scientific reports

OPEN

Implementation of quantum compression on IBM quantum computers

Matej Pivoluska[1,2] & Martin Plesch[1,2 ✉]

Advances in development of quantum computing processors brought ample opportunities to test the performance of various quantum algorithms with practical implementations. In this paper we report on implementations of quantum compression algorithm that can efficiently compress unknown quantum information. We restricted ourselves to compression of three pure qubits into two qubits, as the complexity of even such a simple implementation is barely within the reach of today's quantum processors. We implemented the algorithm on IBM quantum processors with two different topological layouts—a fully connected triangle processor and a partially connected line processor. It turns out that the incomplete connectivity of the line processor affects the performance only minimally. On the other hand, it turns out that the transpilation, i.e. compilation of the circuit into gates physically available to the quantum processor, crucially influences the result. We also have seen that the compression followed by immediate decompression is, even for such a simple case, on the edge or even beyond the capabilities of currently available quantum processors.

Quantum computers, as a theoretical concept, has been suggested in the 1980's independently by Paul Benioff[1] and Yuri Manin[2]. Later they have been popularized by Richard Feynman in his seminal work on simulating quantum physics with a quantum mechanical computer[3], which has inspired a new scientific field, collectively known as quantum information and computation[4]. In the last thirty years, the possibility of quantum computing has been studied in depth and revolutionary advances in computation and information science have been made. It has been shown that aside from the ability to simulate quantum physics efficiently, which is invaluable in chemistry[5,6], quantum computers provide a speedup in interesting computational tasks, such as integer factorization[7], search in unstructured databases[8–10] or random walks[11]. Additionally, quantum information scientists have realized that using quantum features of physical particles, such as entanglement, can be used to implement novel communication protocols providing before unseen efficiency[12–14] and above all else, with unconditional security[15–18].

In spite of all these advances, there has always been a large gap between theory and experiments in quantum computation and information. While there was a steady progress in development of practical quantum-mechanical computers[19–21], in practice it has been lagging behind the theoretical advances and only the most well-known quantum algorithms have obtained a proof-of-principle implementations (see the most recent implementations of Shor's factorization algorithm[22] and Grover search[23,24]). Commonly, however, researchers were, until recently, unable to test their algorithms even on small scale quantum computers. This situation has changed in May 2016, when IBM has made their quantum computers accessible to general public via remote access[25]. This invigorated the field of quantum computation and since then multiple experiments have been conducted on IBM systems and reported on in literature[26–40]. What is more, this inspired a new wave of research, designing algorithms that can take advantage of noisy small scale quantum processors, called "Noisy intermediate-scale quantum (NISQ) algorithms"[41–43].

In this paper we join this effort and implement quantum compression algorithm introduced in[44] and further developed in[45–50]. This algorithm is used to compress n identical copies of an arbitrary pure qubit state into roughly $\log(n)$ qubits. Unlike in classical physics, in quantum world a set of identical states represents a valuable resource in comparison to a single copy of such a state. As quantum states cannot be copied[51–53] and a single copy provides only a limited information about the state when measured[54], several copies can be utilized for repeated use in follow-up procedures or for a more precise measurement.

Storing N identical copies of the same state independently is obviously a very inefficient approach. Whereas it is not possible to compress the states in the classical manner (concentrating entropy into a smaller subspace)

[1]Institute of Physics, Slovak Academy of Sciences, Dúbravská cesta 9, 841 04 Bratislava, Slovak Republic. [2]Institute of Computer Science, Masaryk University, Šumavská 416, 602 00 Brno, Czech Republic. ✉email: martin.plesch@savba.sk
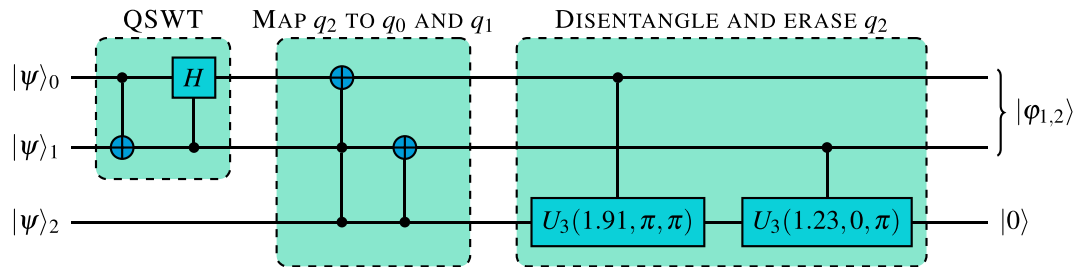
Figure 1. Basic circuit for compression of three qubits into two after a series of optimization comparing to the original results presented in[44]. QSWT stands for Quantum Schur–Weyl Transform, for details see[45]. Before execution on a real quantum processor it needs to be transpiled, i.e. compiled into basis gates.

without measuring the states and disturbing them, laws of quantum mechanics allow to utilize the symmetry of a set of identical states to concentrate all relevant information onto a small, yet not constant subspace. In[44] we have shown that such a procedure can be done in an efficient way (i.e. using a number of elementary quantum gates scaling at most quadratically with the number of compressed states) and this idea was later utilized with a custom designed quantum experiment[45] for the specific case of compressing three identical states of qubits on a subspace of two qubits.

Here we implement the same, simplest non-trivial case, which we call $3 \mapsto 2$ compression. Unfortunately, larger number of compressed qubits is beyond the scope of current quantum processors, because the depth of the required circuit becomes impractical. As we show in the "Results" section, compression followed by immediate decompression is already for this most simple scenario on the edge of capabilities of IBM processors. Scaling up to the next level, i.e. $4 \mapsto 3$ compression, would induce an increase of the number of elementary gates by at least a factor of 5, which would certainly result into a complete noise in the result. Another disadvantage of $4 \mapsto 3$ compression is a large redundancy in the target space (three qubits can accommodate information about as many as seven identical states), leaving space for further errors in the decompression.

