectivity constraints, and corresponds to the critical path of sequential two-qubit gates, which typically provides the dominant contribution to runtime and error accumulation on superconducting hardware.

A. Shift

In the combined operation of shift oracles (Sec. IIIB), the objective is to get nearest-neighbor MCX gates. To achieve this, the choice of F comes from choosing the qubits in $|\text{data}\rangle$ closer to matrix qubits $|j\rangle$. The choice of F determines the mapping $\phi : S_2 \mapsto S_3$. Then, the amplitudes are permuted using A_PERMUTE$_F$ (refer Definition 2). The visualization of this combined shift operation is shown in Fig. 5(a). After shifting, the amplitudes are rearranged back to retain the original order, avoiding confusion in subsequent operations.

We now present some examples for intuitive understanding. Consider block encoding an 8×8 matrix with three matrix qubits ($|j\rangle$) and two data qubits ($|\text{data}_1\rangle, |\text{data}_0\rangle$). Let $v_{\text{data}} = [\psi_0, \psi_1, \psi_2, \psi_3]^T$; basis states for $|\text{data}\rangle$ are $(00, 01, 10, 11)$.

Example 2. The data elements $\{\psi_0, \psi_1\}$ are to be shifted left by one column using the operators $L(01, 0)L(00, 0)$, corresponding to a many-to-one mapping.

Solution. The control set $S_2 = \{00, 01\}$ satisfies the structural constraints in Theorem 2 such that $|S_2| = 2$, fixed index $F = \{|\text{data}_1\rangle\}$ for first composition of MCX gates as in Eq. (26) and Fig. 4. Then, the combined shift operation is given by

$$L(01, 0)L(00, 0) = L(0\mathsf{X}, 0), \tag{41}$$

where X denotes no control gate on qubit $|\text{data}_0\rangle$. The corresponding circuit representation is shown in Fig. 5(b).

Example 3. The data elements $\{\psi_1, \psi_2\}$ are to be shifted left by one column using the operators $L(01, 0)L(10, 0)$, corresponding to a many-to-one mapping.

Solution. The control states $S_2 = \{01, 10\}$ do not satisfy the constraints in Theorem 2. So, we choose the fixed index $F = \{|\text{data}_0\rangle\}$ for first composition of MCX gates as in Eq. (26) and Fig. 4. We apply the A_PERMUTE$_F$ [see Fig. 5(a)], where the amplitudes are swapped $\psi_0|00\rangle + \psi_1|01\rangle \mapsto \psi_1|00\rangle + \psi_0|01\rangle$ using a CNOT (control value is 0) gate, as shown in Fig. 5(c). After this permutation, the combined shift operator can be applied as

$$L(00, 0)L(10, 0) = L(\mathsf{X}0, 0). \tag{42}$$

For the case of $n = 3$ matrix qubits, we performed hardware resource analysis of the circuit [see Fig. 5(c)] before (with only shift oracles) and after (along with coherent permutation and compression) optimization of the index-mapping oracle. As shown in Table II, the optimized construction achieves a significant reduction in two-qubit depth for both heavy-hex and square lattice topologies.

Example 4. Assume a data vector v_{data} of seven values padded with 0: $v_{\text{data}} = [\psi_0, \psi_1, \psi_2, \psi_3, \psi_4, \psi_5, \psi_6, 0]^T$. The task is to shift the data elements $\{\psi_0, \psi_5, \psi_6\}$ left by one column.

Solution. According to Theorem 2, the control set S_2 should be a power of 2 (2^n). Since the task involves three data elements, a naive approach would be to apply shift operation individually. Alternatively, one can exploit the 0 in the data

TABLE II. Hardware resource analysis of Example 3 measured in terms of two-qubit depth. Here, before optimization denotes using only individual shift oracles, and after optimization denotes coherent permutation and compressed shift oracle. Here, "hex" denotes the heavy-hex lattice topology of the Heron r3 processor (IBM Boston), and "square" denotes the square lattice topology of the Nighthawk r1 processor (IBM Miami).

Hardware resource analysis of Example 3	
Before optimization	Hex = 173, square = 144
After optimization	Hex = 36, square = 35

TABLE III. Hardware resource analysis of Example 5 measured in terms of two-qubit depth. Here, before optimization denotes using only delete oracles, and after optimization denotes using coherent permutation and compressed delete oracle. Here, "hex" denotes the heavy-hex lattice topology of the Heron r3 processor (IBM Boston), and "square" denotes the square lattice topology of the Nighthawk r1 processor (IBM Miami).

Hardware resource analysis of Example 5	
Before optimization	Hex = 704, square = 515
After optimization	Hex = 47, square = 47

vector and perform a combined shift on four data elements $\{\psi_0, \psi_5, \psi_6, \psi_7\}$, as shifting 0 does not affect the block-encoded matrix [Eq. (18)]. The combined operation for these basis states requires permutation, as illustrated in Fig. 5(a).

Note that shifting a single data element left and then right results in the identity operation, i.e., $L \times R = I$ [refer Figs. 3(a) and 3(b)]. This property can be exploited when applying combined operations to simplify MCX gates.

