Suppressing Quantum Circuit Errors Due to System Variability

Paul D. Nation* and Matthew Treinish
IBM Quantum, Yorktown Heights, New York 10598, USA

(Received 13 October 2022; accepted 22 February 2023; published 15 March 2023)

We present a quantum circuit optimization technique that takes into account the variability in error rates that is inherent across present-day noisy quantum computing platforms. This method can be run after qubit routing or postcompilation and consists of computing isomorphic subgraphs to input circuits and scoring each using heuristic cost functions derived from system calibration data. Using an independent standard algorithmic test suite, we show that it is possible to recover on average nearly 40% of missing fidelity using better qubit selection via efficient to compute cost functions. We demonstrate additional performance gains by considering qubit placement over multiple quantum processors. The overhead from these tools is minimal with respect to other compilation steps, such as qubit routing, as the number of qubits increases. As such, our method can be used to find qubit mappings for problems at the scale of quantum advantage and beyond.

DOI: 10.1103/PRXQuantum.4.010327

I. INTRODUCTION

Given the limitations of present-day noisy quantum hardware, the implementation of quantum circuit error-suppression techniques [1–4] is vital for obtaining high-fidelity results over nontrivial circuit space-time volumes [5–8]. Indeed, when compiling a quantum circuit to a given quantum system, the choice of physical qubits, basis gates, swap mapping [9], gate optimization [10], and dynamical decoupling [11–13] routines all play a role in helping to reduce errors when executing the circuit.

Consideration of the noise characteristics of the target quantum system, e.g., gate and measurement errors and coherence times, is nominally taken into account *at the beginning* of the compilation process via the choice of virtual to physical qubit mapping. As shown in Fig. 1, present-day quantum systems have substantial variation in their important metrics, making virtual to physical qubit mapping a crucial step in the compilation pipeline. The choice of qubits is even more critical when applying error-mitigation techniques, where the mitigation process is exponentially sensitive to the fidelity of qubit operations [14–16].

For small-width circuits targeting hardware of $\mathcal{O}(10)$ qubits, it is usually possible to hand select a low-noise subset of qubits with reasonable accuracy. However, as hardware quality improves, and qubit counts begin to approach $\gtrsim \mathcal{O}(10^2)$, finding optimal layouts manually becomes exceedingly difficult. While automated noise-aware qubit placement routines do exist [3,17], they often suffer from the fact that the best initial qubit layout in terms of noise characteristics is often not ideal for later qubit routing via swap mapping to the target topology; the error from additional SWAP gates dominates the savings from noise-aware qubit selection. Moreover, the total number and type of gates in the final compiled circuit are not known ahead of time, further complicating qubit placement early in the compilation pipeline. Additionally, it is possible to incorporate quantum processor noise into the qubit routing [2] and approximate local gate compilation [18]. However, these methods may lead to additional SWAP gates and poorly approximated global unitary operators, respectively.

Here, we take a different approach, remapping quantum circuits after qubit routing, or postcompilation, to matching low-noise subgraphs using device-calibration data. Using output circuits generated with SWAP-efficient layout and routing routines that are not sensitive to device parameters, e.g., the SABRE method [9], we compute the possible equivalent qubit mappings ranked by a heuristic cost function. This can be done at the single-device level or across multiple machines of a similar topology. Circuits are then remapped to the lowest error subgraph before execution. A related technique has been considered in Ref. [19] using Google Quantum AI processors [20] but it has been found that the system calibration data fail to capture real-world performance details. In marked contrast, we show, via common independently defined algorithmic test circuits, that meaningful improvements in quantum circuit fidelity

*E-mail: paul.nation@ibm.com

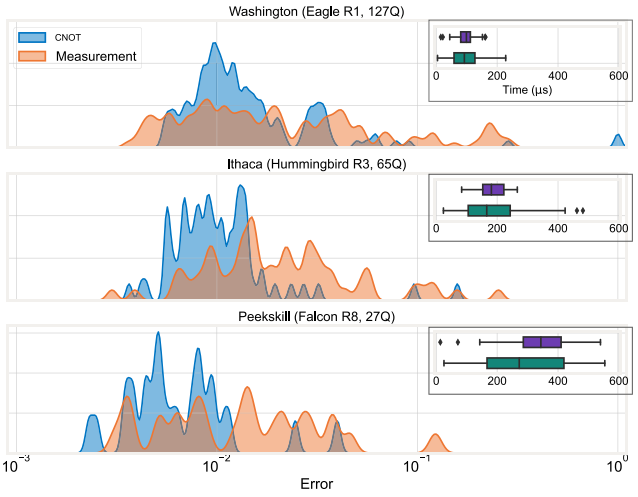

FIG. 1. The density of controlled-NOT (CNOT) gate and measurement errors for three IBM Quantum systems representing different processor families, highlighting the variability of key performance metrics across the devices. The insets show the distribution of T_1 (top) and T_2 (bottom) times.