Implemented algorithm can be defined using a gate model of quantum computation and is given in Fig. 1. Apart from well known standard gates (CNOT gate, Toffoli gate and controlled H gate) the depicted algorithm uses controlled U_3 gates, where

$$U_3(\phi, \theta, \lambda) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{1}$$

Note that U_3 gate is just a specific parametrization of a universal one qubit unitary. Implementing the $3 \mapsto 2$ compression algorithm is in principle possible simply by inserting the circuit from Fig. 1 into the IBM quantum computing platform called Qiskit[55], and running it using a simulator or a real processor. This, however, rarely leads to an optimal, or even acceptable implementation in terms of fidelity of the compressed state to the ideal compressed one. The main reason for this is that controlled H gate, Toffoli gate and the controlled U_3 gates cannot be natively executed on the IBM quantum processors and need to be decomposed into the hardware supported *basis gates*. Procedure to perform this decomposition is called *transpilation*. The basis gates of IBM quantum computers are: $R_z(\theta)$—a rotation around z axis by an angle θ; $\sqrt{X}$—a square root of Pauli X gate; and CNOT—a controlled not gate. The final form of the circuit to be executed is further guided by the connectivity graph of the quantum processor to be used, which contains an information about which pairs of qubits can perform a hardware CNOT operation. There are only two types of connectivity graphs for a connected configuration of 3 qubits: (1) a triangle, a fully connected graph in which CNOT can be implemented between all pairs of qubits and (2) a line, in which one CNOT between one pair of qubits is not available. These are both relevant for practical quantum computing on IBM quantum platform, as at the time of performing the experiments processors of both kinds were available.

The paper is organized as follows. In the first part, we present the results of simulations and experiments for the compression algorithm only, both on the fully connected quantum processor and on the partially connected processor, where a more sophisticated transpilation is needed. In the second part we present the results of a combined compression and immediate decompression algorithm, both for fully connected and partially connected processors. Here the transpilation takes even a bigger role, as the internal IBM system was not able to fully optimize the circuits, unlike in the previous case, so a custom post-processing lead to better results.

Results

We conducted two different $3 \mapsto 2$ compression experiments. First, we performed a compression only algorithm, in which we run the compression algorithm and perform the full tomography of the resulting 2 qubit states to obtain fidelity to the ideal compressed state. Second, we performed compression algorithm followed by decompression algorithm, in which we first compress three input states into two and then proceed to perform the decompression algorithm. This experiment can be seen as a simulation of the whole compression/decompression routine with an assumption of faultless quantum memory. Here we do not need to make a full tomography of the resulting state, as the fidelity is given simply by its 000 state component.

As the input state $|\psi\rangle$ significantly affects the fidelities obtained, each of the two experiments was performed on 6 different input states—eigenvectors of Pauli X, Y and Z operations denoted $|+\rangle$, $|-\rangle$, $|y_+\rangle$, $|y_-\rangle$, $|0\rangle$, $|1\rangle$. Further, we implemented each of these experiments in two ways—one using default calls of transpilation function provided by the IBM programming environment Qiskit[56] and the second using a more sophisticated transpilation algorithm, which first splits the compression circuits into subparts and transpiles them separately before one final transpilation as a whole (see "Methods" section for detailed description). Transpilation is performed using simulators of quantum processors *ibmq_5_yorktown* (triangle connectivity) and *ibmq_bogota* (line connectivity) as backend. Choosing a backend informs the transpilation function about the connectivity and current calibration data which is used in an attempt to find the best decomposition into the elementary quantum gates. We show that using more sophisticated transpilation, we can significantly decrease the number of single- and two-qubit gates needed, which results in decreased depth and increased fidelity in most of the performed experiments (see Figs. 2, 3 and 5).

The first result of this paper is that the implementation on a line connected processor does not require substantially more resources than the fully connected triangle architecture—triangle implementation of the compression circuit requires 9 CNOTs, while the line implementation requires only 10 CNOTs. Thus the overhead of the incomplete connectivity is limited to about 10% and is compensated by the lower noise of the processor with limited connectivity.

Finally, we run both experiments with different starting states and using both efficient and default transpilation on real quantum hardware. This reveals that the simulators are too optimistic as the decrease in fidelity for all cases is rather significant. This effect becomes more pronounced with larger number of gates in the tested circuit, which is apparent from the fact that compression experiment on real hardware produces rather good outcomes even for real hardware (see Fig. 3), while compression/decompression experiment results in very low fidelities of correct decompression (see Fig. 5).

Compression experiment.

In this subsection we present detailed results for the compression only experiment. First we conducted experiments with triangle connectivity, using *ibmq_5_yorktown* quantum processor. Default transpilation with this backend produces a circuit with 9 CNOTs, 35 R_Z operations and 28 $\sqrt{X}$ operations, with depth 46. The efficient transpilation results in a transpiled circuit with 9 CNOTs, 23 R_Z operations and 14 $\sqrt{X}$ operations with depth 37 (see Fig. 2 for a schematic representation of these circuits). This difference results in roughly 1–2% increase of fidelity when simulating the efficiently transpiled compression algorithm, except for starting state $|1\rangle$, where the default solution slightly outperforms the efficient one.

In experiments using line architecture we were using *ibmq_bogota* processor. With this backend the transpilation function produced variable results. Number of CNOTs varied between 10 and 25, while the circuit depth varied between 49 and 105. The reason for this variance is that transpilation procedure uses a stochastic method to find decomposition in case of a missing CNOT connectivity. Roughly 25% of runs find the most efficient solution with 10 CNOTs, 30 R_Z operations and 26 $\sqrt{X}$ operations with depth 49. On the other hand, the efficient transpilation (see "Methods" section for details) resulted in transpiled circuit with 10 CNOTs, 24 R_Z, and 18 $\sqrt{X}$ operations and depth 41 (see Fig. 2 for schematic representation of these circuits). Surprisingly, this difference results only in negligible increase of fidelity of the simulated compression algorithm when using the efficient solution. Even more interestingly, in case of input state $|1\rangle$ the default solution again outperforms the efficient one.

