Quantum Machine Intelligence (2023) 5:40
https://doi.org/10.1007/s42484-023-00124-1

RESEARCH ARTICLE

Interaction graph-based characterization of quantum benchmarks for improving quantum circuit mapping techniques

Medina Bandic[1,2] · **Carmen G. Almudever**[3] · **Sebastian Feld**[1,2]

Received: 15 March 2023 / Accepted: 1 July 2023 / Published online: 6 October 2023

Abstract
To execute quantum circuits on a quantum processor, they must be modified to meet the physical constraints of the quantum device. This process, called *quantum circuit mapping*, results in a gate/circuit depth overhead that depends on both the circuit properties and the hardware constraints, being the limited qubit connectivity a crucial restriction. In this paper, we propose to extend the characterization of quantum circuits by including qubit interaction graph properties using graph theory-based metrics in addition to previously used circuit-describing parameters. This approach allows for an in-depth analysis and clustering of quantum circuits and a comparison of performance when run on different quantum processors, aiding in developing better mapping techniques. Our study reveals a correlation between interaction graph-based parameters and mapping performance metrics for various existing configurations of quantum devices. We also provide a comprehensive collection of quantum circuits and algorithms for benchmarking future compilation techniques and quantum devices.

Keywords Quantum circuits · Compiler · Full-stack quantum computing systems · Quantum circuit mapping · Profiling · Benchmarks

1 Introduction

Quantum technology has experienced rapid development in the past decades and has the potential to solve some classically intractable problems. Its contributions are still in the early stage, as current so-called Noisy Intermediate-Scale Quantum (NISQ) devices can only handle simple, small-sized algorithms considering they are limited by size and noise. They also encompass additional hardware constraints such as low qubit connectivity, reduced supported gate set, and limitations related to classical-control resources, which makes it even more difficult to execute a quantum circuit on these processors successfully.

Quantum algorithms, usually represented as quantum circuits, are hardware-agnostic; that is, when described, they do not consider hardware restrictions. To execute such algorithms (quantum circuits) on a quantum processor, they must be modified to fulfill the processor's limitations through a process called quantum circuit mapping. The quantum circuit mapper, which is part of the compiler, is then at the core of the full-stack quantum computing system, connecting algorithms with quantum devices (Bandic et al. 2022).

Various techniques have been proposed to deal with the mapping of quantum circuits (Li et al. 2019; Murali et al. 2019a; Tannu and Qureshi 2019; Li et al. 2020; Zulehner et al. 2018; Venturelli et al. 2019; Lao et al. 2019a, b; Herbert and Sengupta 2018), which differ in approach (exact or heuristic, local or global solution), methodology (e.g., SMT solver (Lye et al. 2015)), cost functions (optimizing number of gates or circuit depth), and performance metrics (e.g., circuit fidelity). These solutions, however, adopt a bottom-up approach, developing mappers specifically for certain quantum processors and technologies. The majority of quantum circuit mapping techniques have mostly focused on hardware properties (Tannu and Qureshi 2019; Lao et al. 2022) and only considered a rather limited set of algorithm characteristics such as number of qubits, number of quantum gates, two-qubit gate percentage, and qubit interactions (i.e., what pair of qubits perform a two-qubit gate). In addition to this, when mapping outcomes are analyzed, the focus is on the values of the obtained metrics without further evaluating

✉ Medina Bandic
m.bandic@tudelft.nl

[1] Delft University of Technology, Delft, The Netherlands

[2] QuTech, Delft, The Netherlands

[3] Technical University of Valencia, Valencia, Spain

why some circuits show higher or lower overheads. Some works have already pointed out the importance of including more algorithm features in the mapping process (Lao and Browne 2021a). A more complete and in-depth profiling of quantum circuits will help to (i) have a deeper understanding on why specific algorithms have higher fidelity than others when being run on a particular processor using a specific mapping technique; (ii) to categorize (cluster) quantum circuits based on those parameters and predict the performance of additional circuits with similar properties in terms of mapping-related metrics, without actually running them on a given device; and (iii) to develop application-driven and hardware-aware mapping techniques (i.e., mapping techniques tailored for a specific set of algorithms in addition to overcoming hardware constraints) (Bandic et al. 2022; Li et al. 2021a; Lao and Browne 2021b). Note that more broadly, this characterization of quantum circuits will be also crucial for defining a meaningful and complete set of quantum benchmarks to evaluate not only quantum circuit mapping techniques but also full-stack quantum computing systems as well as for having a set of algorithm-level metrics to measure system performance (Tomesh et al. 2022).

One of the most stringent quantum hardware constraints that quantum circuit mapping techniques have to deal with is the limited connectivity of physical qubits, which restricts possible interactions between them. Therefore, in this paper, we propose to extend the profiling of quantum circuits/algorithms by not only extracting "standard" parameters like the number of qubits and gates and percentage of two-qubit gates, but also by performing a deeper analysis of their qubit interaction graphs (i.e., representation of the two-qubit gates or qubit interactions of the circuit). By taking input from graph theory and machine learning, we characterize quantum circuits based on their interaction graph metrics (e.g., average shortest path, connectivity, clustering coefficient). We then map those quantum circuits into several quantum processors using a specific quantum circuit mapping technique. In future work, we will also use different quantum circuit mapping configurations, allowing us to evaluate what quantum circuit features impact the circuit mapping performance the most and identify what combination of mapping technique-quantum hardware works better for a given (set of) algorithm(s). Note that this analysis can in the future help in the codesign of algorithm-driven compilation methods and quantum hardware.

In addition, we present a categorized and, as of now, the most comprehensive set of quantum algorithms (benchmarks) from various sources and platforms and in different quantum programming languages. Most of the currently existing and used quantum algorithms, synthetically generated and application-based circuits are included in this collection and classified based on different criteria. We are hoping that this algorithms/circuits set will be used for benchmarking quantum computing systems as well as parts of it, such as compilation techniques.

The main contributions of this work are as follows:

1. We have performed the first characterization and clustering of quantum circuits that also considers qubit interaction graph parameters in addition to the characteristics related to circuit size (number of gates, number of qubits, amount of two-qubit gates). In-depth profiling and clustering of quantum circuits based on their more structural parameters help to analyze why and when some (families of) quantum algorithms show better performance compared to the rest when being executed on a given quantum processor, as well as which circuit parameters have a higher impact on performance for some hardware-compiler setups. Subsequently, that can also help to predict the mapping performance for additional circuits with similar properties, without actually running them on a given device, and therefore assist in recommending an adequate mapper and hardware configuration to use. Finally, this circuit structural parameters analysis step is crucial for the development of future application-based quantum devices and mappers.
2. We have found that quantum circuits similarly structured in terms of their interaction graph parameters will have comparable results in terms of circuit fidelity and gate overhead when mapped on the same quantum device and by using the same mapping technique. By running these groups of circuits with different hardware configurations, we could make clear suggestions on which group of circuits fits which hardware better.
3. We provide the so-far most comprehensive collection of quantum benchmarks, open-source and available in most currently used high- or low-level quantum languages. The goal is to help the quantum community speed up the research process and in the development of a full-stack quantum system by having an easily accessible, all-in-one-place set of benchmarks that can be used for analyzing the performance of existing and future quantum processors and compilation methods.

The paper is organized as follows: Section 2 presents a short introduction to full-stack quantum computing systems and an overview of the current state-of-the-art quantum circuit mapping techniques as well as benchmark characterization. Section 3 introduces our profiling of quantum algorithms and their clustering based on size and structure. The experimental setup with the details of our benchmark collection is included in Section 4. Section 5 showcases the obtained results on how the mapping performance of quantum circuits when run on a specific chip relates to their structural parameters acquired from the analysis of their interaction graphs and their clusters

from Section 3. Finally, in Sections. 6 and 7, conclusions and future work are presented.

2 Background and related work

2.1 Quantum computers nowadays

Quantum hardware has significantly progressed since its inception, and a wide variety of technologies has been developed for implementing qubits like solid-state spins, trapped-ion qubits, or superconducting qubits (Resch and Karpuzcu 2019). Hardware characteristics like the number of qubits and gate fidelity are continuously improving. However, current NISQ devices are still immensely resource-constrained and error-prone. They are not able to keep up with the development of promising *quantum algorithms*, that might achieve exponential speed-up, as they lack in size (number of qubits), which is required for the implementation of fault-tolerant and error-corrected techniques. Therefore, it was inevitable to develop a set of algorithms that could be successfully executed on current processors, coming from different fields like quantum physics, chemistry, or machine learning (Bharti et al. 2022).

Quantum compilers act like intermediaries between algorithms (expressed as quantum circuits) and quantum processors. They not only translate high-level programming language instructions (e.g., library Qiskit given in Python (Anis et al. 2021)) into low-level ones (quantum assembly-like language, e.g., OpenQASM (Li 2019)), but are also responsible for making transformations and optimizations of the quantum circuit to best fulfill the quantum hardware requirements. The compiler design and complexity highly depend on the constraints imposed by the hardware and chosen technology. In nearest-neighbor architectures (e.g., 2D array of qubits), the primary constraint is the limited connectivity among qubits. As running two-qubit gates requires that the paired qubits are adjacent on the chip, restricted connectivity can become a huge obstacle. The compiler tries to overcome that and other limitations and helps to successfully execute a quantum circuit on a given quantum device through a process called mapping. Note that the mapping of quantum circuits usually results in a gate and latency overhead that in turn decreases the circuit fidelity. Therefore, having efficient mapping techniques is crucial in the NISQ era not only to successfully execute quantum algorithms but also for extracting the most out of constrained NISQ devices.

2.2 Computing with NISQ devices

One of the motivations for building quantum computers in the first place is to run algorithms that solve problems that are intractable for existing classical computers due to limitations in speed and memory. Current NISQ devices can only handle simple algorithms, in terms of the number of qubits and gates and circuit depth, as the presence of noise and limited resources (physical qubits) still constrain them: quantum operations have high error rates and qubits decohere over time resulting in information loss. On top of that, running an algorithm on a NISQ device is not a straightforward process. That is due to hardware constraints that affect the algorithm execution, which can vary between quantum technologies.

One of the restrictions that affects the execution of a quantum algorithm the most is (*limited*) *qubit connectivity*. That applies to most technologies, including superconducting qubits and quantum dots, where qubits are arranged in a 2D grid or some other not-fully connected topology, as shown in the top-right part of Fig. 1, allowing only nearest-neighbor interactions. In order to perform a two-qubit gate in such architecture, the two interacting qubits in the circuit have to be placed in neighboring physical qubits on the chip, which is not always possible (see Fig. 1: two two-qubit gates between virtual qubits 1 and 5, and 5 and 6 cannot be directly performed because they do not share a physical connection in the coupling graph). Other constraints that have to be considered are (i) *primitive gate set*—the gates of the circuit to be executed do not always match the native gate set (supported gates) of the quantum chip. For instance, to run the quantum circuit shown in Fig. 1 on the Surface-17 chip (Lao et al. 2022), its CNOT gates would have to be decomposed into X and Y rotations and CZ-gate supported by the device; (ii) *classical control constraints*—shared electronics help to scale up quantum systems but may limit parallelization of quantum operations during circuit execution. The process of accommodating these requirements imposed by the quantum hardware to efficiently execute a quantum algorithm is called *quantum circuit mapping*.

