

Distributed Kuperberg's algorithm

Peng-Yu Yang(杨鹏宇), Xin Zhang(张新), and Song Lin(林崧)
Citation: Chin. Phys. B, 2026, 35 (**3**): 030303. DOI: 10.1088/1674-1056/adfb58
http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Geometric control of concurrence and quantum gate operations in triangular triple quantum dots

Junqing Li(李俊青), Shuo Dong(董硕), and Jianhua Wei(魏建华)
Chin. Phys. B, 2026, 35 (**2**): 020302. DOI: 10.1088/1674-1056/adec5d

Bayesian phase difference estimation based on single-photon projective measurement

Xu-Hao Yu(余旭豪), Ying Wei(韦颖), Ran Yang(杨然), Wen-Hui Song(宋文慧), Yingning Miao(缪应宁), Wei Zhou(周唯), Xinhui Li(李新慧), Xiaoqin Gao(高小钦), Yan-Xiao Gong(龚彦晓), and Shi-Ning Zhu(祝世宁)
Chin. Phys. B, 2025, 34 (**7**): 070305. DOI: 10.1088/1674-1056/adde33

Progressive quantum algorithm for maximum independent set with quantum alternating operator ansatz

Xiao-Hui Ni(倪晓慧), Ling-Xiao Li(李凌霄), Yan-Qi Song(宋燕琪), Zheng-Ping Jin(金正平), Su-Juan Qin(秦素娟), and Fei Gao(高飞)
Chin. Phys. B, 2025, 34 (**7**): 070304. DOI: 10.1088/1674-1056/addd83

Distributed quantum circuit partitioning and optimization based on combined spectral clustering and search tree strategies

Zilu Chen(陈子禄), Zhijin Guan(管致锦), Shuxian Zhao(赵书娴), and Xueyun Cheng(程学云)
Chin. Phys. B, 2025, 34 (**5**): 050305. DOI: 10.1088/1674-1056/adbada

Automatic architecture design for distributed quantum computing

Ting-Yu Luo(骆挺宇), Yu-Zhen Zheng(郑宇真), Xiang Fu(付祥), and Yu-Xin Deng(邓玉欣)
Chin. Phys. B, 2024, 33 (**12**): 120302. DOI: 10.1088/1674-1056/ad7c2c

---

Distributed Kuperberg's algorithm

Peng-Yu Yang(杨鹏宇)[1], Xin Zhang(张新)[2], and Song Lin(林崧)[1,†]

[1] *College of Computer and Cyber Security, Fujian Normal University, Fuzhou* 350117, *China*
[2] *School of Mathematics and Statistics, Fujian Normal University, Fuzhou* 350117, *China*

(Received 12 June 2025; revised manuscript received 31 July 2025; accepted manuscript online 14 August 2025)

As an important quantum cryptanalysis algorithm, Kuperberg's algorithm efficiently addresses the dihedral hidden subgroup problem with sub-exponential acceleration. However, when dealing with large numbers, the algorithm demands a deeper quantum circuit depth, rendering its implementation on current quantum devices prone to noise interference and thereby significantly reducing its efficiency. To mitigate this challenge, this paper proposes a distributed Kuperberg's algorithm. It decomposes the original function into sub-functions which can be executed on different nodes in parallel. The implementation of these sub-functions necessitates a shallower quantum circuit depth and a reduced number of qubits when contrasted with the execution of the original function. Furthermore, the proposed algorithm can be directly generalized to an arbitrary number of nodes by adjusting the quantity of input qubits. The utilization of multi-node parallel processing makes the proposed algorithm a linear enhancement in query complexity relative to the original algorithm. To validate the feasibility and demonstrate the superiority of our algorithm, experiments are conducted on the Qiskit platform.

Keywords: Kuperberg's algorithm, distributed quantum computing, function decomposition, quantum circuit depth

PACS: 03.67.Ac, 03.67.Lx, 03.67.–a

DOI: 10.1088/1674-1056/adfb58 **CSTR:** 32038.14.CPB.adfb58

1. Introduction

Quantum computing is universally acknowledged as a potential technology for solving computational problems that are intractable for classical computing.[1–4] However, current quantum devices, such as noisy intermediate-scale quantum (NISQ) devices, face some critical constraints: limited qubit count, finite coherence durations, noise-induced error accumulation, and so on.[5] These constraints render the practical deployment of quantum algorithms exceedingly challenging.[6] Nevertheless, distributed quantum computing (DQC)[7–10] has emerged as a promising framework to circumvent NISQ-era limitations. Multiple small-to-medium quantum processors interconnect to collaboratively execute tasks, overcoming the computational limitations of individual devices. Within this framework, distributed quantum algorithms represent a pivotal research point due to their inherent advantage of proactive adaptation to distributed environments.

In 2021, Avron *et al.*[11] proposed a distributed quantum computing architecture through function decomposition. They accomplish distributed implementations of algorithms such as Deutsch–Jozsa, Simon, and Grover. This architecture reduced single-node circuit depth and qubit requirements when functions comply with specific computational paradigms. However, its constrained node scalability and absence of post-decomposition optimization mechanisms lead to performance deterioration.

In 2022, Tan *et al.*[12] advanced a refined function decomposition scheme facilitating arbitrary-scale node expansion within distributed quantum computing architectures. By implementing this framework in the Simon's algorithm with sorting multi-node computational outputs, their work achieved exponential acceleration relative to Avron's distributed Simon's algorithm. In the following, Qiu *et al.*[13] leveraged Tan's function decomposition framework integrated with quantum counting techniques to develop a distributed Grover's algorithm. This methodology reduced qubit resource requirements while achieving linear query complexity advantages. In short, the aforementioned algorithms achieved enhanced scalability and performance through optimized function decomposition and post-processing of computation results.