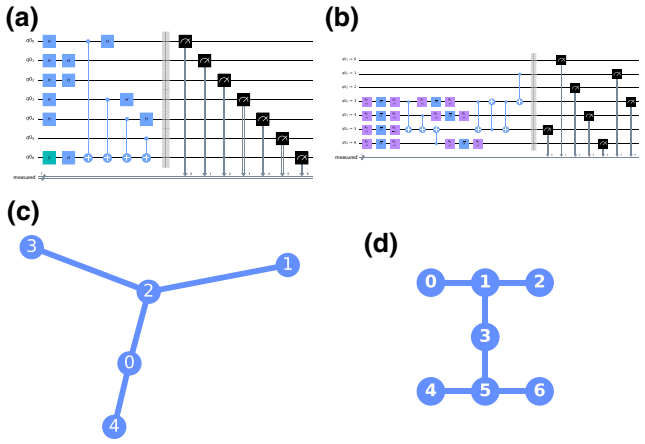

FIG. 2. The stages of the MAPOMATIC algorithm. (a) The input circuit before compilation. (b) The circuit after compilation, targeting the IBM Quantum Nairobi system. (c) The interaction graph for the compiled circuit. (d) The coupling map for the IBM Quantum Nairobi back end.

can be obtained using even the most trivial of, and efficient to compute, cost functions on IBM Quantum hardware. We extend this technique across multiple quantum processors and highlight that additional gains can be found by relaxing the requirement that all circuits be executed on the same quantum system. Although subgraph isomorphism is NP complete [21], we highlight that the cost of computing these graphs is much smaller than the cost of circuit compilation as a whole and thus our method adds little in terms of relative cost. Related machine-learning algorithms have also been put forth [22] but come with substantial overhead in the requirement of gate-set tomography and circuit-learning run times.

This paper is organized as follows. In Sec. II, we detail how quantum processor subgraphs are determined from the entangling-gate topology of a given quantum circuit and discuss how each subgraph is heuristically scored. Section III discusses implementation considerations when integrating our method into complete compilation pipelines. In Sec. IV, we look at the performance gains that are achievable with our technique using standard suites of algorithm test circuits, while Sec. V details additional performance gains that come when looking for optimal layouts across multiple quantum systems. Finally, Sec. VI summarizes our results and looks at possible future improvements. Appendix A looks at the effect of including T_1 and T_2 information in the cost function, while Appendix B details the importance of qubit routing when using our method.

II. METHOD

Our algorithm, called MAPOMATIC [23], is a postrouting routine that assumes that one or more circuits have been compiled to match the native gate set (basis gates) and entangling-gate topology (coupling map) of a target device. This guarantees that the graph structure of the circuits, as defined by their entangling-gate connectivity, is a subgraph of the full device coupling map. Finding an optimal set of qubits then becomes a two-step process. First, a search over the system coupling map is performed to identify subgraphs that are isomorphic to each input circuit. Second, a heuristic cost function is used to score the resultant mappings to find subgraphs with the lowest error. Once identified, the input circuits are remapped to their corresponding minimal-cost subgraphs before execution.

In the first step, we iterate over all the instructions [24] in the circuit after qubit routing, and possibly after additional gate optimization (postcompilation), and build an interaction graph. Edges in this graph represent unique two-qubit gates in the circuit and single-qubit gates are treated as standalone nodes. Typically, a simple graph that is undirected and with no parallel edges is used. We then search for isomorphic subgraphs on the connectivity graph of the quantum processor; a graph where each node represents a physical qubit and each edge indicates support for two-qubit gates between those qubits. Subgraphs isomorphic to the original mapping are bijective layouts between virtual circuit qubits and physical qubits on the quantum processor. Although here we assume that all entangling gates are of the same type, this is not a limitation of our routine. Figure 2 shows an example of the graph construction used for the subgraph-isomorphism problem.

By running after the circuit has been mapped to system topology, there is at least one possible mapping available, as the compiler must rewrite the circuit to match the device. In our implementation, we use the VF2 [25]

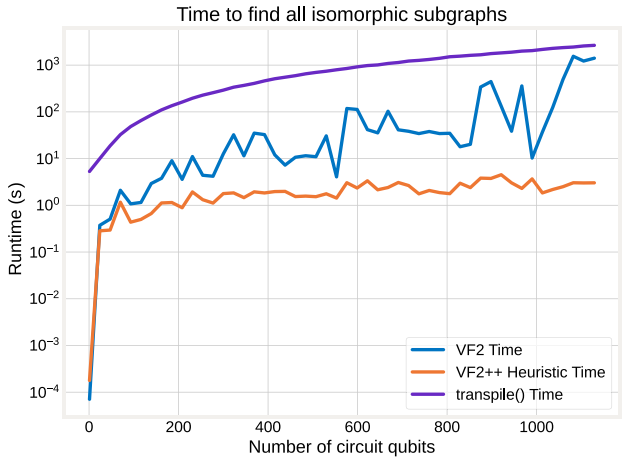

FIG. 3. The search time for finding all isomorphic subgraphs of an interaction graph of a random N-qubit quantum circuit of fixed depth on the coupling map for a target back end with a 1299-qubit heavy-hex coupling map with VF2 and the VF2++ heuristic ordering used by MAPOMATIC. The time is compared to that of the QISKIT TRANSPILE function using the default OPTIMIZATION_LEVEL=1 on the same circuit. Benchmarks are run on a AMD Threadripper 3970x with 128 GB of RAM using QISKIT TERRA 0.23.1 and RUSTWORKX 0.12.1.

