Distributed quantum logic algorithm

Boris Arseniev
Skolkovo Institute of Science and Technology, Moscow, Russian Federation
(Dated: October 15, 2024)

Parallel computation enables multiple processors to execute different parts of a task simultaneously, improving processing speed and efficiency. In quantum computing, parallel gate implementation involves executing gates independently on different registers, directly impacting the circuit depth, the number of sequential quantum gate operations, and thus the algorithm execution time. This work examines a method for reducing circuit depth by adding qubits to enable parallel gate execution, potentially enhancing the performance of quantum simulations on near-term quantum devices. We show that any circuit on n qubits with depth $O(MN)$, where $N = 2^n$, can be transformed into a circuit with depth $O(\log^2(M)N)$ operating on $O(Mn)$ qubits. This technique may be particularly useful in noisy environments, where recent findings indicate that only the final $O(\log n)$ layers influence the quantum state. It may also optimize Trotterization by logarithmically reducing the number of Trotter steps. Additionally, the method may offer advantages for distributed quantum computing, and the perspective used here, treating quantum states as gates and operators as vectors, could have broader applications in quantum computation.

I. INTRODUCTION

The depth of a quantum circuit, defined by the number of sequential gate layers required for execution, presents major challenges for the efficiency, accuracy, and practical implementation of quantum algorithms. Current quantum computers are constrained by short coherence times, limiting their ability to maintain quantum states for extended periods. As circuit depth increases, the execution time grows, which increases the likelihood of qubit decoherence and introduces computational errors.

In this work, we propose a universal method to reduce quantum circuit depth by employing additional qubits. Our approach focuses on parallelizing the application of quantum operators through the use of specialized gates. By defining two types of gates, G and V, we demonstrate how a sequence of M quantum operators can be applied in parallel, resulting in a circuit with logarithmic depth relative to the number of G gates. This approach is particularly relevant in the presence of noise, where recent studies suggest that only the final $O(\log(n))$ layers of deep circuits significantly influence the expectation values of observables [1].

A key application of this algorithm is Hamiltonian simulation, where Trotterization significantly increases circuit depth [2]. Additionally, the parallel execution of gates offers advantages for distributed quantum computing, where multiple quantum processors collaborate on a computation [3]. Moreover, we present an alternative perspective on quantum circuits, treating the quantum state as a gate and operators as vectors, which opens up opportunities for circuit design and optimization.

The structure of this paper is as follows: Section Section II introduces the core components of the algorithm, including the G and V gates. Section III A describes the algorithm in detail, followed by an analysis of the operations in Section III B.

II. BUILDING BLOCKS OF THE ALGORITHM

The central concept utilized in our research is the application of vectorized operators. We represent the vectorization of a matrix M as $\mathrm{vec}\,(M)$, which refers to the column vector formed by stacking the columns of matrix M sequentially on top of each other [4]. For example

$$\mathrm{vec}\left(\begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}\right) = \begin{pmatrix} M_{11} \\ M_{21} \\ M_{12} \\ M_{22} \end{pmatrix}.$$

Throughout this work, we will use square matrices of size $N \times N$, where $N = 2^n$, unless otherwise specified. That is, the operator acts on n qubits. The algorithm comprises three main elements: vectorization, gathering, and multiplication. We will now examine each element individually.

We abalyze complexity in terms of 1 and 2 qubit gates.

A. Vectorization

Consider two matrices A, B of size $N \times N$. To obtain vectorization of these two operators we will use the following equation

$$\mathrm{vec}\,(AB) = (B^T \otimes A)\mathrm{vec}\,(I_N), \tag{1}$$

where I_N is the identity matrix of size $N \times N$ [4].

In this way, we can see that if we initialize the state $|\psi\rangle = \frac{1}{\sqrt{N}}\mathrm{vec}\,(I_N)$ and apply the operator $(B^T \otimes A)$, the resulted state will be proportional to $\mathrm{vec}\,(AB)$.