The quantum circuit mapping process consists of the following steps (not mandatory in this order): (1) *Adapting the gate set* of the circuit to the gates supported by the device; (2) *Scheduling* quantum operations (qubit initialization, gates, and measurements) of the circuit to leverage its parallelism and therefore shorten the execution time; (3) *Placing virtual qubits* (of the circuit) *onto physical qubits* (on the actual chip) so that the previously mentioned nearest-neighbor two-qubit-gate constraint is satisfied as much as possible during algorithm execution; and (4) *Routing* or exchanging positions of virtual qubits on the chip such that all qubits that could not initially interact become adjacent and perform their corresponding two-qubit gates (Fig. 1). This is done by inserting additional quantum gates. How routing is performed and which gates are inserted is technology-dependent with various existing methods (SWAPs, Shuttling). Therefore, the resulting after-mapping circuit will in most cases have more gates and a longer execution time than

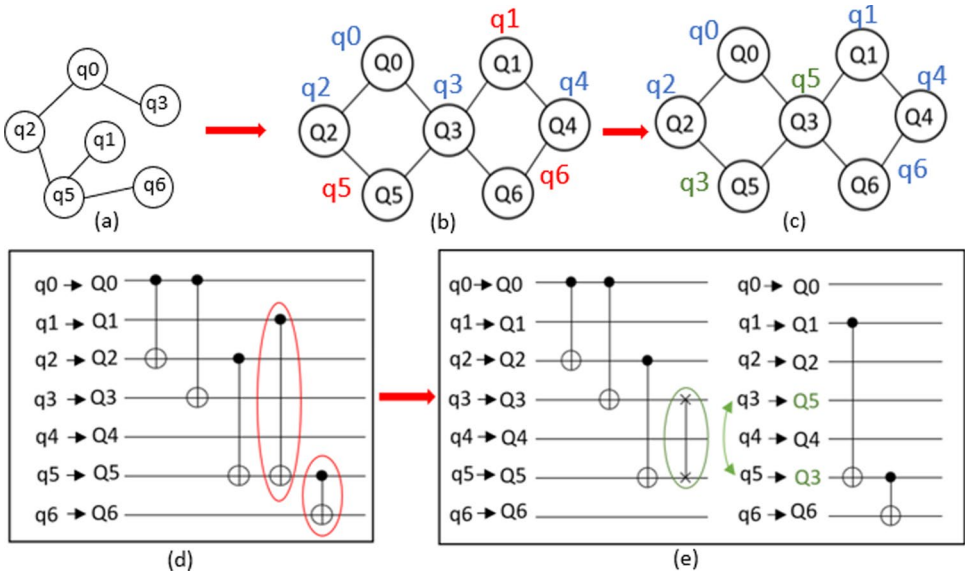

Fig. 1 Running a quantum circuit on a 7-qubit quantum processor. **a** Interaction graph $G_i(V_i, E_i)$ of the circuit shown below; nodes V_i represent virtual qubits, and edges E_i show interactions between qubits (i.e., 2-qubit gates). **b**, **c** The chip's coupling graph $G_c(V_c, E_c)$; nodes V_c represent physical qubits, edges E_c show connections on the chip (i.e., possible two-qubit interactions). **d** Circuit qubits ($qi \in V_i$) are mapped onto physical qubits ($Qi \in V_c$). **e** An extra SWAP gate is required to be able to perform all CNOT gates

originally. Due to the previously mentioned highly-erroneous quantum operations and qubit decoherence, the overhead in terms of number of gates and circuit depth caused by the mapping should be minimal as it ultimately impacts the algorithm fidelity.

Various approaches have been proposed to solve the circuit mapping problem, each using different methods and strategies. Some solutions are optimal (exact), but work in a brute-force style and are thus only suitable for small circuits (Zulehner et al. 2018; Lye et al. 2015; Siraichi et al. 2018). For larger circuits and to allow for scalability, heuristic solutions are a better fit (Li et al. 2019; Lao et al. 2022; Wille et al. 2016; Guerreschi and Park 2018). Some methods proposed by related works include the use of SMT solvers (Murali et al. 2019a; Lye et al. 2015), greedy heuristic (Li et al. 2019; Zulehner et al. 2018; Dousti and Pedram 2012; Bahreini and Mohammadzadeh 2015), and machine learning-based algorithms (Herbert and Sengupta 2018; Venturelli et al. 2018; Pozzi et al. 2020). These solutions all focus on the "routing" part of the mapper. In addition to this, it is possible to deal with the mapping problem by optimizing its other stages like scheduling (Lao et al. 2022; Guerreschi and Park 2018), gate transformation (Pozzi et al. 2020; Guerreschi 2019; Itoko et al. 2020; Tan and Cong 2021), or initial placement (Tannu and Qureshi 2019; Jiang et al. 2021; Li et al. 2021b).

Different metrics are being used to assess the performance of the quantum circuit mapping technique depending on the cost function: some works have the goal of minimizing the number of gates or gate overhead (e.g., number of additional SWAP gates) (Zulehner et al. 2018; Lao et al. 2019a, 2022; Itoko et al. 2020; Tan and Cong 2021; Li et al. 2021b; Bandic et al. 2020; Hillmich et al. 2021), some prioritize low circuit depth or latency (circuit execution time) (Zulehner et al. 2018; Lao et al. 2019a, 2022; Pozzi et al. 2007; Tan and Cong 2021; Bandic et al. 2020), and finally, some focus on the success rate of the circuit (Jiang et al. 2021; Blume-Kohout and Young 2020) and maximizing fidelity (Murali et al. 2019a; Tannu and Qureshi 2019; Tan and Cong 2021) by also considering the different error rates of the quantum device. Note that the overall goal in the current NISQ era is to maximize the fidelity and success rate of quantum circuits, which currently mostly depends on the gate and circuit depth overhead. Figure 2 shows the impact of the number of gates and the gate overhead on the circuit fidelity. However, as shown in Fig. 1, not all the circuits end up with the same decrease in fidelity for the same or similar gate overhead. Note that the circuit fidelity is close to 0% for any circuit with more than 500 gates (Fig. 2a). In addition, a gate overhead of over 200% after mapping leads, in most cases, to a 100% fidelity decrease (Fig. 2b).

These approaches all have in common that they are designed to adapt quantum circuits to the device-specific properties and constraints considering only a reduced set of algorithm properties such as gate and qubit count and two-qubit gate percentage (including qubit interactions). A more in-depth quantum circuit characterization, which for instance could include characteristics of the qubit interaction graph like the number of times each pair of qubits interacts and the distribution of those interactions among the qubits, and of the quantum instruction dependency graph (i.e., graph that represents the dependencies between gates in the circuit and used for scheduling) is still missing. Looking further into interaction graphs is very beneficial for the quantum circuit mapping process, as, like stated before, the most stringent constraint of current quantum hardware is its limited qubit connectivity. Some authors have already pointed out the importance of including application properties (Bandic et al. 2022; Li

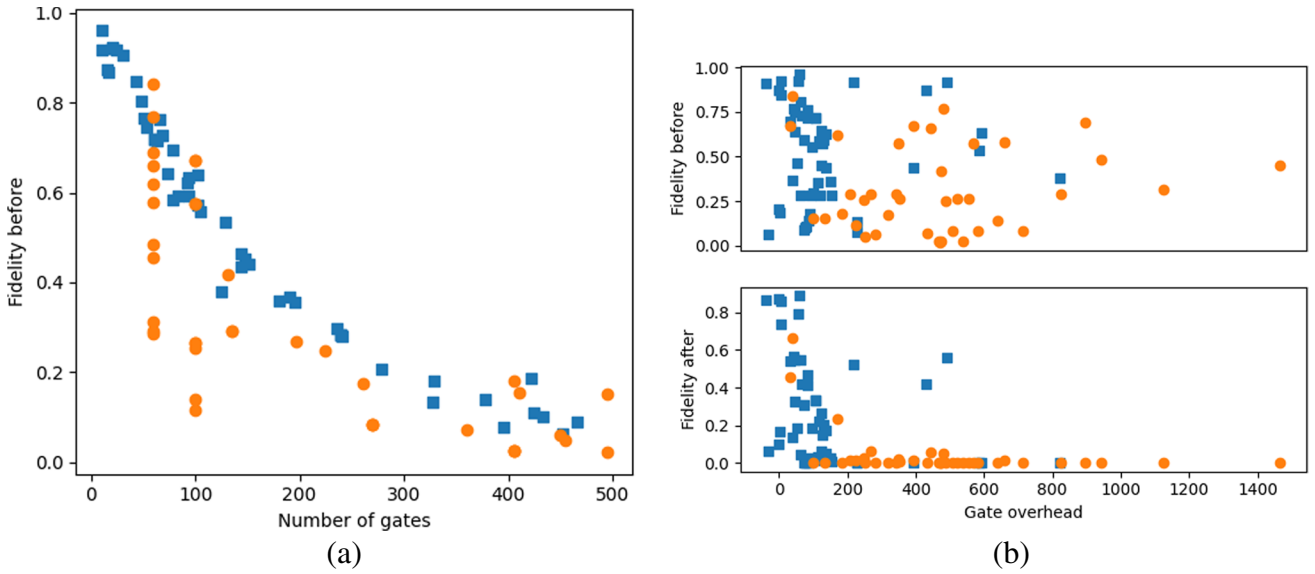

Fig. 2 **a** Circuit fidelity vs. the number of gates. **b** Gate overhead (%) and decrease in fidelity. Synthetically generated circuits are marked with orange circles and real ones (i.e., quantum algorithms and routines) with blue squares. Here, only circuits with up to 500 gates were used

et al. 2020; Lubinski et al. 2021; Mills et al. 2021) and considering the characteristics of the qubit interaction graphs for improving the mapping of quantum circuits (Bandic et al. 2020; Steinberg et al. 2022). Even in classical computing, we notice that different computing resources are necessarily based on what we use the computers for and which applications are executed. For instance, a dedicated GPU can be used for highly parallelizable processes. Likewise, thorough profiling can help to identify which algorithm characteristics are required to execute it successfully on a given device and vice versa. The structural properties of quantum circuits can also help understand why specific algorithms show better success rates than others when being run on a particular processor using a specific mapping technique.

3 Profiling of quantum circuits based on qubit interaction graphs

This section provides an overview of the qubit interaction graph-based benchmark profiling and clustering process, emphasizing why this could be meaningful for improving future quantum circuit mapping techniques.

3.1 On the importance of qubit interaction graphs for quantum circuit mapping

Qubit interaction graph $G(V, E)$ is a graphical representation of the two-qubit gates of a given quantum circuit. It is in general a directed connected graph. Figure 1 shows an example of a quantum circuit (Fig. 1d) along with its interaction graph $G_i(V_i, E_i)$ representation (Fig. 1a). Directed edges E_i represent two-qubit gates, and nodes V_i are the qubits that participate in them. Since the direction of edges in most cases does not influence the execution of the gates, it is sufficient to perceive the interaction graph as undirected for the mapping problem (A quadratic unconstrained binary 2023). If a circuit comprises multiple two-qubit gates between pairs of qubits, it results in a weighted graph (like in Fig. 3), which shows how often each pair of qubits interacts and how those interactions are distributed among qubits.

This additional information can be leveraged to provide more insights into a circuit structure that is otherwise hidden when only considering standard algorithm parameters such as the number of qubits and gates and two-qubit gate percentage. To illustrate this, Fig. 3 shows the interaction graphs of two quantum algorithms, an instance of QAOA and a randomly generated circuit (on the right), which a priori are similar when only characterized in terms of the three common algorithm parameters. What can be noticed is that their qubit interaction graph structure is quite different: the graph of the random circuit is more complex with full connectivity and presents a different distribution of the interactions between qubits, that is, of the weights. This will result in more routing and, therefore, higher overhead, unless we indeed have a fully connected coupling graph of the processor (Section 5).

