npj | Quantum Information

www.nature.com/npjqi

ARTICLE OPEN

Optimization of lattice surgery is NP-hard

Daniel Herr[1,2], Franco Nori[1,3] and Simon J. Devitt[1,4]

The traditional method for computation in either the surface code or in the Raussendorf model is the creation of holes or "defects" within the encoded lattice of qubits that are manipulated via topological braiding to enact logic gates. However, this is not the only way to achieve universal, fault-tolerant computation. In this work, we focus on the lattice surgery representation, which realizes transversal logic operations without destroying the intrinsic 2D nearest-neighbor properties of the braid-based surface code and achieves universality without defects and braid-based logic. For both techniques there are open questions regarding the compilation and resource optimization of quantum circuits. Optimization in braid-based logic is proving to be difficult and the classical complexity associated with this problem has yet to be determined. In the context of lattice-surgery-based logic, we can introduce an optimality condition, which corresponds to a circuit with the lowest resource requirements in terms of physical qubits and computational time, and prove that the complexity of optimizing a quantum circuit in the lattice surgery model is NP-hard.

npj Quantum Information (2017)3:35 ; doi:10.1038/s41534-017-0035-1

INTRODUCTION

Quantum computation is a very promising method to perform information processing. Several types of problems, such as prime factorization[1] or search algorithms[2] can be sped up considerably. The first physical realizations have been built,[3–7] where error rates are small enough to allow for effective error-correction and fault-tolerant quantum computation.[8] Many diverse systems can already run small quantum algorithms and projects such as the IBM Quantum Experience,[9] have connected small prototype computers to the cloud for both educational purposes, and to allow other researchers to test small-scale protocols.

One of the remaining tasks for experimentalists is to scale up the number of qubits, while maintaining low error rates, in order to allow more complex algorithms to be performed. The task for theorists is now to build a quantum compiler,[10–13] that can translate high-level algorithms to individual hardware instructions. This compiler has to be aware of the hardware faults and should introduce error-correction to protect logical qubits from physical influences, in order to ensure a completely fault-tolerant computation. Several parts of such a compiler have already been created for both the offline component (before a quantum computer is initiated)[10–13] and the online component (classical software run in tandem with the quantum algorithm),[14, 15] but a complete software package has yet to be developed. Notably, optimization algorithms,[16] which operate at the error-correction level, are still lacking to optimize physical resources in the most commonly used error-correction models. To this end, we inspect the optimization of a specific topologically based operational model called lattice surgery (LS).[17] This representation was chosen particularly because of its applicability to a wide range of hardware models,[18–22] and the applicability of LS approaches using other topological coding techniques.[23–25]

For a practical fault-tolerant computer using LS, both the physical and logical levels are arranged in a 2D nearest-neighbor array, on which a universal gate set can be realized.[17, 26] This 2D, nearest-neighbor environment is enforced by the connectivity of the physical qubit array. For LS this is the planar code,[27] for braiding it is generally the surface code.[8, 27, 28] The common feature of all these representations is that physical qubits are connected via a graph, that indicates their possible interactions. Even non-fault-tolerant implementations suffer from the restricted connectivity of the underlying physical qubits and methods to perform computation on these had to be developed.[29]

Conceptually, algorithmic compilation and optimization is similar to more traditional measurement-based quantum computation, but at the level of error-corrected qubits. The LS translation[26] of an arbitrary circuit creates an algorithmically specific graph state at the encoded level, using the native parity checks of LS. After this encoded graph is created, a time-ordered sequence of non-Clifford measurements is performed on each encoded node in the graph to realize the algorithm. This is akin to traditional measurement-based quantum computation[30] (which is not error-corrected), where a 2D, universal graph state (commonly referred to as a cluster state) is prepared, a quantum circuit mapped to this 2D array and all associated Clifford measurements are performed. The 2D cluster state is then converted to an algorithmically specific graph state where the only subsequent operations needed are a time-ordered sequence of non-Clifford measurements and feed-forward.[26, 31]