Kuperberg's algorithm[14] is a quantum algorithm of subexponential query complexity for the dihedral hidden subgroup problem and hidden shift problem. However, its practical implementation demands deep quantum circuits and a large number of qubits, limiting its applicability on near-term devices. Inspired by the above algorithms, we propose a distributed Kuperberg's algorithm. The algorithm decomposes the injective function into subfunctions, enabling parallel execution across nodes. We develop a 2-node distributed algorithm and extend it to t nodes, where t is arbitrary and can be adaptively adjusted based on practical requirements. Notably, the implementation of subfunctions on each node features shallower circuit depth, which enhances the algorithm's noise resistance. Furthermore, due to the parallel execution

†Corresponding author. E-mail: lins95@fjnu.edu.cn

© 2026 Chinese Physical Society and IOP Publishing Ltd. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

of tasks by multiple nodes, our algorithm reduces the query complexity from $2^{O(\sqrt{n})}$ to $2^{O(\sqrt{n-t})}$ with 2^t distributed nodes (n is the number of bits of hidden information). Experiments on the Qiskit platform demonstrate a reduction in single-node quantum circuit depth relative to the original Kuperberg's algorithm. In addition, an improvement in measurement success rates confirms the algorithm's practical viability.

The remainder of this paper is organized as follows. Section 2 briefly reviews the Kuperberg's algorithm. Section 3 presents the distributed Kuperberg's algorithm and its details. Section 4 provides experiments to prove the feasibility of the algorithm. In Section 5 we conclude with a summary.

2. Review of Kuperberg's algorithm

Kuperberg's algorithm[14] is a quantum algorithm for solving the hidden shift problem, particularly over the dihedral group D_N. This problem is formally characterized as a generalization of the hidden subgroup problem within a non-abelian group,and it is connected to certain non-standard discrete logarithm problems. Compared to Ref. [15], this algorithm has a subexponential time complexity $O(2^{\sqrt{n}})$, showing significant quantum advantages in dealing with such non-abelian group problems. Therefore, the algorithm is a potential threat to some non-abelian group (lattice-based) cryptosystems and has important application value in quantum cryptanalysis.[16]

In the following, we present a concise exposition of Kuperberg's algorithm. Define two injective functions $f, g : Z_N \to R$, where Z_N is an additive group of integers and R is the output set. Then $f(x) = g(x+a)$ is satisfied for any $x \in Z_N$. The goal of Kuperberg's algorithm is to determine the shift a by performing a minimum number of queries to oracles, which can efficiently query the values of functions f and g as follows:

$$|b\rangle|x\rangle|y\rangle \xrightarrow{o} \begin{cases} |0\rangle|x\rangle|y \oplus f(x)\rangle, & \text{if } b = 0, \\ |1\rangle|x\rangle|y \oplus g(x)\rangle, & \text{if } b = 1. \end{cases} \tag{1}$$

The detailed steps of Kuperberg's algorithm are formally outlined in Algorithm 1.

Algorithm 1: Kuperberg's algorithm

Input: Oracles about functions $f, g : Z_N \to R$, where $N{=}2^n$, R for any set.

Output: a_0, the last bit of the hidden shift a

Step 1: Prepare $|0\rangle_1|0^{\otimes n}\rangle_2|0^{\otimes m}\rangle_3$, apply $H^{\otimes(n+1)}$ to the first and second registers

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle_1 + |1\rangle_1) \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle_2|0\rangle_3.$$

Step 2: Query the oracles about f, g

$$|\psi\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} \left(|0\rangle_1|x\rangle_2|f(x)\rangle_3 + |1\rangle_1|x\rangle_2|g(x)\rangle_3\right).$$

Step 3: Measure the third register to get y_0. The first and second registers turn into

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_1|x_0\rangle_2 + |1\rangle_1|x_0 + a\rangle_2\right).$$

Step 4: Apply quantum Fourier transform (QFT) to the second register

$$|\psi\rangle = \frac{1}{\sqrt{2^{n+1}}}\left(\sum_{j=0}^{2^n-1} \mathrm{e}^{2\pi i j x_0/2^n}|0\rangle_1|j\rangle_2 + \sum_{k=0}^{2^n-1} \mathrm{e}^{2\pi i k(x_0+a)/2^n}|1\rangle_1|k\rangle_2\right).$$

Step 5: Measure the second register to get l, then the first register turns into

$$|\psi_l\rangle = \frac{1}{\sqrt{2}}\left(\mathrm{e}^{2\pi i l x_0/2^n}|0\rangle_1 + \mathrm{e}^{2\pi i l(x_0+a)/2^n}|1\rangle_1\right) = |0\rangle_1 + \mathrm{e}^{2\pi i l a/2^n}|1\rangle_1.$$

Step 6: $|\psi_{2^{n-1}}\rangle$ is obtained by the sieve method proposed by Kuperberg, with l equal to 2^{n-1}, we get

$$|\psi_{2^{n-1}}\rangle = |0\rangle_1 + \mathrm{e}^{\pi i a}|1\rangle_1.$$

Step 7: Apply H to the first register and measure it. When the last bit of a is even, it yields $|0\rangle_1$, otherwise $|1\rangle_1$.

