scientific reports

Check for updates

OPEN

Quantum circuits from non-unitary sparse binary matrices

Krishnageetha Karuppasamy✉, Varunteja Puram, K. M. George & Thomas P. Johnson

Quantum computing leverages unitary matrices to perform reversible computations while preserving probability norms. However, many real-world applications involve non-unitary sparse matrices, posing a challenge for quantum implementation. This paper introduces a novel method for transforming a class of non-unitary sparse binary matrices into higher-dimensional permutation matrices, ensuring unitarity. Our approach is efficient in both space and time, ensuring practical applicability to large-scale problems. We demonstrate the utility of this transformation in constructing quantum gates and apply the method to model quantum finite state machines (QFSMs) derived from classical deterministic finite automata (DFAs). This work offers a practical pathway for integrating non-unitary transformations into quantum systems, with implications for the many applications that are based on sparse, non-unitary matrices. The significance of this work for automata theory and quantum computation is outlined.

Quantum computing has the potential to revolutionize a wide range of scientific fields, including cryptography, drug discovery, climate modeling, finance, and artificial intelligence. Unlike classical computing, which relies on binary bits, quantum computing uses qubits, which can exist in superposition, allowing them to represent multiple states simultaneously. This unique property enables quantum computers to perform complex computations at exceptional speeds compared to classical computers.

Unitary matrices are crucial in quantum computing, where quantum gates are represented as unitary operators acting on qubits. Unitary matrices preserve two key properties of quantum systems: reversibility and probability conservation. In quantum mechanics, the evolution of a system must be reversible to ensure that no information is lost over time[1]. Additionally, the sum of probabilities for all possible states of a quantum system must always equal 1(the norm of any quantum state is 1). These properties are embedded in unitary matrices, whose structure ensures that quantum gates can be inverted, and that the quantum state's total probability remains unchanged throughout the computation process. This preservation is achieved by unitary transformations as unitary matrices preserve the norm of vectors representing quantum states.

The motivation for this work can be found in non-Hermitian quantum Physics which is rooted in the observation that Hermiticity is a sufficient (not necessary) condition for real eigenvalues. Recently, there have been many research papers published on this topic highlighting theory and applications. The works presented in[2,3] are based on the parity-time-symmetric (PT-symmetric) Hamiltonian theory. PT-symmetric Hamiltonians do not guarantee the evolution operator is unitary. References[4–6] are based on open system formalism. In this case, a closed quantum system interacts with the environment. The significance of non-Hermitian or open system time evolution is that it can be non-unitary. Hence from the perspective of quantum computation, representation of non-unitary matrices as quantum circuits is necessary.

The duality computer, introduced in[7], is a theoretical model that extends the conventional quantum computer by allowing wave functions to coherently split and recombine along multiple paths. In this framework, a *multi-dubit* (duality bit) system propagates along two spatially identical paths. When these sub-waves recombine at the Quantum Wave Combiner (QWC), they interfere constructively because their spatial modes remain in phase. If only a single path is used, the duality computer effectively reduces to a standard quantum computer. This architectural extension provides additional flexibility in manipulating quantum information and lays the foundation for new algorithmic strategies.

An algorithm named LCU (linear combination of unitaries) algorithm is developed based on a duality computer to synthesize a quantum circuit. The LCU algorithm expresses a non-unitary matrix A as the linear combination of unitary matrices. The resulting circuit is equivalent to embedding the matrix A as the principal block of a higher dimensional unitary matrix $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$. Zheng[8] applies the LCU algorithm to simulate a

Department of Computer Science, Oklahoma State University, Stillwater 74075, USA. ✉email: kkarupp@okstate.edu

single qubit non-unitary operator. A quantum circuit with two qubits is demonstrated using the LCU algorithm where the 2×2 non-unitary matrix is decomposed as a linear combination of Pauli matrices. All these works suggest that non-unitary operations and embedding them into quantum circuits are necessary to synthesize quantum systems. In our work, the non-unitary matrix A is assumed to be sparse and binary.

Despite these advances, many practical systems such as those arising from Partial Differential Equations (PDEs) involve non-unitary boundary conditions due to errors, noise, or model approximations[9,10]. PDEs are foundational in modeling real-world phenomena, and quantum computing offers the potential to reduce the cost of solving them. However, to leverage quantum algorithms for such applications, efficient conversion of non-unitary matrices into unitary forms is essential.

Our proposed approach provides a structurally simple and resource-efficient alternative for a special class of non-unitary matrices. Specifically, we address non unitary sparse binary matrices with norms greater than one and show how they can be embedded into higher-dimensional permutation matrices, which are naturally unitary.

1. Our main contribution in this paper is the introduction of a novel method for converting non-unitary sparse binary matrices with a norm greater than one into higher-dimensional unitary matrices. Specifically, we focus on transforming $n \times n$ square matrices that have atmost n nonzero entries, with each row containing no more than one nonzero element, into higher-dimensional permutation matrices. To the best of our knowledge, no prior work addresses this specific class of transformations.
2. An important advantage of our method is that the resulting permutation matrices are not only unitary but also allow for efficient quantum gate construction. Each permutation can be decomposed into a sequence of transpositions, which can be implemented as a series of SWAP or CNOT operations acting on binary encodings of the indices. This facilitates direct synthesis of permutation-based unitaries into hardware-efficient quantum circuits.
3. As an application, we demonstrate the implementation of a Quantum Finite State Machine (QFSM) where the initial transformation matrix is non-unitary. Our results provide an effective solution to this challenge, ensuring the preservation of unitary properties essential for quantum computations.