Here, we want to evaluate the complexity of the creation of such an encoded graph state. In complexity theory, problems are divided into categories, which determine their hardness. A famous class consists of non-deterministic polynomial complete (NP-complete) problems.[32] Such problems lie in the complexity class NP, such that a solution can be verified in polynomial time, and are at least as hard as the most difficult problems in NP. A common way to determine NP-completeness is to map an already known NP-complete problem to the problem of interest.[32]

[1]Quantum Condensed Matter Research Group, CEMS, RIKEN, Wako-shi 351-0198, Japan; [2]Computational Physics, ETH Zurich, Zurich 8093, Switzerland; [3]Department of Physics, University of Michigan, Ann Arbor, MI 48109-1040, USA and [4]ARC Centre for Engineered Quantum Systems, Department of Physics and Astronomy, Macquarie University, North Ryde, NSW 2109, Australia
Correspondence: Daniel Herr (herrd@phys.ethz.ch)

Received: 27 March 2017 Revised: 21 July 2017 Accepted: 26 July 2017
Published online: 11 September 2017

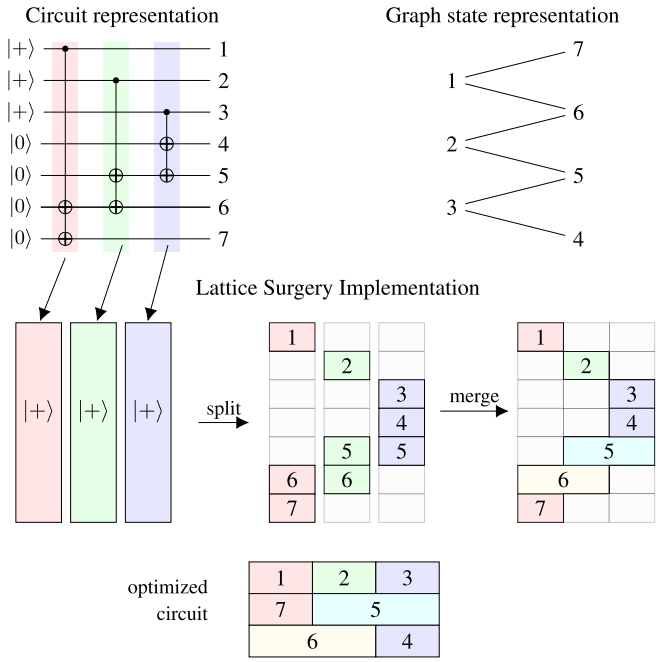

Fig. 1 LS translation. Here, multi-target CNOTs are implemented using LS. During initialization three patches of surface code with *N* by 7*N* qubits are created, which are then split and merged to perform the computation. The faded boxes indicate ancillary qubits. An optimized version of this circuit is shown at the *bottom* where the placement of the patches ensures a minimal bounding box of the whole circuit area. This circuit can achieve the theoretical optimality

We were inspired by the proof of NP-hardness of Tetris,[33] and map the 3-partitioning problem[34] to the optimization of LS patches using the translation devised in ref. 26. This implies that it is also NP-hard to optimize the complete problem including measurements, because this only adds an additional layer of complexity to the system.

We will prove that the circuit optimization of a particular fault-tolerant implementation of topological error-correction is NP-hard. Similarly to our result, it has been shown that it can be NP-hard for a compiler to optimize classical code, such that its execution is time optimal.[35] Our results, thus, urge the development of heuristics that can optimize quantum circuits not exactly, but at least reasonably well, for implementation on realistic quantum hardware. Furthermore, we derive general estimates on best and worst performance. We also discuss the benefit of optimization given a sample algorithm and estimate the hardness of the optimization problem for an exact, classical solver.