This shows the importance of quantum circuit structure when developing mapping techniques and the necessity of characterizing the circuits in terms of their qubit interaction graphs. A few works have already pointed out how interaction graph along with

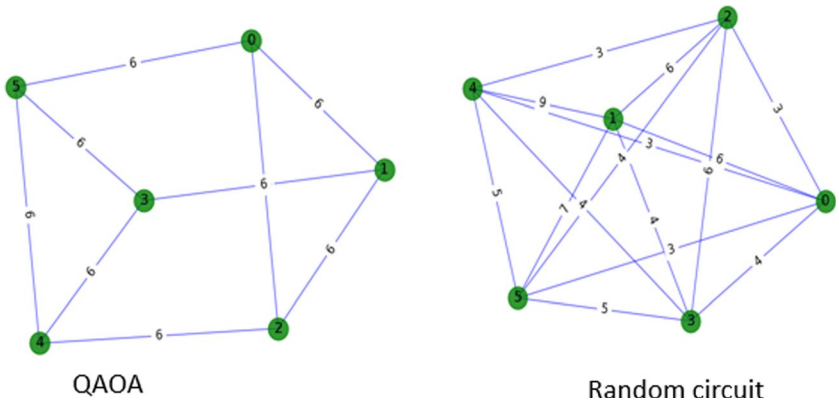

Fig. 3 Interaction graphs of circuits with the same size-related parameters: no. of qubits = 6, no. of gates = 456, two-qubit gate percentage = 0.135

quantum instruction dependency graph can be used as a baseline for designing better mapping techniques (Li et al. 2019; Lao et al. 2022; Baker et al. 2020; Bandic et al. 2023). In those works, gate dependency graphs are used as core information for scheduling optimization and look-ahead techniques, whereas interaction graphs are usually only used for the initial placement of qubits of the routing procedure. Considering that the primary constraint affecting the fidelity of the circuit execution is nearest-neighbor connectivity required for performing two-qubit gates, it would be valuable to know in advance how they are distributed among qubits and not only their quantity.

In this paper, we perform profiling of quantum circuits by focusing on interaction-graph properties and their relation to quantum circuit mapping. To that purpose, we took input from graph theory and analyzed qubit interaction graphs based on metrics described in Hernández and Mieghem (2011) with a focus on those that are relevant to the mapping problem.

Quantum circuit profiling in our work consists of the following steps:

1. Benchmark collection—collecting benchmarks (quantum circuits) from various sources, translating them to the same quantum language, and extracting their interaction graphs (Section 4).
2. Parameter selection and extraction—choosing and extracting graph-theory-based parameters from the qubit interaction graph that are relevant to the mapping of quantum circuits.
3. Benchmark clustering—clustering benchmarks based on their size- and interaction graph-related parameters.

After performing these steps, we compiled the quantum circuits using OpenQL (Khammassi et al. 2021) and analyzed the relation between their performance and extracted parameters, as well as clusters (Sections 4 and 5).

3.2 Parameter selection for quantum algorithm profiling

There exists a vast amount of metrics used for describing graphs, which can be classified into different groups and classes. However, not all of these metrics are relevant to our goal in terms of qubit interaction graph analysis. After thoroughly investigating all metrics described in Hernández and Mieghem (2011), we chose those that are key for the circuit mapping problem. These metrics, when calculated from the qubit interaction graphs, should represent features of quantum circuits that have a correlation with the mapping performance metrics (e.g., number of SWAPS). For instance, the node degree distribution is a relevant metric as it defines the connectivity of the graph (i.e., density of qubit interactions). The more connected the graph, the higher the node degrees. In case there is an all-to-all connected interaction graph, all degrees would be $n-1$, (n being the number of qubits) and that graph would be more challenging to map onto limited connectivity device topologies, which would result in the insertion of a higher number of additional SWAP gates. Table 1 shows the selected metrics subset and how they relate to the quantum circuit mapping process.

We noticed, however, that a large amount of these metrics are correlated, i.e., they scale in the same manner. Therefore, the parameter space was reduced by using a Pearson correlation matrix as shown in Fig. 4 (−1/1 meaning maximally-correlated, 0 meaning not correlated) (Freedman et al. 2007). For instance, note that a minimal node degree of a graph strongly relates to maximal clique and edge connectivity, so in that case, just using one of the parameters, instead of all three, is sufficient. This method allowed us to reduce our previous metric set to average shortest path (average hopcount), maximal and minimal node degree, and adjacency matrix (interaction graph edge-weight distribution) standard deviation. These metrics and the common circuit parameters can be used to cluster quantum circuits. It is expected that quantum algorithms with similar properties should show similar performance when run on specific chips using a given mapping strategy.

Table 1 Selected metrics for characterizing interaction graphs and their relation to the quantum circuit mapping

Metric	Description	Relation to quantum mapping
Hopcount/closeness	Number of links in shortest path between two nodes. / Avg. hopcount (shortest path) between nodes $\mathrm{H_{A\to B}} = \min_{\mathrm{k}\in[1\ldots\mathrm{N}-1]}\left(P_{A\to B}(k)\right)$ hence the hopcount of the path $P_{A\to B}(k) = n_A \to n_2 \to n_3 \to \cdots \to n_{k+1}$, is $H_{A\to B} = k$.	The larger the average hopcount between the nodes, the less connected the graph. A simpler interaction graph is easier to map
Diameter	Maximum hopcount in the graph or the longest shortest path $D = \max_{n_i\in\mathcal{N}}\left(\epsilon_i\right)$	For graphs with the same number of nodes: the larger the diameter, the simpler and less connected the graph. Easier to map
Persistence	Smallest number of links whose removal disconnects the graph	The smaller the persistence, the less connected the graph (also with lower link weights). Makes the quantum circuit easier to map
Betweenness/central point of dominance	Number of shortest paths between nodes that traverse some node or edge. / Maximum betweenness of any point in the graph (ranges from 0 for complete graphs to 1 for star-shaped graphs) The betweenness of node or link k is $\mathrm{B_k} = \frac{\sigma_{\mathrm{ij}}(\mathrm{k})}{\sigma_{\mathrm{ij}}}$, where $\sigma_{ij}(k)$ is the number of shortest paths going between i and j	Values near 0 or 1 are undesirable from the perspective of quantum circuit mapping. 0 reports a graph too much connected, and 1 indicates one qubit being involved in all gates, making the circuit hard to parallelize
Maximal and minimal degree	Maximum and minimum value of degree. Degree d represents the number of nodes to which some node n is connected: $d_i = \sum_{j=1}^{N} a_{ij}$	The lower the minimal and maximal degree, the smaller the interaction of a qubit. This makes the quantum circuit easier to map
Clique/maximal clique	Subset of nodes in which all elements are directly connected. / The largest clique in a graph	The smaller the number of nodes in the largest clique, the smaller the amount of fully connected parts of the graph. Worst case: graph is fully-connected, making quantum circuit mapping more difficult
Clustering coefficient	Measures cliquishness of neighborhood. Between 0 and 1, where 1 is fully-connected graph: $c_i = \frac{y_i}{\binom{d_i}{2}}$	Worst case: Interaction graph is fully-connected, i.e., value of 1
Vertex/edge connectivity (Reliability)	Number of removed nodes/edges that disconnects the graph	The lower the reliability of edges and nodes, the less connected the graph. This consequently leads to an easier qubit mapping process
Adjacency matrix/max and minimal value/weight distribution/mean value/standard deviation/ Variance	An adjacency matrix is a square matrix used for graph representation. It shows which nodes are connected and with how many edges	Trade-off: A large variance means some specific pairs of qubits interact much more than others and there is less additional movement involved

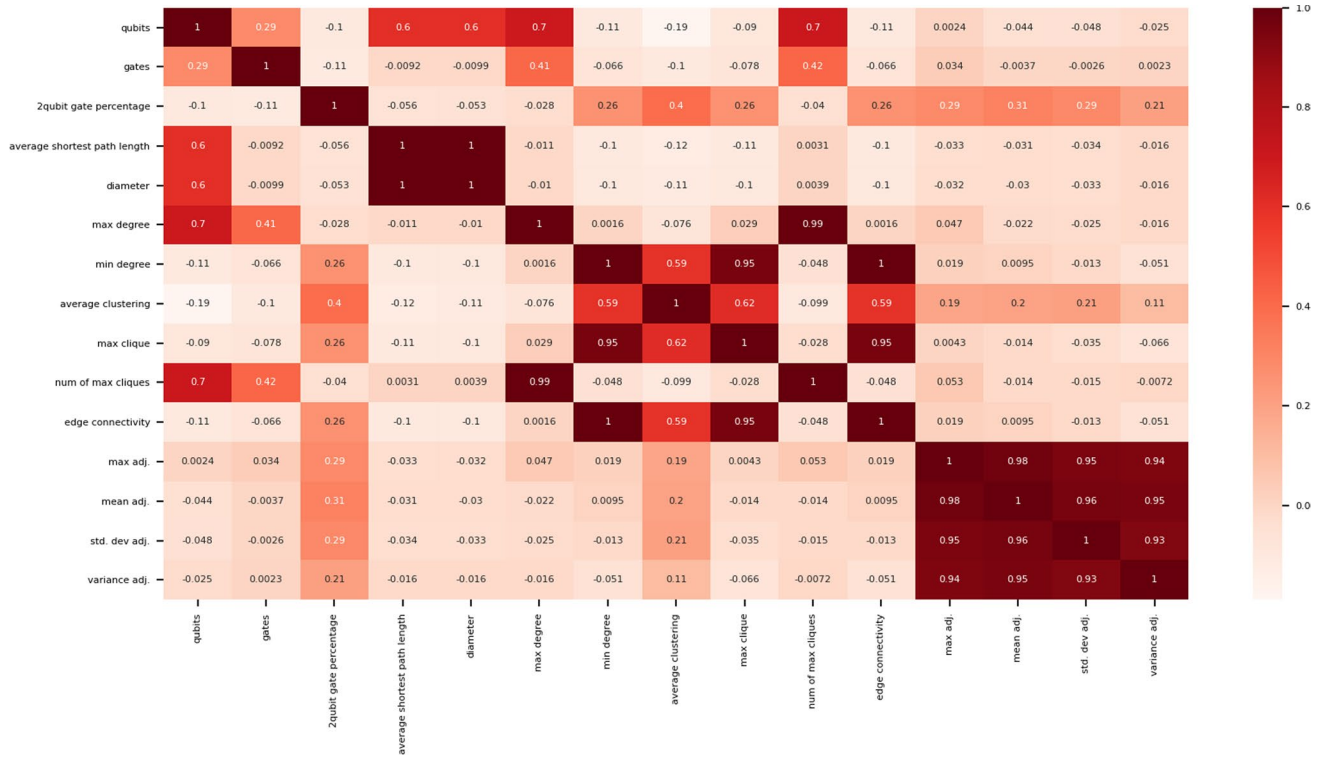

Fig. 4 Heatmap of a Pearson correlation matrix for quantum circuit and interaction graph metrics selected for mapping

3.3 Clustering benchmarks outcomes and evaluation

As mentioned earlier, one of our goals is to find structural similarities among quantum circuits and create some sort of "circuit families," whose elements (quantum circuits) will show similar compilation behavior and require similar hardware resources. The two criteria we have used for clustering benchmarks are properties based on circuit size and qubit interaction graph. Note that we performed a two-step clustering: circuits were first clustered based on size parameters (number of qubits and gates and percentage of two-qubit gates) and then on qubit interaction graph metrics. The reason behind this was to avoid the former to become the most significant criteria of our clustering algorithm. Figure 5 shows the five clusters (different colors) in which a set of 300 selected benchmarks (Section 4) have been divided by using the kmeans (Lloyd 1982) clustering algorithm. Benchmarks are represented as lines in this parallel-coordinates plot. The x-axis contains a list of three different parameters with their values shown in y-axes.