implementation of the RUSTWORKX [26,27] library, which includes support for using the search-order heuristic from VF2++ [28]. This node-ordering heuristic, of which MAPOMATIC takes advantage, orders the search tree based on the degree of each node and in some situations can make the isomorphism search faster, minimizing the cost of the search. Figure 3 shows the run time of the VF2 mapping function of the RUSTWORKX library to find all the isomorphic subgraphs of the coupling map with and without the VF2++ ordering heuristic compared with the time it takes to execute the full QISKIT compilation pipeline without MAPOMATIC. Being 3 orders of magnitude faster at the largest qubit counts, it is clear that the overhead from subgraph finding is minimal compared to the rest of the compilation work flow. This difference widens if, unlike the constant circuit depth used in Fig. 3, we consider situations where the circuit depth increases with width. In these cases, the circuit routing and optimization time increase with depth, whereas subgraph finding is only minimally affected because the device topology remains unchanged. Additionally, limitations can be set on the number of internal state visits used in VF2 to bound the overall run time for finding a set of isomorphic mappings [29].

For the second step, we apply an efficient heuristic scoring function to each found mapping (see Algorithm 1). The scoring function bases its output on the reported calibration data for the processor, from which we can define an error map $\mathcal{E}$ that maps physical instructions provided by a layout mapping $\mathcal{M}$ to error rates. For IBM Quantum systems,

1: **procedure** SCORE($C, \mathcal{M}, \mathcal{E}$) ▷ Quantum circuit C with a set of instructions $\{C\}$, layout mapping $\mathcal{M}$, and error map $\mathcal{E}$
2: fidelity $\leftarrow 1.0$
3: **for** instr $\in$ {C} **do**
4: fidelity $\leftarrow$ fidelity $*$ $(1 - \mathcal{E}[\mathcal{M}[\text{instr}]])$
5: **end for**
6: **return** $1 -$ fidelity
7: **end procedure**

Algorithm 1. Algorithm for scoring layout mappings

this includes, amongst other information, single- and two-qubit gate errors, measurement infidelities, and T_1 and T_2 times across the device. This data are nominally updated daily and thus the scoring can change on a similar time scale. For each isomorphic mapping, we estimate the overall fidelity of the circuit with that layout applied by taking the product of instruction fidelities corresponding to the physical qubits over which those instructions are applied. That the resultant fidelity approaches zero in the large error limit is not a concern here, as we are looking for relative differences, not absolute values (for a discussion, see Ref. [30]). The returned scores for each layout are then used to rank all the possible mappings in order of their estimated error and the layout with the least error is used. Note that the choice of cost function is not hardcoded into MAPOMATIC and users are free to define cost functions based on arbitrary input information.

Missing from Algorithm (1) are T_1 and T_2 times, from which one can define approximate error rates associated with qubit idle times. We do not include this information in our default cost function as it has been empirically found not to have a large impact on the layout order from scoring; it nominally permutes qubits within a given mapping but does not modify the mapping ordering (see Appendix A). Additionally, adding timing information to quantum circuits is currently an unoptimized transformation in QISKIT [31] and greatly increases the run time of layout selection. We do, however, include the cost function with idle errors as an example of a custom scoring function at the MAPOMATIC web site [23].

It is also important to note that scoring returns floating-point values for each layout. The difference between these values can be smaller than the uncertainty from device fluctuations and ultimately finite-sampling statistics, effectively leading to a tie between one or more scored layouts. In cases such as these, there is a need for more complex cost functions and/or information beyond device-calibration data in order to break the scoring degeneracy.

III. INTEGRATION INTO A COMPILER

While MAPOMATIC was originally designed to run as a standalone postcompilation routine, it is also possible

to directly integrate it into a compilation pipeline. When doing this, the only additional constraint is to ensure that any changes in layout do not prevent the circuit from running on the device. When running as a standalone post-compilation routine, this is not a constraint because we can always rerun the compilation with a fixed optimal layout to update the circuit. There are two techniques for integrating the algorithm into a compiler: the first is to perform MAPOMATIC immediately after routing and the second is to run MAPOMATIC after all optimization routines but prior to any physical scheduling of the circuit. There are trade-offs between the two approaches.

Running immediately after routing works with looser constraints, typically ignoring directionality (by using an undirected interaction graph and coupling map) and with no guarantee that the circuit has been converted to the native gate set. This means that for error evaluation, we can only apply an inexact scoring, typically using average error rates for the different types of operations available (i.e., one-qubit gates, two-qubit gates, etc.) on the device instead of using the exact error rates for each instruction from the calibration data. However, running with looser constraints provides the flexibility to evaluate more potential layouts and potentially yield better results, as later compilation stages in the pipeline will be able to transform the circuit as needed.