B. Gathering

Using equation (1), we can obtain the state, which is proportional to the tensor product of vectorized oper-

ators. Since our objective is to ultimately compute the product of the considered operators, we introduce the following gathering operation G, which has a size of $N^2 \times N^4$

$$G\left(\mathrm{vec}\left(A\right) \otimes \mathrm{vec}\left(B\right)\right) = \mathrm{vec}\left(BA\right). \tag{2}$$

In fact, we note that these operator maybe written as

$$G = I_N \otimes \mathrm{vec}\left(I_N\right)^T \otimes I_N. \tag{3}$$

We now introduce a square matrix G_u of size $N^2 \times N^2$, whose first row is given by $\mathrm{vec}\left(I_N\right)^T$. There are various methods for constructing such a matrix; for example, one approach to construct unitary G_u is to generate a complete basis, starting with $\frac{1}{\sqrt{N}}\mathrm{vec}\left(I_N\right)$, and use these vectors as the rows of the matrix G_u. Our proposed approach is detailed in Section III. For the moment, we note that after applying this matrix, the desired state will reside in the first and fourth registers (or the first and the last n qubits), i.e.

$$\begin{aligned}&\left[I_N \otimes G_u \otimes I_N\right]\left(\mathrm{vec}\left(A\right) \otimes \mathrm{vec}\left(B\right)\right)\\ &= \mathrm{vec}\left(BA\right)_1 \otimes |0\rangle \otimes |0\rangle \otimes \mathrm{vec}\left(BA\right)_2\\ &+ \sum_{k,l=1}^{N} \vec{r}_k \otimes |k\rangle \otimes |l\rangle \otimes \vec{r}_l,\end{aligned} \tag{4}$$

where $\mathrm{vec}\left(BA\right)_1$ and $\mathrm{vec}\left(BA\right)_2$ are such that $\mathrm{vec}\left(BA\right) = \mathrm{vec}\left(BA\right)_1 \otimes \mathrm{vec}\left(BA\right)_2$, and $\vec{r}_{k,l}$ are some residual vectors, which we are not concerned with. On can consider the application of a swap operator between the second and the last registers to obtain $\mathrm{vec}\left(BA\right) \otimes |0\rangle \otimes |0\rangle$.

C. Multiplication

The only part left to build an algorithm is the multiplication of a vector by a matrix. Thus, for a vector $\vec{v}$ of size N, and a matrix A of size $N \times N$, where $N = 2^n$ we introduce the following operator V of size $N \times N^2$

$$V\mathrm{vec}\left(A^T\right) = A\vec{v} \tag{5}$$

This operation can be implemented as

$$V = I_N \otimes \vec{v}^T \tag{6}$$

Similarly to the gathering operator, we can extend the row $\vec{v}^T$ into a square matrix V_u of size $N \times N$, whose first row is given by $\vec{v}^T$. Our method for constructing this matrix is provided in Section III. At this stage, we note that after applying V_u, the desired state will be located in the first register (or the first n qubits), i.e.

$$\left[I_N \otimes V_u\right]\mathrm{vec}\left(A^T\right) = \left(A\vec{v}\right) \otimes |0\rangle + \sum_{k=1}^{N} \vec{r}_k \otimes |k\rangle\,, \tag{7}$$

where $\vec{r}_k$ are some residual vectors, which we are not concerned with.

III. CIRCUIT FOR THE ALGORITHM

In this subsection, we first provide the circuit in terms of G_u and V_u and then we show how to realize them. Based on equations (1) and (7) it is easy to see how one can create a circuit for parallelization of two gates. We formilize this in the following Proposition

Proposition 1 (2 parallel gates) *Consider a quantum circuit with two gates, U_1 and U_2, each of size $N \times N$, acting on an initial state $|\vec{v}\rangle$. The circuit for parallel implementation can be expressed as:*

$$V(U_2^T \otimes U_1)\mathrm{vec}\left(I_N\right) = U_1 U_2 \left|\vec{v}\right\rangle \otimes |0\rangle + \sum_{k=1}^{N} \vec{r}_k \otimes |k\rangle\,, \tag{8}$$

where $V = I_N \otimes V_u$, and V_u is a unitary matrix whose first row is given by $|\vec{v}\rangle^T$. The terms $\vec{r}_k$ represent residual vectors, and $\mathrm{vec}\left(I_N\right)$ denotes the vectorization of the identity matrix of size N.