The main idea of the LS translation[26] is to encode an algorithmically specific graph state in the square lattice of the planar code, which will then use a measurement-based quantum computational approach to perform any calculation. The implementation of this encoded state needs to respect the underlying structure of the planar code. Many square patches that encode individual qubits[17] are aligned on a 2D lattice. Connections between nearest-neighbor logical qubits are possible using physical qubits that lie on the boundary between the patches. These operations constitute merges and splits that act as parity checks between the two encoded qubits, and can be used together with injection to enable universal quantum computation.[26]

The analysis performed here is rooted in the LS translation given in ref. 26. First, patches are initialized to $|+\rangle$, then using parity checks an algorithmically specific stabilizer state is generated. This stabilizer state is measured in the bases $|Z\rangle$, $|X\rangle$, $|Y\rangle = P|+\rangle$, and $|A\rangle = T|+\rangle$, where $P = \sqrt{Z}$ and $T = \sqrt{P}$. However, for planar codes the rotated basis measurements (e.g., *Y*, *A*) are not protected fault-tolerantly, and magic states must be injected.[17, 26] Our description is only concerned with the creation of the initial algorithmically specific stabilizer state and shows that even the optimization of this less complicated problem is already NP-hard.

An arbitrary circuit can be rearranged into the ICM format,[12] which is already divided into (I)nitializations, (C)NOTs, and (M)easurements. The first two steps can be interpreted as a circuit to generate the stabilizer state. The translation to LS first merges all CNOTs of this circuit into multi-target CNOTs, which can then be easily implemented in LS: For each multi-target CNOT a column in the planar code is created, which is later split into individual encoded qubits (Fig. 1). Then, the qubits that are targeted by two or more CNOTs have to be combined through LS merge operations.

Due to different CNOTs targeting the same qubits multiple times in a general quantum circuit, the compiler naturally produces ancillary qubits, which are inherent to the structure of the algorithm and the compiler. During our calculations these are disregarded and we only study the problem of how to optimally place the patches in the 2D-nearest neighbor environment of the planar code error-correction model using LS.

RESULTS

We will prove that the problem of deciding if a perfectly optimizable layout in LS exists, by mapping a known NP-complete problem to the problem of interest.

Inspired by the proof of NP-hardness of Tetris,[33] we chose to encode the 3-partition problem into a circuit, which then gets translated to LS. We will show that, with polynomial overhead, a solution to the 3-partition problem can be obtained by the optimization of the placement of LS patches.

The proof whether theoretical optimality is reachable implies that the optimization problem itself is NP-hard. This can be explained by using the optimization problem as a subroutine to the decision problem. If the optimization problem was easier, the decision problem would be solvable in polynomial time. With the described mapping, this would mean that any problem in NP could be solved polynomially, which is widely assumed to be false.

The proof itself is given in the methods section.

DISCUSSION

We want to give an estimate on how much of an improvement can be expected from the optimization of a double $|Y\rangle$-state distillation circuit.[8, 36] This circuit is only illustrative for the optimization and we are aware of better proposals to implement $|Y\rangle$-states in surface codes.[37] For reference, we provide the circuit of one distillation step in Fig. 2. In our calculation we give bounds for the best case by calculating the theoretical optimum.

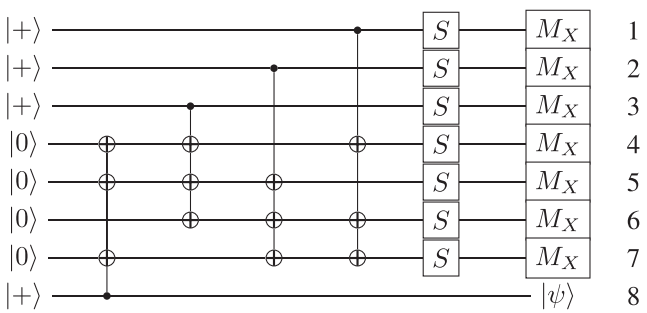

Fig. 2 Steane code. This circuit is the Steane code, to be used for the distillation of $|Y\rangle$ states. This is an iterative procedure where the error-prone *Y* are used during the application of the *S*-gates

Furthermore, the worst case bounds are given by calculating an unoptimized placement, where each qubit corresponds to one row of patches in LS. The definitions for both worst-case and best-case bounds are given in the "Methods" section. However, a previous manual optimization[26] has shown that the $|Y\rangle$-state distillation circuit cannot reach theoretical optimality, such that the best possible solution lies somewhere in between these bounds. In the following back-of-the-envelope calculation we assume that the basis transformation of the measurement step can be applied without movement, which would correspond to a solution of a more complex optimization problem.