Related works

Our work aims to construct permutation matrices from a certain class of non-unitary matrices. With that focus, we review related work. In[11] Robert M. Gingrich and Colin P. Williams, addressed the problem of computing non-unitary operators probabilistically and presented a method to convert a non-unitary matrix to a unitary matrix. They construct the quantum circuit for the operation $\rho \rightarrow \frac{M\rho M^\dagger}{tr\left(M\rho M^\dagger\right)}$ where M is non-unitary and ρ a density matrix and † is the complex conjugate transpose. The method presented first converts the non-unitary matrix M into a high dimensional unitary matrix by padding zeros. The unitary matrix U is obtained by the transformation $U = e^{i\varepsilon \begin{bmatrix} 0 & -iN \\ iN^\dagger & 0 \end{bmatrix}}$. The computation introduces an ancilla qubit and approximates the computation as $\rho' = U\left(|1><1| \otimes \rho\right) M^\dagger$. However, the most significant drawback is its non-deterministic nature. The success probability depends on the norm of the operator, often requiring multiple repetitions that increase circuit depth and error accumulation. Additionally, constructing the required higher-dimensional unitary embedding incurs gate overhead and may disturb the quantum state upon failure. These factors limit the scalability and efficiency of the approach, especially for large systems or complex operators.

Childs and Wiebe[12] introduced the LCU method. In this approach, a non-unitary matrix is expressed as a weighted sum of unitary operators: $A = \sum_i a_i U_i$, where each U_i is a unitary (e.g., a Pauli string). The LCU framework constructs a quantum circuit that uses ancilla qubits to encode the coefficients a_i, performs controlled-unitary operations, and applies oblivious amplification (OAA) to boost the success probability of the correct evolution. This method provides a powerful, general-purpose way to simulate time evolution e^{-iHt} where t is time evaluation and H is Hermitian matrix. But, since it is probabilistic and requires ancilla-driven controlled operations and post-selection or amplitude amplification, which introduces additional circuit depth and ancilla overhead. In contrast, our proposed method, enabling deterministic and low-depth circuit implementations without probabilistic post-selection. This offers a resource-efficient and hardware-friendly alternative for representing non-unitary operations within a fully unitary framework.

Lin[13] proposed a block encoding method. This approach embeds a general matrix (which may not be unitary) into a higher-dimensional unitary matrix, enabling quantum algorithms to process non-unitary matrices through additional quantum operations. This technique has been foundational in quantum algorithms for efficiently representing complex matrices. Given a matrix A, which may not be unitary, block encoding allows constructing a unitary matrix U such that A is a submatrix of U. Formally, we express U = $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$ In this approach, *A* represents the original matrix, while *B*, *C*, and *D* are selected to ensure that *U* is unitary. This transformation allows quantum algorithms to handle non-unitary matrices by simulating unitary operations within a higher-dimensional space. While the size of *U* increases linearly, the method relies on singular value decomposition method, which has a computational complexity of $O(n^3)$. In contrast, our proposed method delivers significant improvements in both computational efficiency and practical implementation as described later.

In[14], George Cybenko addresses the challenge of simplifying complex quantum computations into sequences of elementary quantum operations. Re et al.[6] demonstrates any general unitary operation can be represented

as a sequence of elementary quantum gates. The paper also discusses the importance of maintaining specific properties, like unitarity and control, during the decomposition process. The work also raises questions about the efficiency and feasibility of implementing these reductions in practice, particularly regarding the exponential number of operations required as the number of qubit increases, which remains a significant challenge in the field of quantum computing.

Planat et al.[15] in the section "From permutations to quantum gates", establish a link between classical permutation matrices and quantum gates. Permutation matrices, which rearrange elements by placing 1 s in specific positions and 0 s elsewhere, can describe certain quantum gates, especially the Controlled-NOT (CNOT) gate. The authors introduce a particular subset of these matrices, termed "magic" permutation matrices, which have 1 s on the main diagonal. These matrices correspond to essential quantum gates, such as the Pauli X gate, the CNOT gate, and the Toffoli gate, which are foundational in constructing multi-qubit quantum gates and generating quantum states like stabilizer states and "magic" states. However, this paper focuses only on a limited group of permutation matrices, those that correspond directly to these specific quantum gates. It does not extend the analysis to the general class of permutation matrices, nor does it explore the broader applicability of permutation matrices in quantum gate design.

Weber[16] provides several examples of quantum permutation matrices, illustrating how these matrices can arise from combinations of classical operations but with quantum behavior embedded. One of the key examples discussed involves Pauli matrices combined with unitary transformations, generating a quantum permutation matrix from their tensor products. The study of quantum permutation matrices extends to their application in quantum isomorphisms of graphs, which allows quantum analogs of graph isomorphisms. These quantum isomorphisms provide a broader symmetry framework for graph structures in quantum settings, indicating that quantum symmetries can go beyond classical permutations.

One application of our work in constructing permutation matrices from non-unitary matrices can be seen in the domain of quantum automata. Similar to the work on Quantum Automata and Quantum Grammars by Moore and Crutchfield[17], where unitary matrices are used to model quantum versions of classical computational structures like finite state machines and pushdown automata, our approach provides a method for handling non-unitary matrices. In their models, unitary matrices were connected to alphabets and grammar symbols, which are applied during state transitions in Hilbert space.

In this article, we present a method to convert non-unitary sparse binary matrices into permutation matrices, enabling their use in quantum implementations of finite state machine (QFSM) models with initial non-unitary transformation matrices. Our approach offers a practical solution for integrating non-unitary transformations within a quantum framework and provides a pathway for mapping permutation matrices to quantum gates.