Running the algorithm after physical optimization (but before scheduling) means that the circuit is guaranteed to be matching the directionality of the entangling-gate topology and all operations are in the native gate set. This allows Algorithm (1) to apply a more exact scoring, as the exact instructions used in the output circuit have already been determined, leading to a potentially more accurate selection of the best-performing layout. However, running in this mode places more constraints on the available layouts, as at this point in a typical compilation pipeline the algorithm must conform to all the constraints of the target device. This means that the algorithm needs to use directed graphs for the interaction graph and the coupling map and also only evaluate isomorphic mappings where the instructions performed on the circuit qubits are available on the physical qubits selected, as the selection of a layout that violates these constraints would result in an invalid output from the compilation.

We integrate the MAPOMATIC algorithm into the QISKIT compiler as the VF2POSTLAYOUT pass [32]. The VF2POSTLAYOUT pass supports running in both modes; however, it is integrated into the default compilation pipelines only in the first mode, which runs immediately after routing. While in most cases when compiling for current hardware there is no difference when running the different techniques, there are some situations where being able to evaluate more potential mappings can yield better results.

IV. BENCHMARKING RESULTS

In order to validate the performance benefit of our method, we take advantage of the wide variety of test-circuit libraries that are available [33–36]. In particular, here we use circuits from the Quantum Economic Development Consortium (QED-C) [34], comparing runs of this suite with and without MAPOMATIC. We point out that this test suite is developed independent of this work and, unlike a few hand-selected examples, represents a true test of our technique. Rather than focusing on all the test results, some of which are well beyond what today's quantum processors are capable of, we only look at those algorithmic tests and numbers of qubits where at least one of the two fidelities used in the comparison is $\geq 1/e$. This prevents inflated claims of success when dealing with differences in fidelity values the overall magnitude of which is not indicative of meaningful experimental outcomes. In addition, rather than looking at fidelities directly, we ask what fraction of the fidelity missing in the baseline result can be recovered through better qubit selection. That is, given a baseline fidelity f_{base}, we compute the value

$$\eta = \frac{f - f_{\text{base}}}{1 - f_{\text{base}}}, \tag{1}$$

where f is the comparison fidelity.

In Fig. 4(a), we use the fidelities obtained on the QED-C test suite using QISKIT as the baseline and look at the fraction of recoverable fidelity when using MAPOMATIC after compilation to remap the same circuit on the IBM Quantum Peekskill device. Specifically, all baseline circuits are transpiled once in QISKIT using OPTIMIZATION_LEVEL=3 and APPROXIMATION_DEGREE=0, which enables SABRE layout and routing and disables approximate unitary synthesis. We see that qubit remapping provides a substantial performance benefit over the qubit assignment of SABRE layout. Across the benchmarking results, an average (median) of 37% (34%) of the fidelity is recoverable on the system via simple qubit remapping. This emphasizes the importance of not only choosing the correct quantum system to execute circuits but also the correct qubits on that system. The small number of test results with negative values indicates a loss in fidelity from the remapping process; the calibration data no longer faithfully represent a subset of qubits (device parameter drift) and/or the simple cost function used here does not capture enough of the underlying contributions to circuit errors. However, it is clear that in the vast majority of cases, there are marked performance gains from using our technique to remap quantum circuits.

While Fig. 4(a) highlights fidelity improvements versus QISKIT without MAPOMATIC, it is also beneficial to look at comparisons against other compilation pipelines. To this end, in Fig. 4(b) we look the differences in fidelity when using QISKIT with MAPOMATIC integrated as a transpiler pass versus the compiler in PyTket [37], used as the base

