Benchmarking Enhanced Surface Codes for Grover's Algorithm

Spencer Dearman[1, 2, *] and Victory Omole[1, †]
[1] *Infleqtion, Chicago, IL 60604, USA*
[2] *University of Chicago, Chicago, IL 60637, USA*
(Dated: July 29, 2024)

Question: How can minimizing the error correction overhead in quantum computing, specifically through the use of long-range-enhanced surface codes on a neutral atom architecture enhance the practical feasibility and effectiveness of achieving quadratic speedups with Grover's algorithm on k-SAT problems?

I. INTRODUCTION

Integrating Grover's Algorithm into quantum computing shows promise in unordered search problems where it can outperform classical algorithms. Traditional algorithms for boolean satisfiability often use brute-force or heuristics, but Grover's Algorithm reduces the number of checks, solving problems more efficiently. However, anticipated quadratic speedups are limited by surface code error correction overhead, slowing down logical operations [1]. It is estimated that bridging the performance gap will require over 10 billion function calls for practical quantum speedup [2]. This research aims to reduce error correction overhead by leveraging an enhanced-nearest-neighbor surface code [3] for faster speeds and more dense logical qubit encoding rates compared to standard surface codes.

II. ALGORITHM

Grover's Algorithm can evaluate a black-box quantum oracle function in $O(\sqrt{N})$ iterations while only using $O(log_2 N)$ qubits, where N is the number of possible input combinations to the function [4]. The following steps explain the execution of Grover's Algorithm on a k-SAT boolean satisfiability problem [5].

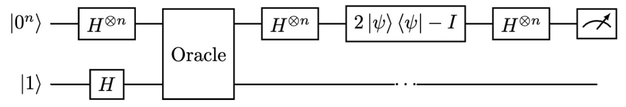

FIG. 1: Grover's Algorithm Diagram

1. **Initialization** Hadamard gates are applied to all qubits, initially in the $|0\rangle^{\otimes n}$ state, to create an equal superposition of all possible states [6]:

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{2^n-1}^{x=0} |x\rangle \tag{1}$$

2. **Oracle Application** The applied oracle is designed to recognize the correct solutions to the problem. For k-SAT problems, this involves checking whether a given assignment satisfies the boolean clause formula. The oracle flips the amplitude of the solution states, which is typically implemented using phase kickback [7]. Essentially, a phase of π (equivalent to multiplying by -1) is applied to the amplitude of the states that satisfy the formula. This marks the correct solutions.

   Several oracles are defined that are essential for evaluating SAT clauses within Grover's Algorithm. The OracleOr function implements a logical OR operation on a register of qubits, flipping a target qubit if any qubit in the register is in the $|1\rangle$ state. Next, the OracleXor function implements a logical XOR operation using CNOT gates. The OracleSATClause function then evaluates individual SAT clauses, which are disjunctions of literals (including variables or their negations), and combines these evaluations together using auxiliary qubits. Finally, the OracleSAT function uses these three oracle functions together to mark the states that satisfy the SAT problem. The oracle checks each clause and marks the states where all clauses are satisfied. The marked state will receive a phase flip if x is a solution:

$$|x\rangle \to -|x\rangle \tag{2}$$

3. **Diffusion Operator** The Grover diffusion operator amplifies the probability amplitude of the marked states. It is a reflection about the average amplitude of the current state vector. Mathematically, this is performed as:

$$D = 2|\psi\rangle\langle\psi| - I \tag{3}$$

   Where $|\psi\rangle$ is the equal superposition state.

4. **Iterations** Steps 2 and 3 are repeated approximately $\sqrt{N}$ times, where $N = 2^n$, and n is the

* spencer.dearman@infleqtion.com
† victory.omole@infleqtion.com

number of qubits. Each iteration increases the amplitude of the correct solutions while decreasing the amplitude of incorrect ones.

5. **Measurement** After a sufficient number of iterations, the system's state is measured. Due to the increased amplitude of the correct solutions, measuring the system will most likely yield one of the correct solutions.