In order to confirm the results obtained in simulation we also ran the same circuits on real hardware. For *ibmq_5_yorktown* the obtained fidelities are significantly lower than the simulation suggests, with an average drop of 20–30% and in case of default transpiled circuit, ranging even to 40% for the $|-\rangle$ input state (see Fig. 3). The length of the circuit is the likely reason for this decrease, as the coherence times for the *ibmq_5_yorktown* processor were rather short compared to the newer generation of processors. This intuition is confirmed by inspecting the experiment results for *ibmq_bogota*. In this set of experiments with a newer generation of the processor with longer coherence times the fidelity decrease compared to the simulation was only 7–20% (see Fig. 3). Interestingly, also in this case we can see that the best default transpiled circuit outperformed the efficient one for some input states.
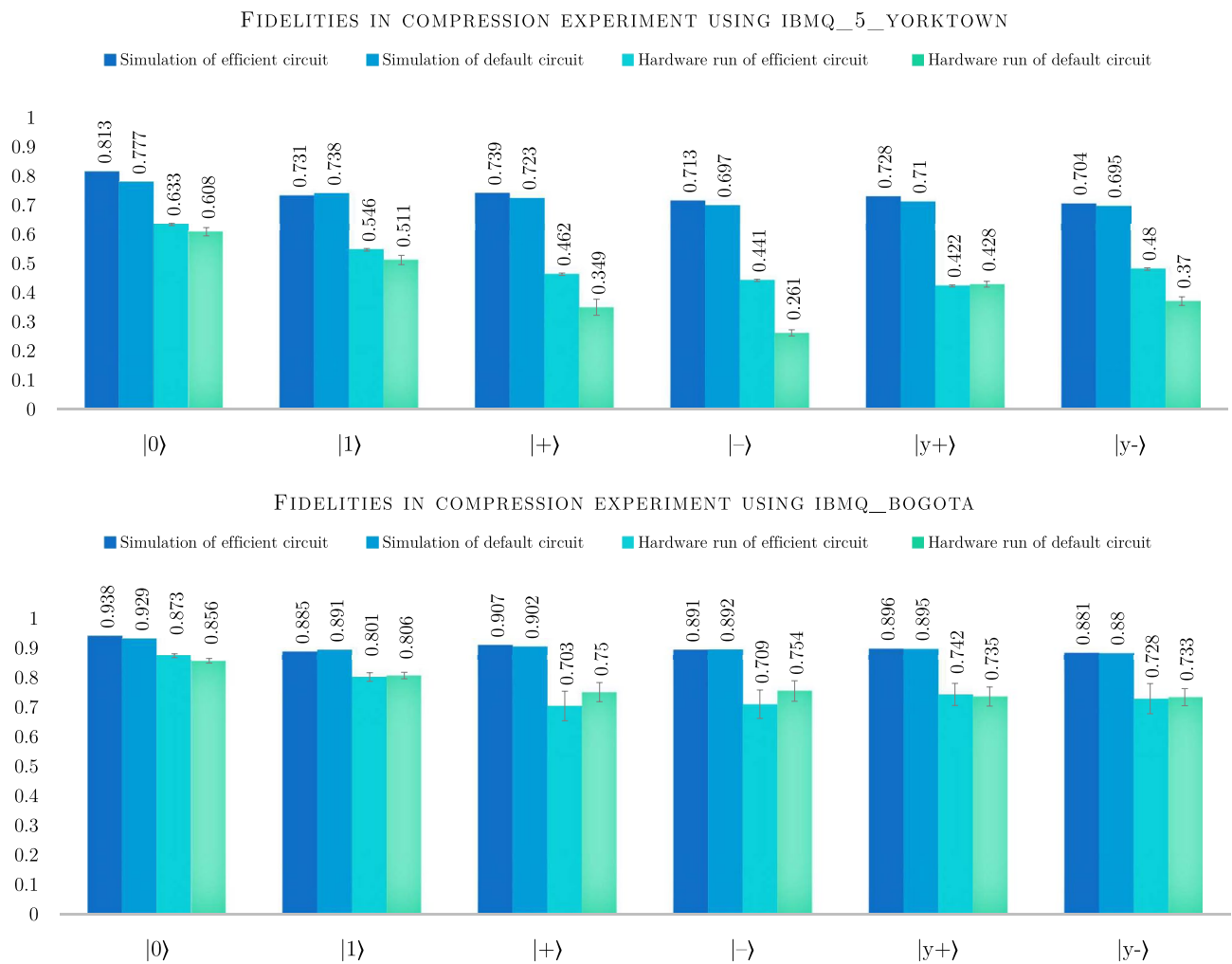
Compression and decompression experiment.

In the second experiment we let the default transpiler produce the circuits for both compression and decompression (see Fig. 4). In case of *ibmq_5_yorktown* the default transpiler could not find a decompression circuit with 9 CNOTs and the complete compression/decompression circuit therefore had 21 CNOTs, 62 R_Z operations and 46 $\sqrt{X}$ operations with total depth of 90 including the operation, which resets the third qubit. This compares to efficient circuit for triangle connectivity, which uses complex conjugate of the efficient compression algorithm for decompression with 18 CNOTs, 46 R_Z and 28 $\sqrt{X}$ with total depth of 77 including the reset operation. This more significant difference results in larger advantage of simulating the efficient solution compared to the compression only experiment, with efficient circuit reaching roughly 5–6% better fidelities. Again, the outlier is the prepared state $|1\rangle$, where the advantage of the efficient solution is only roughly 1% (see Fig. 5).