In particular, Kuperberg's sieve method (Step 6) crucially reduces query complexity to sub-exponential. In summary, a large number of quantum states with identical final $\lceil\sqrt{n-1}\rceil$ bits are generated, then two of these states are selected to perform a controlled combination operation. The resulting composite states then serve as inputs for subsequent iterations of the process. Once a number of states is achieved, states with identical last $2\lceil\sqrt{n-1}\rceil$ bits are selected and subjected to further combination procedures. This iteration persists until a target quantum state is synthesized. The first bit of the target quantum state is uncertain, while all remaining bits are 0. Remarkably, a single execution of Algorithm 1 suffices to deterministically recover the least significant bit a_0 of a. Then, the full reconstruction of the bit string a is achieved by recursively applying the algorithm with appropriately redefined functions $f'(x) = f(2x)$ and $g'(x) = g(2x + a_0)$.

Finally, the query complexity of Kuperberg's algorithm is discussed. The algorithm's query complexity is primarily

determined by two parameters. One is the bit length n of the hidden shift a. Another is the sieving procedure, which incurs a query complexity of $2^{O(\sqrt{n-k})}$ per iteration, where k denotes the iteration count. Consequently, the total complexity is $2^{O(\sqrt{n})}$, and the space complexity is $2^{O(\sqrt{n})}$.[14]

3. Distributed Kuperberg's algorithm

Before presenting the distributed Kuperberg's algorithm in detail, we first introduce the function decomposition, which serves as a foundation for understanding the algorithm's structure. We then describe the algorithmic procedure in both two-node and multi-node settings. Finally, we analyze the computational complexity and compare the results with the original Kuperberg's algorithm.

3.1. Function decomposition

Based on the function established,[11] we propose a detailed exposition of the function decomposition employed in the distributed Kuperberg's algorithm.

Definition 1 Set the original injective functions $f,g: Z_N \to R$ satisfy $f(x)=g(x+a)$, where $x \in Z_N$, $a \in Z_N$, $x,a = n = \lceil \log_2 |Z_N| \rceil$. To facilitate function decomposition, the input is decomposed into two distinct parts, such as $x = w||u$, $u \in \{0,1\}^{n-t}, w \in \{0,1\}^t$, where $||$ refer to the string concatenation operation. Define the function decomposition:

$$f_{\mathrm{w}}(u) = f(w||u), \quad g_{\mathrm{w}}(u) = g(w||u). \tag{2}$$

Assume there are 2^t distributed quantum computing nodes, which are denoted as N_w, $w \in \{0,1\}^t$. Each subfunction f_w and g_w is allocated to an isolated quantum node N_w to handle $(n-t)$ bits of input. Finally, the outputs of all subfunctions, computed in parallel across the nodes, are subsequently collected and aggregated into a set.

Definition 2 For each input $u \in \{0,1\}^{n-t}$, there exist sets $H(u)$ and $R(u)$, respectively:

$$\begin{aligned} H(u) &= \{f_{\mathrm{w}}(u) \mid w \in \{0,1\}^t\}, \\ R(u) &= \{g_{\mathrm{w}}(u) \mid w \in \{0,1\}^t\}. \end{aligned} \tag{3}$$

Due to the injective properties of f and g, the sets $H(u)$ and $R(u)$ are devoid of duplicate elements. However, the hidden shift a remains non-recoverable directly from these sets for two critical reasons. (i) The computational outputs across nodes are mutually independent, preventing direct establishment of inter-node correlations. (ii) The elements in $H(u)$ and $R(u)$ exhibit no explicit correlation with the hidden shift a. To establish the connection between subfunction values and the hidden shift, we impose a global ordering on the set elements. The strings $F(u)$ and $G(u)$ are defined through the concatenation of lexicographically sorted elements from of $H(u)$ and $R(u)$, respectively.

Definition 3 For each input $u \in \{0,1\}^{n-t}$, the sorted strings $F(u)$ and $G(u)$ are as follows:

$$\begin{aligned} F(u) &= f(w_0||u)||f(w_1||u)||\cdots||f(w_{2^t-1}||u) \in \{0,1\}^{2^t m}, \\ G(v) &= g(w_0||v)||g(w_1||v)||\cdots||g(w_{2^t-1}||v) \in \{0,1\}^{2^t m}. \end{aligned} \tag{4}$$

$f(w_0||u) \leqslant f(w_1||u) \leqslant \cdots \leqslant f(w_{2^t-1}||u), g(w_0||v) \leqslant g(w_1||v) \leqslant \cdots \leqslant g(w_{2^t-1}||v), w_i \in \{0,1\}^t$, when $i \neq j$, $w_i \neq w_j$.

The ordering process ensures that the construction of $F(u)$ and $G(u)$ depends on the function values and not on the specific computing nodes, thereby removing any data dependency between nodes. It then suffices to demonstrate the relationship between the ordered results and the hidden shift a, which can be obtained through measurements of the sorted outputs.