Each of the five size-related clusters can then be further divided into sub-clusters based on previously explained graph parameters: average shortest path length, maximal and minimal degree, and adjacency matrix standard deviation. In this case, we have again selected the kmeans algorithm among several others by evaluating different methods and parameter setups with the silhouette coefficient method (Rousseeuw 1987). In Fig. 6 is an example of when one of the size-parameters-based clusters (cluster 0 from Fig. 5) is divided into sub-clusters based on the interaction graph parameters. It is also pretty straightforward for additional future circuits to be assigned to a specific cluster (size- and interaction graph-based) as each of the clusters and sub-clusters covers the specific range of combinations of parameters (e.g., cluster 4 in Fig. 5 covers benchmarks with less than 25% percentage of two-qubit gates, and cluster 3 in Fig. 6 covers the highest minimal degree values (over 6)). Those circuits should then have similar expected fidelity and gate overhead outcomes as the other circuits in that cluster. How exactly do the mapping performance metrics correlate with our clusters from Fig. 6, and the possible reason for that will be described in the next sections.

4 Experimental setup

This section describes all the necessary elements for performing our experiments: (i) our newly created benchmarks collection (qbench benchmark suite 2021) and a subset used in this paper; (ii) the OpenQL compiler with its Qmap mapper (Lao et al. 2022) and Surface-97 platform, IBM Rochester

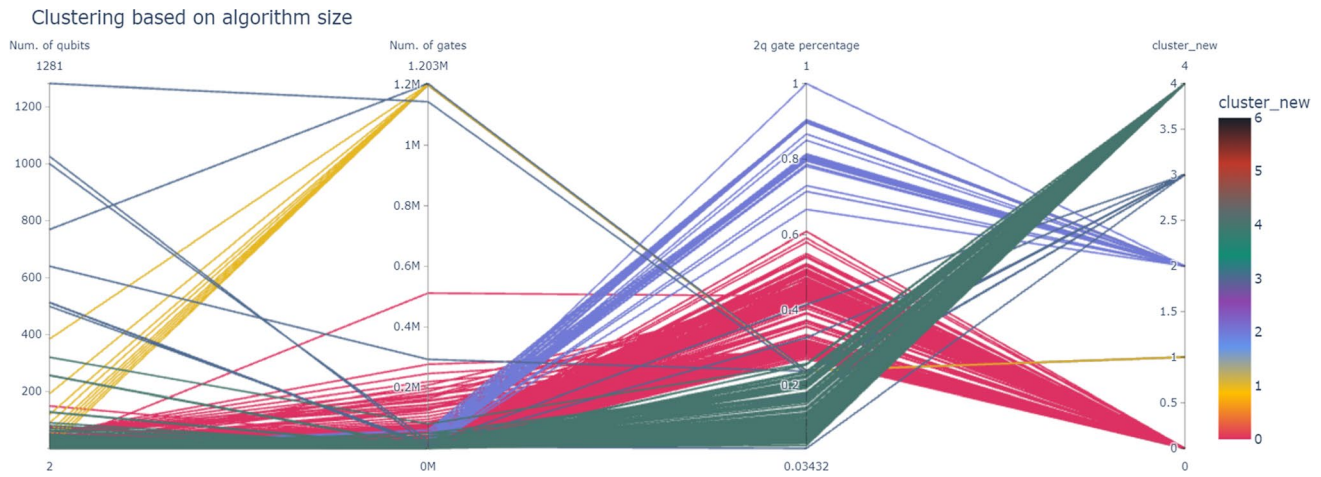

Fig. 5 Clustering of quantum algorithms based on size-related parameters

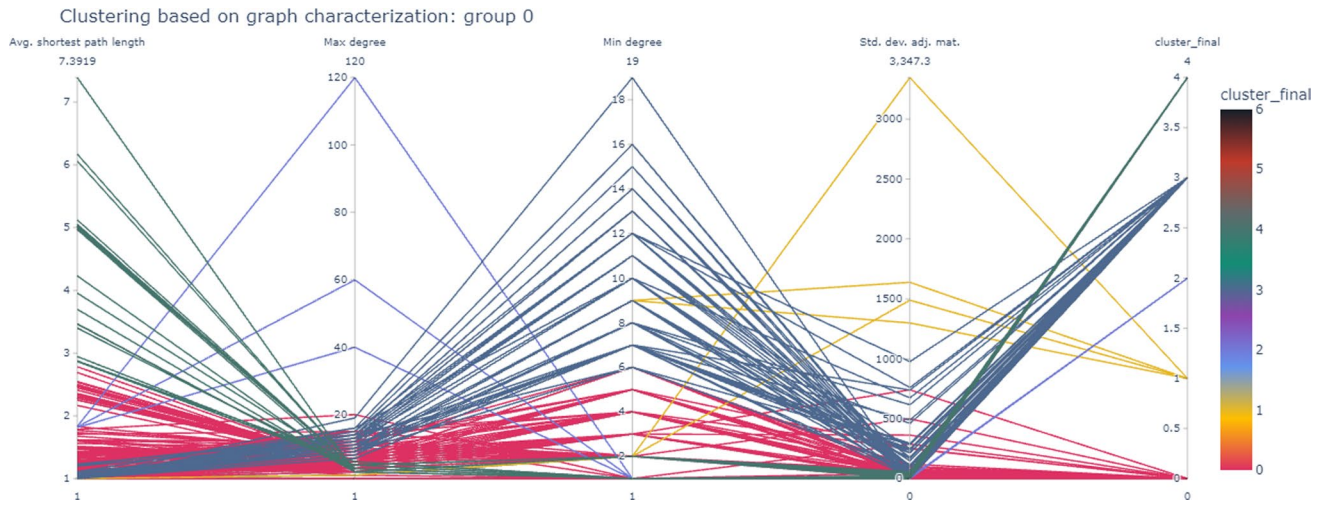

Fig. 6 Sub-clustering of quantum algorithms of cluster 0 (Fig. 5) based on interaction graph parameters

and Aspen-16 configuration files, and (iii) chosen set of metrics for evaluating the performance of the quantum circuit mapping technique.

4.1 Quantum benchmarks collection and classification

The fast development of quantum computing systems dictates the necessity for an all-including and standardized benchmark suite that can serve to test quantum devices as well as compilation techniques and, in general, any part(s) of the full stack. To address this issue, we collected various types of quantum circuits used as benchmarks from a large number of sources (Anis et al. 2021; Li 2019; Li and Krishnamoorthy 2020; UCLA 2020; JKU 2018; Möller and Schalkers 2020; Valada 2020; Microsoft 2020; QuTech n.d; Developers n.d; Smith et al. 2016; Sivarajah et al. 2020; Cross 2018; Last et al. 2020; Wille et al. 2008) written in and translated to different available high- and low-level languages. An overview of our open-source benchmark suite called QBench (qbench benchmark suite 2021) is shown in Fig. 7.

Benchmarks are first divided into two high-level groups: real vs. synthetic quantum circuits. The first ones are then further split into two categories depending on whether they are based on quantum algorithms or are simple reversible arithmetic circuits. In the second group, we can find three different subgroups based on how they are generated. According to Nielsen and Chuang (2002), currently used benchmarks based on *real algorithms* (QFT, search algorithms, application-based algorithms) are the ones that are of the highest importance when measuring the performance of all

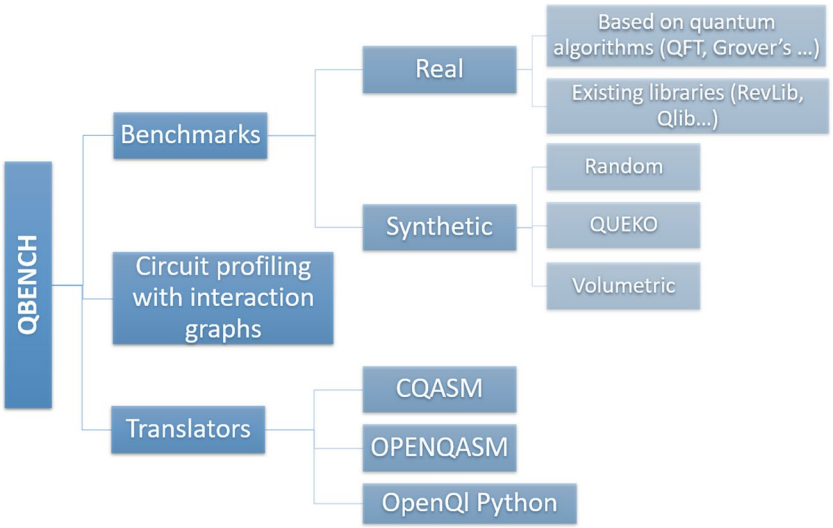

Fig. 7 Overview of our QBench repository

future quantum systems as they are scalable, meaningful, and can show the advantage in quantum systems comparing to classical counterparts (Tomesh et al. 2022). For the current NISQ era, however, there is a need for benchmark libraries like RevLib (Wille et al. 2008) that are within the domain of reversible and quantum circuit design. *Synthetic benchmarks* represent the group of randomly generated quantum circuits, which provide a larger variety in terms of their parameters (e.g., number of qubits, gates, two-qubit gate ratio, circuit depth), and are mainly used to test the performance of quantum devices and explore their computational power to the fullest. For this paper, we mainly focused on (i) randomly generated quantum circuits that are created by uniformly randomly choosing single- and two-qubit gates from a predefined set and then applying them on arbitrarily chosen qubits or qubit pairs in the circuit (Valada 2020); (ii) QUEKO circuits (UCLA 2020), which are designed to be optimal for specific devices (e.g., with optimal depth); and (iii) Quantum volume square circuit (Cross et al. 2019) that is used in general for benchmarking quantum system architectures. A summary of all the real-algorithm-based or synthetic circuits that are part of our benchmark set can be found in qbench benchmark suite (2021).

Benchmarks in our set are also classified based on their size (large-, middle-, and small-scale and parameterized ones) and on the higher- or lower-level language they are written in qbench benchmark suite (2021). *Note that a parameterized (scalable) version* of the circuits allows the creation of new circuits of a desired size, which will be very meaningful for future quantum systems (Tomesh et al. 2022). Furthermore, *different translators* from one quantum language to another, *interaction graphs*, *and interaction graph-based profiling* are also part of this benchmarks suite.

For our experiments, we selected a subset of 300 benchmarks from QBench covering different types (previously described in this section) and qubit number ranges (2–1281 qubits for clustering, 3–54 qubits for mapping experiments).

Note that this benchmark set is to become open source not only for other researchers to use it for the future development of quantum systems, but also for others to participate in its future extensions. There will always be new benchmarks that can be added or quantum languages to translate the current benchmarks to, as we are in the era where we witness a continuous development of new quantum algorithms, compilers, simulators, and programming languages.