Using *ibmq_bogota* with line connectivity, we again observe that decompression algorithm poses a problem for the default transpiler. The solutions vary considerably with circuits using between 26 and 49 CNOTs with depths between 107 and 201. In this experiment only roughly 3% of transpiler runs resulted in the best solution with 26 CNOTs, 69 R_Z and 60 $\sqrt{X}$ operations with total depth of 99 including the three reset operations. For comparison, the efficient solution always results in 20 CNOTs, 48 R_Z and 36 $\sqrt{X}$ operations with total depth of 87. This decrease in complexity results in advantage for the simulated efficient solution with average increased fidelity between 3–7%, depending on the input state.
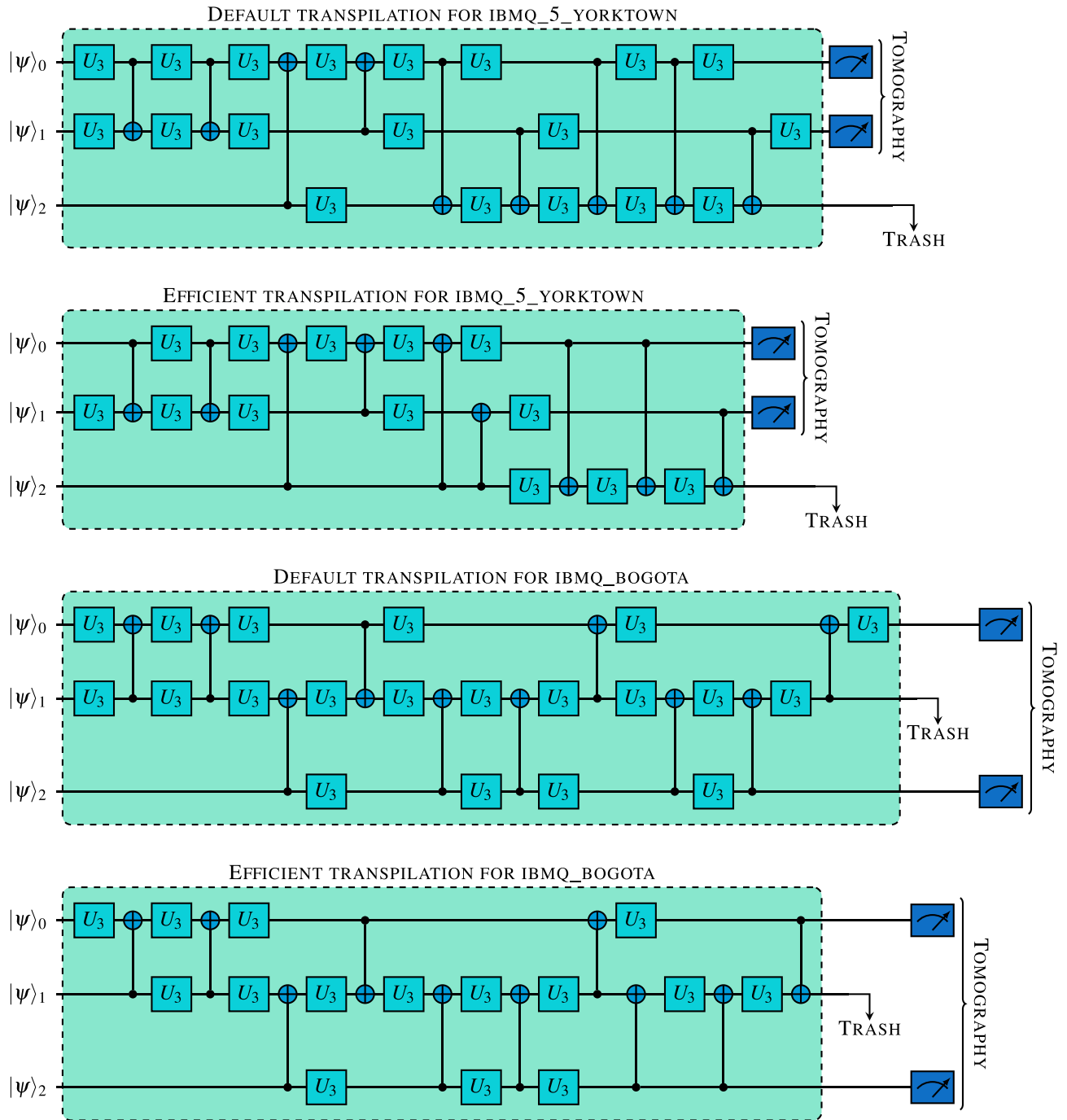
Figure 2. Schematic representation of transpiled compression circuits. Here, single qubit rotations labeled U_3 (see Eq. (1) for definition) are implemented by 1–5 basis gates, i.e. R_z rotations and $\sqrt{X}$. In total, the default transpiled circuit using a triangle architecture contains 9 CNOTs, 35 R_Z operations and 28 $\sqrt{X}$ operations, with depth 46. Efficiently transpiled circuit using triangle architecture consists of 9 CNOTs, 23 R_Z and 14 $\sqrt{X}$ operations and has depth 37. The best circuit produced by default transpiler on a line architecture consists of 10 CNOTs, 30 R_Z operations and 26 $\sqrt{X}$ operations with depth 49. Efficient transpilation always finds a solution with 10 CNOT, 24 R_Z and 18 $\sqrt{X}$ operations and also has depth 41. These circuits are used in the compression experiment, where full tomography is performed on the qubits containing the compressed state, while the third qubit is discarded. Note that in the circuit for the triangle connectivity qubit 2 is discarded and for the line connectivity qubit 1 is discarded.

Moving on to experiments with hardware backend, we see that *ibmq_5_yorktown* suffers from a substantial performance drop. In particular, the results in the default transpilation case are consistent with random outcomes,

FIDELITIES IN COMPRESSION EXPERIMENT USING IBMQ_5_YORKTOWN

■ Simulation of efficient circuit   ■ Simulation of default circuit   ■ Hardware run of efficient circuit   ■ Hardware run of default circuit

FIDELITIES IN COMPRESSION EXPERIMENT USING IBMQ_BOGOTA

■ Simulation of efficient circuit   ■ Simulation of default circuit   ■ Hardware run of efficient circuit   ■ Hardware run of default circuit

Figure 3. Column chart representing fidelities of the compressed state in the compression experiment using both *ibmq_5_yorktown* and *ibmq_bogota* quantum processors. Simulations were performed with 1 million shots each, using error parameters provided by IBM for each of the processors, while the hardware run fidelities are averages calculated over 10–20 runs with 8192 shots each. We use standard deviation for error bars.

suggesting the experimental state fully decohered before the calculation could finish. Similarly, in the case of experiments using line connected *ibmq_bogota* backend we can observe a significant drop in fidelities. The difference between default and efficient circuits is more substantial than in the compression only experiment, which is caused by a more substantial difference between the two circuits. Here, clearly the efficient solution outperformed the default one, however, both suffered 20–45% drop in obtained fidelities comparing to simulations. This suggests that the length of the circuits currently exceeds the possibilities of even the newest generation of IBM quantum computers.