Theorem 1 Let $a = a_1 \parallel a_2$, where $a_1 \in \{0,1\}^t$, $a_2 \in \{0,1\}^{n-t}$, for all $u \in \{0,1\}^{n-t}$, $v \in \{0,1\}^{n-t}$, there exists a_2 such that:

$$F(u) = G(v) \quad \text{if and only if} \quad u + a_2 = v. \tag{5}$$

Proof By the injectivity of f and g, the elements in $H(u)$ and $R(v)$ are distinct for all $u, \upsilon \in \{0,1\}^{n-t}$. $H(u) = R(v)$ if and only if $F(u)$=$G(v)$, is equivalent to $F(u) = G(v)$ if and only if $u + a_2 = v$. Let $w + a_1 = w'$:

Necessity $\forall z \in H(u)$, $\exists w \in \{0,1\}^t$, $z = f(w||u)$. Since $f(w||u) = g(w||u+a)$, $z = g(w||u+a)$. When $u + a_2$ is added without carry-over, $z = g(w+a_1||u+a_2) = g(w'||v) \in R(v)$. If $u + a_2$ is added with carry-over, then $z = g(w+a_1||u+a_2) = g(w'+1||v) \in R(v)$. Hence, $H(u) \subseteq R(v)$. Similarly, $R(v) \subseteq H(u)$ implies $F(u) = G(v)$.

Sufficiency If $F(u) = G(v)$, $\forall z \in H(u)$, $\exists w \in \{0,1\}^t$, $z = f(w||u) = g(w'||u+a)$. When $u + a_2$ is added without carry-over, $w||u + a = w'||v, w + a_1 = w', u + a_2 = v$. When $u + a_2$ is added with carry-over, $w||u+a = w'+1||v, u+a_2 = v$. Therefore, $u + a_2 = v$.

3.2. The distributed Kuperberg's algorithm

Based on the function decomposition described above, we present the implementation of the distributed Kuperberg's algorithm. The fundamental idea of the algorithm is first introduced.

(i) The original function in Kuperberg's algorithm is decomposed into 2^t subfunctions $\{f_w(u), g_w(u)\}$, where $w \in \{0,1\}^t$. Each node N_w is responsible for processing the corresponding pair of subfunctions $f_w(u)$ and $g_w(u)$.

(ii) Subfunction queries are executed in parallel across nodes, generating the quantum state

$$\frac{1}{\sqrt{2}} \left(|0\rangle|u\rangle|f_w(u)\rangle + |1\rangle|u\rangle|g_w(u)\rangle \right).$$

Sort the data from all distributed nodes in the quantum sorting register.

(iii) The sorting register is measured, and the quantum Fourier transform (QFT) is applied to extract the final bit of a_2.

(iv) New functions are subsequently defined as $f_{\rm w}'(u) = f_{\rm w}(2u)$ and $g_{\rm w}'(u) = g_{\rm w}(2u + a_0)$, where a_0 is the least significant bit of a_2. By recursively applying this procedure, the remaining bits of a_2 and a_1 are determined, leading to the full reconstruction of the hidden shift parameter a.

Following the algorithmic process, we present the distributed Kuperberg's algorithm for the two-node (case $t = 1$), as specified in Algorithm 2.

Algorithm 2: Distributed Kuperberg's algorithm (2 nodes)

Input: the sub-function oracles O_{f_w} and O_{g_w}, $w \in \{0,1\}^2$.

Output: the last bit a_0 of the hidden shift a_2 in the shift $a = a_1||a_2$, $a_1 \in \{0,1\}^1, a_2 \in \{0,1\}^{n-1}$.

Step 1: Prepare the quantum state $|0\rangle_1|0^{\otimes n-1}\rangle_2|0^{\otimes m}\rangle_3|0^{\otimes m}\rangle_4|0^{\otimes 2m}\rangle_5$ and apply $H^{\otimes n}$ to registers 1 and 2

$$\frac{1}{\sqrt{2}}(|0\rangle_1 + |1\rangle_1)\frac{1}{\sqrt{2^{n-1}}}\sum_{u\in\{0,1\}^{n-1}}|u\rangle_2|0\rangle_3|0\rangle_4|0\rangle_5.$$

Step 2: Query the oracles O_{f_w} and O_{g_w}

$$\frac{1}{\sqrt{2}}\frac{1}{\sqrt{2^{n-1}}}\sum_{u\in\{0,1\}^{n-1}}(|0\rangle_1|u\rangle_2|f_1(u)\rangle_3|f_0(u)\rangle_4|0\rangle_5 + |1\rangle_1|u\rangle_2|g_1(u)\rangle_3|g_0(u)\rangle_4|0\rangle_5)\,.$$

Step 3: Using the operator $\boldsymbol{U}_{\rm sort}$ to sort the data in registers 3 and 4, and the outcome is stored in register 5

$$\frac{1}{\sqrt{2}}\frac{1}{\sqrt{2^{n}}}\sum_{u\in\{0,1\}^{n-1}}(|0\rangle_1|u\rangle_2|f_1(u)\rangle_3|f_0(u)\rangle_4|S(u)\rangle_5 + |1\rangle_1|u\rangle_2|g_1(u)\rangle_3|g_0(u)\rangle_4|T(u)\rangle_5).$$

Step 4: Measure register 5. The remaining registers turn into

$$\frac{1}{\sqrt{2}}(|0\rangle_1|u_0\rangle_2 + |1\rangle_1|u_0 + a_2\rangle_2).$$

Step 5: Apply QFT to register 2

$$\frac{1}{\sqrt{2^n}}\left(\sum_{j=0}^{2^{n-1}-1}\mathrm{e}^{2\pi\mathrm{i}ju_0/2^{n-1}}|0\rangle_1|j\rangle_2 + \sum_{k=0}^{2^{n-1}-1}\mathrm{e}^{2\pi\mathrm{i}k(u_0+a_2)/2^{n-1}}|1\rangle_1|k\rangle_2\right)$$