Another key contribution of our work is the ability to map the resulting permutation matrices directly to quantum gate sequences. Each permutation is decomposed into a series of transpositions, which can be implemented using a small set of hardware-efficient gates such as CNOT and SWAP. This decomposition allows for the systematic construction of low-depth quantum circuits, making our method highly suitable for noisy intermediate-scale quantum (NISQ) devices.

Our work offers a novel solution to the challenge of maintaining unitary properties essential for quantum computation when starting from non-unitary matrices. Our method produces a unitary matrix of size $np \times np$, significantly reducing resource requirements compared to other approaches. Additionally, our method enables effective handling of non-unitary matrices within quantum systems, specifically in QFSM models, where unitary matrices are associated with alphabets and grammar symbols that facilitate state transitions in a Hilbert space.

Proposed method for sparse to permutation matrix conversion

In this section we outline our proposed approach to create a permutation matrix from a non-unitary sparse binary matrix. The class of matrices we consider are sparse binary matrices that have at most one nonzero entry in a row. Our method is presented via the two propositions stated below.

Proposition 1 *Let* $T \epsilon R^{n\times n}$ *matrix with entries* $T_{ij} = \left\{ \begin{array}{c} 0; 1 \leq j \leq n-1 \\ 1; j = 0 \end{array} \right\}$. *Then there is a permutation matrix* $M \epsilon R^{m\times m}$ *where* $m = n^2 \times n^2$ *such that* $M = \left[\begin{array}{ccc} M_{00} & \cdots & M_{0(n-1)} \\ \vdots & \ddots & \vdots \\ M_{(n-1)0} & \cdots & M_{(n-1)(n-1)} \end{array} \right]$ *and* $T = \sum_{i=0}^{n-1} M_{01}$ *where* $M_{kl} is\ a\ n \times n$ *block matrices.*

Proof Given T, construct a matrix A of dimension $m = n^2 \times n^2$ using tensor product $A = I_{n\times n} \otimes T_{n\times n}$ where I is the identiy matrix. Observe that A is a diagonal block matrix with every diagonal block being T. Also, only the first column of T is nonzero with all elements 1. So, we can split A into $m \times \mathrm{m}$ matrices A_t such that $A = \sum_{t=0}^{m-1} A_t$, where all elements of A_t are 0's except the element $A_t\,[t, k]\,, where\ k = n \times \mathrm{floor}\ \left(\frac{t}{n}\right)$. Apply mod m column permutation on each A_t to move the 1's into distinct columns. Let $M = A_0 + \sum_{t=1}^{n-1} A_t\ \mathrm{P}\,(0,\ \mathrm{nt} + 0) + \sum_{t=1}^{\mathrm{n}} \mathrm{A}_t\ \mathrm{P}(\mathrm{n}, (\mathrm{nt} + 1)\ \mathrm{mod\ m}\ \ \ldots . + \sum_{t=1}^{n} \mathrm{A}_t \mathrm{P}\,(\mathrm{m} - \mathrm{n})\,, ((\mathrm{m} - \mathrm{n})\,\mathrm{t} + (\mathrm{n} - 1)\,\mathrm{m}\ \mathrm{mod\ m},$ where $P \epsilon R^{m\times m}$ is a permutation matrix. M is a permutation matrix as each column and row has exactly one nonzero entry which is 1 by construction and is unitary. Hence, we can represent M as an $m \times \mathrm{m}$ block matrix with block size of n and having exactly one nonzero element. Also, for the first row of the block matrix the nonzero entries are $M_{0l}\,[l, 0] = 1$ where the first row of blocks is $M_{0l}, 0 \leq l \leq n-1$.

Furthermore, each step of the construction of M is well defined and hence given a final $n^2 \times n^2$ permutation matrix, we can determine the initial $n \times n$ matrix. If all entries of column k rather than column 0 are 1's, then we can apply the permutations $P(k, nt + k)$, $P(n + k, nt + k + 1)$ etc. to construct the matrix M. Thus, we can extend Proposition 1 to any matrix T with all columns but one are zeros and the nonzero column is $[\ 1 \quad 1 \quad \ldots, \quad 1\ ]^T$.

Properties of M

1. The matrix M is an $n^2 \times n^2$ block matrix, represented as follows:

$$M = \begin{bmatrix} M_{00} & \cdots & M_{0(n-1)} \\ \vdots & \ddots & \vdots \\ M_{(n-1)0} & \cdots & M_{(n-1)(n-1)} \end{bmatrix}$$, where each $M_{kl}, 0 \leq k, l \leq n-1$ is an $n \times n$ matrix. Each block M_{kl} contains exactly one nonzero entry, located at position $[x, y]$, with $M_{kl}[x, y] = 1$. For a fixed k, if $l = k$ then $x = 0$ and $y = k$, if $l = k + 1$, then $x = 1$ and $y = k$ and so on. If $l = n$, it is reset to $l = 0$ and the process continue until $l = k - 1$. So, in the overall matrix, the first n rows each have exactly one nonzero entry positioned at columns $0, n, 2n \ldots$, respectively. Matrix M is obtained by first taking the tensor product of the matrix T with identity matrix, creating a block diagonal matrix A with diagonal blocks equal to T. Then, columns in A are permuted in such a way that only one row in each n-row block is shifted per permutation. As a result, each n -row block of M corresponds to a column-permuted version of T, with distinct permutations applied across blocks.

$$\sum_{0}^{n-1} M_{0l} = T$$

2. Hence if v is an n-dimensional vector then $(\sum_0^{n-1} M_{0l})v = Tv$. The matrices $\sum_0^{n-1} M_{kl}$ are column permutations of T starting with the first column.
3. Let v be an n-dimensional vector and $\overrightarrow{1} = [1, 1, \cdots, 1]^T$ be the n-dimensional vector with all 1's. Then $M\left(\vec{1} \otimes v\right) = \left[\left(\sum_{0}^{n-1} M_{0l}, \sum_{0}^{n-1} M_{1l}, \cdots, \sum_{0}^{n-1} M_{(n-1)l}\right) v\right]^T$.
4. From the above properties, Tv constitutes the first *n* elements of $M\left(\vec{1} \otimes v\right)$.