B. Delete

We have seen how a single data element can be deleted in a specific row [Fig. 3(d)]. In this section, we generalize this to deletion in multiple rows. Consecutive delete operations of a single data element across multiple rows result in a composition of MCX gates with control and target on the same qubits. When the control states of such a composition satisfy the constraints in Theorem 2, they can be combined into a single MCX gate. Otherwise, one can choose a set of fixed indices $F \subset G$ and determine $\phi : S_2 \mapsto S_3$ (refer Sec. V). The choice of F comes from choosing the qubits in $|j\rangle$ close to $|\text{data}\rangle$ and $|\text{del}\rangle$ to achieve nearest-neighbor connectivity. The combined delete operation along with the permutation operator $\text{A\_PERMUTE}_\text{F}$ is presented in Fig. 5(d).

Example 5. Consider three matrix qubits $|j_2 j_1 j_0\rangle$. The task is that the kth data element is to be deleted in rows $\{0, 1, 4, 7\}$ using the operators $D^{|k\rangle}_{|0\rangle} D^{|k\rangle}_{|1\rangle} D^{|k\rangle}_{|4\rangle} D^{|k\rangle}_{|7\rangle}$, corresponding to a one-to-many mapping.

Solution. The set of control states $S_2 = \{000, 001, 100, 111\}$ does not have F and hence does not satisfy the constraints in Theorem 2. We choose $F = \{|j_2\rangle\}$ and obtain $\phi : S_2 = \{0, 1, 4, 7\} \mapsto S_3 = \{0, 1, 2, 3\}$ (see Sec. V) using the linear sum assignment algorithm [37]. The corresponding permutation of amplitudes (see Example 1) is given as

$$\begin{aligned}
000 &\to 000 && \to 000,\\
001 &\to 001 && \to 001,\\
100 &\to C^{|10\rangle}_{|j_1 j_0\rangle} X_{|j_2\rangle} \times C^{|10\rangle}_{|j_2 j_0\rangle} X_{|j_1\rangle} && \to 010,\\
111 &\to C^{|11\rangle}_{|j_1 j_0\rangle} X_{|j_2\rangle} && \to 011.
\end{aligned}$$

The combined deletion is then given by

$$D^{|k\rangle}_{\{|0\rangle,|1\rangle,|4\rangle,|7\rangle\}} = C^{|k\rangle}_{|\text{data}\rangle} C^{|0\rangle}_{|j_2\rangle} X_{|\text{del}\rangle}. \tag{43}$$

After the deletion, the rows are permuted back to their original order. The circuit for this example is shown in Fig. 5(e).

For $|k\rangle = |10\rangle$, we performed the hardware resource analysis for the circuit [see Fig. 5(e)] reported in Table III. The results indicate once again that optimized construction achieves a significant reduction in two-qubit depth for both heavy-hex and square lattice topologies.

Furthermore, a many-to-many mapping is also possible, where more than one data element can be deleted in multiple rows, provided that the set of rows to be deleted is common for all data elements.

C. Insert

In this section, we introduce an insert operator that places the data elements into one or few rows without requiring exponential MCX gates as illustrated in Example 6.

Example 6. Consider an example of block encoding a matrix of $2^n \times 2^n$. The data elements $\{\psi_0, \psi_1\}$ are to be inserted in different rows $\{r_a, r_b\}$, respectively, corresponding to a one-to-one mapping.

Solution. To insert a single data element ψ_0 in row r_a, we define

$$I^{|\psi_0\rangle}_{|r_a\rangle} = D^{|\psi_0\rangle}_{|r_a\rangle} \left(\prod_{k=0}^{2^n-1} D^{|\psi_0\rangle}_{|k\rangle} \right) = D^{|\psi_0\rangle}_{|r_a\rangle} \left(C^{|\psi_0\rangle}_{|\text{data}\rangle} X_{|\text{del}\rangle} \right), \tag{44}$$

where $(\prod_{k=0}^{2^n-1} D^{|\psi_0\rangle}_{|k\rangle})$ means deleting in all rows and it is compressed to single MCX gate as presented in Eq. (27) and Theorem 2. Since the proposed block-encoding method places each data element in every row [Eq. (18)], we first delete the data element from all rows and then apply deletion on the desired row $|r_a\rangle$. This effectively inverts the deletion on the desired row, resulting in the insertion of the data element in that row alone.

To insert a set of data elements $\{\psi_0, \psi_1\}$ into rows $\{|r_a\rangle, |r_b\rangle\}$, respectively, we formulate

$$\begin{aligned}
I^{|\psi_0\rangle}_{|r_a\rangle} I^{|\psi_1\rangle}_{|r_b\rangle} &= D^{|\psi_0\rangle}_{|r_a\rangle} \left(\prod_{k=0}^{2^n-1} D^{|\psi_0\rangle}_{|k\rangle} \right) D^{|\psi_1\rangle}_{|r_b\rangle} \left(\prod_{k=0}^{2^n-1} D^{|\psi_1\rangle}_{|k\rangle} \right)\\
&= D^{|\psi_0\rangle}_{|r_a\rangle} D^{|\psi_1\rangle}_{|r_b\rangle} \left(\prod_{k=0}^{2^n-1} D^{\{|\psi_0\rangle,|\psi_1\rangle\}}_{|k\rangle} \right).
\end{aligned} \tag{45}$$

The circuit representation for this task is shown in Fig. 5(f), where $\text{A\_PERMUTE}_\text{F}$ (represented as P_F) is included (if needed) for permutation of amplitudes for a chosen F.