III. RESOURCE ESTIMATION APPROACH

The Azure Quantum Resource Estimator (AQRE) offers a comprehensive suite of tools for testing Grover's Algorithm with various QEC configurations on Inflection's Sqorpius architecture. The AQRE was initialized with the values in Table I based on projections for the Sqorpius architecture in 2028 [8].

TABLE I: Sqorpius Qubit Configuration

Metric	Sqorpius Qubit
1-Q Measurement Time	5 ms
1-Q Gate Time	10 μs
2-Q Gate Time	4 μs
1-Q Measurement Error Rate	2.5%
1-Q Gate Error Rate	0.01%
2-Q Gate Error Rate	0.05%

A. QEC Configuration

To simulate the Sqorpius configuration, specific parameters were input as seen in Table II. The logical cycle time represents the time required for syndrome extraction, and does not account for the amount of time required for lattice surgery operations. In addition, due to the dual-species architecture Sqorpius uses, there is no need to account for movement during syndrome extraction. Therefore, the logical cycle time is represented by the following formula:

$$(4 \times \text{2-Qubit GT} + 2 \times \text{1-Qubit MT}) \times \text{Code Distance}$$

Where GT is gate time and MT is measurement time. This formula is based on the surface code diagram provided in [9]. The error correction threshold, p, was reverse calculated in Eq. 4 using initial values provided from [9].

$$0.02 = 0.03(\frac{p}{0.0057})^4 \tag{4}$$

$P_L = 4$ was found based on Fig. 4 in [9], the crossing prefactor a was given in equation 11 from [9], and p_{th} was given in section VII to be 0.57% [9]. For d value, based on [3] a value of $d = 8$ was used. Once solved, the error correction threshold value, p, was used, as seen in Table II.

$$p = 0.00515 \tag{5}$$

The next value to be calculated is a, the crossing prefactor [10]. This value is calculated differently for the standard surface code versus the LRESC, and is described specifically in the following two sections. Finally, the physical qubits per logical qubit is detailed in [3] by using a [[61, 1, 6]] surface code and a [[244, 4, 8]] LRESC, leading to a consistent rate of 61.

TABLE II: Surface and LRESC Code Configuration

Metric	Value
Logical Cycle Time [9]	(4*2Q GT + 2*1Q MT) * d
Error Correction Threshold	0.00515
Crossing Prefactor	a
Physical Qubits Per Logical [3]	61

1. Surface Code

The Surface Code is a type of quantum error-correcting code specifically designed for fault-tolerant quantum computation. This code is a planar version of the toric code initially proposed by Alexei Kitaev [11]. It utilizes a two-dimensional array of physical qubits, which are stabilized through local interactions to form logical qubits that are more robust against errors. The surface code achieves error correction by measuring a set of stabilizers—specific operator products that help detect and correct errors without directly measuring the qubits' state, thereby preserving quantum information.

One of the significant advantages of the surface code is its relatively high error tolerance, with threshold error rates of up to approximately 1% per operation. This tolerance allows for practical implementation of quantum computers, as it simplifies the physical layout to two dimensions and uses only nearest-neighbor interactions. This design makes the surface code one of the most promising approaches for scalable and fault-tolerant quantum computation, as it can handle a wide range of errors while maintaining the integrity of the quantum information being processed [9].

To determine the crossing prefactor, a, for the surface code it was first necessary to find the logical error rate, P_L, in order to satisfy Eq. 6. In order to find the correct crossing prefactor, a, it was first necessary to find the P_L value in order to solve. This first required using the Noisy Syndromes graph from Fig. 4 in [3] and finding the intersection point between each of the three [[61, 1, 6]] surface codes, and a physical error rate of 0.0057 [9]. This produced the P_L values seen in Table III. Using these given P_L values, a was calculated using the following formula:

$$P_L = a\left(\frac{0.00515}{0.0057}\right)^3 \qquad (6)$$

Where $d = 6$ based on the [[61, 1, 6]] [3] code being used. Eq. 6 produced the a values in Table III.