The first round of the $|Y\rangle$-state distillation circuit consists of seven distillations. Each distillation consists of four CNOTs from the Steane-code with three target qubits each. Furthermore, for the application of one *S*-gate an additional qubit is needed for the injection procedure. Thus, an additional 7×7 qubits are needed. In a second round, an additional distillation needs to be performed, requiring eight more qubits and four more CNOTs. Furthermore, each distillation circuit consists of eight qubits, and initially 7×7 noisy $|Y\rangle$-states need to be injected. Thus, the total number of qubits needed in this double distillation is $N_Q = 8 \times 7 + 7 \times 7 + 8 = 113$.

The optimal costs can be calculated with Eq. (2) and lead to $32 \times 4 + 7 \times 7 = 177$ encoded patches of the planar code. A suboptimal placement, Eq. (3), where each qubit is fixed to one row, requires $32 \times 113 = 3616$ patches. This difference is a factor of roughly 20, with the difference only growing for larger circuits.

Since this optimization has to be performed by a compiler, which is likely to run on classical hardware, the nature of the optimization being a NP-hard problem will restrict the size of the exactly optimizable instance. To show that it would be unfeasible to exactly optimize even a small amount of individual CNOTs, we look at an exact solution of the number-partitioning problem. An exact algorithm has to loop through all valid configurations to find the best one. We do not consider a dynamic programming solution here, because such an algorithm is unlikely to be devised for the optimization of LS. The reason is that dynamic programming relies on the solution of subproblems. However, connections between different surface code patches required by merges break the structure exploited by dynamic programming approaches. Furthermore, one should note that this optimization needs to be general, such that each circuit can be optimized. Any circuit-specific optimization is therefore discouraged, which makes our claims valid despite the symmetry of the current exemplary circuit. Assuming the same double $|Y\rangle$-state distillation circuit as before, we would have 46 numbers and want to partition these into 15 subsets. The nature of the 3-partitioning problem only allows three numbers per set, such that we only have to consider these configurations. The assignment of $3N$ elements to N sets such that each set contains exactly three elements has

$$N_{\text{config}} = \prod_{i=0}^{N-1} \frac{(3N-3i)!}{3!(3N-3i-3)!} = \frac{(3N)!}{(3!)^N} \tag{1}$$

configurations. These would equal $\sim 10^{44}$ possible configurations for the distillation circuit. A computer with 3.5 GHz and an ability to check one configuration per cycle would still need $\sim 10^{34}$ processor hours to complete this task. Thus, it is not feasible to find the optimal solution with exact algorithms. The scaling of LS should be even worse, because individual qubits of the CNOTs have to be checked for eventual merges with horizontal neighbors adding an additional layer of complexity. Thus, this rough calculation indicates that the NP-hardness of this problem makes it impossible to optimize any meaningful quantum algorithms exactly and efficient heuristic algortihms have to be developed.

We have proven that the decision problem of whether a circuit is perfectly optimizable using the LS-translation devised in ref. 26 is NP-complete and that the optimization problem has to be NP-hard. Furthermore, we have given some rough estimates on how hard exact optimization for LS would be, and have shown that even small circuits cannot be optimized exactly. For practical purposes, however, the optimal configuration is not needed because an optimization protocol can get reasonably close. This urges further research in the development of efficient heuristics to optimize circuits in LS, that are as close to optimality as possible. Furthermore, the inclusion of the measurement step introduces an additional layer of complexity which has not yet been considered in our analysis. This will increase the space of possible configurations and likely decrease even further the efficiency of prospective optimization algorithms.

METHODS

In this section we will detail the proof of this work. In order to do so, we need to define optimality in the context of the LS translation. Furthermore, the optimization problem is presented in an abstracted, non-physical description.