For the instance $|\psi_0\rangle = |01\rangle$, $|\psi_1\rangle = |10\rangle$, $n = 3$, $|r_a\rangle = |101\rangle$, $|r_b\rangle = |011\rangle$, we performed the hardware resource

TABLE IV. Hardware resource analysis of Example 6 measured in terms of two-qubit depth. Here, before optimization denotes using only delete oracles, and after optimization denotes using insert and coherent permutation operators. Here, "hex" denotes the heavy-hex lattice topology of the Heron r3 processor (IBM Boston), and "square" denotes the square lattice topology of the Nighthawk r1 processor (IBM Miami).

Hardware resource analysis of Example 6	
Before optimization	Hex = 2234, square = 2069
After optimization	Hex = 330, square = 272

analysis of the circuit [Fig. 5(f)] comparing the original implementation, consisting only of delete oracles, with the optimized implementation incorporating insert and coherent permutation operators. The results, reported in Table IV, show that for this instance the introduction of insert and coherent permutation operations leads to a significant reduction in the transpiled two-qubit depth.

VIII. COMPLETE CIRCUIT

In this section, we present the circuit architectures and resource accounting for complete block encoding of sparse matrices. The workflow is summarized in Algorithm 1.

A. Circuit architectures

A first architecture employing the optimized index-mapping oracle (see Sec. VII) is presented in Fig. 6(a). In this design, amplitudes are restored to a fixed ordering after every combined operation. Let the coherent permutation operator consist of n MCX gates. Then,

$$\text{A\_PERMUTE} = \prod_{i=0}^{n-1} \text{MCX}_i,$$
$$\text{A\_PERMUTE}^\dagger = \prod_{i=0}^{n-1} \text{MCX}_{n-1-i}, \tag{46}$$

where the inverse permutation is implemented by applying the same MCX gates in reverse order.

A second architecture is illustrated in Fig. 6(b), where the state-preparation oracle initializes amplitudes in an ordering already compatible with the first combined operation. In this case, intermediate inverse permutations are avoided; instead, the forward permutation effect is tracked implicitly through an evolving ordering of amplitudes, and a complete inverse permutation $\text{A\_PERMUTE}^\dagger$ is applied at the end of the circuit. Whether this strategy yields a reduction in overall implementation cost compared with Fig. 6(a) remains an open question.

B. Resource accounting

In this section, we discuss the cost comparison of full block-encoding circuit. Let $\mathcal{C}(\cdot)$ denote an implementation cost metric, such as transpiled two-qubit depth, native two-qubit gate count, or a hardware-dependent weighted cost function.

For an original implementation consisting only of shift and delete operators,

$$\mathcal{C}_{\text{orig}} = \mathcal{C}(\text{PREP}) + \mathcal{C}(\text{UNPREP}) + \sum_{r=1}^{R} \mathcal{C}(O_r), \tag{47}$$

where $O_r \in \{O_{\text{shift}}, O_{\text{delete}}\}$.

ALGORITHM 1. Block encoding of sparse matrix via coherent permutation.

Input: Sparse matrix $A \in \mathbb{C}^{N\times N}$
Output: Block-encoding unitary U_A satisfying $\alpha\langle 0|U_A|0\rangle = A$
Data preprocessing
1. Extract the nonzero entries S_d along each diagonal offset $d \in$ Diag [see Eq. (5)].
2. Construct v_{data}, v_{sign} [Eqs. (10) and (11)].
3. Associate with each data element $(v_{\text{data},k}, v_{\text{sign},k}, d_k, S_r(k), |k\rangle_{\text{data}})$ (see Table I).

State preparation
1. Construct PREP and UNPREP oracles (Sec. III A).

Index-mapping optimization
1. Determine the required shift, delete, and insert operators for each data element (Sec. III B).
2. Identify compressible control patterns and apply MCX compression (Theorem 2).
3. If a compressed operator cannot be implemented directly, construct a target set S_3 from a source set S_2 using the mode-based heuristic (Theorem 3).
4. Solve the linear assignment problem [Eq. (29)] to obtain the bijection $\phi : S_2 \to S_3$.
5. Determine admissible routing paths and synthesize the coherent permutation A_PERMUTE (Sec. VI).

Circuit synthesis
1. Get the combined shift, delete, and insert operators with the help of permutation operators to obtain the optimized index-mapping oracle (see Sec. VII).
2. Assemble the PREP/UNPREP with optimized index-mapping oracles to get the complete block-encoding circuit [see Fig. 6(a)].
3. Multiply by the normalization factor α to recover the original matrix A (Theorem 1).