Step 6: Measure register 2 to get l, then register 1 becomes

$$\begin{aligned}|\psi_l\rangle &= \frac{1}{\sqrt{2}}\left(\mathrm{e}^{2\pi\mathrm{i}lu_0/2^n}|0\rangle_1 + \mathrm{e}^{2\pi\mathrm{i}l(u_0+a_2)/2^n}|1\rangle_1\right)\\ &= \frac{1}{\sqrt{2}}\mathrm{e}^{2\pi\mathrm{i}lu_0/2^n}\left(|0\rangle_1 + \mathrm{e}^{2\pi\mathrm{i}la_2/2^n}|1\rangle_1\right) = |0\rangle_1 + \mathrm{e}^{2\pi\mathrm{i}la_2/2^n}|1\rangle_1.\end{aligned}$$

Step 7: Use the sieve method to get $|\psi_{2^{n-2}}\rangle$, which means l is equal to 2^{n-2}. Apply H to the first register and measure it. When the last bit of a is even, it yields 0, otherwise 1.

Here $\boldsymbol{U}_{\rm sort}$ compares the values of two registers and performs a lexicographical-order-dependent XOR operation on the third register:

$$\boldsymbol{U}_{\rm sort}|q\rangle|p\rangle|r\rangle = \begin{cases}|q\rangle|p\rangle|r\otimes(q||p)\rangle, & p \geq q,\\ |q\rangle|p\rangle|r\otimes(p||q)\rangle, & p < q,\end{cases} \quad p,q\in\{0,1\}^m, \quad r\in\{0,1\}^{2m}. \tag{6}$$

Evidently, $\boldsymbol{U}_{\rm sort}$ executes a controlled operation contingent upon the states of the quantum registers. To sort 2^t elements with $O(t)$ complexity, the implementation of $U_{\rm sort}$ can be adapted to employ a sort network.[17] Specifically, this can be achieved by loading the target registers for sorting into a sorting register, and then applying a sorting network to the sorting register. This optimization serves as a foundation for the distributed Kuperberg's algorithm over 2^t nodes, which is formally outlined in Algorithm 3 and illustrated schematically in Fig. 1.

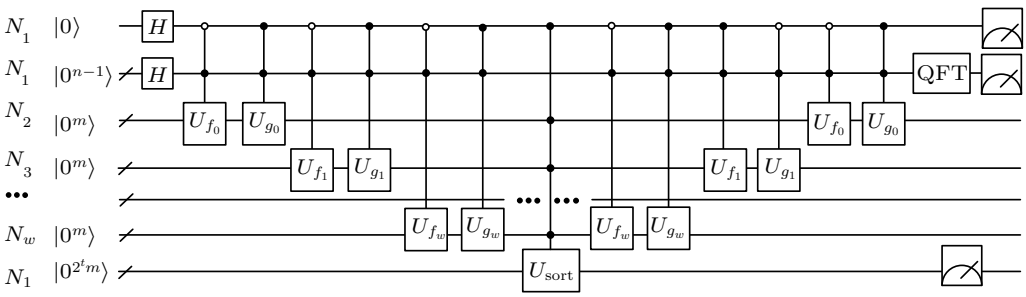


Fig. 1. Quantum circuit of the distributed Kuperberg's algorithm with 2^t nodes. Each node $N_{\rm w}$ processes subfunction oracles $f_w(u)$ and $g_w(u)$. The circuit includes state preparation, oracle queries, global sorting via $\boldsymbol{U}_{\rm sort}$, and QFT-based extraction of the hidden shift bits.

Algorithm 3: Distributed Kuperberg's algorithm (2^t nodes)

Input: the sub-function oracles O_{f_w} and O_{g_w}, $w \in \{0,1\}^t$.

Output: the last bit a_0 of the hidden shift a_2 in the shift $a = a_1||a_2$, $a_1 \in \{0,1\}^t, a_2 \in \{0,1\}^{n-t}$.

Step 1: Prepare the quantum state$|0\rangle_1|0\rangle_2|0^{\otimes m}\rangle_3|0^{\otimes m}\rangle_4\ldots|0^{\otimes m}\rangle_{2^t+2}|0^{\otimes 2^m}\rangle_{2^t+3}$ and apply $H^{\otimes n}$ to registers 1 and 2

$$\frac{1}{\sqrt{2}}(|0\rangle_1+|1\rangle_1)\frac{1}{\sqrt{2^{n-t}}}\sum_{u\in\{0,1\}^{n-t}}|u\rangle_2|0^{\otimes m}\rangle_3\otimes|0^{\otimes m}\rangle_4\ldots\otimes|0^{\otimes 2^t m}\rangle_{2^t+2}|0^{\otimes 2^m}\rangle\otimes{}_{2^t+3}.$$

Step 2: Query the oracle

$$\frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2^{n-t}}}\sum_{u\in\{0,1\}^{n-t}}\Big(|0\rangle_1|u\rangle_2|f_{w_0}(u)\rangle_3|f_{w_1}(u)\rangle_4\cdots|f_{w_{2^t-1}}(u)\rangle_{2^t+2}|0\rangle_{2^t+3}$$

$$+|1\rangle_1|u\rangle_2|g_{w_0}(u)\rangle_3|g_{w_1}(u)\rangle_4\cdots|g_{w_{2^t-1}}(u)\rangle_{2^t+2}|0\rangle_{2^t+3}\Big).$$