Discussion

Compression of unknown quantum information in its simplest scenario, compressing of three identical states into two, is a nice toy example for testing of abilities of emerging quantum computers. In this work we present the implementation of the quantum compression algorithm on two different IBM processors. In both cases we simulate the procedure using classical computers and run real quantum computations.

The first result is the comparison of two different types of quantum processor connectivity—full triangle and a line connectivity. Our implementations reveal that triangle connectivity does not result in a significant advantage for the $3 \mapsto 2$ compression, as only one additional CNOT is needed to compensate the missing connection. In other words, the higher quality of the newer generation of processors fully compensated the lower level of connectivity. As a result, we have seen that the most recent generation of IBM quantum processors can attain fidelity of compression of $70\% - 87\%$, depending on the state to be compressed. On one hand this is a rather impressive technical feat, because the implemented circuits are non-trivial, on the other, it is still likely below the levels needed for practical use of the compression algorithm.

There are also several results that have a general validity for basically any computation performed on quantum computers. First, we have shown that the current qiskit transpiler needs to be used wisely, with some

COMPRESSION AND DECOMPRESSION EXPERIMENT
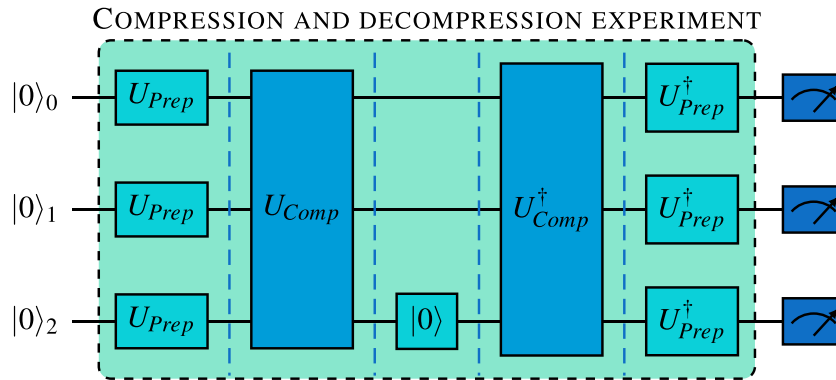
Figure 4. Schematics of the compression and decompression experiment. Three copies of input state $|0\rangle$ are first prepared into one of the desired starting states $|\psi\rangle$ from the set $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle, |y_+\rangle, |y_-\rangle\}$ using the preparation unitary U_{Prep}. Subsequently, three copies of $|\psi\rangle$ are compressed using the compression algorithm U_{Comp}. Then, the last qubit is restored to state $|0\rangle$. This marks the end of the compression part, after which the first two qubits plus the $|0\rangle$ state are first decompressed using the complex conjugation of U_{Comp}, and rotated using the complex conjugation of the preparation unitary. The expected result is $|000\rangle$ state and the probability of obtaining this result is the fidelity of the experimental compression and decompression experiment. Dashed vertical lines represent barriers that divide the circuits into parts, which the transpiler processes separately and independently.

sophistication. This is demonstrated by the fact that the default setting of the transpiler finds the most efficient solution in case of line connectivity with only a very small probability. Even worse, the default transpiler does not find the best solution for decompression circuit at all. This suggests that in order to find optimal transpiled circuits for any algorithm described by a unitary U, it generally might be a good strategy to transpile both U and $U^\dagger$ and choose the more efficient one. It also turns out that it is advantageous to transpile more complicated circuits first in smaller blocks to get rid of unsupported gates and connections and then optimize the whole circuit in order to minimize the total number of gates.

As a very important point it turns out that the simulators implemented for IBM quantum computers are far too optimistic. Most probably only a part of the decoherence sources is sufficiently modeled, which leads to a far lower simulated noise if compared to reality. This in particular limits its usability for testing the performance of the available processors on complicated tasks.

Methods

In this section we briefly describe the tools and the algorithm used to obtain efficient circuits. The main tool that crucially influenced the quality of the output results was `transpile` function from qiskit. It translates all gates that are not directly supported by the computer into gates from its library and also bridges CNOT gates acting between not-connected qubits into a series of gates on connected qubits. It should, to some level, also optimize the circuit for the least number of gates and get use of the higher quality qubits and connections.

In its basic form, `transpile` function takes as inputs `circuit`, `backend` and `optimization_level`. Input `circuit` contains the information about the circuit to be transpiled and `backend` contains information about the quantum processor to be used—connectivity of given quantum processor (i.e. line or triangle in our case), as well as error parameters of individual qubits. The last input defines what kind of optimization is performed on the circuit. There are four basic levels qiskit offers, described in the Qiskit tutorial as:

```
optimization_level=0:
```
just maps the circuit to the backend, with no explicit optimization (except whatever optimizations the mapper does).
```
optimization_level=1:
```
maps the circuit, but also does light-weight optimizations by collapsing adjacent gates.
```
optimization_level=2:
```
medium-weight optimization, including a noise-adaptive layout and a gate-cancellation procedure based on gate commutation relationships.
```
optimization_level=3:
```
heavy-weight optimization, which in addition to previous steps, does resynthesis of two-qubit blocks of gates in the circuit.