For $M > 2$, the gathering gate G, as described in equation (4), can be applied to generalize the previous proposition to any $M > 1$. In the following, we assume M is even; if not, we introduce $U_{M+1} = I$ to ensure even parity.

A. Distributed quantum logic algorithm

The algorithm for constructing the quantum circuit comprises three primary stages: vectorization, gathering, and multiplication. We also note that the use of swap operators is not necessary; we use them for the illustration purposes only.

a. Vectorization stage During this stage we construct $M/2$ vectorization circuits as defined by equation (1). Specifically, the circuit is initialized in the state $\underbrace{\mathrm{vec}\left(I_N\right) \otimes \cdots \otimes \mathrm{vec}\left(I_N\right)}_{M/2}$, and operators are applied, alternating transpositions, as $U_1 \otimes U_2^T \otimes U_3 \otimes U_4 \ldots \otimes U_{M-1} \otimes U_M^T$. At the conclusion of this stage, the quantum state becomes proportional to the tensor product of the vectorized operators:

$$\mathrm{vec}\left(\left(U_1 U_2\right)^T\right) \otimes \cdots \otimes \mathrm{vec}\left(\left(U_{M-1} U_M\right)^T\right).$$

b. Gathering stage Gathering operators $G = I_N \otimes G_u \otimes I_N$ are employed to combine the vectorized states, where G_u acts on two registers. We consider a swap gate between the second and fourth output registers of the operator G, producing the desired state in the first two output registers. Let us define the combination of G and the swap gate as $\tilde{G}$. For simplicity, we ignore the terms involving $\vec{r}_i$ in equation (4), focusing solely on the first term.

First let us consider the case $M = 2^\kappa$ (with $\kappa > 1$). After applying $\underbrace{G \otimes \cdots \otimes G}_{M/4}$, the resulting state is proportional to:

$$\text{vec}\left((\prod_{j=1}^{4} U_j)^T\right) \otimes |00\rangle \otimes \cdots \otimes \text{vec}\left((\prod_{j=M-3}^{M} U_j)^T\right) \otimes |00\rangle\,.$$

We then apply swap gates to bring the vectorized operators together, yielding the state:

$$\text{vec}\left((\prod_{j=1}^{4} U_j)^T\right) \otimes \text{vec}\left((\prod_{j=5}^{8} U_j)^T\right) \otimes |0000\rangle \otimes \ldots.$$

This process is repeated until the state $\text{vec}\left((\prod_{j=1}^{M} U_j)^T\right) \otimes \underbrace{|0\rangle \ldots |0\rangle}_{M-2}$ is obtained. The process requires $\log_2(M) - 1$ layers. In the first layer, there are $M/4 = 2^{\kappa-2}$ operators $\tilde{G}$; the second layer contains $M/(4 \times 2) = 2^{\kappa-3}$ operators, and so on, until the final layer, which consists of a single $\tilde{G}$ operator.

For any even M, the same method can be generalized by decomposing M into sums of powers of two. The algorithm constructs circuits for each contributing 2^κ, starting with the largest power. These circuits are then connected layer by layer, using swap and $\tilde{G}$ gates in a similar fashion as described for powers of two. The number of additional $\tilde{G}$ gates required corresponds to the number of 1's in the binary decomposition of M, minus 1.

c. Multiplication stage The operator $V = I_N \otimes V_u$ is applied to the first two registers, completing the computation. The resulted circuit yields the final state, proportional to $\left(\prod_{j=1}^{M} U_j\right)\vec{v} \otimes \underbrace{|0\rangle \ldots |0\rangle}_{M-1}$.