Step 3: Using the operator $\boldsymbol{U}_{\text{sort}}$ to sort the data in registers 3 to 2^t+2 and the outcome is stored in the last register

$$\frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2^{n-t}}}\sum_{u\in\{0,1\}^{n-t}}\Big(|0\rangle_1|u\rangle_2|f_{w_0}(u)\rangle_3|f_{w_1}(u)\rangle_4\cdots|f_{w_{2^t-1}}(u)\rangle_{2^t+2}|S(u)\rangle_{2^t+3}$$

$$+|1\rangle_1|u\rangle_2|g_{w_0}(u)\rangle_3|g_{w_1}(u)\rangle_4\cdots|g_{w_{2^t-1}}(u)\rangle_{2^t+2}|T(u)\rangle_{2^t+3}\Big).$$

Step 4: Measuring the last register, the remaining registers are becoming

$$\frac{1}{\sqrt{2}}(|0\rangle_1|u_0\rangle_2+|1\rangle_1|u_0+a_2\rangle_2).$$

Step 5: Apply QFT to register 2

$$\frac{1}{\sqrt{2^n}}\Big(\sum_{j=0}^{2^{n-t}-1}\mathrm{e}^{2\pi iju_0/2^{n-t}}|0\rangle_1|j\rangle_2+\sum_{k=0}^{2^{n-t}-1}\mathrm{e}^{2\pi ik(u_0+a_2)/2^{n-t}}|1\rangle_1|k\rangle_2\Big).$$

Step 6: Measure register 2 to get l, register 1 becomes

$$|\psi_l\rangle=\frac{1}{\sqrt{2}}\left(\mathrm{e}^{2\pi ilu_0/2^n}|0\rangle_1+\mathrm{e}^{2\pi il(u_0+a_2)/2^n}|1\rangle_1\right)=|0\rangle_1+\mathrm{e}^{2\pi ila_2/2^n}|1\rangle_1.$$

Step 7: Use the sieve method to get $|\psi_{2^{n-2}}\rangle$, which means l is equal to 2^{n-t-1} . Apply H the first register and measure it. When the last bit of a is an even measurement, it yields 0, otherwise 1.

Then, redefining the original functions as $f'_w(u)=f_w(2u)$ and $g'_w(u)=g_w(2u+a_0)$, execute Algorithm 3 to iteratively extract the remaining bits of a_2. Similarly, the parameter a_1 is recursively deduced via the aforementioned function construction process. Through this iterative process, the complete hidden shift a can be obtained.

3.3. Complexity analysis

The query complexity of the proposed distributed algorithm is predominantly determined by the two components of a, a_1 and a_2. Specifically, the computation of a involves a sieving process with a query complexity of $2^{O(\sqrt{n-t})}$, where $n-t$ is the bit length of a_2. Then $O(n-t)$ time iterations required to resolve all bits of a_2. Therefore, the query complexity for calculating a_2 is $2^{O(\sqrt{n-t})}+2^{O\left(\sqrt{n-t-1}\right)}+\cdots+2^{O\left(\sqrt{2}\right)}=2^{O(\sqrt{n-t})}$. Similarly, the complexity for resolving a_1 is $2^{O(\sqrt{t})}+2^{O\left(\sqrt{t-1}\right)}+\cdots+2^{O\left(\sqrt{2}\right)}=2^{O(\sqrt{t})}$. Consequently, the total complexity is $2^{O\left(\sqrt{n-t}\right)}+2^{O(\sqrt{t})}$. A key question in terms of query complexity is how to optimally choose t for a given n. To address this, we analyze the query complexity with the function $f(t)=2^{\sqrt{n-t}}+2^{\sqrt{t}}$. Obviously, the function $f(t)$ is strictly decreasing on $(0,n/2)$ and strictly increasing on $(n/2,n)$. Thus, the optimal choice of t can be taken as $n/2$ and the query complexity is $2^{O(\sqrt{n/2})}$. However, choosing $t=n/2$ requires $2^{n/2}$ nodes, which is not practical. Therefore, we consider the case $t\ll n$, where $2^{O(\sqrt{t})}\ll 2^{O(\sqrt{n-t})}$. Under this case, the overall query complexity can be approximated by $2^{O(\sqrt{n-t})}$.

In addition to query complexity, the cost of inter-node quantum operations is generally a bottleneck in distributed architectures. However, in our algorithm, the inter-node cost remains negligible. This is because the inter-node operations occur during the data transfer phase of $\boldsymbol{U}_{\text{sort}}$. Specifically, $\boldsymbol{U}_{\text{sort}}$ performs 2^t quantum teleportations of m-qubit registers.[10] Due to the independence of different nodes, these 2^t transfers can be performed in parallel. Therefore, the total cost of inter-node operation is $O(m)$. Because m is independent of n, we can assume $m=O(1)$. As a result, the impact on algorithm efficiency is minimal.

Moreover, gate complexity and circuit depth of $\boldsymbol{U}_{\text{sort}}$ are important aspects to consider. As previously discussed, $\boldsymbol{U}_{\text{sort}}$ consists of two steps: data transfer and sorting. The transfer step has already been analyzed. Here, we focus on the sorting step, which requires $O(t)$ depth and $O(2^t\cdot t)$ comparators.[17] Each comparator is implemented as a reversible quantum circuit based on the construction described in Appendix B of Ref. [18]. The comparator has a gate complexity and T-count of $O(m)$, and a circuit depth of $O(\log m)$. Therefore, the gate complexity and T-count of $\boldsymbol{U}_{\text{sort}}$ scale as $O(2^t\cdot t\cdot m)$, and the circuit depth scales as $O(t\cdot\log m)$. The detailed resource esti-

mates are listed in Table 1. Since $m = O(1)$, it is evident that the resource cost of $\boldsymbol{U}_{\text{sort}}$ depends primarily on t. This further supports our choice of setting $t \ll n$ to ensure the algorithm remains practically implementable.