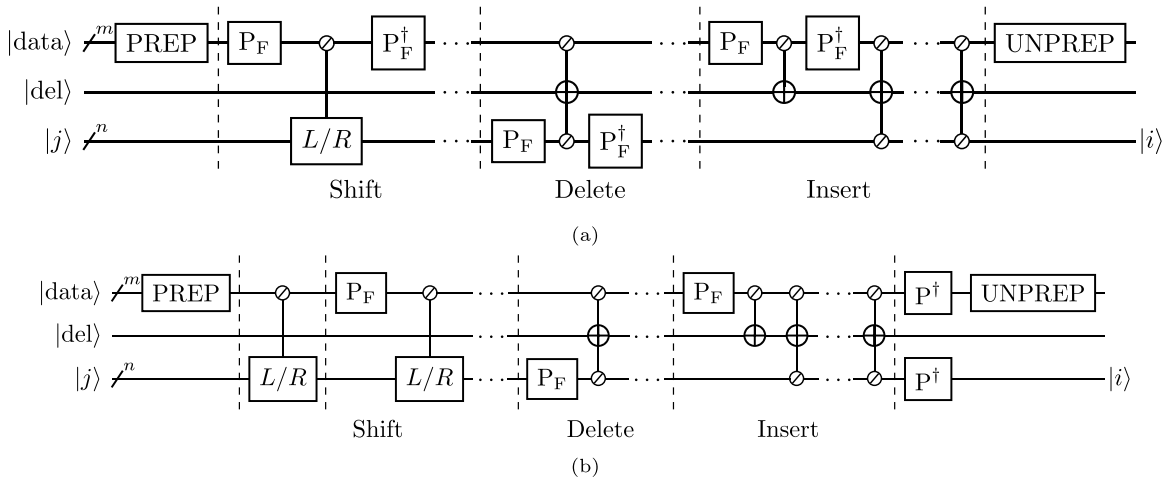


FIG. 6. (a) Circuit representation for block encoding with coherent permutation. The circuit is initialized with m $|\text{data}\rangle$, one $|\text{del}\rangle$, and n matrix ($|j\rangle$) qubits. It includes the PREP and UNPREP operators from the state-preparation oracle (see Sec. III A). The combined shift, delete, and insert operators [Figs. 5(a), 5(d), and 5(f)] are incorporated along with the corresponding A_PERMUTE$_\text{F}$ (represented as P$_\text{F}$) operators for permutation of amplitudes for a chosen F. The dots indicate repeating structures. (b) A second circuit architecture allows the amplitude order to evolve throughout the computation and avoids intermediate inverse permutations, and a single inverse permutation P† is applied at the end.

The cost for architecture of Fig. 6(a) is

$$\mathcal{C}_{\text{opt}}^{(a)} = \mathcal{C}(\text{PREP}) + \mathcal{C}(\text{UNPREP}) + \sum_{r=1}^{R_c} \left(\mathcal{C}(P_r) + \mathcal{C}\left(O_r^{\text{comp}}\right) + \mathcal{C}(P_r^\dagger)\right), \tag{48}$$

where $R_c \leqslant R$, O_r^{comp} denotes the rth compressed index-mapping operator obtained by combining shift, delete, and insert operations, and P_r denotes the coherent permutation operator (Theorem 4).

Similarly, the architecture of Fig. 6(b) can be expressed as

$$\mathcal{C}_{\text{opt}}^{(b)} = \mathcal{C}(\text{PREP}) + \mathcal{C}(\text{UNPREP}) + \sum_{r=1}^{R_c} \left(\mathcal{C}(P_r) + \mathcal{C}\left(O_r^{\text{comp}}\right)\right) + \mathcal{C}(P^\dagger), \tag{49}$$

where intermediate inverse permutations are avoided, rather a single inverse permutation $P^\dagger$ is applied at the end.

The implementation cost of coherent permutation layers is generally nontrivial, since A_PERMUTE itself may admit further compression through the same MCX-reduction techniques (see Theorem 5) and coherent permutation introduced in this work. Consequently, the resource analysis depends on

(1) decomposition of MCX gates into native one- and two-qubit gates,
(2) ancilla requirements,
(3) routing overhead induced by hardware connectivity,
(4) possible further compression of coherent permutation networks.

At the level of the complete block-encoding circuit, the relevant comparison is therefore between

$$\mathcal{C}_{\text{opt}}^{(a)}, \quad \mathcal{C}_{\text{opt}}^{(b)}, \quad \text{and} \quad \mathcal{C}_{\text{orig}}.$$

For the representative examples considered in this work, hardware resource analyses (see Tables II–IV) indicate that the proposed optimization framework can lead to significant reductions in transpiled two-qubit depth for combined shift, delete, and insert operations, including connectivity-aware coherent permutation layers. More generally, the framework provides a systematic methodology for constructing and evaluating hardware-aware block-encoding circuits, where the overall implementation cost is determined by the interplay between MCX compression, permutation synthesis, and hardware connectivity constraints.

IX. APPLICATIONS

In this section, we present two examples of block encoding of sparse matrices: a complex tridiagonal matrix and a structured real matrix.

A. Complex tridiagonal matrix

Consider a complex tridiagonal matrix $A \in \mathbb{C}^{2^n \times 2^n}$ of the form

$$A = \begin{bmatrix} z_2 & z_3 & & & \\ z_1 & z_2 & z_3 & & \\ & \ddots & \ddots & \ddots & \\ & & z_1 & z_2 & z_3 \\ & & & z_1 & z_2 \end{bmatrix}, \tag{50}$$

where $z_1 = \psi_0 + \psi_1 \text{i}, z_2 = \psi_2 + \psi_3 \text{i}, z_3 = \psi_4 + \psi_5 \text{i}$, and $\psi_i \in \mathbb{R}$. The corresponding data vector is $v_{\text{data}} = [|\psi_0|, |\psi_1|, |\psi_2|, |\psi_3|, |\psi_4|, |\psi_5|]^T$ [refer Eq. (10)], with sign vector

$$v_{\text{sign}} = [\text{sgn}(\psi_0), \text{sgn}(\psi_1)\text{i}, \text{sgn}(\psi_2), \text{sgn}(\psi_3)\text{i}, \text{sgn}(\psi_4), \text{sgn}(\psi_5)\text{i}]^T \text{ [refer Eq. (11)]}.$$