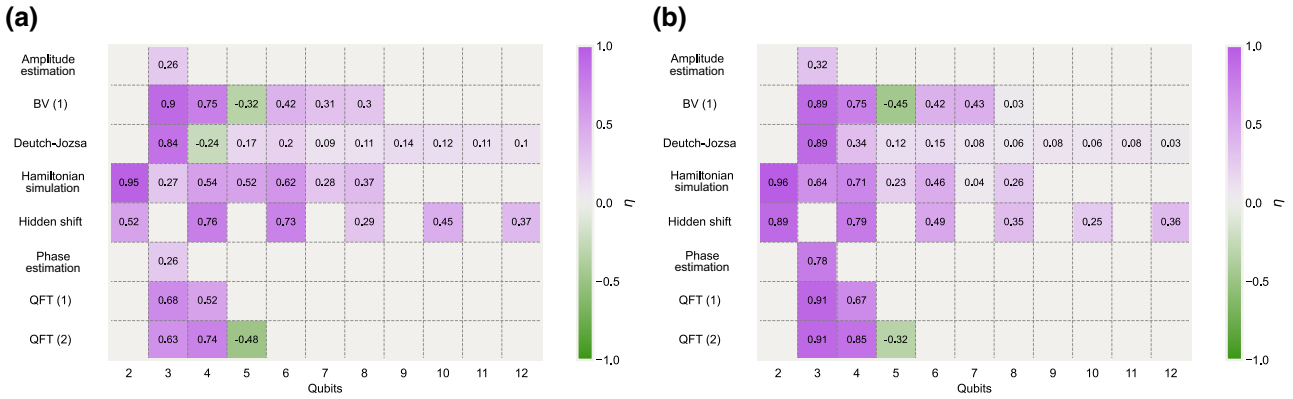

FIG. 4. The fraction of the fidelity [see Eq. (1)] that is recoverable on the QED-C test circuits when using (a) MAPOMATIC to remap circuits generated by the QISKIT transpiler at optimal settings and (b) QISKIT with MAPOMATIC as a transpiler pass versus a baseline value corresponding to circuits compiled using PyTket. The circuits are executed on the IBM Quantum Peekskill system, taking 10, 000 samples, with no gap between subsequent runs of the test suite. Results colored purple show improvement over baseline, whereas green indicates degraded performance. The results in (a) are produced using QISKIT TERRA 0.20.2 with MAPOMATIC, whereas (b) uses QISKIT TERRA 0.21.0, in which MAPOMATIC is introduced as a default transpiler pass.

fidelity. In this investigation, we use PyTket version 1.3.0 with DEFAULT_COMPILATION_PASS(2). This comparison is of interest not only because PyTket is a commonly used alternative to QISKIT but also because its qubit layout and routing methods are deterministic rather than stochastic. As with Fig. 4(a), we see that a sizable portion of the fidelity missing from the PyTket results is recoverable with MAPOMATIC utilized in the QISKIT compilation stack.

Although the results presented in Fig. 4 highlight the possible gains when using MAPOMATIC, they do not constitute a panacea and other steps of the compilation process still play an important role in the final output fidelity. In particular, the stochastic nature of the qubit-routing (SWAP-mapping) methods used in QISKIT gives rise to a variable number of SWAP gates in the final compiled circuits. The variance on the number of added gates means that it is possible that a poor routing choice cannot be remapped with a higher fidelity than would otherwise be possible with a more careful routing selection utilizing repeated routing attempts or using a nonstochastic routing routine such as that found in PyTket. An example highlighting poor routing on the overall fidelity is shown in Appendix B. As a corollary, our results show the challenge when evaluating the performance of quantum systems; the fidelity of the final results depends greatly on the circuit-rewriting pipeline used before execution. This applies even more so to cross-platform comparisons where different compilation work flows, both client-side and server-side, further complicate the interpretation.

V. BEST-DEVICE SELECTION

To date, standard work flows for quantum computation involve first selecting a good target system on which to compile and execute a set of quantum circuits. Until now, we have followed this procedure in this work as well, choosing the IBM Quantum Peekskill system based on prior knowledge of its performance characteristics. However, as the number of available quantum systems grows and the complexity of algorithms that they faithfully execute increases, it becomes challenging to ascertain which of the myriad of system- and qubit-layout combinations are good candidates. Moreover, when running many different algorithms, or the same algorithm over a varying number of qubits, the optimal system and qubit layout can span several devices. Therefore, it can be valuable to look for optimal circuit layouts across multiple quantum processors.

It is possible to use the tools presented here to determine good initial device and layout candidates across multiple quantum systems in the following manner. To begin, circuits must be compiled against one of the systems within the set of possible target processors. To obtain the largest number of matching subgraphs, it is ideal for the systems to share a common entangling-gate topology, e.g., all IBM Quantum systems are based on the heavy-hex architecture [38]. The MAPOMATIC routine can then be run across the set of quantum processors returning the device name, the best layout, and the associated error value for each processor. The device and layout corresponding to the lowest overall error value is then selected for remapping and execution. The matching subgraphs for systems with the same topology need only be computed once, whereas the cost function for each layout must be evaluated on each individual system. If a quantum system does not have enough qubits to accommodate a circuit, then there are no matching subgraphs and the device is skipped.

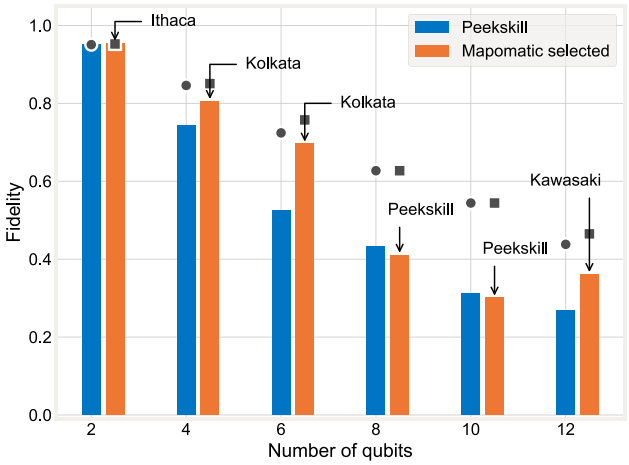

FIG. 5. The fidelity of the QED-C Hamiltonian-simulation circuits for the Peekskill system used in Fig. 4 compared to execution of the same circuits on the optimal target system as selected by MAPOMATIC from the entire set of IBM Quantum devices. The markers show the estimated fidelity for the Peekskill (circle) and MAPOMATIC chosen (square) systems. Circuits are executed in the same manner as Fig. 4.