Example 1 To illustrate the construction described above, we provide an example demonstrating how to derive a unitary matrix from a non-unitary sparse binary matrix. Consider a matrix T of dimensions 2×2 In this matrix, exactly 2 entries are 1's, all of which are in the first column, while the remaining entries are 0.

Given $n = 2$ and $T = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$ by applying Proposition 1, we get:

Step 1 compute $A = I_{n \times n} \otimes T_{n \times n}$ which gives

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Step 2 Compute the individual matrices $A_i s$

$$A_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} A_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} A_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} A_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Step 3 Compute $A_i P$'s

$$A_1 P_{(0,2)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} A_2 P_{(2,3)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} A_3 P_{(2,1)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Step 4 Finally, Build the matrix M

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The general case

In the previous section, we presented and demonstrated a method for constructing a unitary matrix from a non-unitary sparse binary matrix, where all non-zero entries are confined to the first column. In this section, we extend our discussion to the general case of an $n \times n$ sparse binary matrix that contains at most n non-zero entries. These non-zero entries are distributed such that some columns contain only zeros, while others contain more than one non-zero entry. The method for constructing a unitary matrix in this context is outlined in the following proposition.

Proposition 2 *Let $T \epsilon R^{n\times n}$ be a $n \times n$ matrix in which every row has exactly one nonzero element, with entries $T_{ij} \in \{0, 1\}$. The matrix T has exactly n entries are 1's and some of its columns consist entirely of zero entries. Then there is a permutation matrix $M \epsilon R^{m\times m}$ where $m = np$ and p is the maximum number of nonzero elements in any column of T . Further, M is structured as a block matrix: $\begin{bmatrix} M_{00} & \cdots & M_{0(p-1)} \\ \vdots & \ddots & \vdots \\ M_{(p-1)0} & \cdots & M_{(p-1)(p-1)} \end{bmatrix}$ where each M_{kl} is an $n \times n$ block matrix. Additionally, we can express $T = \sum_{i=0}^{p-1} M_{0i}$.*

Proof We will prove the statement by constructing the matrix M based on the given matrix T as follows:

Let T be a $n \times n$ binary matrix with q nonzero columns and let p = maximum number of nonzero entries in a column of T. We construct $n \times n$ matrices $T_0, T_1 \ldots, T_{q-1}$, where each T_j, has only one nonzero column. Specifically:

- The nonzero column of T_0 is corresponds to the first nonzero column of T
- The nonzero column of T_1 is corresponds to the second nonzero column of T and so on.

Then, we can express T as sum of these matrices: $T = \sum_{i=0}^{q-1} T_i$.

For each $i = 0 \cdots q-1$, construct the $np \times np$ matrix $A_i = I \otimes T_i$, where I is the $p \times p$ identity matrix. Since T contains exactly one nonzero element in each row, every distinct pair T_i and T_j will have the property that if one row of one matrix is nonzero, the corresponding row in the other will be zero, then A_i block diagonal matrix equal to $\begin{bmatrix} T_i & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & T_i \end{bmatrix}$. If is the first nonzero column of A_i, then the other nonzero columns are $n + l, 2n + l, \ldots (p-1)\, n + l$ Each nonzero segment of the nonzero column occurs in the diagonal blocks. We follow the method used in the proof of 'Proposition 1' to express A_i as the sum of m matrices. Let A_i^j be the matrix whose j^{th} row is a copy of the matrix A_i and other rows are zeros. Viewed as a block matrix, all blocks are 0 matrices except the one enclosing the j^{th} row which has exactly one nonzero element. Each A_i^j has exactly one nonzero element in one of the columns $l, n + l, 2n + l, \ldots (p-1)\, n + l$ in one of the diagonal blocks. Then $A_i = \sum_{j=0}^{m-1} A_i^j$. Nonzero columns of A_i^j may be the same if A_i has more than one nonzero element in a column. We use column permutations to align nonzero element of A_i^j so that no two matrices have nonzero elements in the same column. For the elements in column l (those are in the first diagonal block), use the permutations $P(l, ns + s + 1)$ where $s = 0, \ldots, n-1$. For elements of column $nr + l$, where $r = 1, \ldots p-1$(those are in the r^{th} diagonal block), use permutations $P(nr + l, (nr + l + ns + r)\, mod m$. The relative positions of the nonzero elements in the column from the first are given by s. Let B_i^j denote A_i^j following the outlined permutation. Let $B_i = \sum_{j=0}^{m-1} B_i^j$. Then B_i has at most one nonzero element (which is 1) in each row and column. Also, by construction for any pair B_i and B_j there is no nonzero intersection of elements. Let $M = \sum_{i=0}^{q-1} B_i$. Since T has exactly n elements, M has exactly m = np elements and no row or column contains more than one nonzero element. So, M is a permutation matrix. If we represent M as a $p \times p$ block matrix, $M = \begin{bmatrix} M_{00} & \cdots & M_{0(p-1)} \\ \vdots & \ddots & \vdots \\ M_{(p-1)0} & \cdots & M_{(p-1)(p-1)} \end{bmatrix}$ and each B_i as

block matrices, $B_i = \begin{bmatrix} B_i^{00} & \cdots & B_i^{0(p-1)} \\ \vdots & \ddots & \vdots \\ B_i^{(p-1)0} & \cdots & B_i^{(p-1)(p-1)} \end{bmatrix}$, then $M_{0j} = \sum_{i=0}^{p-1} B_i^{0j}$ is an n-by-n matrix. Hence $T = \sum_{i=0}^{p-1} M_{0i}$.