State preparation requires $\lceil \log_2 6 \rceil = 3$ data qubits [refer Eq. (12)], where the state is padded with zeros. For block encoding, the data elements $\{\psi_0, \psi_1\}$ in basis states $\{|000\rangle, |001\rangle\}$ must be shifted left by one column and deleted

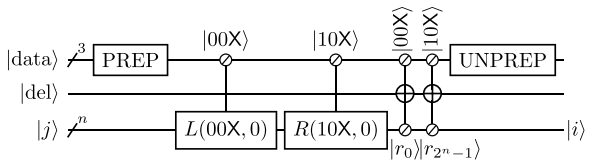


FIG. 7. Circuit representation for block-encoding complex tridiagonal matrix [Eq. (50)]. Basis states $\{|000\rangle, |001\rangle\}$ compressed into $|00\mathsf{X}\rangle$ and $\{|100\rangle, |101\rangle\}$ compressed into $|10\mathsf{X}\rangle$ (refer Theorem 2).

in row $|r_0\rangle$ as in Eq. (23). If required, amplitudes can be permuted to achieve nearest-neighbor MCX gate connectivity. Similarly, the data elements $\{\psi_4, \psi_5\}$ in basis states $\{|100\rangle, |101\rangle\}$ must be shifted right by one column and deleted in row $|r_{2^n-1}\rangle$ as in Eq. (24).

The circuit representation of this construction is shown in Fig. 7 and provides a practical gate-level realization that can be directly employed within quantum algorithms. Note that zeros in the state vector can also be leveraged for combined shifting (see Example 4), thereby further reducing the control overhead of the MCX gates.

B. Structured real matrix

Consider a sparse real matrix $A = \mathbb{R}^{32\times 32}$ with the structure shown in Fig. 8(a). Following the block-encoding procedure outlined in Sec. VIII, the corresponding data vector is $v_{\text{data}} = [|\psi_i|]^T,\ i \in [0, 13]$ and sign vector is $v_{\text{sign}} = [\text{sgn}(\psi_i)]^T,\ i \in [0, 13]$. Since $\dim(v_{\text{data}}) = 14$, we require $m = \lceil \log_2(14) \rceil = 4$ data qubits, and the remaining two basis states are introduced as zero-amplitude padding states.

The block-encoding operations (shift, delete, insert) are summarized in Table V (refer Sec. VII). Note that two distinct values (ψ_2, ψ_3) occur on the main diagonal. Within the block-encoding framework, these appear as the combined diagonal value $\psi_2 + \psi_3$ [see Eq. (18)]. To address this case, we outline three possible encoding strategies with an objective to reduce MCX gates and subnormalization factor:

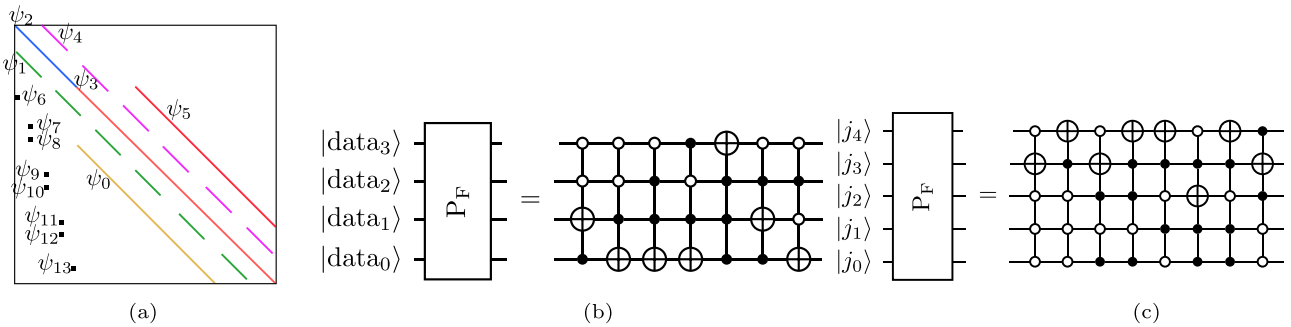
(1) Without modification: ψ_2, ψ_3.
Delete operations: $\psi_2 : D^{|0010\rangle}_{\{5-31\}}$, $\psi_3 : D^{|0011\rangle}_{\{0-4\}}$.
Contribution to α: $|\psi_2| + |\psi_3|$.
(2) With modification: $\tilde{\psi}_2 = \psi_3 - \psi_2$, $\tilde{\psi}_3 = \psi_2$.
Delete operations: $\tilde{\psi}_2 : D^{|0010\rangle}_{\{0-4\}}$, $\tilde{\psi}_3$: Nothing,
Contribution to α: $|\tilde{\psi}_2| + |\tilde{\psi}_3|$.
This is advantageous when $|\psi_3 - \psi_2| < |\psi_3|$.
(3) With modification: $\tilde{\psi}_2 = \psi_3$, $\tilde{\psi}_3 = \psi_2 - \psi_3$.
Delete operations: $\tilde{\psi}_2$: Nothing, $\tilde{\psi}_3 : D^{|0011\rangle}_{\{5-31\}}$.
Contribution to α: $|\tilde{\psi}_2| + |\tilde{\psi}_3|$.
This is advantageous when $|\psi_2 - \psi_3| < |\psi_2|$.