4.2 Quantum compiler and targeted quantum devices

To analyze how the previously shown clusters of circuits (Section 3) relate with their after-mapping outcomes, we compiled the 300 selected quantum circuits using as target quantum processor an extended 97-qubit version of the Surface-17 chip (like in Fig. 8a). Surface-17 is a quantum processor with a surface code architecture (Lao et al. 2022), designed to be easily scalable. The device characteristics and all its constraints are included in a configuration file, which is then used as input for the compiler OpenQL (Khammassi et al. 2021). The configuration file of our chosen back-end includes information like error rates, primitive gate set, gate-decomposition rules, and processor qubit topology/connectivity. In addition to this, and in order to compare the performance of the mapper for different groups of circuits, we performed the same experiments for two more quantum processors: the IBM Rochester and the Rigetti 16q-Aspen chips that are shown in Fig. 8b and c, respectively. We selected these device configurations because they are currently commonly used in other research on quantum circuit mapping and provide realistic and different connectivity patterns in their coupling graphs. Note that in our experiments, we do not

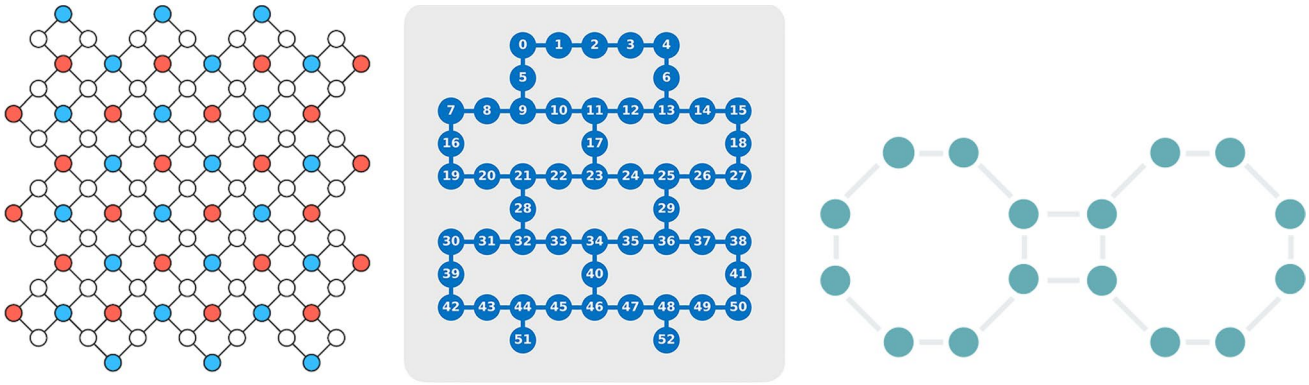

Fig. 8 Topologies of the quantum architectures used for experiments: **a** Surface-97; **b** IBM Rochester, and **c** Rigetti 16-q Aspen. Figures taken from (Overwater et al. 2022; IBM n.d; Rigetti n.d)

execute the quantum circuits on actual devices, but instead, they are just mapped into the different quantum processors; that is, their hardware constraints are considered in the compiling process.

At the core of the OpenQL compiler is its Qmap mapper, which has many options and strategies allowing to create a sort of custom-made compilation technique. The Qmap quantum circuit mapper considers several types of hardware constraints: limited connectivity, primitive gate set, and restrictions derived from classical control electronics. It supports several options for circuit optimization, routing, initial placement as well as scheduling. In addition, it outputs different circuit mapping performance metrics such as the number of additional gates and circuit latency. The routing strategy we opted for was MinExtend (Lao et al. 2022), which, among other features, includes looking back to previously mapped gates and strives to minimally extend the latency of the circuit. It also includes different but common gate transformation and optimization strategies such as gate cancelation or commutation.

4.3 Metrics

The most commonly used metrics for quantum circuit mapper evaluations are the number of added SWAPs, circuit depth, and fidelity/reliability. In our case, we have used the additional gates and extended depth information retrieved from the compiler to calculate the following metrics:

1. **Gate overhead** is calculated as $G_{overhead} = \frac{(G_{after} - G_{before})}{G_{before}}$, where G_{before} and G_{after} represent the number of gates before and after compilation.
2. **Latency overhead** is defined as: $L_{overhead} = \frac{(L_{after} - L_{before})}{L_{before}}$, where L_{before} and L_{after} represent the circuit latency before and after compilation. Latency is calculated as the number of cycles of the circuit, which also considers variations in gate duration, making it different from circuit depth in which all gates are considered to take one time-step.
3. **Circuit fidelity** is defined as the product of error rates of the gates in the circuit. When mapping a circuit, the main goal is to maximize this metric (Murali et al. 2019b; Nishio et al. 2020). We assumed that all one-qubit and two-qubit gates have the same error rates, respectively, for which we used average values of the Starmon-5 chip (QUTECH 2020).
4. **Fidelity decrease** is calculated as $F_{decrease} = \frac{(F_{before} - F_{after})}{F_{before}}$, where F_{before} and F_{after} represent the circuit fidelity before and after compilation.

In the following section, we will discuss the relation of the structural parameters of circuits with the above-stated obtained metrics after mapping them into the Surface-97, IBM Rochester, and Rigetti Aspen-16 devices.

5 Results

5.1 Mapping the circuits to Surface-97 chip architecture

In this section, we evaluate and compare the mapping outcomes of our selected circuits and analyze how the circuit parameters impact the results. Additionally, we compare the performance of different clusters of circuits when using the same mapping technique and processor design (Surface-97).

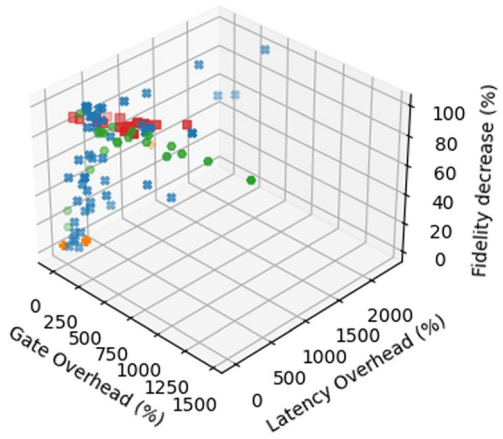

Fig. 9 Mapping performance metrics: gate overhead, latency overhead, and fidelity decrease (all in %) for all groups of benchmarks. We differentiate (i) synthetic circuits: randomly generated circuits (hexagons) and QUEKO circuits (UCLA 2020) (squares) and (ii) real, algorithm-based circuits: simpler arithmetic circuits ("x") and circuits based on quantum algorithms ("+") (e.g., QFT or Grover's algorithm, see Section 4). Only circuits with up to 500 gates are shown

As previously shown in Section 2 (Fig. 2), the gate overhead and circuit fidelity decrease is, on average, higher for our type of synthetic (randomly generated) circuits than for those based on real algorithms, even when they are in the same range of size.[1] Furthermore, as shown in Fig. 9, these two groups of circuits (real and synthetic) are further divided into a total of four differently-structured groups that include randomly generated circuits, QUEKO benchmarks, quantum algorithm-based circuits, and reversible arithmetic circuits. Note in Fig. 9 the difference between these groups in terms of three defined mapping performance metrics. Reversible arithmetic circuits showed on average the lowest gate overhead (∼ 120%) and therefore decrease in fidelity. Randomly generated circuits have on average the best latency overhead (∼ 88%). To give an example, QUEKO circuits show an average gate overhead of ∼ 348%, latency overhead of ∼ 153%, and fidelity decrease of nearly 100%. This all clearly shows the importance of including the structure of the quantum circuit in the mapping process and leads us to using that information to our advantage when choosing an appropriate pair of device and mapping technique.

Subsequent to this, we unveil how size-related parameters: number of qubits, number of gates, and two-qubit gate percentage relate to gate overhead and fidelity decrease, respectively, as shown in Fig. 10. Each point in the graphs represents a benchmark mapped to the Surface-97 processor, and just like in Fig. 9, different groups of benchmarks are shown using different symbols and in the same way. In this case, we only considered circuits with up to 500 gates, as all those above that threshold had negligible fidelity even before mapping. Note that these three mentioned parameters are correlated with the mapping results of the circuits on the chip: the closer the points in graphs are to 0 in all axes simultaneously, the lower the overhead and fidelity decrease. Another point that can be made from these figures is that synthetic circuits (QUEKO and random circuits) perform in this setup, on average, worse than the algorithm-based circuits in terms of after-mapping fidelity and gate overhead (just like in Fig. 9).

We have noticed earlier (Section 2) that the size of a circuit, even though an important feature, is not the only reason why some circuits have lower after-mapping overheads than others. Figure 11 shows how the parameters minimal degree, maximal degree, and average shortest path of the interaction graph influence fidelity and gate overhead of circuits. As observed before, the closer the points in graphs are to 0 in all axes simultaneously, the lower the overhead and fidelity decrease. The graph shows a strong correlation of both the increase in gate overhead (Fig. 11a) and fidelity decrease (Fig. 11b) with the increase in maximal and minimal node degree and average shortest path. 2D cuts of Fig. 11 are shown in Fig. 12 for a better visualization. The following observations can be made: (1) the higher all three circuit parameters, average shortest path, minimal and maximal node degree are simultaneous, the higher the gate overhead (Fig. 12a) and fidelity decrease (Fig. 12b). This means the fidelity is the highest and overhead the lowest if all three circuit parameters are close to 0. (2) Some patterns for circuits belonging to the same group can be observed based on how they are created. For instance, QUEKO circuits (squares) have a high average shortest path (∼ 3), random circuits (hexagons) have a high average node degree (∼ 8), whereas RevLib and algorithm-based circuits (x in graph) have on average low values of the same parameters (∼ 1.5 for average shortest path and ∼ 4.5 for node degree).

In Section 3, quantum circuits have been clustered based on size and interaction graph parameters. In Fig. 13, we can see how the clusters based on interaction graph similarity (example shown in Fig. 6) relate to the mapping performance metrics gate overhead, latency overhead, and fidelity decrease. As mentioned in Section 4, the lower these metrics are, the better the mapping performance. One can notice that circuits belonging to cluster 0 outperform other circuits in terms of gate overhead and fidelity decrease (up to 200% for gate overhead, and an average of ∼ 89% for fidelity decrease), whereas clusters 3 and 4 show the best performance in terms of latency (up to ∼ 150%). What we can further conclude when comparing Figs. 9 and 13a is that clusters mostly consist of

[1] The details on how much the fidelity dropped for each benchmark and how much it differs between the two groups are shown in Fig. 18 in the Appendix.