Figure 1 illustrates the circuit configurations for $M = 14$ in detail. The next proposition provides the scaling for the circuit constructed using this algorithm.

Proposition 2 (Distributed Qubit Logic (DQL)) *Consider $M > 1$ unitary operators $U_1, \ldots, U_M$, each of size $N \times N$ with $N = 2^n$, acting on an initial state $|\vec{v}\rangle$. To achieve a state proportional to $(U_1 \cdots U_M)|\vec{v}\rangle$ in the first register, one can construct a quantum circuit with a depth d defined by:*

$$d = d_U + (\lceil \log_2(M) \rceil - 1)\, d_G + d_V, \tag{9}$$

where d_U represents the maximum depth of the operators U_j for $j = 1, \ldots, M$, d_G denotes the depth of the operator $G = I_N \otimes G_u \otimes I_N$, as outlined in equation (4), and d_V is the depth of the operator V_u, detailed in equation (7). The number of G_u gates required for the implementation of this algorithm is $M/2 - 1$, in addition to one V_u gate. Furthermore, the total number of qubits required is nM.

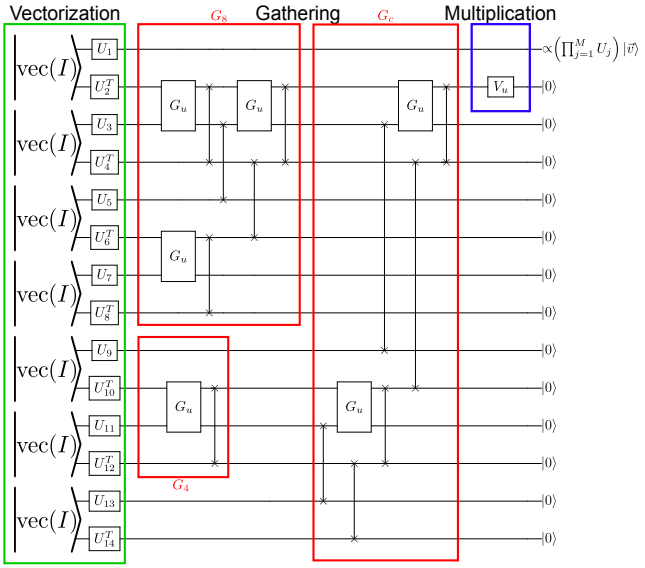

FIG. 1. Circuit for parallel computation of $M = 14$ gates. The vectorization stage is highlighted by the green rectangle, while the gathering stage is represented by three red rectangles: G_8, G_4, and G_c. Here, G_8 and G_4 are gathering circuits for 8 and 4 gates, respectively, and G_c is the gathering circuit that combines the remainders. The multiplication stage is indicated by the blue rectangle. The desired quantum state is illustrated on the right.

B. Implementation of G_u and V_u

First, we present the implementation of the operator G_u. To construct it, we use the sum of all Pauli strings containing only X and I. We use the matrices $P_j = \bigotimes_{k=1}^{n} X^{(j-1)_k}$, with $(j-1)_k$ being the k-th bit of $j-1$ in binary form using reverse encoding (the least significant bit is the leftmost) and $X^0 = I$. Matrix G_u with the first row given by $\vec{j}^T$ is given by

$$G_u = \sum_{j=1}^{N} P_j \otimes P_j. \tag{10}$$

This matrix is not unitary, so to implement it we will rewrite G_u as

$$G_u = \frac{N}{2}\left(I - e^{i\frac{\pi}{N}G_u}\right). \tag{11}$$

Now, since all Pauli strings are commute, we can simultaneously diagonalize them, obtaining

$$G_u = \frac{N}{2}\left(I - (D^\dagger \otimes D^\dagger) e^{i\frac{\pi}{N}\sum_{j=1}^{N} Z_j \otimes Z_j} (D \otimes D)\right), \tag{12}$$

where Z_j are diagonal matrices (namely Pauli string, which contains Z and I operators only) and D are diagonalization operators, namely $D = \underbrace{H \otimes \cdots \otimes H}_{n}$, where H is the Hadamard gate.