Example 2 To illustrate the construction described above, we provide an example demonstrating how to derive a unitary matrix from a sparse binary matrix. Consider a matrix T of dimensions 4×4 In this matrix, exactly 4 entries are 1's, some columns are entirely composed of zero entries.

Given $T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$, we aim to construct a permutation matrix M by applying.

Proposition 2 *where* $p = 2$. *The steps are as follows*:

Step 1 Express T as the sum of matrices with isolated nonzero columns, $T = \sum_{i=0}^{q-1} T_i$

Since T has two nonzero columns with two 1 s each, we write T as a sum: $T = T_0 + T_1 = \mathrm{T_0} + \mathrm{T_1}$ where

$$T_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } T_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Step 2 Construct matrices $A_i = I \otimes T_i$, for each T_i

$$A_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$A_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 3 Decompose each A_i into matrices A_i^j with only one 1 per matrix.

For each A_i separate it as a sum of matrices A_i^j, each containing at most one 1 A_0:

$$A_0 = A_0^0 + A_0^1 + A_0^2 + A_0^3$$

where each A_0^j has one nonzero entry. For example:

$$A_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 4 Apply column permutations to ensure unique positions.

For each A_0^j apply column permutations (as in Proposition 1) to avoid overlaps among nonzero entries. Let the resulting matrices be B_0^j

$$B_0^0 = \begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix} B_0^1 = \begin{bmatrix} 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix} B_0^2 = \begin{bmatrix} 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix} B_0^3 = \begin{bmatrix} 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

Step 5 Combine all B_0^j to form B_0

$$B_0 = \begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

Likewise from T_1 we construct $B_1 = \begin{bmatrix} 0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1 \end{bmatrix}$

Step 6 Finally construct $M\ as\ the\ sum\ of\ B_i$

$M = B_0 + B_1$.

$$M = \begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&1&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1 \end{bmatrix}$$

Space complexity

The total space complexity for constructing the matrix M is $O(np)$ where n is the dimension of the given sparse matrix T, $p \leq n$ is the maximum number of nonzero entries in any column of T.

Time complexity

For a given sparse binary matrix of dimension n, constructing the corresponding permutation matrix requires n^2 assignments. This results in a time complexity of $O(n^2)$, reflecting the computational effort needed to complete the transformation efficiently.

Comparison of methods

Our proposed method, with a space complexity of $O(np)$ and a time complexity of $O(n^2)$, demonstrates efficient scaling for sparse matrices. Here, n is the matrix dimension, p is the maximum number of nonzero entries in any column, ensuring practical applicability to large-scale problems. This contrasts with the method by Gingrich and Williams in[11], which constructs a unitary matrix through probabilistic computations and computational overhead unsuitable for large-scale systems. Similarly, the block encoding method in[13] embeds a non-unitary matrix into a higher-dimensional unitary matrix using singular value decomposition, with a space complexity that is linear in the size of the output matrix but a time complexity of (n^3), limiting scalability due to the computationally expensive decomposition. While the latter methods are versatile and can handle general non-unitary matrices, their resource demands make them impractical for many applications. In comparison, our method not only ensures unitary transformation but also reduces the time complexity, making it particularly advantageous for sparse matrices and applications requiring efficient quantum computations. Table 1 shows the comparison between the proposed methods with other methods.

Methods	Time complexity	Scalability	Advantages
Proposed method	$O(n^2)$	High	Efficient for sparse matrices; error-free numeric computation
Gingrich and Williams[11]	$O(n^3)$	Low	Probabilistic framework for non-unitary operators
Block encoding[13]	$O(n^3)$	Low	Enables handling general matrices; foundational for quantum algorithms

Table 1. comparison between different approaches.

Permutation matrices to gates

From the previous section, we conclude that any non-unitary sparse binary matrix with exactly n nonzero entries (where n is the size of the matrix) can be transformed into a permutation matrix, which is inherently unitary. Though permutation matrices are unitary, their practical implementations are restricted to a particular set of quantum gates. In this section, we demonstrate how to decompose any permutation into a product of elementary quantum gates.

Observation on permutations matrices

In[18], the system model defines a parameterized quantum circuit as a sequence of unitary operations acting on an input state, expressed as:

$$U\left(\overrightarrow{\theta}\right) = U_L\left(\theta_L\right) U_{L-1}\left(\theta_{L-1}\right) \ldots U_1(\theta_1) \tag{1}$$

This structure underlies many variational quantum algorithms, where each $U_i(\theta_i)$ represents a layer or gate applied sequentially to the quantum register. Our method aligns naturally with this model because the permutation matrices constructed from sparse binary matrices can be directly translated into such unitary operations, with each permutation corresponding to a swap gate or CNOT gate.

Let $a_{1,}a_{2,}\ldots a_{n,}$ represent an ordered sequence. Consider a permutation $P_{(i,j),}$ which swaps the elements at positions i and j. The permutation $P_{(i,j)}$ can be expressed as a product of transpositions of neighboring elements as follows

$$P_{(i,j)} = P_{(i,i+1),}P_{(i+1,i+2),}\ldots P_{(j-1,j),}P_{(j-2,j-1),}\ldots P_{(i,i+1)} \tag{2}$$

Each of these adjacent transpositions corresponds to a hardware-efficient gate, such as a Swap or CNOT, and fits directly into the layered structure in Eq. (1). Moreover, these permutation operations can be interpreted as unitaries derived from Hermitian generators, using the exponential form:

$$U_i\left(\theta_i\right) = \exp(-i\theta_i P_i) \tag{3}$$

where P_i is a Hermitian matrix representing a basic transposition, and θ_i is a tunable parameter. This provides a formal mapping from permutation logic to the standard unitary framework used in quantum circuits.