Optimality

A usual definition of optimality is reaching a (computational) goal with minimal physical requirements. In our case these physical requirements correspond to a minimal space-time volume, which is defined by the product of error-correcting cycles and physical qubits. We will further restrict this definition such that the bounding box of this space-time volume (within which all computation happens) needs to be minimal, while the placement still retains the same output state. Another way at looking at this definition is that every patch of the surface code inside the bounding box is initialized to a computational qubit and no ancillary patches are needed. We focus on the generation of the algorithmically specific stabilizer-state and prove that even the optimization of this part is NP-hard. Such a stabilizer-state can be prepared in constant time (as all circuit elements are Clifford), which allows a simplified optimality condition to be the mapping that results in the "least surface area".

(Non-physical) problem description

The LS translation creates a problem where each CNOT has to be fitted into a surface code area that contains all computations. This area should be minimized. However, this can be viewed as an abstract problem, completely detached from the LS picture. We will now introduce the problem that needs to be solved.

The problem consists of minimizing the surface area of a square lattice which consists of individual patches. Some of these patches are assigned an integer q_{ij}, but they do not necessarily need one. Furthermore, multiple patches can have the same integer. A horizontal (vertical) neighbor of a patch is defined as the next non-empty patch (i.e., a patch that contains an integer) to the right or left (up or down). A set of boxes C_i containing patches with integers q_{ij} are given and can be implemented on the lattice by a chain of vertical neighbors, where the order of the $\{q_{ij}\}_j$ can be chosen freely. Furthermore, empty patches can be added freely. The following criteria have to be met to obtain a valid configuration: (1) Patches for all boxes need to be placed (vertical neighbors); (2) Patches with the same integers need to be placed such that they are horizontal neighbors.

Thus, the problem consists of an optimal placement of these numbered patches such that the area of the bounding box of the total arrangement is minimized. The less empty patches are required, the more optimized is the configuration.

For a circuit that has been prepared in the universal, inverted ICM-representation each multi-target CNOT operation will contribute to one box C_i. The numbers q_{ij} of box C_i are given by the qubits that partake in this operation. Figure 1 shows how a sample circuit is mapped to the LS representation.

If a circuit reaches optimality, the number of patches needed in LS can be calculated by:

$$N_{\text{Patch}} = \sum_{i=1}^{\#\text{CNOT}} \left(N_{\text{target}_i} + 1\right). \tag{2}$$

Here, #CNOT denotes the number of multi-target CNOTs with different qubits as their control, in the original circuit specification, and N_{target_i} is the number of target qubits for the *i*th CNOT. However, due to incompatibility

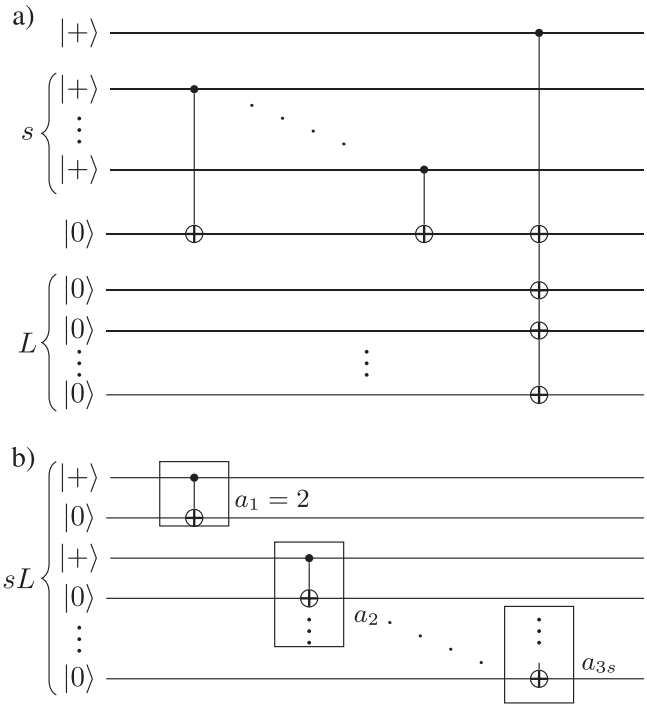

Fig. 3 Circuit for the optimization problem. The optimization for both parts of this circuit corresponds to solving the 3-partitioning problem. **b** Implements the 3-partitioning problem only. Each CNOT corresponds to a number a_i and will be translated into a separate patch of variable height in the LS representation of Fig. 4. The compute area is the area in the LS representation which only consists for the CNOTs from **b**. **a** (of this circuit) Used to force the compute area to be a rectangle of height L and width s and the qubits in the compute area are responsible of encoding the original NP-complete problem

during the merge step, this can (in the worst case) lead to a non-optimal placement with a patch-requirement of

$$N_{\text{Patch}} = N_Q \cdot \#\text{CNOT}, \tag{3}$$

where N_Q represents the total number of qubits in the circuit.