To demonstrate the value in looking across multiple systems, we repeat the Hamiltonian-simulation test circuits from the QED-C suite and use MAPOMATIC to find the system and layout with minimal cost across the entire fleet of IBM Quantum systems ranging from five to 127 qubits. The experiments presented in Fig. 5 show the resultant fidelity from the quantum system selected from the entire IBM Quantum processor line-up compared to the best layout on the IBM Quantum Peekskill system used in Fig. 4. Although the Peekskill system is a newer high-coherence (average $T_1 \sim 300$ μsec) 27-qubit Falcon R8 system (see Fig. 1), we see that previous-generation Kolkata and Kawasaki Falcon R5 systems are selected and give better results for several circuit widths. The same is true for the two-qubit case, where the 65-qubit Ithaca system does better but with gains limited by the overall high fidelities at such small circuit space-time volumes. We also see that the estimated fidelities returned by MAPOMATIC do not match the results from hardware execution. First, as mentioned previously, we do not include qubit idle-time errors in the cost function. Although they contribute to the overall circuit error, we empirically find that they do not greatly change the ordering of layouts; contributions from T_1 and T_2 in the instruction errors already capture much of much of these effects in the scoring process. Second, the calibration data returned do not include sources of error such as spectator qubits or crosstalk. Thus the fidelities corresponding to scored layouts should be loosely interpreted as upper bounds. The performance gains shown in Fig. 5 are non-negligible given that the Peekskill system is one of the best-performing IBM Quantum systems to date, and is selected because of this. This highlights the fact that even with detailed device knowledge, looking across multiple quantum systems with MAPOMATIC can provide performance gains that would otherwise be overlooked.

VI. CONCLUSIONS

We show that it is possible to account for system variability in near-term quantum processors using circuit remapping applied postcompilation, or after qubit routing. Using simple quick-to-evaluate cost functions, we demonstrate that sizable fractions (approximately 40%) of result fidelity can be recovered on a wide variety of standard quantum application test circuits and as compared to other circuit transformation pipelines, using our MAPOMATIC method. Using performant subgraph-isomorphism routines, our technique adds little in terms of additional cost to the overall compilation process. Given that much of the overall wall-clock time of executing circuits is waiting in a queue, this remapping overhead can likely be amortized over this duration. This low overhead also allows for layout scoring across multiple quantum processors, alleviating the need for users to identify a target system ahead of time and uncovering additional performance improvements that are missed if device selection is done ahead of time. MAPOMATIC is easy to integrate into quantum work flows and, because of the performance advantages it offers, is incorporated into the transpilation process by default in QISKIT TERRA 0.21+. Thus users are capable of leveraging these techniques immediately and in many cases have been doing so without knowing it.

There are several possible future improvements to MAPOMATIC that should be investigated. Chief among them is looking for improved heuristics for scoring layouts that are more accurate while at the same time being efficient to evaluate. In particular, is there information outside of standard device-calibration data that can yield more accurate cost analysis and break ties between layouts? Additionally, qubit selection over multiple quantum systems has received little attention to date but is likely beneficial for algorithms that can be executed in parallel over multiple systems, e.g., variational algorithms [39,40] and circuit cutting [41,42]. More broadly, as the performance improvements seen with MAPOMATIC apply across the board, it is possible to integrate it into cloud-based work flows that would allow for the abstraction of device selection away from users. This would not only lead to performance gains but it would also remove the need for users not interested in device characterization to understand detailed system information.

As the field of quantum computation approaches the frontier of quantum advantage and users increasingly make use of error-mitigation techniques with run times exponentially sensitive to qubit quality, MAPOMATIC and related qubit selection methods will undoubtedly play a pivotal

TABLE I. The lowest-cost layouts found using the default MAPOMATIC cost function, as well as when including idle errors due to T_1 and T_2 in the scoring process, for the Hamiltonian-simulation experiments on IBM Peekskill presented in Sec. V.

Number of qubits	Default cost function	Cost function with T_1 and T_2
2	[5, 8]	[5, 8]
4	[9, 8, 5, 3]	[9, 8, 5, 3]
6	[16, 14, 11, 8, 5, 3]	[16, 14, 11, 8, 5, 3]
8	[2, 3, 5, 8, 11, 12, 13, 14]	[12, 13, 14, 11, 8, 2, 3, 5]
10	[2, 3, 5, 7, 8, 10, 11, 12, 13, 14]	[3, 5, 8, 4, 11, 7, 14, 10, 12, 13]
12	[23, 21, 18, 15, 12, 11, 14, 13, 8, 5, 3, 2]	[24, 23, 21, 18, 15, 14, 13, 12, 11, 8, 5, 3]

role in early demonstrations of practical quantum applications.