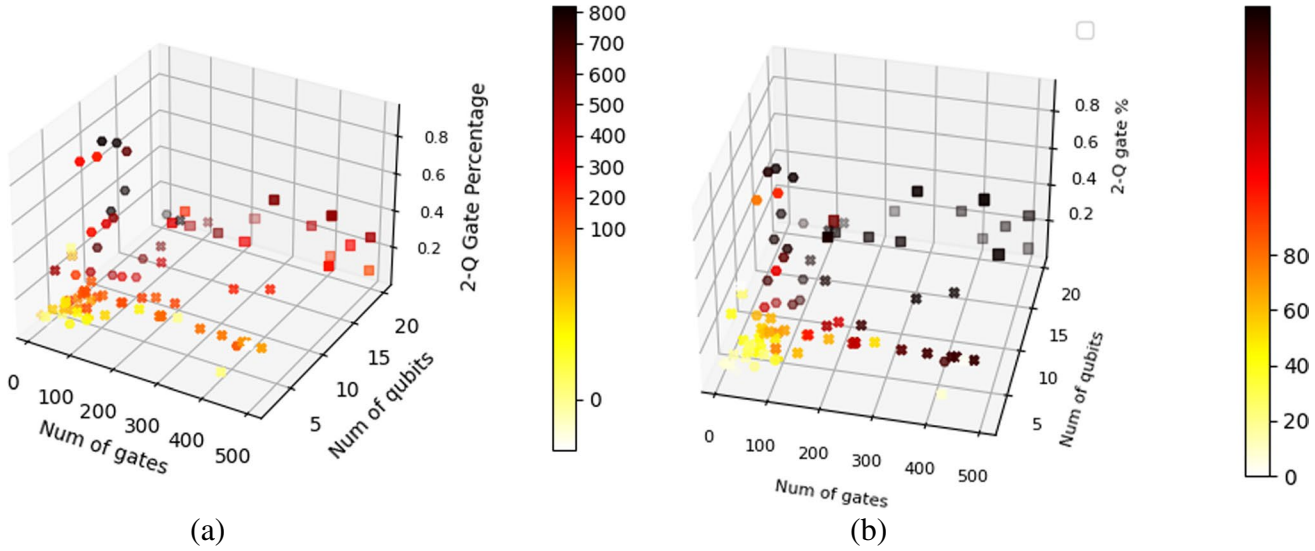

Fig. 10 **a** Gate overhead and **b** fidelity decrease in % (color bar) vs. size-related parameters: number of qubits, number of gates, and two-qubit gate percentage. We differentiate (i) synthetic circuits: randomly generated circuits (hexagons) and QUEKO circuits (UCLA 2020) (squares) and (ii) real, algorithm-based circuits: simpler arithmetic circuits ("x") and circuits based on quantum algorithms ("+")

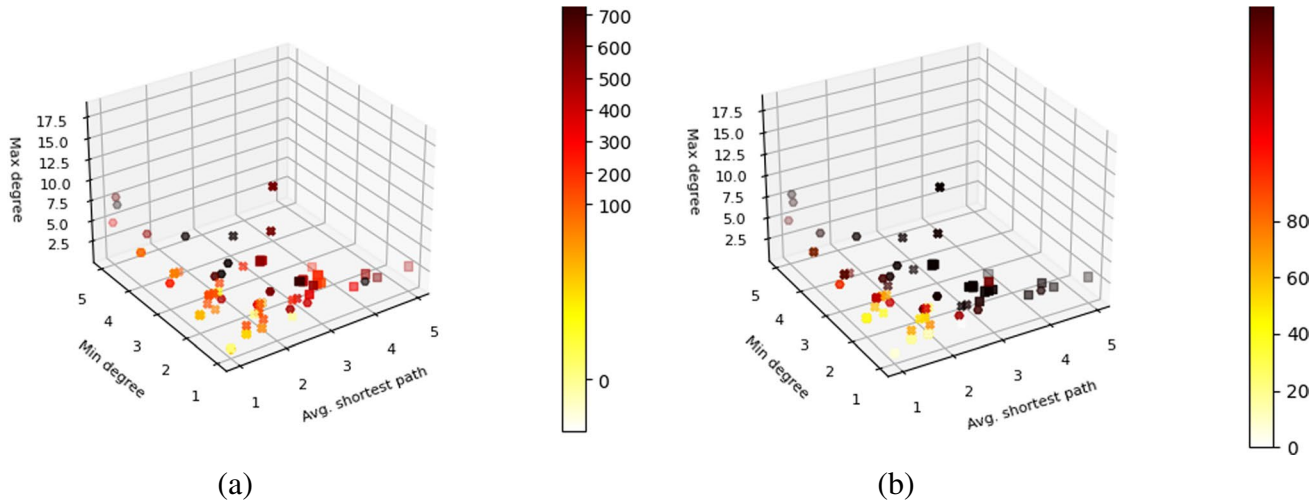

Fig. 11 **a** Gate overhead and **b** fidelity decrease in % (color bar) vs. interaction graph-related parameters: minimal node degree, maximal node degree, and average shortest path

benchmarks of the same type: cluster 0 mostly has real circuits, cluster 3 random ones, and cluster 2 QUEKO circuits. That shows, for instance, that real quantum circuits, especially those from cluster 0, present some pattern in the structure that is easier to map without requiring too many additional gates. Finally, Fig. 13b, which represents a 2D cut of Fig. 13a, clearly shows differences in the range of gate and latency overhead for different clusters. For instance, clusters 3 and 4 have almost constant circuit latency overhead, on average lower than for other clusters, whereas circuits in cluster 0 have low and similar gate overhead. Gate overhead values of cluster 2 scale linearly with latency overhead.

5.2 Quantum chip topology as one rationale behind results

To further look into the reasoning behind the relation between quantum circuit parameters and mapping performance metrics, we first into the device topology. Thus, we

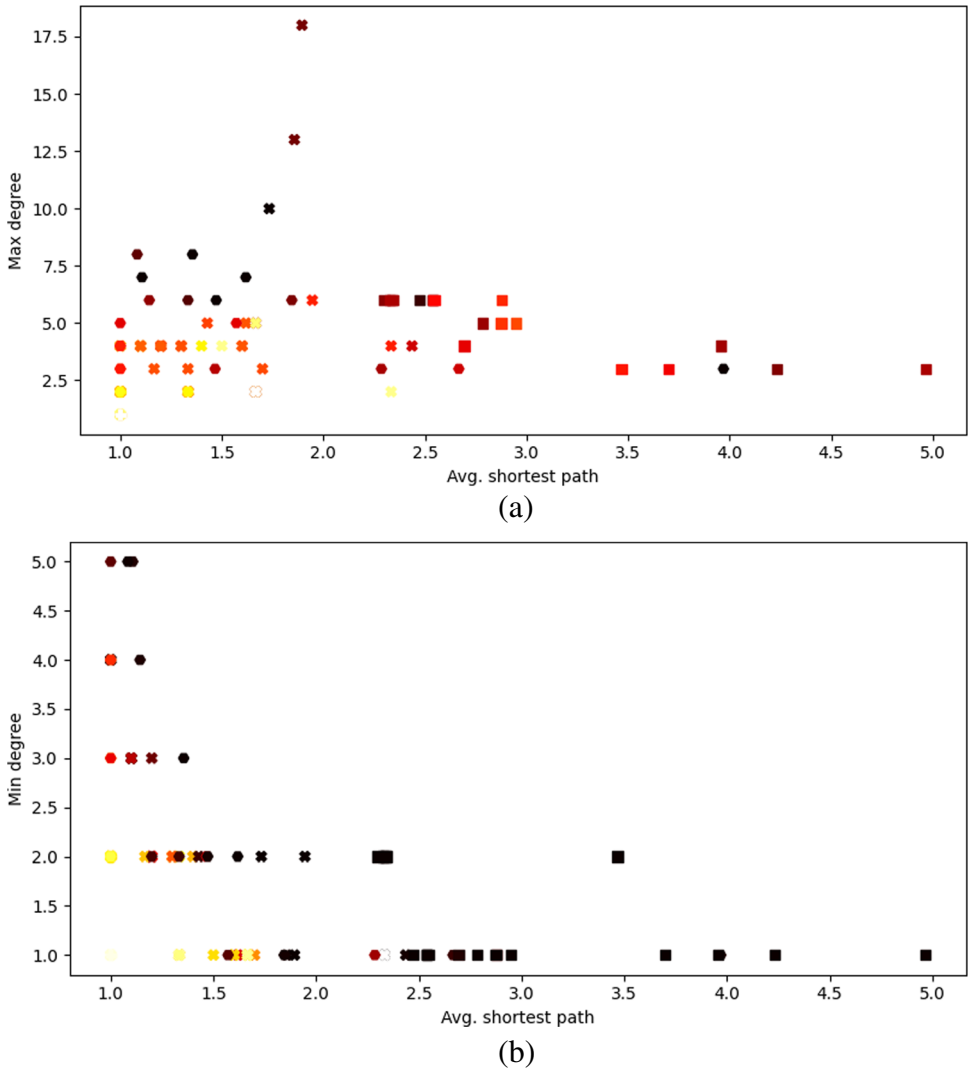

Fig. 12 2D plots of the graphs shown in Fig. 10: **a** interaction graph-based metrics vs. gate overhead (color) and **b** interaction graph-based metrics vs. fidelity decrease (color)

map the same groups of circuits on two additional quantum platforms: the IBM Rochester and Aspen-16 quantum devices (Fig. 8). The outcomes are shown in Figs. 14 and 15. Figure 15 showcases detailed information on how much each structural parameter influences the three mapping performance metrics: gate overhead, latency overhead, and fidelity decrease for all three device configurations. In Fig. 19 (see Appendix), additional details can be found.

From the figures, we can derive the following:

(i) Different groups of benchmarks based on their origin and structure perform differently when executed on different device topologies. The main value of the figures comes from the fact that we can clearly choose a preferred quantum processor topology for each of the benchmark groups (e.g., Surface-97 is preferred for arithmetic reversible circuits, whereas IBM Rochester might be chosen for random ones, as shown in Figs. 9 and 14).

(ii) The impact of structural parameters on the results varies depending on the topology. For example, in the case of the two new topologies, the number of qubits was not as strongly correlated with gate overhead, whereas the degree of the graph played a more significant role. The correlation matrix shown in Fig. 15 highlights that certain parameters are more relevant for specific quantum devices. For the IBM Rochester device, the most important parameter for *gate overhead* is the *two-qubit gate percentage*, whereas for Aspen-16 is *the maximal degree.* The most important parameters for *fidelity decrease* of both devices are the *maximal and minimal degree of the qubit interaction graph*, *the number of qubits and gates*, and *the two-qubit gate percentage*. In contrast, for the Surface-97 device, the most important parameters for *gate overhead* are the *number of qubits* and *the two-qubit gate percentage*, while the most important parameters