Due to the structure of the high-level circuit that needs to be compiled, it is not always possible to reach the theoretical optimum. A general optimized algorithm needs resources between the two bounds given above.

Proof

In the 3-partitioning problem[34] a set of non-negative integers $\{a_i\}_{1\le i\le 3s}$ is given. With another non-negative integer L, two further requirements are: (i) $\frac{L}{4} \le a_i \le \frac{L}{3}$ $\forall i$, such that $1 \le i \le 3s$, and (ii) $\sum_{i=1}^{3s} a_i = sL$.

The NP-complete decision problem for 3-partitioning answers the following question: Can $\{a_i\}_{1\le i\le 3s}$ be partitioned into s disjoint subsets A_1, …, A_s, such that $\sum_{i\in A_j} a_i = L$ for $j\in\{1, \ldots, s\}$?

Mapping

We can translate the problem of 3-partitioning to the problem of deciding whether a corresponding circuit can reach optimality in LS. The main idea of this mapping is to encode each of the integers of the 3-partitioning problem a_i into a single multi-target CNOT, where the number of qubits that partake in the ith CNOT is given by a_i. Therefore, a box C_i of the non-physical problem description contains a_i integers which will then be translated to blocks of width 1 and height a_i in the LS model. Furthermore, each qubit is only acted on by one CNOT, such that no further constraints apply to the placement of these boxes. The solution of the 3-partitioning problem is given by finding an arrangement of these CNOT blocks in a rectangle of height L and width s. We will call this rectangle the compute area. In Fig. 3 we show a possible circuit, where the qubits in part (b) implement the CNOTs corresponding to a_i.

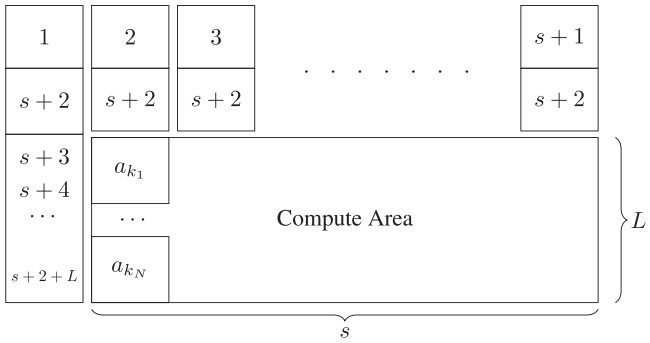

Fig. 4 Translated circuit. The circuit from Fig. 3 is now translated to the LS model of quantum computing. Here, the numbers indicate which qubit of the original circuit each patch represents. If the circuit can reach the theoretical optimum, the last sL qubits can be fit in the compute area space. Each column then consists of L patches of surface code, which are all filled with qubits that partake in CNOT operations. If each of these columns is completely filled, s sets are found, which have elements that sum to L

The qubits from part (a) are needed to ensure that a compute area of L by s is optimal. To ensure a width of at least s, one can devise a chain of CNOTs that have different control qubits but operate on the same target qubit. This is encoded in the qubits starting from the second and ending at qubit $(s+2)$ of part (a). An additional column has to be created here, because one also has to ensure a height of L in the compute area.

This can be performed by adding a single multiple target qubit CNOT with $L+1$ target qubits. One of these qubits is used to link the vertical with the horizontal constraints. This qubit is the $(s+2)$nd qubit in the circuit of Fig. 3. The following L qubits are used to increase the height by L. This results in the optimal placement of LS patches shown in Fig. 4. If that circuit cannot reach theoretical optimality, the compute area cannot contain all 3-partitioning CNOT-patches and thus additional qubits are needed. Holes (i.e., patches of surface code that do not correspond to any qubit in the circuit) are created and the bounding box of the calculation increases.