An additional advantage of our approach is its exploitation of Hamming distance: any permutation of binary strings can be decomposed into transpositions between strings that differ by a Hamming distance of 1. This means the overall permutation matrix can be realized as a sequence of minimal bit-flip operations, further reducing circuit depth and enhancing efficiency for NISQ devices[19].

In summary, by embedding non sparse binary matrices into structured permutation matrices and decomposing them into transpositions with minimal Hamming distance, we enable an efficient realization of non-unitary operations in the unitary model of Eq. (1).

Transposition to quantum gates

We construct quantum gates to implement transpositions represented by the matrices whose rows as binary integers differ by 1 in hamming distance. Since a transposition with a Hamming distance of 1 involves swapping two elements that differ in only one bit, it can be realized using a controlled quantum operation that acts conditionally based on that bit.

It is well known that any permutation can be expressed as a product of such transpositions; hence we can systematically construct a sequence of quantum gates to realize any desired permutation gate. This approach allows for the decomposition of any arbitrary permutation matrix into a series of elementary gates that operate on transpositions, thereby enabling efficient implementation in quantum circuits.

We start with a given $2^n \times 2^n$ permutation matrix. First, we analyze the matrix to identify the indices that have been swapped by the permutation. This can be achieved by comparing the rows and columns of the permutation matrix to determine which positions map to each other.

Once the swapped indices are identified, we construct quantum gates to implement these swaps. For each swap, we check if the indices differ by a Hamming distance of 1. If they do, the corresponding transposition can be directly realized using a single gate designed for Hamming distance 1 swaps. If the Hamming distance is greater than 1, we decompose the swap into a sequence of transpositions with neighboring elements (i.e., intermediate swaps with Hamming distance 1), as shown in the earlier observation.

By repeating this process for all swaps in the permutation matrix, we construct a sequence of quantum gates that faithfully implements the given $2^n \times 2^n$ permutation matrix. This method ensures a systematic and efficient translation of any permutation matrix into a quantum circuit.

The method is implemented in Python 3 using Qiskit, and the pseudo code listings are provided in the "Supplementary Materials"). At a high level, the method involves the following steps:

1. Preprocessing the matrix to identify the swapped elements and the corresponding qubit controls.
2. Checking the Hamming distance between the binary representations of the indices.

- If the Hamming distance is 1, a direct multi-controlled- X (MCX) gate is inserted.
- If greater than 1, the permutation matrix is factorized into simpler transpositions.

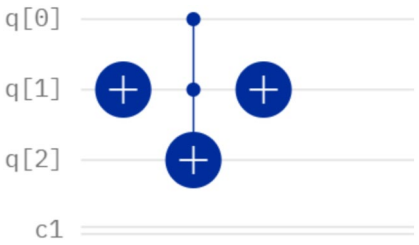

Fig. 1. Quantum circuit for transposition M.

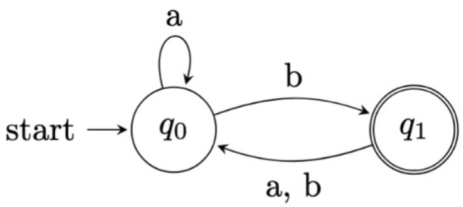

Fig. 2. DFA over $\Sigma = \{\mathrm{a},\ \mathrm{b}\}$ recognizes the language L.

3. Recursive construction of the circuit until the full permutation is realized.

This method guarantees that any $2^n \times 2^n$ transposition matrix can be translated into a corresponding n-qubit quantum circuit through a systematic construction of controlled gates.

Illustrative example

We demonstrate the method with a simple example transposition matrix:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$ is a transposition between states 4 and 5 (with indexing from 0 to $n-1$).

The binary indices of the states are: $bin_{s1} = 100$ and $bin_{s2} = 101$. The Hamming distance between bin_{s1} and bin_{s2} is 1. Therefore, the swap can be implemented using a multi-controlled X (MCX) gate with controls on Qubit 0 and Qubit 1, and the target on Qubit 2. Negative control handling is required for Qubit 1, as it is initially 0 in both bin_{s1} and bin_{s2}. The resulting quantum circuit is shown in Fig. 1. Quantum circuit for transposition M. The circuit uses controls on qubits $q[0]$(positive control) and $q[1]$(negative control, created by add X gate before and after the controls) to apply an X (NOT) gate on the target qubit $q[2]$. The negative control ensures that the gate is triggered when $q[1]$ is in the $|0\rangle$.

Applications

Sparse binary matrices have extensive real-world applications due to their efficiency in representing large, structured, and often sparse datasets. Examples include representations of social networks, web graphs, telecommunication networks, finite state machines, and so on. These matrix representations in general are not unitary. However, within the quantum system, operations are unitary matrices. So, to make use of these representations nonunitary-to-unitary transformations are necessary. To demonstrate the application of our method, we consider classical deterministic finite state automata (DFA). DFA is a computational model with a wide range of applications including compiler design, text processing and pattern matching. We show how a quantum finite state system can be built from a classically defined DFA.