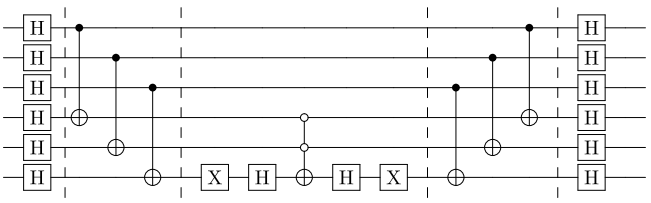

FIG. 2. Circuit for implementation of $e^{i\frac{\pi}{N}G_u}$, where G_u is of size $N^2\times N^2$, with $N=2^n$ and $n=3$. Here C^2Z implemented as $(I\otimes I\otimes H)C^2X(I\otimes I\otimes H)$.

This equation can be written as

$$G_u = \frac{N}{2}\left(I-(H^\dagger)^{\otimes 2n}S\left(I_N\otimes X_nC^nZX_n\right)SH^{\otimes 2n}\right), \tag{13}$$

$$S=\prod_{j=1}^{n}CX(j,j+n). \tag{14}$$

Now, we can apply a technique called the linear combination of unitaries [5]. Using this approach, we can see that the number of gates required to implement G_u is $O(N+n)$. Additionally, we need to use one extra qubit.

Thus, the complexity of implementation of this gate is: $4n$ controlled Hadamard gates, $2n$ CCX gates, 2 CX gates and one $C^{n+1}Z$ gate. Controlled Hadamard gate can be implemented as $CH=(I\otimes R_y(\frac{\pi}{4}))CZ(I\otimes R_y(-\frac{\pi}{4}))$, CCX can be implemented with 6 CX gates and $C^{n+1}Z$ can be implemented recursively with $O(n^2)$ gates and an auxiliary gubit []. Taking all this into account, we have the complexity of $O(n^2)$. The circuit for the case of $n=2$ is presented in Figure 2.

For the case of V_u we can do almost the same approach. First, we note that

$$V_u=\sum_{j=1}^{N}v_jP_j. \tag{15}$$

After this we can diagonolize $V_u = D^\dagger\Lambda_V D$, where the diagonalization operator is the same as in equation (14). Next, we apply arccos function to the elements of $\Lambda_V/\max|\lambda_j|$, where λ_j are the elements of Λ_V.

In order to implement them, one results from [6] where it is shown that the gate count for a circuit which implements the diagonal exponent can be reduced to $O(N')$ gates with $N'<2^n$, without ancillary qubits.

The resulted matrix $\Lambda_{\arccos}$ will have elements $\arccos(\lambda_j/\max|\lambda_j|)$ on the diagonal. Now we can reconstruct Λ_V as a sum of 2 unitary matrices as

$$\Lambda_V=\frac{1}{2}\left(e^{i\Lambda_{\arccos}}+e^{-i\Lambda_{\arccos}}\right). \tag{16}$$

Similarly to G_u the implementation of each term in this formula requires $O(N)$ gates, without ancillary qubits, making the total cost $O(N+n^2)$. We formulate this result in the Proposition 3.

Proposition 3 (Implementation of G_u and V_u) *Gate G_u of size $N^2\times N^2$, with $N=2^n$ such that it has 1 as a first row can be implemented with $O(n^2)$ gates and 2 ancillary qubits or with $O(N)$ gates and one ancillary qubit.*

Gate V_u of size $N\times N$, with $N=2^n$ such that it has $\vec{v}$ as a first row can be implemented with $O(N)$ gates and 1 ancillary qubit.

We can combine this result with the result from Proposition 2 to obtain the following Corollary 1.

We note that in general implementation of initial condition $|\vec{v}\rangle$, where $\vec{v}$ is pf size $N=2^n$ have the scalability of $O(N)$ [**?**]. The initial condition $|\mathrm{vec}\,(I_N)\rangle$ can be implemented with S gate, i.e n CX gates acting on state $|0\rangle$. Thus, if consider this fact we can calim that we can transform any circuit to the circuit of depth $O(n^2)$.