ACKNOWLEDGMENTS

We thank Doug McClure and David McKay for helpful discussions. This material is based upon work supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, Co-design Center for Quantum Advantage (C2QA) under Contract No. DE-SC0012704.

APPENDIX A: INCLUDING T_1 AND T_2 IN COST FUNCTION

The default cost function in MAPOMATIC does not include errors on idle qubits due to incoherent errors from T_1 and T_2 processes. As mentioned in the main text, this is due to two reasons. First, it is currently inefficient to add repeatedly generate timing information in QISKIT and doing so would greatly increase the run time of the scoring process. Second, even when this information is included, the resulting qubit layouts are to a large extent nominally simple permutations of the layouts computed without these error processes; effects from T_1 and T_2 are, to large extent, already accounted for in the gate and measurement errors included in the device-calibration data. In Table I, we give an explicit example of this, looking at the resulting optimal qubit layouts generated by MAPOMATIC, both with and without idle errors included in the cost function, from the Hamiltonian-simulation test circuits from Sec. V.

FIG. 6. MAPOMATIC as a QISKIT transpiler pass versus PyTket. The SABRE SWAP mapper is run 10 times for each QISKIT circuit, with the circuit with the largest number of CNOT gates selected for execution. All other execution parameters are the same as in Fig. 4(b).

APPENDIX B: SWAP-MAPPING VARIABILITY

MAPOMATIC will remap any circuit passed to it that matches hardware topology and is written in terms of the supported basis gates. For compilation routines that contain stochastic components, such as the qubit-routing methods in QISKIT, this means that the properties of the output circuits will follow a distribution of values. For routing, this is a distribution in the total number of SWAP gates added to the circuit in order to satisfy the topology constraints of the target system. Because each SWAP gate (equal to three CNOT gates) is costly to execute, there is a corresponding distribution in output fidelity when executing the same input circuit multiple times. It can be the case that the variance in result fidelities is greater than the benefit of remapping and that optimization of routing before remapping with MAPOMATIC is still an important part of the compilation work flow. To overcome this, circuits should nominally be routed multiple times, which can be done in parallel, with the output circuit comprised of the fewest CNOT gates and then passed on to MAPOMATIC.

To highlight the effect that routing has on the overall fidelity, we do the opposite of what is proposed above and compile circuits multiple times using QISKIT and the SABRE routing method, taking the one with the *greatest* number of CNOT gates as the chosen circuit. Figure 6 shows the effect that this has on the QED-C test circuits, using the deterministic routing method in PyTket as the baseline result. The detrimental consequence that added SWAP gates have on the resultant fidelity is clear when comparing against the results in Fig. 4(b).

[1] Y. Ding, P. Gokhale, S. F. Lin, R. Rines, T. Propson, and F. T. Chong, in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (2020), p. 201.

[2] S. Niu, A. Suau, G. Staffelbach, and A. Todri-Sanial, A hardware-aware heuristic for the qubit mapping problem in the NISQ era, IEEE Trans. Quantum Eng. **1**, 1 (2020).

[3] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers (2019), ArXiv:1901.11054.

[4] W. Finigan, M. Cubeddu, T. Lively, J. Flick, and P. Narang, Qubit allocation for noisy intermediate-scale quantum computers (2018), ArXiv:1810.08291.

[5] N. Sundaresan, T. J. Yoder, Y. Kim, M. Li, E. H. Chen, G. Harper, T. Thorbeck, A. W. Cross, A. D. Córcoles, and M. Takita, Matching and maximum likelihood decoding of a multi-round subsystem quantum error correction experiment (2022), ArXiv:2203.07205.

[6] J. R. Glick, P. Gujarati, Tanvi, A. D. Córcoles, Y. Kim, A. Kandala, J. M. Gambetta, and K. Temme, Covariant quantum kernels for data with group structure (2021), ArXiv:2105.03406.

[7] Y. Kim, C. J. Wood, T. J. Yoder, S. T. Merkel, J. M. Gambetta, K. Temme, and A. Kandala, Scalable error mitigation for noisy quantum circuits produces competitive expectation values (2021), ArXiv:2108.09197.

[8] P. Jurcevic, *et al.*, Demonstration of quantum volume 64 on a superconducting quantum computing system (2020), ArXiv:2008.08571.

[9] G. L. Li, Y. Ding, and Y. Xie, Tackling the qubit mapping problem for NISQ-era quantum devices (2018), ArXiv:1809.02573.

[10] F. Vatan and C. Williams, Optimal quantum circuits for general two-qubit gates, Phys. Rev. A **69**, 032315 (2004).

[11] N. Ezzell, B. Pokharel, L. Tewala, G. Quiroz, and D. A. Lidar, Dynamical decoupling for superconducting qubits: A performance survey (2022), ArXiv:2207.03670.