In the theory of computation, DFAs are powerful models that recognize languages by processing input symbols and transitioning deterministically between states. Traditionally, a deterministic finite state automaton is defined as a five tuple M = (K, Σ, Δ, s, F), where: K is a finite set of states, Σ (the alphabet) is a finite set of symbols, s ∈ K is the initial state, F ⊆ K is the set of accepting or final states, and Δ is the transition function. Δ is a function from K × Σ to K. DFAs consist of a finite set of states, a transition function defined as a mapping from the current state and input symbol to the next state, and a set of final or accepting states that define the acceptance condition of the automaton. Adapting DFAs for quantum systems bridges classical automata with

quantum algorithms, laying the foundation for advanced quantum computational models and enhancing our ability to tackle probabilistic or complex-pattern languages.

To translate a classical DFA into a quantum framework, DFA states can be encoded as binary vectors in Hilbert space. Transitions are then represented by unitary matrices that transform the quantum state vector. Hence, DFA transitions can be realized as quantum gates. By associating unitary matrices to input symbols, the quantum DFA processes input strings by applying the associated unitary matrices/quantum gates, representing each input symbol in sequence. The system's final state vector after processing the string indicates whether the input is accepted.

Figure 2 DFA over $\Sigma = \{\mathrm{a}, \mathrm{b}\}$ recognizes the language L, illustrates a two-state deterministic finite automaton (DFA) over the alphabet $\Sigma = \{\mathrm{a}, \mathrm{b}\}$ designed to recognize the language $L = \{w | \mathrm{w\ is\ a\ string\ of\ a's\ and\ b's\ ending\ in\ b}\}$. The start state q_0 is shown as a single-bordered circle, while the accepting (final) state q_1 is indicated by a double-bordered circle. There is a self-loop on q_0 for input a, allowing the automaton to remain at q_0 upon reading a,. Two horizontal transitions connect the states: An upper arrow labeled a, b sends the automaton from $q_1 to q_0$ on reading either a or b. A lower arrow labeled b sends the automaton back from $q_0 to q_1$ when a b is read.

To illustrate the application in a quantum-inspired framework, we represent the DFA states q_0 and q_1 as two-dimensional column vectors: $q_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $q_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, The input symbols a and b are represented as $a = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$, and $\mathrm{b} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. A state transition upon reading an input symbol is defined as a matrix–vector multiplication: $v^T = u^T A$, where u represents the current state, v represents the next state, and A represents the current input symbol either a or b.

To realize the method outlined in this paper, complete the list of steps to construct quantum state transition symbol 'a' equivalent to the classical state transition $v^T = u^T A$ as shown below:

1. Input preparation.

$$\mathrm{U} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \hat{u} = e_0 \otimes \mathrm{u},\ where\ e_0 = [10]^T,\ \hat{u} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

2. Construct unitary matrix M from the given 'a' (steps shown in proposition 1).

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

3. State transition with the unitary matrix.

$$\begin{aligned} \hat{v}^T &= \hat{u}^T M \\ &= [\ 0 \quad 1 \quad 0 \quad 0\ ] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \\ &= [\ 0 \quad 0 \quad 1 \quad 0\ ] \end{aligned}$$

The state vector obtained here is a permutation of the actual vector. The permutation is realized by the permutations performed to construct the unitary matrix. Affected permutation needs to be reversed.

4. Reverse the effect of permutations (affected permutation is columns 1 and 3).

$$\hat{v}^T = [\ 1 \quad 0 \quad 0 \quad 0\ ] = [\ 1 \quad 0\ ]^T \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = e_0 \otimes v$$

Thus $\widehat{v}^T = \widehat{u}^T M$ is equivalent to $v^T = u^T T$.

5. Convert this M into Quantum gate.

   5.1. Express M as the product of Transposition.

Cycles in M [1, 1–3] so M can be expressed as:

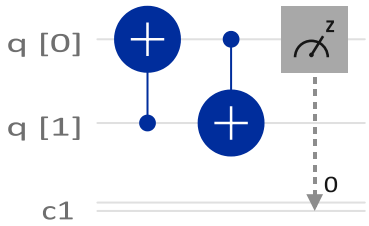

Fig. 3. Quantum circuit for DFA symbol a.

$$M = P_0\,[1,3]\,.P_1\,[1,2]$$

5.2 Convert the index into binary.

$$M = P_0\,[01,11]\,.P_1\,[01,10]$$

In P_1 Haming distance between 01 and 10 is not equal to one so we need to decompose it further,

$$\begin{aligned} P_1 &= P_{10}\,[01,11]\,.P_{11}[11,10]P_{12}\,[01,11] \\ M &= P_0\,[01,11]\,.P_{10}\,[01,11]\,.P_{11}[11,10]P_{12}\,[01,11] \\ &= P_{11}[11,10]P_{12}\,[01,11] \end{aligned}$$

5.3. P_{11} represent by a CNOT with control on qubit 1 and target on qubit 0.
5.4. P_{12} represent by a CNOT with control on qubit 0 and target on qubit 1.

Figure 3 Quantum circuit implementing the DFA symbol $a = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$. The circuit applies a two-qubit controlled-X (CNOT-like) operation, followed by a measurement in the computational (Z) basis on q [0]. The measurement outcome is recorded into the classical register c [0], indicated by the downward dashed arrow from the measurement box to the classical bit.

This arrow signifies the transfer of information from the quantum state to a classical bit, allowing the result to be processed classically after measurement.

Though DFA operations are irreversible, each transition deterministically moves from one state to another without storing information about prior states. However, in quantum computing, unitary matrices (the primary transition operators) are inherently reversible. This seemingly contradictory behavior occurs because the DFA transition matrix is embedded into a larger unitary matrix.