The number of qubits that are needed for this mapping is $s(L+1)+L+2$. With the algorithmic ancillary patches, the circuit requires $(L+2)(s+1)$ patches in LS. Thus, this mapping only needs resources linearly in the number of integers of the original problem.

By construction, each column in the compute area corresponds to one of the sets A_i, such that the requirement of each set summing to L is equivalent to the requirement that each column in the compute area has a height of exactly L. Furthermore, checking whether each column exactly contains L qubits can be performed in polynomial time, such that the problem is in the complexity class NP.

Data availability

This work does not rely on any further data.

ACKNOWLEDGEMENTS

We like to thank Michael J. Bremner and Austin G. Fowler for fruitful discussions and remarks. D.H. is supported by the RIKEN IPA program. S.J.D. acknowledges support from the JSPS Grant-in-aid for Challenging Exploratory Research and support by the Australian Research Council Centre of Excellence in Engineered Quantum Systems EQUS (Project CE110001013). S.J.D. and F.N. were supported by the IMPACT program of JST, CREST grant No. JPMJCR1676. F.N. was partially supported by the RIKEN iTHES Project, MURI Center for Dynamic Magneto-Optics via the AFOSR Award No. FA9550-14-1-0040, the Japan Society for the Promotion of Science (KAKENHI), and the John Templeton Foundation.

AUTHOR CONTRIBUTIONS

All authors (D.H., S.D., and F.N.) contributed to the conception of the proof. D.H. developed the details of the proof, S.D. checked these, and all authors helped writing the manuscript.

ADDITIONAL INFORMATION

Competing interests: The authors declare that they have no competing financial interests.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