TABLE V. Block-encoding operations for Fig. 8(a), with data vector $v_{\text{data}} = [|\psi_i|]_{i=0}^{15}$. Here, $\tilde{\psi}_2, \tilde{\psi}_3$ denote modified values of ψ_2, ψ_3 in the data vector. Shift operations are given by $O_{\text{shift}}(k, d)$ [refer Eq. (19)]. Deletion operations are given by $O_{\text{del}}(k, r) = D^{|k\rangle}_{r_i}$ [refer Eq. (25)]. Similarly, insert operations are represented as $I^{|k\rangle}_{r_i}$.

Block-encoding operations
$\psi_0,\ O_{\text{shift}}(0000, d = 5),\ D^{\|0000\rangle}_{\{0-9,30,31\}}$
$\psi_1,\ O_{\text{shift}}(0001, d = 1),\ D^{\|0001\rangle}_{\{0,5,10,15,20,25,30,31\}}$
$\tilde{\psi}_2,\ D^{\|0010\rangle}_{\{0-4\}}$
$\tilde{\psi}_3$
$\psi_4,\ O_{\text{shift}}(0100, d = -1),\ D^{\|0100\rangle}_{\{4,9,14,19,24,29,30,31\}}$
$\psi_5,\ O_{\text{shift}}(0101, d = -5),\ D^{\|0101\rangle}_{\{0-4,25-31\}}$
$\psi_6,\ O_{\text{shift}}(0110, d = 6),\ I^{\|0110\rangle}_{6}$
$\psi_7,\ O_{\text{shift}}(0111, d = 9),\ I^{\|0111\rangle}_{10}$
$\psi_8,\ O_{\text{shift}}(1000, d = 11),\ I^{\|1000\rangle}_{12}$
$\psi_9,\ O_{\text{shift}}(1001, d = 14),\ I^{\|1001\rangle}_{16}$
$\psi_{10},\ O_{\text{shift}}(1010, d = 16),\ I^{\|1010\rangle}_{18}$
$\psi_{11},\ O_{\text{shift}}(1011, d = 19),\ I^{\|1011\rangle}_{22}$
$\psi_{12},\ O_{\text{shift}}(1100, d = 21),\ I^{\|1100\rangle}_{24}$
$\psi_{13},\ O_{\text{shift}}(1101, d = 24),\ I^{\|1101\rangle}_{28}$
$\psi_{14} = 0$
$\psi_{15} = 0$

For demonstration, we choose the second approach as illustrated in Table V. Next, we determine the common shift operators (refer Sec. VII), shown in Table VI.

We demonstrate the optimized index-mapping oracle for the block-encoding operations in Table V. For demonstration purposes, we consider the shift $L(k, 0)$ in Table VI. For $L(k, 0)$, the control states S_2 does not satisfy the structural requirements in Theorem 2. Therefore, we determine $F = \{|\text{data}_0\rangle\}$ and apply the permutation operator A_PERMUTE$_F$ as shown in Fig. 5(a). Following Theorem 3, the mapping $\phi : S_2 \mapsto S_3$ is obtained by solving the linear sum assignment problem [37]. Finally, the gates are generated to permute



(a) (b) (c)

FIG. 8. (a) Visual representation of the example sparse matrix structure (see Sec. IX B). The data elements are denoted by $[\psi_i]_{i=0}^{13}$. Continuous lines of the same color indicate the repeating pattern of a data element (ψ_i) along a diagonal d, while black square dots denote single, nonrepeating data elements. (b) Coherent permutation on the data qubits [Eq. (51)]. (c) Coherent permutation on the matrix qubits [Eq. (52)].

TABLE VI. Common shift operations of the data elements (Table V). Here, $(\psi_k, |k\rangle)$ denotes kth data element in $|k\rangle$ basis state represented in binary. Note that the $L(k,0)$ shift involves six data elements; however, as illustrated in Example 4, the zeros in ψ_{14} and ψ_{15} can be exploited to extend this to eight shifts.

	Common shift operators
$L(k,0)$	$(\psi_0, 0000)$, $(\psi_1, 0001)$, $(\psi_7, 0111)$, $(\psi_8, 1000)$, $(\psi_{11}, 1011)$, $(\psi_{12}, 1100)$, $(\psi_{14}, 1110)$, $(\psi_{15}, 1111)$
$L(k,1)$	$(\psi_6, 0110)$, $(\psi_8, 1000)$, $(\psi_9, 1001)$, $(\psi_{11}, 1011)$
$L(k,2)$	$(\psi_0, 0000)$, $(\psi_6, 0110)$, $(\psi_9, 1001)$, $(\psi_{12}, 1100)$
$L(k,3)$	$(\psi_7, 0111)$, $(\psi_8, 1000)$, $(\psi_9, 1001)$, $(\psi_{13}, 1101)$
$L(k,4)$	$(\psi_{10}, 1010)$, $(\psi_{11}, 1011)$, $(\psi_{12}, 1100)$, $(\psi_{13}, 1101)$
$R(k,0)$	$(\psi_4, 0100)$, $(\psi_5, 0101)$
$R(k,2)$	$(\psi_5, 0101)$