Corollary 1 (Efficient parallelization) *Any circuit on n qubits of depth $O(M(N+n^2))$, with $N=2^n$ can be represented as a circuit of depth $O(\log_2(M)(N+n^2))$ acting on $O(Mn)$ qubits.*

This result allows for the construction of a circuit with linear depth relative to the system size, implying that any problem represented as a sequence of operations can be solved in linear time, provided there are sufficient qubits available.

We note

IV. CONCLUSION

In this work, we have introduced a universal method for reducing the depth of quantum circuits by leveraging additional qubits, referred to as the Distributed Quantum Logic (DQL) algorithm. This method distributes quantum gates across different registers, enabling parallel execution of a sequence of M operators U through the use of specialized gates, denoted as G and V. The number of G gates required is $(M/2-1)$ in addition to one V gate, resulting in an overall gate count of $\frac{3}{2}M$ for G, U, and V gates. The main result is a reduction in circuit depth to $(\lceil\log_2(M)\rceil+1)$ in terms of these gates. This is demonstrated for $M=2$ in Proposition 1 and generalized in Proposition 2.

Additionally, we provide a method for implementing the G and V gates, formalized in Proposition 3. The circuit required for this implementation consists of $O(N)$ gates, where $N=2^n$ is the size of the initial operators U, and it requires one additional qubit. As a result, for any circuit with depth $O(MN)$, where $M>1$, our algorithm reduces the circuit depth to $O(\log_2(M)N)$, as shown in Corollary 1.

During the development of this algorithm, we adopted a distinct perspective on quantum circuits, where the initial state is treated as a gate and operators are represented as vectors. This approach offers potential advantages for a range of quantum computing applications.

Specifically, the DQL algorithm could be beneficial for Hamiltonian simulation, where the Trotter formula often leads to long circuits. For instance, in our previous work [7], we achieved a scalability of $O(N^2 n^{3/2})$ for solving the wave equation on n qubits, which can be improved to $O(\log(Nn^{5/2})N) = O(nN)$ using the DQL algorithm with $O(Nn^{7/2})$ additional qubits. Furthermore, this method may have applications in variational quantum algorithms, where it has been shown that in most of the circuits with n qubits, only the final $O(\log(n)$ layers significantly influence the expectation values of observables [1].

The PYTHON code for the numerical experiment presented in all Figures is available on GitHub [8].

[1] A. A. Mele, A. Angrisani, S. Ghosh, S. Khatri, J. Eisert, D. S. França, and Y. Quek, Noise-induced shallow circuits and absence of barren plateaus, arXiv preprint arXiv:2403.13927 (2024).

[2] D. W. Berry, G. Ahokas, R. Cleve, and B. C. Sanders, Efficient quantum algorithms for simulating sparse hamiltonians, Communications in Mathematical Physics **270**, 359 (2007).

[3] M. Caleffi, M. Amoretti, D. Ferrari, J. Illiano, A. Manzalini, and A. S. Cacciapuoti, Distributed quantum computing: a survey, Computer Networks **254**, 110672 (2024).

[4] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation* (Wiley Publishing, 2009).

[5] A. M. Childs and N. Wiebe, Hamiltonian simulation using linear combinations of unitary operations, Quantum Information and Computation 10.26421/qic12.11-12 (2012).

[6] J. Welch, D. Greenbaum, S. Mostame, and A. Aspuru-Guzik, Efficient quantum circuits for diagonal unitaries without ancillas, New Journal of Physics **16**, 033040 (2014).

[7] B. Arseniev, D. Guskov, R. Sengupta, J. Biamonte, and I. Zacharov, Tridiagonal matrix decomposition for hamiltonian simulation on a quantum computer, Physical Review A **109**, 052629 (2024).

[8] Github page with numerical experiments, `https://github.com/barseniev/DQL_algorithm`.