Table 1. Resource cost of a m-qubit comparator and $\boldsymbol{U}_{\text{sort}}$.

Resource cost	Gate complexity	T-count	Circuit depth
m-qubit comparator	$O(m)$	$O(m)$	$O(\log m)$
$\boldsymbol{U}_{\text{sort}}$	$O(2^t \cdot t \cdot m)$	$O(2^t \cdot t \cdot m)$	$O(t \cdot \log m)$

To further illustrate the advantages of the proposed algorithm, we compare it with original algorithm. In terms of query complexity, the Kuperberg's algorithm requires a query complexity of $2^{O(\sqrt{n})}$, whereas our distributed architecture achieves only $2^{O(\sqrt{n-t})}$. This demonstrates a reduction in oracle queries compared to the original algorithm. Furthermore, the distributed algorithm optimizes quantum circuit depth at each node. Specifically, the decomposition of the original function enables a substantial reduction in per-node circuit depth relative to the original implementation. The shallower circuit depth at each node enhances operational feasibility on current quantum devices and improves resilience to noise.

4. Experiments

To illustrate the efficacy of the proposed distributed algorithm, we perform experimental simulations on the Qiskit platform (v0.44). In the experiments, we construct injective functions: $f, g : \{0,1\}^3 \to \{0,1\}^4$, satisfying $f(x) = g(x+a \mod N)$, The corresponding truth tables of f and g are presented in Table 2. We implement the original Kuperberg's algorithm on a 4-qubit node and the distributed Kuperberg's algorithm on a dual-node where each node consists of 3 qubits, with 1 ancilla qubit shared between them. Each of the two algorithms was carried out 2,048 times to get the last bit of the hidden shift $a = 111_2 = 7_{10}$ (binary to decimal).

Table 2. Truth table.

x	$g(x)$	$f(x)$	x	$g(x)$	$f(x)$
000	1001	1000	100	0111	0101
001	1100	1001	101	0011	0111
010	1010	1100	110	0001	0011
011	0101	1010	111	1000	0001

To construct the function oracle components required for both the original and the distributed Kuperberg's algorithm, the DORCIS framework[19] was utilized, as illustrated in Figs. 2 and 3.

In contemporary universal fault-tolerant quantum architectures, the "Clifford + T" gate set is conventionally adopted to achieve universal quantum computation.[20] Notably, the fault-tolerant implementation of T gates entails significantly higher resource overhead compared to Clifford group operations, often exceeding the latter by an order of magnitude. As a result, the T-depth serves as a critical determinant of circuit execution time.[21] Reducing the number of T gates decreases the required number of ancilla qubits, significantly minimizing the circuit footprint while enhancing fidelity. Table 3 quantifies the T-depth and total circuit depth for oracles implemented under both the original function and its decomposed subfunctions. These results demonstrate the depth reduction achieved through distributed function decomposition.

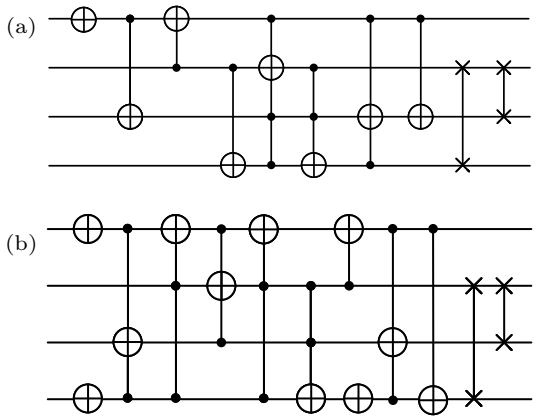


Fig. 2. Quantum circuit of original Kuperberg's algorithm oracles. (a) Quantum circuit of O_f. (b) Quantum circuit of O_g.

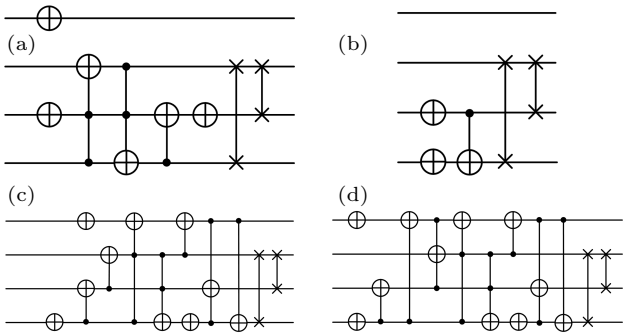


Fig. 3. Quantum circuits of distributed Kuperberg's algorithm oracles. (a) Quantum circuit of O_{f_0}. (b) Quantum circuit of O_{f_1}. (c) Quantum circuit of O_{g_0}. (d) Quantum circuit of O_{g_1}.

Table 3. Circuit depth of oracle: before and after functional decomposition.

	O_f	O_g	O_{f_0}	O_{f_1}	O_{g_0}	O_{g_1}
T-depth	24	24	8	0	12	16
Full depth	58	57	25	9	35	43

Functional decomposition of the original function simplifies the subfunction oracle implementations, enabling the proposed distributed algorithm to achieve significant T-depth reduction and improved compatibility with contemporary quantum devices. This finding generalizes to broader computational scenarios.