REFERENCES

1. Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Sci. Stat. Comput.* **26**, 1484 (1997).
2. Grover, L. K. A fast quantum mechanical algorithm for database search. In Proceedings, 28th Annual ACM Symposium on the Theory of Computing, 212–219. Preprint at http://arxiv.org/pdf/quant-ph/9605043 (ACM, 1996).
3. Barends, R. et al. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature* **508**, 500–503 (2014).
4. Veldhorst, M. et al. An addressable quantum dot qubit with fault-tolerant control fidelity. *Nat. Nanotechnol.* **9**, 981 (2014).
5. Takeda, K. et al. A fault-tolerant addressable spin qubit in a natural silicon quantum dot. *Sci. Adv.* **2**, e1600694 (2016).
6. Ballance, C. J., Harty, T. P., Linke, N. M., Sepiol, M. A. & Lucas, D. M. High-fidelity quantum logic gates using trapped-ion hyperfine qubits. *Phys. Rev. Lett.* **117**, 060504 (2016).
7. Linke, N. M. et al. Experimental demonstration of quantum fault tolerance. Preprint at http://arxiv.org/pdf/quant-ph/1611.06946 (2016).
8. Fowler, A. G., Mariantoni, M., Martinis, J. M. & Cleland, A. N. Surface codes: towards practical large-scale quantum computation. *Phys. Rev. A* **86**, 032324 (2012).
9. IBM. Quantum experience. http://www.research.ibm.com/quantum/ (2016).
10. Wecker, D. LIQUi|⟩ and SoLi|⟩: simulation and compilation of quantum algorithms. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15 URL http://dl.acm.org/citation.cfm?id=2807591.2897789 (ACM, 2015).
11. Häner, T., Steiger, D. S., Svore, K. & Troyer, M. A software methodology for compiling quantum programs. Preprint at http://arxiv.org/pdf/quant-ph/1604.01401 (2016).
12. Paler, A., Polian, I., Nemoto, K. & Devitt, S. J. Fault-tolerant, high-level quantum circuits: form, compilation and description. *Quantum Sci. Technol.* **2**, 025003 (2017).
13. Paler, A., Devitt, S. J., Nemoto, K. & Polian, I. Mapping of topological quantum circuits to physical hardware. *Sci. Rep.* **4**, 4657 (2014).
14. Devitt, S. J., Fowler, A. G., Tilma, T., Munro, W. J. & Nemoto, K. Classical processing requirements for a topological quantum computing system. *Int. J. Quantum Inf.* **8**, 121–147 (2010).
15. Paler, A., Devitt, S. J., Nemoto, K. & Polian, I. Software-based pauli tracking in fault-tolerant quantum circuits. *Proceedings of the Conference on Design, Automation & Test in Europe* 124 (European Design and Automation Association, 2014).
16. Paetznick, A. G. F. Quantum circuit optimization by topological compaction in the surface code. Preprint at http://arxiv.org/pdf/quant-ph/1304.2807 (2013).
17. Horsman, C., Fowler, A. G., Devitt, S. & v. Meter, R. Surface code quantum computing by lattice surgery. *New J. Phys.* **14**, 123011 (2012).
18. Jones, N. C. et al. Layered architecture for quantum computing. *Phys. Rev. X* **2**, 031007 (2012).
19. Nemoto, K. et al. Photonic architecture for scalable quantum information processing in diamond. *Phys. Rev. X* **4**, 031022 (2014).
20. Hill, C. D. et al. A surface code quantum computer in silicon. *Sci. Adv.* **1**, e1500707 (2015).
21. Li, Y., Humphreys, P. C., Mendoza, G. J. & Benjamin, S. C. Resource costs for fault-tolerant linear optical quantum computing. *Phys. Rev. X* **5**, 041007 (2015).
22. Lekitsch, B. et al. Blueprint for a microwave trapped ion quantum computer. *Sci. Adv.* **3**, e1601540 (2017).
23. Terhal, B. Quantum error correction for quantum memories. *Rev. Mod. Phys.* **87**, 307 (2015).
24. Delfosse, N., Iyer, P. & Poulin, D. Generalized surface codes and packing of logical qubits. Preprint at http://arxiv.org/pdf/quant-ph/1606.07116 (2016).
25. Brown, B. J., Nickerson, N. H. & Browne, D. E. Fault-tolerant error correction with the gauge color code. *Nat. Commun.* **7**, 12302 (2016).
26. Herr, D., Nori, F. & Devitt, S. J. Lattice surgery translation for quantum computation. *New J. Phys.* **19**, 013034 (2017).
27. Dennis, E., Kitaev, A., Landahl, A. & Preskill, J. Topological quantum memory. *J. Math. Phys.* **43**, 4452 (2002).
28. Fowler, A. G., Stephens, A. M. & Groszkowski, P. High threshold universal quantum computation on the surface code. *Phys. Rev. A* **80**, 052312 (2009).
29. Beals, R. et al. Efficient distributed quantum computing. *Proc. R. Soc. Lond. A Math. Phys. Eng. Sci.* **469**, 2153 (2013).
30. Raussendorf, R., Browne, D. E. & Briegel, H. J. Measurement-based quantum computation with cluster states. *Phys. Rev. A* **68**, 022312 (2003).
31. Fowler, A. G. Time-optimal quantum computation. Preprint at http://arxiv.org/pdf/quant-ph/1210.4626 (2012).
32. Moore, C. & Mertens, S. *The Nature of Computation* (Oxford University Press, 2011).
33. Breukelaar, R. et al. Tetris is hard, even to approximate. *Int. J. Comput. Geom. Appl.* **14**, 41–68 (2004).
34. Garey, M. R. & Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman & Co., 1979).
35. Goss, C. F. *Machine Code Optimization—Improving Executable Object Code*. PhD dissertation, Technical Report 246, 13–14 (1986/2013).
36. Steane, A. Multiple-particle interference and quantum error correction. *Proc. R. Soc. Lond. A Math. Phys. Eng. Sci.* **452**, 2551–2577 (1996).
37. Brown, B. J., Laubscher, K., Kesselring, M. S. & Wootton, J. R. Poking holes and cutting corners to achieve clifford gates with the surface code. *Phys. Rev. X* **7**, 021029 (2017).