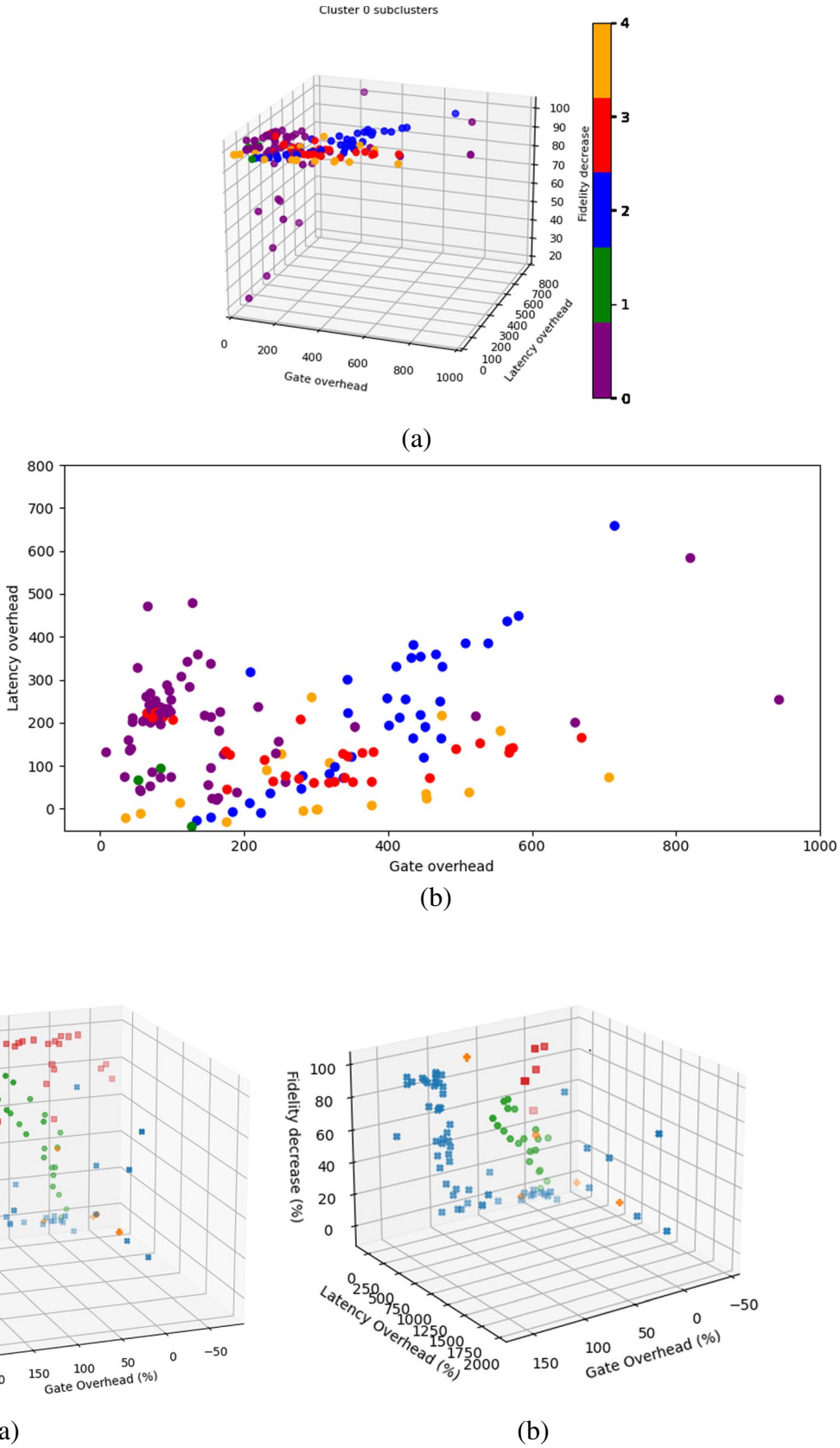

Fig. 13 Relation of clusters of circuits (that are shown in Fig. 6) with the parameters of their interaction graphs: **a** Gate and latency overhead and fidelity decrease and **b** gate and latency overhead

Fig. 14 Results of the circuit compilation when mapping different quantum circuits (Random, QUEKO, Reversible arithmetic circuits—RevLib, Quantum-algorithm based circuits) to the IBM Rochester (**a**) and Aspen-16 (**b**) device topologies using the MinExtend mapper

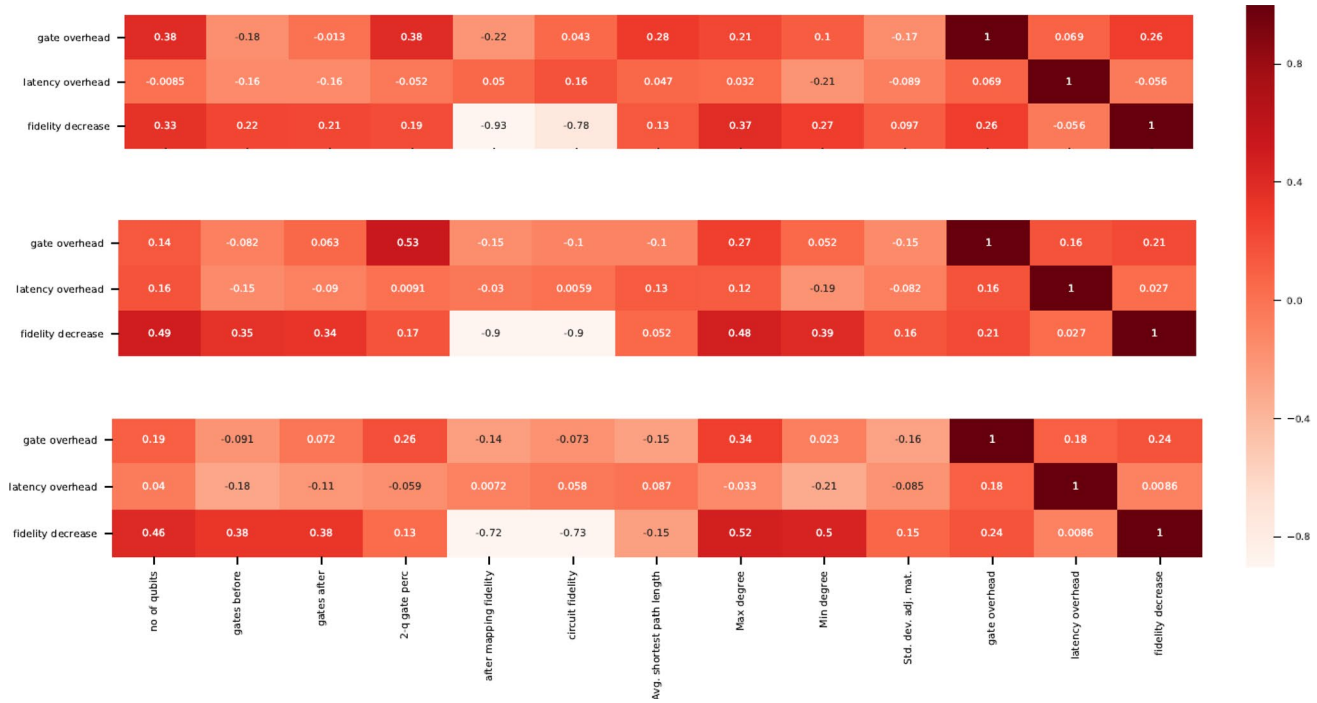

Fig. 15 Correlation matrices showing correlations of mapping performance metrics (gate overhead, latency overhead, fidelity decrease) with extracted metrics of the circuit for the three device configurations: Surface-97, IBM Rochester, and Rigetti Aspen 16-q (top-down)

for *fidelity decrease* are *the number of qubits* and *the maximal degree* of the qubit interaction graph. *Latency overhead* does not appear to be related to these structural parameters, so we will investigate this metric further in future work with other parameters. These observations suggest that

- *Interaction graph parameters* are more relevant for the mapping outcomes of Aspen-16 and IBM Rochester devices than for Surface-97. We can see that the majority of structural parameters are highly correlated with the circuit fidelity decrease. The main reasoning behind this is that these processors have much less connected coupling graphs; in other words, the sparser the coupling graph, the strongest the correlation with the interaction graph parameters. In our case, Aspen-16 has the most restricted coupling graph connectivity, and consequently, its mapping metrics have the highest correlation with interaction graph properties.
- Two-qubit gate percentage, as expected, shows a very high correlation with the gate overhead metric regardless of the device. Other *size-related parameters* (number of qubits and gates) are highly correlated with the fidelity decrease of Aspen-16 and IBM Rochester devices due to again limited connectivity of their coupling graphs as well as smaller device size. On the other hand, the number of qubits only correlates with the gate overhead of Surface-97, which can be attributed to the fact it is a much larger device where we could run much bigger and more complex circuits that would then lead to inevitably long routing paths between at least some of the qubits.

(iii) The two new topologies used for these experiments have quite similar structures (just in different scales of qubit range), and consequently, experiments showcased similar patterns. In future work, we plan to expand our analysis by including additional device topologies.

(iv) In cases where there is no correlation between interaction graph parameters and certain results (such as latency overhead and minimal degree), it suggests that other structural parameters may have played a more significant role. In our future work, we plan to investigate additional parameters such as gate-dependency critical paths and parallelism, which are discussed in Section 6. Similar findings were also observed in a previous study (Tomesh et al. 2022), demonstrating differences between topologies.

To further investigate the benchmark cluster-device relationship, we continued by observing the circuits belonging to the same clusters. We noticed that (Fig. 16) cluster 0 consists of sparse, low-degree graphs and mostly RevLib circuits; cluster 1 is composed of circuits of a very large standard deviation

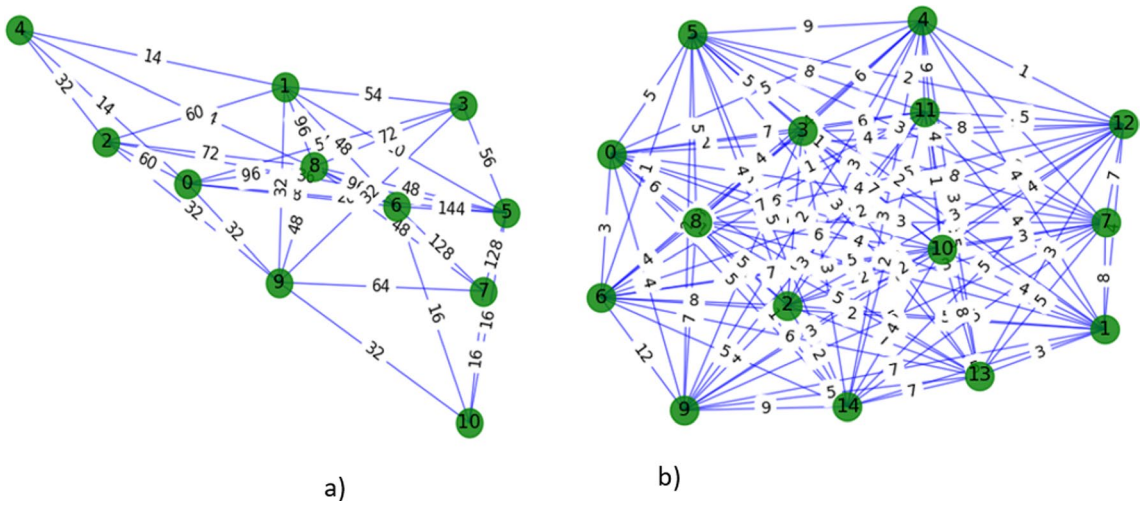

Fig. 16 Qubit interaction graphs for circuits belonging to cluster 0 (**a**) and to cluster 3 (**b**)

of weight distribution; cluster 2 includes grid-like-shaped circuits with mostly QUEKO benchmarks; cluster 3 has the densest graphs with highest node degree, mostly consisting of randomly generated circuits; and finally, cluster 4 contains circuits with large average shortest path, mostly QUEKO circuits based on some existing algorithms (UCLA 2020).

As expected, the sparse graphs of low node degree in cluster 0, which are easier to map to the 2D-grid-resembling qubit topology, required the lowest amount of additional SWAPs, but due to specific, algorithm-based structure could not be well optimized in terms of depth (more difficult to parallelize operations). Cluster 0 is the only cluster with circuits whose fidelity did not drop 100%

On the other hand, the 2D-grid qubit topology, which is the most common state-of-the-art for quantum chips, could not handle well the dense graphs belonging to cluster 3, most of which are random circuits. However, they did perform fine in terms of their latency. What is also interesting, based on these outcomes, is that having, for instance, high average shortest path (like circuits in cluster 4) leads to low latency overhead—as explained in Section 1, which means that the circuit depth was not extended so much. That was as expected, considering that it means that those circuits are much less connected and easier to parallelize.

Furthermore, we have also analyzed the relationship between different circuit clusters and the mapping performance metrics for the experiments performed with the latter two quantum devices, the 53q Rochester and the 16q Aspen processor (see Fig. 17). This time, we clearly see different outcomes. For instance, cluster 0 is not anymore outperforming the others in terms of gate overhead—cluster 4 shows the lowest gate overhead of $\sim$ 12%; cluster 3 fluctuates much more in terms of latency—it goes up to $\sim$ 450% instead of the previous $\sim$ 150%; and cluster 4 is doing way better in terms of fidelity decrease—$\sim$ 90% instead of previous $\sim$ 100%. This is more evident for the Rochester device as the number of circuits included is significantly larger. As 16q-Aspen is on a smaller scale (lower number of qubits) similar to Rochester device in terms of connectivity, we also notice that they have similarly distributed clusters regarding mapping metrics. The data points in Fig. 17b could even be a subset of those in Fig. 17a. This outcome means that other devices with similar topology and higher numbers of qubits would still show similar patterns.