For all settings, the approach of the transpiler is stochastic. Thus it does not necessarily ends up with the same solution every time it is called. Moreover, for more complicated circuits and higher optimization levels it might not find a solution at all, most probably due to reaching a threshold in the number of iterations or computer load.

The default circuits for both experiments with *ibmq_5_yorktown* were obtained by transpiling the default circuit presented in Fig. 1 three times in a row, each time with decreasing the value of the optimization level, starting from value 3. The default circuits for *ibmq_bogota* were obtained in the same way. The transpilation was run 100 times for both experiments and the most efficient circuits were used.
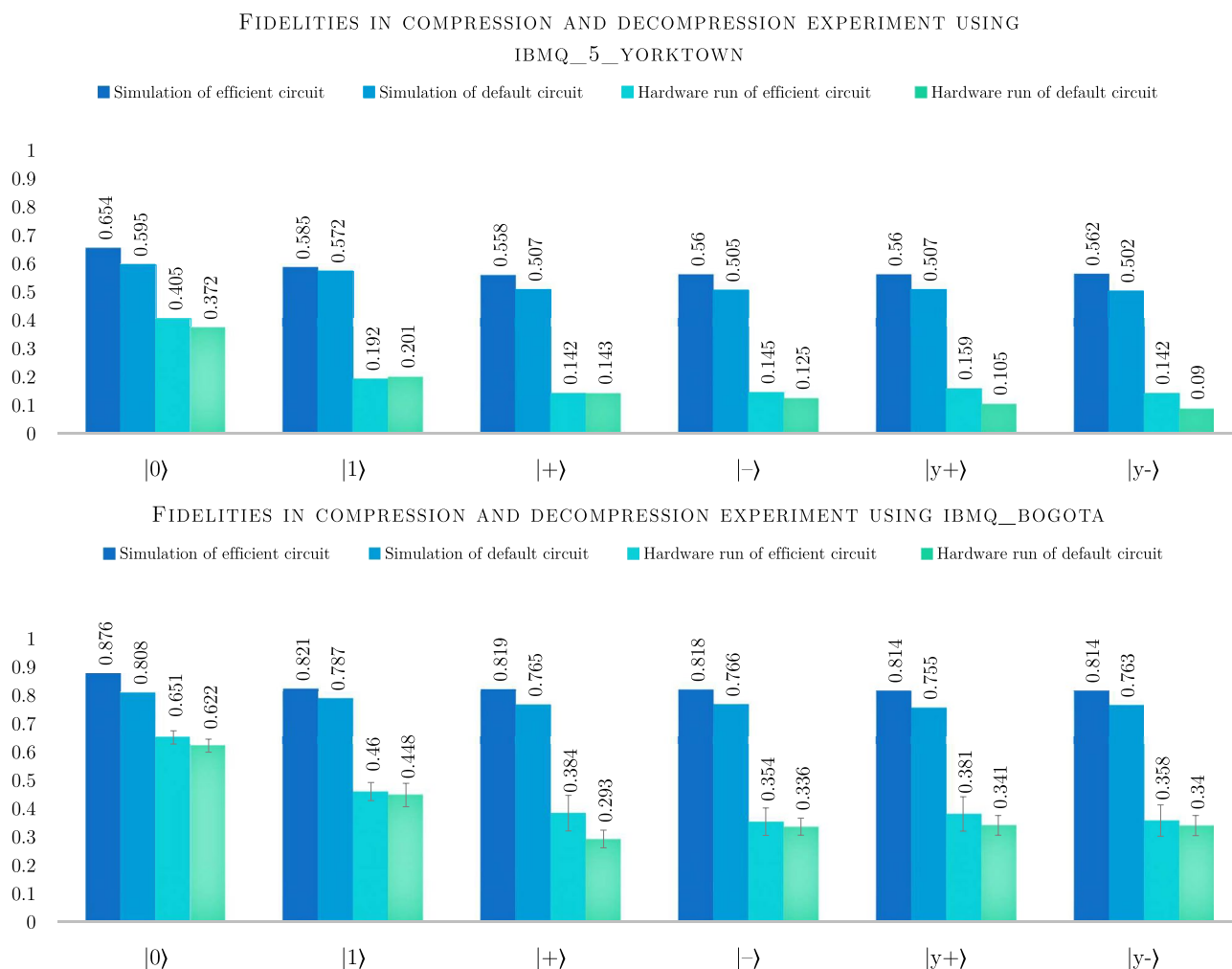
Figure 5. Column chart representing results of the compression and decompression experiment. We plot the fidelity of the decompressed states to $|\psi^{\otimes 3}\rangle$, where $|\psi\rangle$ is the input state. Simulations were performed with 1 million shots each, using error parameters provided by IBM for each of the hardware backends (*ibmq_5_yorktown* and *ibmq_bogota*). Due to the expected low fidelity on *ibmq_5_yorktown*, the experiments were run only once with 8192 shots. On the other hand, in case of hardware run on *ibmq_bogota* we present averages calculated over 10 runs with 8192 shots each. We use standard deviation for error bars.

The efficient circuits for compression experiment were obtained by first splitting circuit presented in Fig. 1 into three parts, in order to transpile both controlled U_3 operations and the "Disentangle and erase" part separately. First this was done using *ibmq_5_yorktown*, with `optimization_level = 3` for $U_3(1.23, 0, \pi)$ and `optimization_level = 2` for $U_3(1.91, \pi, \pi)$, to produce a circuit with 1 CNOT and 2 CNOTs respectively. Then the rest of the circuit from Fig. 1 was transpiled with `optimization_level = 1`. Finally all three parts were joined together and again transpiled with `optimization_level = 3`, followed by `optimization_level = 1` to produce the final result.

In order to produce an efficient circuit for *ibmq_bogota*, the efficient circuit for *ibmq_yorktown* was transpiled three times with a new backend, with *optimization_level* starting at value 3, followed by value 2 and finally value 1. Again, the result of this transpilation was stochastic, but in roughly 10% of the trials the final circuit with 10 CNOTS was produced.