TABLE III: [[61, 1, 6]] Surface Code P_L and Crossing Prefactor a values

	w	P_L	a
[[61, 1, 6]] Surface Code	1	0.0300	0.0407
	2	0.0056	0.0076
	3	0.0020	0.0027

2. *Long-Range-Enhanced Surface Code*

The Long-Range-Enhanced Surface Codes (LRESCs) represent a significant advancement in quantum error-correcting codes, crucial for quantum computing. These codes integrate long-range interactions to enhance the robustness and capacity of logical qubits, enabling the encoding of additional logical qubits without compromising error-resilience. This approach overcomes the spatial locality constraints of conventional surface codes, which typically require more physical qubits to boost error correction.

LRESCs are designed for implementation with current quantum technologies, like neutral atoms. By incorporating non-local parity checks, LRESCs maintain and can improve fault tolerance, offering a scalable solution for quantum error correction without a prohibitive increase in physical qubits. [3].

Similar to the surface code, determining the crossing prefactor, a, for the LRESC requires finding the P_L value. Using the Noisy Syndromes graph from Fig. 4 in [3], the intersection point between each of the three [[244, 4, 8]] LRESCs and a physical error rate of 0.0057 [9] provided the P_L values in Table IV. Using these P_L values, a was calculated using Eq. 7:

$$P_L = a\left(\frac{0.00515}{0.0057}\right)^4 \qquad (7)$$

Where $d = 6$ based on the [[244, 4, 8]] [3] code. Eq. 7 produced the a values shown in Table IV.

IV. RESULTS

The results demonstrate that the long-range-enhanced surface code (LRESC) provided a slight runtime speedup

TABLE IV: [[244, 4, 8]] LRESC P_L and Crossing Prefactor a values

	w	P_L	a
[[244,4,8]] LRESC	1	0.0270	0.0405
	2	0.0038	0.0057
	3	0.0012	0.0018

when running Grover's algorithm on k-SAT problems. As the number of variables increased, the benefits compounded, and the fastest runtimes were observed when $w = 3$.

TABLE V: Surface Code and LRESC Runtimes (Seconds)

	Variables	$w = 1$	$w = 2$	$w = 3$
Surface Code	8	83.78	74.21	69.42
	16	442.28	399.13	355.98
	24	208.98	187.55	176.83
	32	295.26	266.46	237.65
	40	341.26	307.96	291.32
	48	1093.0	995.84	898.69
	56	1224.61	1115.75	1061.33
	64	2919.13	2670.70	2546.48
LRESC	8	83.78	74.21	64.63
	16	442.28	377.55	355.98
	24	208.98	187.55	166.12
	32	295.26	252.05	237.65
	40	341.26	307.96	274.67
	48	1093.0	947.26	898.69
	56	1224.61	1115.75	1006.90
	64	2919.13	2670.70	2422.26

Qubit usage remained nearly identical between the surface code and the LRESC, aligning with the LRESC's goal of increasing logical qubits without significantly affecting the number of physical qubits. These results are shown in Table VI.

V. OBSERVATIONS

The crossover times between Surface Code and Long-Range-Enhanced Surface Code (LRESC) runtimes reveal significant insights into the efficiency and scalability of quantum error correction methods. For more than 40 variables, LRESC demonstrates faster runtimes compared to Surface Code, particularly at higher values of w. This is highlighted in Fig. 2.