$(S_2 \mapsto S_3)$ according to Definition 2, resulting in the following local walk operators:

$$\begin{aligned}
&\psi_0 \;\; 0000,\\
&\psi_1 \;\; 0001 \to 0011 \to 0010,\\
&\psi_7 \;\; 0111 \to 0110,\\
&\psi_8 \;\; 1000,\\
&\psi_{11} \;\; 1011 \to 1010,\\
&\psi_{12} \;\; 1100,\\
&\psi_{14} \;\; 1110,\\
&\psi_{15} \;\; 1111 \to 0111 \to 0101 \to 0100. \qquad (51)
\end{aligned}$$

The MCX gates implementing the mapping in Eq. (51) (see Theorem 4) correspond to the operator A_PERMUTE$_F$, as illustrated in Fig. 8(b). Alternatively, one may employ multiswapping strategies as discussed in Theorem 5.

Considering the combined delete $D^{|0001\rangle}_{\{0,5,10,15,20,25,30,31\}}$ (refer Table V), the control states S_2 require permutation. Therefore, we determine $F = \{|j_4 j_3\rangle\}$ and implement A_PERMUTE$_F$, as shown in Fig. 5(d). Following the same protocol as for the previous case, we obtain the mapping $\phi : S_2 \mapsto S_3$, leading to the following local walk operators:

$$\begin{aligned}
&0 \;\; 00000 \to 01000 \to 11000,\\
&5 \;\; 00101 \to 01101 \to 11101,\\
&10 \;\; 01010 \to 11010,\\
&15 \;\; 01111 \to 01011 \to 11011,\\
&20 \;\; 10100 \to 11100,\\
&25 \;\; 11001,\\
&30 \;\; 11110,\\
&31 \;\; 11111. \qquad (52)
\end{aligned}$$

The MCX gates corresponding to the mapping in Eq. (52) (see Theorem 4) are implemented through A_PERMUTE$_F$, as illustrated in Fig. 8(c). The complete circuit for block encoding the matrix Fig. 8(a) is obtained by combining the shift, delete, insert, and permutation operators, as shown in Fig. 6(a). This circuit provides a practical gate-level realization that can be directly employed within quantum algorithms.

X. DISCUSSION

In this work, we developed a framework for block encoding of sparse matrices with hardware-aware circuit constructions and local reductions in MCX control complexity. The construction provides an explicit bridge from oracle-level block encoding to gate-level implementations. Using existing state-preparation primitives, we focus primarily on optimizing index-mapping oracles, including shift, delete, insert, and coherent permutation operations.

The framework allows an entrywise construction for general sparse matrices. However, its principal efficiency benefits are expected for exploitable structure. Highly irregular sparse matrices with predominantly distinct entries and nonrepetitive diagonal structure may exhibit higher index-mapping overhead due to limited opportunities for MCX compression.

At the circuit level, we show that several shift, delete, and insert operations can be combined through MCX compression when the associated control structures admit favorable grouping. When such direct compression is not available, we formulate the resulting basis-state reordering as a coherent permutation problem and propose a routing-based construction. This yields an optimized index-mapping oracle that is compatible with hardware-aware circuit synthesis and can reduce long-range interactions by enabling nearest-neighbor-friendly MCX structures under appropriate connectivity assumptions. However, the overall resource cost depends not only on MCX compression but also on the decomposition of MCX gates into native operations and the routing overhead induced by the underlying hardware connectivity graph.

The hardware resource analyses on representative examples (see Tables II–IV) indicate that the proposed optimization framework can lead to significant reductions in circuit depth. The complete block-encoding construction was illustrated through two circuit architectures, and representative examples were used to demonstrate the end-to-end pipeline from oracle-level formulations to explicit gate-level implementations. The resulting circuits are directly applicable to key quantum algorithms—including generalized frameworks such as QSVT as well as specific applications like Hamiltonian simulation and quantum linear system solvers.

ACKNOWLEDGMENTS

This research was funded through the European Union's Horizon Programme (HORIZONCL4-2021-DIGITALEMERGING-02-10), Grant Agreement No. 101080085 (QCFD).

DATA AVAILABILITY

There are no publicly available research data or software supporting this manuscript. Requests for further information or data should be sent to the authors.

[1] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics, in *Proceedings of the 51st annual ACM SIGACT Symposium on Theory of Computing* (ACM, 2019), pp. 193–204.

[2] J. M. Martyn, Z. M. Rossi, A. K. Tan, and I. L. Chuang, Grand unification of quantum algorithms, PRX Quantum **2**, 040203 (2021).

[3] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. **103**, 150502 (2009).

[4] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, Simulating Hamiltonian dynamics with a truncated Taylor series, Phys. Rev. Lett. **114**, 090502 (2015).

[5] A. M. Childs, R. Kothari, and R. D. Somma, Quantum algorithm for systems of linear equations with exponentially improved dependence on precision, SIAM J. Comput. **46**, 1920 (2017).

[6] A. M. Childs and N. Wiebe, Hamiltonian simulation using linear combinations of unitary operations, Quantum Inf. Comput. **12**, 901 (2012).