[12] V. Tripathi, H. Chen, M. Khezri, K.-W. Yip, E. M. Levenson-Falk, and D. A. Lidar, Suppression of crosstalk in superconducting qubits using dynamical decoupling (2021), ArXiv:2108.04530.

[13] B. Pokharel, N. Anand, B. Fortman, and D. A. Lidar, Demonstration of Fidelity Improvement Using Dynamical Decoupling with Superconducting Qubits, Phys. Rev. Lett. **121**, 220502 (2018).

[14] E. van den Berg, K. Minev, Zlatko, A. Kandala, and K. Temme, Probabilistic error cancellation with sparse Pauli-Lindblad models on noisy quantum processors (2022), ArXiv:2201.09866.

[15] P. D. Nation, H. Kang, N. Sundaresan, and J. M. Gambetta, Scalable Mitigation of Measurement Errors on Quantum Computers, PRX Quantum **2**, 040326 (2021).

[16] K. Temme, S. Bravyi, and J. M. Gambetta, Error Mitigation for Short-Depth Quantum Circuits, Phys. Rev. Lett. **119**, 180509 (2017).

[17] https://qiskit.org/documentation/stubs/qiskit.transpiler.passes.denselayout.html (2022).

[18] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, Validating quantum computers using randomized model circuits, Phys. Rev. A **100**, 032328 (2019).

[19] E. Peters, P. Shyamsundar, A. C. Y. Li, and G. Perdue, Noise-aware qubit assignment on NISQ hardware using simulated annealing and Loschmidt Echoes (2022), ArXiv:2201.00445.

[20] https://quantumai.google/hardware (2022).

[21] S. A. Cook, in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71 (Association for Computing Machinery, New York, 1971), p. 151.

[22] L. Cincio, K. Rudinger, M. Sarovar, and P. J. Coles, Machine Learning of Noise-Resilient Quantum Circuits, PRX Quantum **2**, 010324 (2021).

[23] https://github.com/qiskit-partners/mapomatic (2022).

[24] Here, instructions include gates and nonunitary operations such as qubit reset and measurement but not timing and alignment expressions such as barriers.

[25] L. Cordella, P. Foggia, C. Sansone, and M. Vento, A (sub)graph isomorphism algorithm for matching large graphs, IEEE Trans. Pattern Anal. Mach. Intell. **26**, 1367 (2004).

[26] M. Treinish, I. Carvalho, G. Tsilimigkounakis, and N. Sá, RUSTWORKX: A high-performance graph library for PYTHON, J. Open Source Softw. **7**, 3968 (2022).

[27] https://github.com/qiskit/rustworkx (2022).

[28] A. Jüttner and P. Madarasi, Vf2++—an improved subgraph isomorphism algorithm, Discrete Appl. Math. **242**, 69 (2018). computational Advances in Combinatorial Optimization,.

[29] The set need not be complete if the number of calls is not large enough.

[30] T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, Measuring the capabilities of quantum computers, Nat. Phys. **18**, 75 (2022).

[31] QISKIT, https://qiskit.org (2022).

[32] https://qiskit.org/documentation/stubs/qiskit.transpiler.passes.vf2postlayout.html (2022).

[33] T. Tomesh, P. Gokhale, V. Omole, G. S. Ravi, K. N. Smith, J. Viszlai, X.-C. Wu, N. Hardavellas, M. R. Martonosi, and F. T. Chong, SupermarQ: A scalable quantum benchmark suite (2022), ArXiv:2202.11045.

[34] T. Lubinski, S. Johri, P. Varosy, J. Coleman, L. Zhao, J. Necaise, C. H. Baldwin, K. Mayer, and T. Proctor, Application-oriented performance benchmarks for quantum computing (2021), ArXiv:2110.03137.

[35] R. Blume-Kohout and K. C. Young, A volumetric framework for quantum computer benchmarks, Quantum **4**, 362 (2020).

[36] A. Li, S. Stein, S. Krishnamoorthy, and J. Ang, QASMBench: A low-level QASM benchmark suite for NISQ evaluation and simulation (2020), ArXiv:2005.13018.

[37] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, t—ket>: A retargetable compiler for NISQ devices, Quantum Sci. Technol. **6**, 014003 (2020).

[38] C. Chamberland, G. Zhu, T. J. Yoder, J. B. Hertzberg, and A. W. Cross, Topological and Subsystem Codes on Low-Degree Graphs with Flag Qubits, Phys. Rev. X **10**, 011022 (2020).

[39] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C.

Kwek, and A. Aspuru-Guzik, Noisy intermediate-scale quantum algorithms, Rev. Mod. Phys. **94**, 015004 (2022).

[40] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms, Nat. Rev. Phys **3**, 625 (2021).

[41] T. Peng, A. W. Harrow, and X. Wu, Simulating Large Quantum Circuits on a Small Quantum Computer, Phys. Rev. Lett. **125**, 150504 (2020).

[42] S. Bravyi, G. Smith, and J. A. Smolin, Trading Classical and Quantum Computational Resources, Phys. Rev. X **6**, 021043 (2016).