This improvement is crucial for practical quantum computations, reducing computational time and enhancing efficiency. Although the speedup with 64 variables

TABLE VI: Surface Code and LRESC Qubits (Physical)

	Variables	$w = 1$	$w = 2$	$w = 3$
Surface Code	8	35,014	32,574	32,574
	16	34,770	32,330	34,770
	24	36,417	36,417	33,977
	32	36,112	36,112	38,552
	40	37,942	37,942	37,942
	48	42,517	40,077	40,077
	56	44,591	42,151	42,151
	64	46,665	46,665	44,225
LRESC	8	35,014	32,574	32,574
	16	34,770	34,770	34,770
	24	36,417	33,977	33,977
	32	36,112	38,552	36,112
	40	37,942	37,942	37,942
	48	42,517	40,077	40,077
	56	44,591	42,151	42,151
	64	46,665	46,665	46,665

may seem minor, it highlights quantum computing's potential for solving large-scale satisfiability problems, which are important in fields like hardware and software verification, electronic design automation, cryptography, bioinformatics, and AI planning. While not yet suitable for everyday tasks, quantum computers show significant promise for complex problems, underscoring their importance in advanced computation [12].

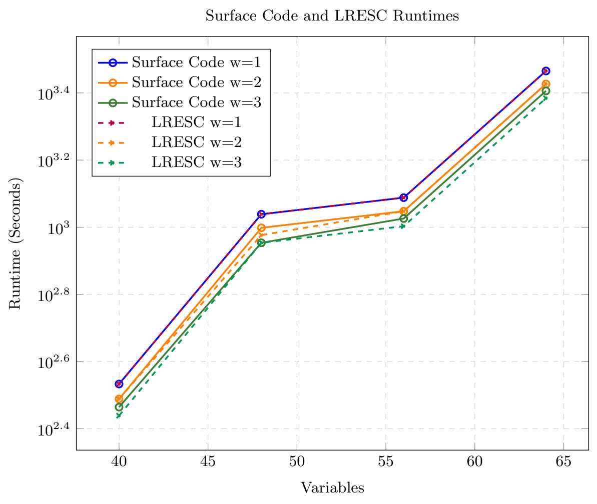

FIG. 2: Surface Code and LRESC Runtimes

VI. CONCLUSION

Overall, the adoption of Long-Range-Enhanced Surface Codes presents a promising advancement in runtime performance for Grover's algorithm on k-SAT problems as the number of variables increases. The observed improvements, particularly for $w = 3$, highlight the potential for LRESC to enhance quantum algorithm efficiency without significantly altering qubit usage. These findings underscore the scalability and effectiveness of LRESC in practical quantum computing applications, making it a promising approach for future developments in quantum error correction and algorithm optimization.

[1] R. Babbush, J. R. McClean, M. Newman, C. Gidney, S. Boixo, and H. Neven, Focus beyond quadratic speedups for error-corrected quantum advantage, PRX Quantum **2**, 10.1103/prxquantum.2.010103 (2021).

[2] T. Hoefler, T. Haener, and M. Troyer, Disentangling hype from practicality: On realistically achieving quantum advantage (2023), arXiv:2307.00523.

[3] Y. Hong, M. Marinelli, A. M. Kaufman, and A. Lucas, Long-range-enhanced surface codes (2024), arXiv:2309.11719 [quant-ph].

[4] L. K. Grover, A fast quantum mechanical algorithm for database search (1996), arXiv:quant-ph/9605043 [quant-ph].

[5] V. Portilheiro, Applying grover's algorithm to unique-k-sat (2018).

[6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).

[7] J. Watrous, Grover's algorithm (2023).

[8] B. Wang, Infleqtion 1600 qubit array today and five year roadmap to fault tolerant quantum computers (2024).

[9] I. A. Rajeev Acharya and R. A. et al., Suppressing quantum errors by scaling a surface code logical qubit (2022), arXiv:2207.06431 [quant-ph].

[10] W. van Dam, M. Mykhailova, and M. Soeken, Using azure quantum resource estimator for assessing performance of fault tolerant quantum computation (2024), arXiv:2311.05801.

[11] A. Kitaev and C. Laumann, Topological phases and quantum computation (2009), arXiv:0904.2771 [cond-mat.mes-hall].

[12] J. Marques-Silva, Practical applications of boolean satisfiability, in *2008 9th International Workshop on Discrete Event Systems* (2008) pp. 74–80.