Moreover, in gate-model quantum neural networks (QNNs), the ability to embed sparse structures into unitary matrices is also highly valuable. Training QNNs often encounters challenges like barren plateaus and slow convergence, which are exacerbated by dense or poorly structured parameterizations. The results in[20] highlight the importance of initialization strategies that maintain structure while supporting trainability. Our method allows for the construction of permutation-based unitary matrices from sparse binary inputs. These can serve as lightweight and expressive layers in QNNs, facilitating both better optimization during training and improved generalization performance. This aligns well with current trends in designing noise-resilient and resource-efficient machine learning models.

Our method also extends to quantum networking applications. In emerging quantum internet infrastructures, sparse matrices frequently model the distribution of entanglement, routing paths, or connectivity graphs between network nodes. As discussed in[21,22], unitary transformations are required to preserve entanglement during distributed operations while minimizing communication overhead. By embedding sparse binary matrices into structured unitary permutation matrices, our method offers an efficient tool for implementing routing protocols, optimizing quantum channel usage, and supporting scalable quantum network architecture.

Conclusions

This work presents an efficient and scalable method for converting non-unitary sparse binary square matrices into unitary matrices, enabling their application in quantum computational frameworks. The proposed approach addresses a significant bottleneck in embedding non-unitary transformations into quantum systems. The proposed approach is significantly less complex than previously proposed methods. Furthermore, a method to translate permutation matrices into quantum circuits is outlined, providing a practical pathway for implementation. The application of this method in quantum finite state machines demonstrates its potential to bridge classical automata with quantum algorithms. While this study focuses on square matrices, future work will explore the embedding of rectangular matrices into unitary matrices and their integration into diverse quantum computing paradigms.

Data availability

All data generated or analyzed during this study are included in this published article.

Received: 4 December 2024; Accepted: 20 May 2025
Published online: 02 July 2025

References

1. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information* (Cambridge University Press, 2010).
2. Bender, C. M. & Boettcher, S. Real spectra in non-Hermitian Hamiltonians having P T symmetry. *Phys. Rev. Lett.* **80**(24), 5243 (1998).
3. Zheng, C., Hao, L. & Long, G. L. Observation of a fast evolution in a parity-time-symmetric system. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **371**(1989), 20120053 (2013).
4. Breuer, H. P. & Petruccione, F. *The Theory of Open Quantum Systems* (OUP Oxford, 2002).
5. Hu, Z., Xia, R. & Kais, S. A quantum algorithm for evolving open quantum dynamics on quantum computing devices. *Sci. Rep.* **10**(1), 3301 (2020).
6. Del Re, L., Rost, B., Kemper, A. F. & Freericks, J. K. Driven-dissipative quantum mechanics on a lattice: Simulating a fermionic reservoir on a quantum computer. *Phys. Rev. B* **102**(12), 125112 (2020).
7. Gui-Lu, L. General quantum interference principle and dualitycomputer. *Commun. Theor. Phys.* **45**(5), 825 (2006).
8. Zheng, C. Universal quantum simulation of single qubit nonunitary operators using duality quantum algorithm. *Sci. Rep.* **11**(1), 3960 (2021).
9. Krantz, P. et al. A quantum engineer's guide to superconducting qubits. *Appl. Phys. Rev.* https://doi.org/10.1063/1.5089550 (2019).
10. Sato, Y., Kondo, R., Koide, S., Takamatsu, H. & Imoto, N. Variational quantum algorithm based on the minimum potential energy for solving the Poisson equation. *Phys. Rev. A* **104**(5), 052409 (2021).
11. Gingrich, R. M. & Williams, C. P. Non-unitary probabilistic quantum computing. In *Proceedings of the Winter International Symposium on Information and Communication Technologies (WISICT '04). Trinity College Dublin* 1–6 (2004).
12. Childs, A. M. & Wiebe, N. Hamiltonian simulation using linear combinations of unitary operations. arXiv preprint arXiv:1202.5822 (2012).
13. Lin, L. Lecture Notes on Quantum Algorithms for Scientific computing. quant-ph (2022).
14. Cybenko, G. Reducing quantum computations to elementary unitary operations. *Comput. Sci. Eng.* **3**(2), 27–32. https://doi.org/10.1109/5992.908999 (2001).
15. Planat, M. & Haq, R. U. The magic of universal quantum computing with permutations. *Adv. Math. Phys.* **2017**, 1–9. https://doi.org/10.1155/2017/5287862 (2017).
16. Weber, M. Quantum permutation matrices. *Complex Anal. Oper. Theory* **17**, 37. https://doi.org/10.1007/s11785-023-01335-x (2023).
17. Moore, C. & Crutchfield, J. P. Quantum automata and quantum grammars. *Theor. Comput. Sci.* **237**(1–2), 275–306. https://doi.org/10.1016/S0304-3975(98)00191-1 (2000).
18. Gyongyosi, L. & Imre, S. Circuit depth reduction for gate-model quantum computers. *Sci. Rep.* **10**(1), 11229 (2020).
19. Gyongyosi, L. & Imre, S. Scalable distributed gate-model quantum computers. *Sci. Rep.* **11**(1), 5172 (2021).
20. Gyongyosi, L. & Imre, S. Training optimization for gate-model quantum neural networks. *Sci. Rep.* **9**(1), 12679 (2019).
21. Gyongyosi, L. & Imre, S. Advances in the quantum internet. *Commun. ACM* **65**(8), 52–63 (2022).
22. Gyongyosi, L., Imre, S. & Nguyen, H. V. A survey on quantum channel capacities. *IEEE Commun. Surv. Tutor.* **20**(2), 1149–1205 (2018).

Author contributions

K.K. conception and design of the research, article writing; V.P. Algorithm design and article reviewing; K.G. research supervision, study design, article writing; J.P research supervision, article revising; All authors have read and approved the final manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-025-03424-7.

Correspondence and requests for materials should be addressed to K.K.