Efficient circuits for the compression and decompression experiment were obtained by using previously obtained efficient circuits for compression experiment and their complex conjugation for decompression part.

Data availability
Data and programs used to derive the results presented in this paper are available from the corresponding author upon reasonable request.

References

1. Benioff, P. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by turing machines. *J. Stat. Phys.* **22**, 563–591. https://doi.org/10.1007/BF01011339 (1980).
2. Manin, Y. *Computable and Uncomputable* (Sovetskoye Radio, 1980) **(in Russian)**.
3. Feynman, R. P. Simulating physics with computers. *Int. J. Theor. Phys.* **21**, 467–488. https://doi.org/10.1007/BF02650179 (1982).
4. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information* (Cambridge University Press, 2000).
5. Cheng, H.-P., Deumens, E., Freericks, J. K., Li, C. & Sanders, B. A. Application of quantum computing to biochemical systems: A look to the future. *Front. Chem.* **8**, 1066. https://doi.org/10.3389/fchem.2020.587143 (2020).
6. Cao, Y. *et al.* Quantum chemistry in the age of quantum computing. *Chem. Rev.* **119**, 10856–10915. https://doi.org/10.1021/acs.chemrev.8b00803 (2019).
7. Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **26**, 1484–1509. https://doi.org/10.1137/S0097539795293172 (1997).
8. Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pp. 212–219 (Association for Computing Machinery, 1996). https://doi.org/10.1145/237814.237866
9. Long, G. L. Grover algorithm with zero theoretical failure rate. *Phys. Rev. A* **64**, 022307. https://doi.org/10.1103/PhysRevA.64.022307 (2001).
10. Toyama, F. M., van Dijk, W. & Nogami, Y. Quantum search with certainty based on modified Grover algorithms: optimum choice of parameters. *Quantum Inf. Process.* **12**, 1897–1914. https://doi.org/10.1007/s11128-012-0498-0 (2013).
11. Reitzner, D., Nagaj, D. & BuŽek, V. Quantum walks. *Acta Phys. Slov. Rev. Tutor.* **61**. https://doi.org/10.2478/v10155-011-0006-6 (2011).
12. Bennett, C. H. *et al.* Teleporting an unknown quantum state via dual classical and Einstein–Podolsky–Rosen channels. *Phys. Rev. Lett.* **70**, 1895–1899. https://doi.org/10.1103/PhysRevLett.70.1895 (1993).
13. Bennett, C. H. & Wiesner, S. J. Communication via one- and two-particle operators on Einstein–Podolsky–Rosen states. *Phys. Rev. Lett.* **69**, 2881–2884. https://doi.org/10.1103/PhysRevLett.69.2881 (1992).
14. Bäuml, S., Winter, A. & Yang, D. Every entangled state provides an advantage in classical communication. *J. Math. Phys.* **60**, 072201. https://doi.org/10.1063/1.5091856 (2019).
15. Bennett, C. H. & Brassard, G. Quantum cryptography: Public key distribution and coin tossing. *Theor. Comput. Sci.* **560**, 7–11. https://doi.org/10.1016/j.tcs.2014.05.025 (2014). Theoretical Aspects of Quantum Cryptography—celebrating 30 years of BB84).
16. Pirandola, S. *et al.* Advances in quantum cryptography. *Adv. Opt. Photon.* **12**, 1012. https://doi.org/10.1364/aop.361502 (2020).
17. Long, G. L. & Liu, X. S. Theoretically efficient high-capacity quantum-key-distribution scheme. *Phys. Rev. A* **65**, 032302. https://doi.org/10.1103/PhysRevA.65.032302 (2002).
18. Pan, D., Li, K., Ruan, D., Ng, S. X. & Hanzo, L. Single-photon-memory two-step quantum secure direct communication relying on Einstein–Podolsky–Rosen pairs. *IEEE Access* **8**, 121146–121161. https://doi.org/10.1109/ACCESS.2020.3006136 (2020).
19. Blatt, R. & Roos, C. F. Quantum simulations with trapped ions. *Nat. Phys.* **8**, 277–284. https://doi.org/10.1038/nphys2252 (2012).
20. Bruzewicz, C. D., Chiaverini, J., McConnell, R. & Sage, J. M. Trapped-ion quantum computing: Progress and challenges. *Appl. Phys. Rev.* **6**, 021314. https://doi.org/10.1063/1.5088164 (2019).
21. Huang, H.-L., Wu, D., Fan, D. & Zhu, X. Superconducting quantum computing: A review. *Sci. China Inf. Sci.* **63**, 180501. https://doi.org/10.1007/s11432-020-2881-9 (2020).
22. Monz, T. *et al.* Realization of a scalable Shor algorithm. *Science* **351**, 1068–1070. https://doi.org/10.1126/science.aad9480 (2016).
23. Figgatt, C. *et al.* Complete 3-qubit Grover search on a programmable quantum computer. *Nat. Commun.* **8**. https://doi.org/10.1038/s41467-017-01904-7 (2017).
24. Long, G. *et al.* Experimental NMR realization of a generalized quantum search algorithm. *Phys. Lett. A* **286**, 121–126. https://doi.org/10.1016/S0375-9601(01)00416-9 (2001).
25. *IBM Quantum* (2021). https://quantum-computing.ibm.com/
26. Rundle, R. P., Mills, P. W., Tilma, T., Samson, J. H. & Everitt, M. J. Simple procedure for phase-space measurement and entanglement validation. *Phys. Rev. A* **96**, 022117. https://doi.org/10.1103/PhysRevA.96.022117 (2017).
27. Huffman, E. & Mizel, A. Violation of noninvasive macrorealism by a superconducting qubit: Implementation of a Leggett–Garg test that addresses the clumsiness loophole. *Phys. Rev. A* **95**, 032131. https://doi.org/10.1103/PhysRevA.95.032131 (2017).
28. Deffner, S. Demonstration of entanglement assisted invariance on IBM's quantum experience. *Heliyon* **3**, e00444. https://doi.org/10.1016/j.heliyon.2017.e00444 (2017).
29. Huang, H.-L. *et al.* Homomorphic encryption experiments on IBM's cloud quantum computing platform. *Front. Phys.* **12**, 120305. https://doi.org/10.1007/s11467-016-0643-9 (2016).
30. Wootton, J. R. Demonstrating non-abelian braiding of surface code defects in a five qubit experiment. *Quantum Sci. Technol.* **2**, 015006. https://doi.org/10.1088/2058-9565/aa5c73 (2017).
31. Fedortchenko, S. A quantum teleportation experiment for undergraduate students. arXiv:1607.02398 (2016).
32. Li, R., Alvarez-Rodriguez, U., Lamata, L. & Solano, E. Approximate quantum adders with genetic algorithms: An IBM quantum experience. *Quantum Meas. Quantum Metrol.* **4**, 1–7. https://doi.org/10.1515/qmetro-2017-0001 (2017).
33. Hebenstreit, M., Alsina, D., Latorre, J. I. & Kraus, B. Compressed quantum computation using a remote five-qubit quantum computer. *Phys. Rev. A* **95**, 052339. https://doi.org/10.1103/PhysRevA.95.052339 (2017).
34. Alsina, D. & Latorre, J. I. Experimental test of Mermin inequalities on a five-qubit quantum computer. *Phys. Rev. A* **94**, 012314. https://doi.org/10.1103/PhysRevA.94.012314 (2016).
35. Devitt, S. J. Performing quantum computing experiments in the cloud. *Phys. Rev. A* **94**, 032329. https://doi.org/10.1103/PhysRevA.94.032329 (2016).
36. Mandviwalla, A., Ohshiro, K. & Ji, B. Implementing Grover's algorithm on the IBM quantum computers. In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 2531–2537. https://doi.org/10.1109/BigData.2018.8622457 (2018).
37. Acasiete, F., Agostini, F. P., Moqadam, J. K. & Portugal, R. Implementation of quantum walks on IBM quantum computers. *Quantum Inf. Process.* **19**, 426. https://doi.org/10.1007/s11128-020-02938-5 (2020).
38. Zhang, K., Rao, P., Yu, K., Lim, H. & Korepin, V. Implementation of efficient quantum search algorithms on NISQ computers. *Quantum Inf. Process.* **20**, 233. https://doi.org/10.1007/s11128-021-03165-2 (2021).
39. Bharti, K. *et al.* Noisy intermediate-scale quantum (NISQ) algorithms. arXiv:2101.08448 (2021).
40. Oliveira, A. N., de Oliveira, E. V. B., Santos, A. C. & Villas-Bôas, C. J. Quantum algorithms in IBMQ experience: Deutsch–Jozsa algorithm. arXiv:2109.07910 (2021).
41. Peruzzo, A. *et al.* A variational eigenvalue solver on a photonic quantum processor. *Nat. Commun.* **5**, 4213. https://doi.org/10.1038/ncomms5213 (2014).
42. Wei, S., Li, H. & Long, G. A full quantum eigensolver for quantum chemistry simulations. *Research* **2020**, 1486935. https://doi.org/10.34133/2020/1486935 (2020).
43. Bharti, K. *et al.* Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.* **94**, 015004. https://doi.org/10.1103/RevModPhys.94.015004 (2022).