The experimental methodology encompassed superposition state initialization, oracle queries, quantum sorting operation $\boldsymbol{U}_{\text{sort}}$, and QFT measurement. As demonstrated in Fig. 4,

the distributed implementation achieved a measurement success rate of 22.6%, significantly higher than the 10.6% observed for the original algorithm. These results empirically confirm the efficacy of the distributed Kuperberg's algorithm in reducing quantum resource overhead and enhancing measurement success probabilities.

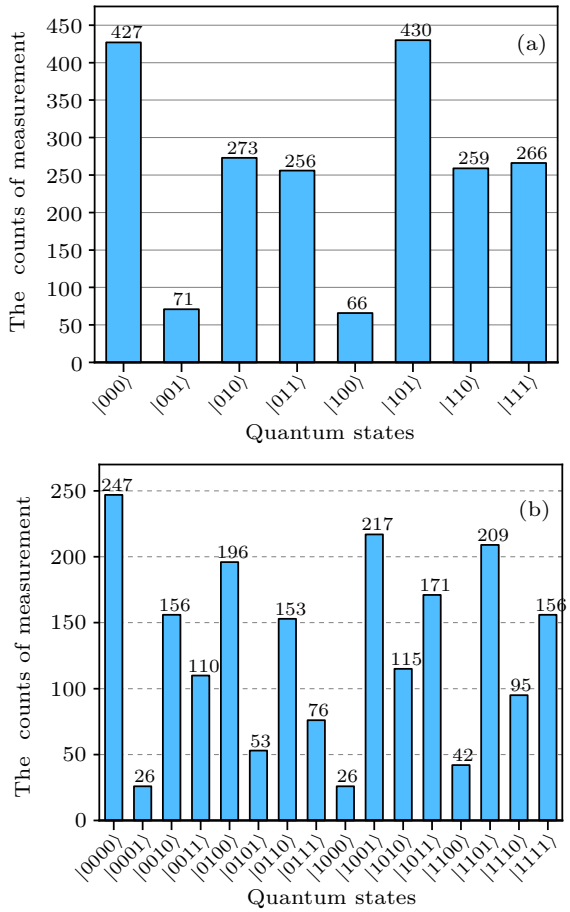


Fig. 4. Comparisons of experimental results: (a) distributed Kuperberg's algorithm with input bit 3 and correct result 101, (b) Kuperberg's algorithm with input bit 4 and correct result 1001.

5. Conclusion

In this paper, we present a distributed variant of Kuperberg's algorithm, optimized via functional decomposition. By distributing computation across multiple nodes operating in parallel, the number of oracle queries required per node is significantly reduced. This reduction not only mitigates noise but also facilitates the practical realization of subexponential speedups on current NISQ devices. Furthermore, to validate the efficiency of the proposed algorithm, we implemented it on the Qiskit platform and compared its performance against the original Kuperberg's algorithm. Experimental results demonstrate that our distributed approach achieves notably higher measurement success rates and significantly reduced circuit depth, thereby making it more suitable for execution on NISQ devices.

Acknowledgements

Project supported by the National Natural Science Foundation of China (Grant No. 62171131), the Natural Science Foundation of Fujian Province, China (Grant Nos. 2022J01186 and 2023J01533), and the Innovation Program for Quantum Science and Technology (Grant No. 2021ZD0302901).

References

[1] Shor P W 1994 *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, November 20–22 1994 Santa Fe USA p. 124
[2] Grover L K 1997 *Phys. Rev. Lett.* **79** 325
[3] Harrow A W, Hassidim A and Lloyd S 2009 *Phys. Rev. Lett.* **103** 150502
[4] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 *Nature* **549** 195
[5] Preskill J 2018 *Quantum* **2** 79
[6] Greinert F, Müller R, Goorney S R, Laurenza R, Sherson J and Ubben M S 2024 *European competence framework for quantum technologies Geneva* (Zenodo) pp. 1–30
[7] Caleffi M, Cacciapuoti A S and Bianchi G 2018 *Proceedings of the 5th ACM International Conference on Nanoscale Computing and Communication*, September 5–7 2018 Reykjavik Iceland p. 3
[8] Chen Z L, Guan Z J, Zhao S X and Cheng X Y 2025 *Chin. Phys. B* **34** 050305
[9] Luo T Y, Zheng Y Z, Fu X and Deng Y X 2024 *Chin. Phys. B* **33** 120302
[10] Tian L, Sun L L, Zhu X Y, Song X K, Yan L L, Liang E J, Su S L and Feng M 2020 *Chin. Phys. B* **29** 050306
[11] Avron J, Casper O and Rozen I 2021 *Phys. Rev. A* **104** 052404
[12] Tan J, Xiao L, Qiu D, Luo L and Mateus P 2022 *Phys. Rev. A* **106** 032417
[13] Qiu D, Luo L and Xiao L 2024 *Theor. Comput. Sci.* **993** 114461
[14] Kuperberg G 2005 *SIAM J. Comput.* **35** 170
[15] Ettinger M, Hoyer P and Knill E 1999 arXiv:quant-ph/9901034 [quant-ph]
[16] Regev O 2004 *SIAM J. Comput.* **33** 738
[17] Paterson M S 1990 *Algorithmica* **5** 75
[18] Berry D W, Kieferová M, Scherer A, Sanders Y R, Low G H, Wiebe N, Gidney C and Babbush R 2018 *npj Quantum Inf.* **4** 22
[19] Chun M, Baksi A and Chattopadhyay A 2023 *Cryptol. ePrint Arch.* **286**
[20] Bravyi S and Kitaev A 2005 *Phys. Rev. A* **71** 022316
[21] Fowler A G 2012 arXiv:1210.4626 [quant-ph]