[7] F. G. Brandao and K. M. Svore, Quantum speed-ups for solving semidefinite programs, in *IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE, Berkeley, CA, USA, 2017), pp. 415–426.

[8] J. Van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf, Quantum SDP-solvers: Better upper and lower bounds, Quantum **4**, 230 (2020).

[9] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, Encoding electronic spectra in quantum circuits with linear T complexity, Phys. Rev. X **8**, 041015 (2018).

[10] B. D. Clader, A. M. Dalzell, N. Stamatopoulos, G. Salton, M. Berta, and W. J. Zeng, Quantum resources required to block-encode a matrix of classical data, in *IEEE Transactions on Quantum Engineering* (IEEE, 2022), Vol. 3, pp. 1–23.

[11] S. Chakraborty, A. Gilyén, and S. Jeffery, The power of block-encoded matrix powers: Improved regression techniques via faster Hamiltonian simulation, in *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming* (ICALP, 2019).

[12] D. Camps and R. Van Beeumen, FABLE: Fast approximate quantum circuits for block-encodings, in *IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, Broomfield, CO, USA, 2022), pp. 104–113.

[13] P. Kuklinski and B. Rempfer, S-FABLE and LS-FABLE: Fast approximate block-encoding algorithms for unstructured sparse matrices, arXiv:2401.04234.

[14] Z. Li, X.-M. Zhang, C. Yang, and G. Zhang, Binary tree block encoding of classical matrix, in *IEEE Transactions on Quantum Engineering* (IEEE, 2025), Vol. 7.

[15] D. Camps, L. Lin, R. Van Beeumen, and C. Yang, Explicit quantum circuits for block encodings of certain sparse matrices, SIAM J. Matrix Anal. Appl. **45**, 801 (2024).

[16] C. Sünderhauf, E. Campbell, and J. Camps, Block-encoding structured matrices for data input in quantum computing, Quantum **8**, 1226 (2024).

[17] C. Yang, H. Yao, Z. Li, Z. Fan, G. Zhang, and J. Liu, Block encoding of sparse structured matrices coming from ocean acoustics in quantum computing, Quantum **9**, 1805 (2024).

[18] N. M. Linke, D. Maslov, M. Roetteler, S. Debnath, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, Experimental comparison of two quantum computing architectures, Proc. Natl. Acad. Sci. USA **114**, 3305 (2017).

[19] R. Beals, S. Brierley, O. Gray, A. W. Harrow, S. Kutin, N. Linden, D. Shepherd, and M. Stather, Efficient distributed quantum computing, Proc. R. Soc. A **469**, 20120686 (2013).

[20] S. A. Kutin, D. P. Moulton, and L. M. Smithline, Computation at a distance, arXiv:quant-ph/0701194.

[21] V. V. Shende, S. S. Bullock, and I. L. Markov, Synthesis of quantum logic circuits, in *Proceedings of the 2005 Asia and South Pacific Design Automation Conference* (2005), pp. 272–275.

[22] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits, in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (IEEE, 2013), Vol. 32, pp. 818–830.

[23] A. Cowtan, S. Dilkes, R. Duncan, A. Krajenbrink, W. Simmons, and S. Sivarajah, On the qubit routing problem, in *14th Conference on the Theory of Quantum Computation, Communication and Cryptography*, Leibniz International Proceedings in Informatics (TQC, 2019).

[24] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers, in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (ACM, 2019), pp. 1015–1029.

[25] R. W. Hamming, Error detecting and error correcting codes, Bell Syst. Tech. J. **29**, 147 (1950).

[26] J. Li, S. Lin, K. Yu, and G. Guo, Quantum k-nearest neighbor classification algorithm based on Hamming distance, Quantum Inf. Proc. **21**, 18 (2022).

[27] M. Mottonen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, Transformation of quantum states using uniformly controlled rotations, Quantum Inf. Comput. **5**, 467 (2004).

[28] M. Möttönen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, Quantum circuits for general multiqubit gates, Phys. Rev. Lett. **93**, 130502 (2004).

[29] R. Iten, R. Colbeck, I. Kukuljan, J. Home, and M. Christandl, Quantum circuits for isometries, Phys. Rev. A **93**, 032318 (2016).

[30] X.-M. Zhang, T. Li, and X. Yuan, Quantum state preparation with optimal circuit depth: Implementations and applications, Phys. Rev. Lett. **129**, 230504 (2022).

[31] H. W. Kuhn, The Hungarian method for the assignment problem, Nav. Res. Logist. Q. **2**, 83 (1955).

[32] J. Munkres, Algorithms for the assignment and transportation problems, J. Soc. Ind. Appl. Math. **5**, 32 (1957).

[33] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*, Other Titles in Applied Mathematics (SIAM, 2012).

[34] L. A. Wolsey, *Integer Programming* (John Wiley & Sons, 2020).

[35] A. Schrijver *et al.*, *Combinatorial Optimization: Polyhedra and Efficiency* (Springer, Berlin, Heidelberg, 2003), Vol. 24.

[36] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms* (Springer, Berlin, Heidelberg, 2008).

[37] D. F. Crouse, On implementing 2D rectangular assignment algorithms, in *IEEE Transactions on Aerospace and Electronic Systems* (IEEE, 2016), Vol. 52, pp. 1679–1696.