44. Plesch, M. & Bužek, V. Efficient compression of quantum information. *Phys. Rev. A* **81**, 032317. https://doi.org/10.1103/PhysRevA.81.032317 (2010).
45. Rozema, L. A., Mahler, D. H., Hayat, A., Turner, P. S. & Steinberg, A. M. Quantum data compression of a qubit ensemble. *Phys. Rev. Lett.* **113**, 160504. https://doi.org/10.1103/PhysRevLett.113.160504 (2014).
46. Yang, Y., Chiribella, G. & Ebler, D. Efficient quantum compression for ensembles of identically prepared mixed states. *Phys. Rev. Lett.* **116**, 080501. https://doi.org/10.1103/PhysRevLett.116.080501 (2016).
47. Yang, Y., Chiribella, G. & Hayashi, M. Optimal compression for identically prepared qubit states. *Phys. Rev. Lett.* **117**, 090502. https://doi.org/10.1103/PhysRevLett.117.090502 (2016).
48. Huang, C.-J. *et al.* Realization of a quantum autoencoder for lossless compression of quantum data. *Phys. Rev. A* **102**, 032412. https://doi.org/10.1103/PhysRevA.102.032412 (2020).
49. Bai, G., Yang, Y. & Chiribella, G. Quantum compression of tensor network states. *N. J. Phys.* **22**, 043015. https://doi.org/10.1088/1367-2630/ab7a34 (2020).
50. Fan, C.-R., Lu, B., Feng, X.-T., Gao, W.-C. & Wang, C. Efficient multi-qubit quantum data compression. *Quantum Eng.* **3**, e67. https://doi.org/10.1002/que2.67 (2021).
51. Park, J. L. The concept of transition in quantum mechanics. *Found. Phys.* **1**, 23–33. https://doi.org/10.1007/BF00708652 (1970).
52. Wootters, W. K. & Zurek, W. H. A single quantum cannot be cloned. *Nature* **299**, 802–803. https://doi.org/10.1038/299802a0 (1982).
53. Dieks, D. Communication by EPR devices. *Phys. Lett. A* **92**, 271–272. https://doi.org/10.1016/0375-9601(82)90084-6 (1982).
54. Holevo, A. S. *Statistical Structure of Quantum Theory* (Springer, 2001).
55. *Qiskit: Open Source Quantum Development* (2021). https://qiskit.org/
56. *Qiskit: Transpiler documentation* (2021). https://qiskit.org/documentation/apidoc/transpiler.html

Acknowledgements

Author contributions

Both authors designed the experiment based on the algorithm developed by M.Pl., M.Pi. implemented the programs and both authors analyzed the data. M.Pi. prepared the first draft of the manuscript and both authors edited and finalized it.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to M.P.

Reprints and permissions information is available at www.nature.com/reprints.

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