We discuss other possible reasoning behind the results in *Future work* section.

6 Discussion and future work

In Section 3, we mentioned that for completing the description of the structure of quantum circuits, in addition to the interaction graph, we also require gate dependency graph properties. Gate dependency graphs can give insight into how a circuit evolves in time. The critical path within the graph is the most relevant property as it is related to the parallelization degree of the gates, which directly influences the circuit depth. This would also help to explore the oracles or other patterns and repetitions within the circuit. In addition to gate dependency graphs, properties like the amount of parallelism in the circuit (gate density), measurement, and idle gates are influencing the success rate of the circuit a lot (Tomesh et al. 2022).

In addition to this, we must not underestimate the role of the mapping technique in these outcomes. For example, including features like look-ahead/back approaches or optimal initial qubit placement would probably have a stronger influence in terms of mapping results when used on circuits with already predefined,

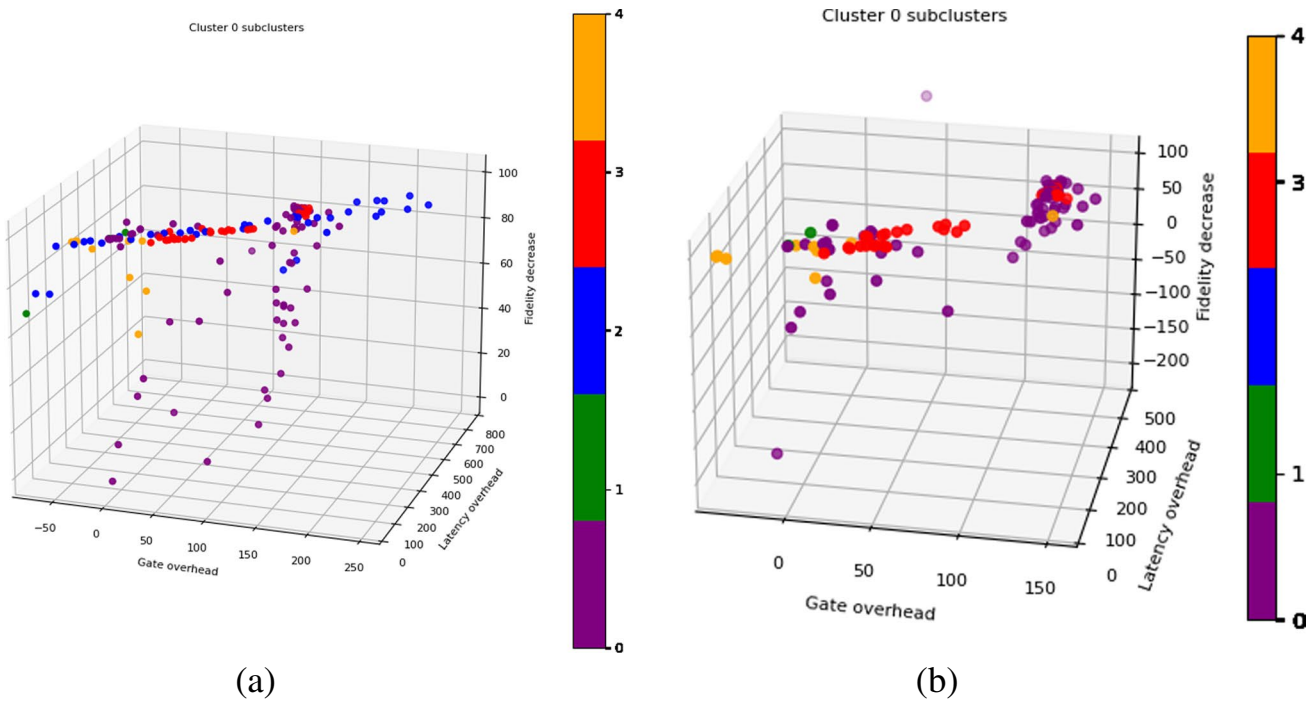

Fig. 17 Quantum circuit mapping metrics vs. clusters of quantum circuits when targeting IBM Rochester (**a**) and Aspen-16 (**b**) topologies

steady, and repetitive structures. To verify this assumption, we plan to compare the performance of quantum circuits when using different types of mappers and optimization properties to investigate the mapper-circuit relationship in contrast to the device-circuit relationship demonstrated in this paper. That could then lead to providing guidelines for designing and optimizing algorithm-aware mapping techniques. To this purpose, structured design space exploration methodologies can be used as pointed out in Bandic et al. (2020).

To conclude, in our future work, we would like to explore further (i) other structural parameters of quantum circuits based on gate dependency graphs such as critical path, the density of gates per layer, and the amount of measurement and idle gates. With this, we will ensure to encapsulate all structural perspectives of quantum circuits when performing benchmark clustering and profiling; (ii) how these observed patterns (with current parameters and additional ones) can help us to predict the mapping performance of new circuit samples assigned to our clusters, without actually running them on the device; (iii) how exactly the interaction graph and coupling graph similarity relate to the mapping result; and (iv) investigate a relationship between interaction graphs and gate dependencies with the chosen mapping technique and to which extend that affects the circuit mapping performance on-chip. For this, we will include more compiling options when performing comparisons. This insight into a circuit structure could help us compare and improve currently existing mapping techniques and enable us to have algorithm-driven mappers and quantum devices.

7 Conclusion

Current quantum devices are still bounded by size and noise and can only handle small and simple quantum algorithms. To execute quantum algorithms, expressed as quantum circuits, on these error-prone and resource-constrained devices, they need to be adapted to overcome those limitations and therefore prevent additional errors. That process is referred to as the mapping of quantum circuits and represents a complex optimization problem that is dependent on both, processor and algorithm properties. In addition to hardware properties, in this paper, we have analyzed how the structure of quantum circuits affects their mapping performance. Our selected quantum circuits were characterized in terms of not only standard parameters, such as the number of qubits and gates and percentage of 2-qubit gates, but also in terms of their interaction graph (i.e., graph theory-based) parameters that include average shortest path, minimal and maximal node degree, and standard deviation of the edge-weight distribution. Our results show a strong correlation between these parameters and circuit mapping metrics: gate overhead, latency overhead, and fidelity decrease increased with the increase in all the chosen parameters. The effect of these parameters varies across different devices and metrics. For example, the degree parameter has a larger impact on fidelity decrease for the IBM Rochester device than for the Surface-97 device. From these findings, we can identify the preferred devices for an

algorithm with specific individual metrics. Furthermore, after clustering the circuits based on mentioned parameters, we found patterns in mapping performance (in terms of the three mentioned metrics) of the circuits belonging to the same cluster, when mapped using the same technique on the same device. For instance, clusters with simpler, low node-degree graphs showed better performance when targeting a 2D-grid topology regarding gate overhead, whereas clusters consisting of complex and dense circuits outperformed others in latency. On the other hand, different performance results were noted when running the same groups of circuits on two other less-connected devices: size parameters like the number of qubits were far less relevant, and synthetic circuits outperformed real ones (which was not the case for Surface-97), and finally, the correlation between clusters of benchmarks and mapping results was unlike to the previously obtained ones. It was also shown that the way circuits were created is very related to their structure and impacts the results (e.g., if they were uniformly randomly generated circuits), as those circuits were in most cases grouped in the same clusters. Finally, we could see how the clusters scale with different mapping metrics. For instance, in one of the clusters, gate overhead scales linearly with latency overhead; in another, gate overhead is constantly within a specific range regardless of the increase in latency.

The proposed method and current findings will help to enhance circuit mapping techniques by including information about the structure of the circuit as well as to have a deeper understanding on the disparity of the observed outcomes when executing different quantum algorithms. In addition, structural parameters of circuits could be used to predict their fidelity decrease and gate and latency overhead for some specific processor and compilation technique without running them on actual devices. This could help to analyze and perform a design space exploration as well as codesign of current compilers, quantum processors, and quantum applications. Ultimately, this process contributes to the development of application-specific quantum systems, where algorithms will be run with higher performances.

Quantum circuits are also used as benchmarks for evaluating mapping and quantum processors. However, the quantum community still does not agree on one benchmark set used, which resulted in an overwhelming amount of sources of quantum circuits. In this work, we have created a soon-to-be open-sourced easy-to-use benchmark collection having benchmarks from various sources cataloged in folders based on how they are implemented (e.g., based on a real algorithm, random, application-based), the language they are written in, and their size. The set also contains various scripts for translating circuits from one language to another, circuit interaction graphs, and profiling results, as described in this paper. We hope this collection will be useful for testing new quantum processors, updated regularly by the research community to keep up with the new technologies, compilers, programming languages, and most importantly applications, and eliminate the over-the-top amount of benchmark sources.

Appendix 1

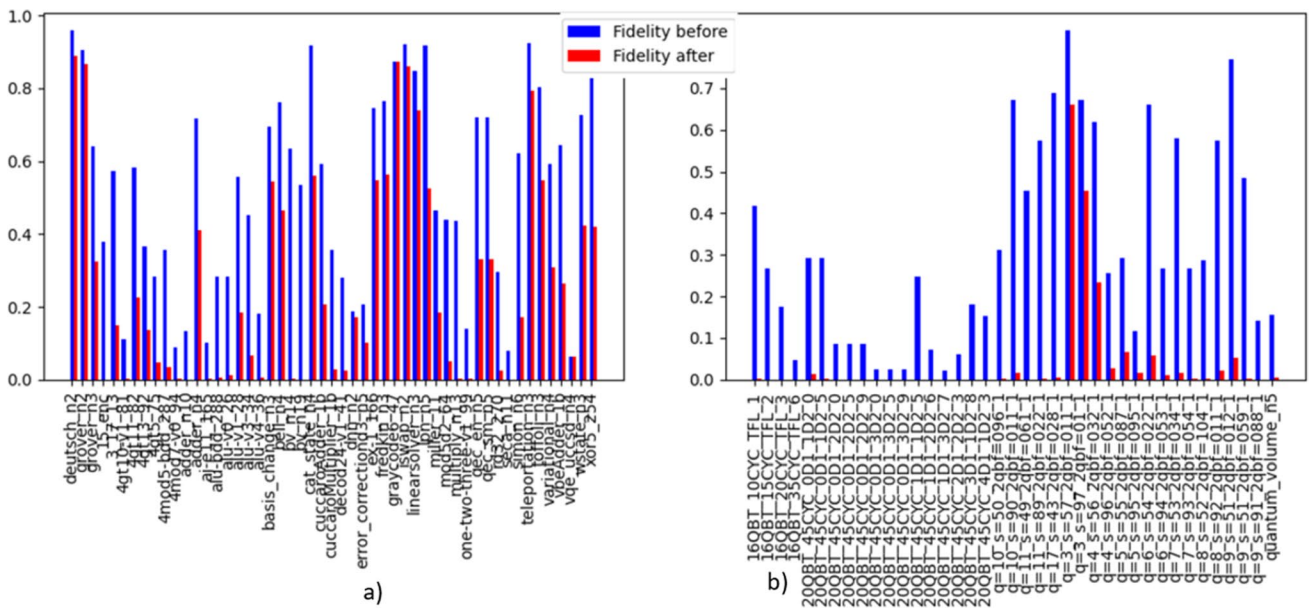

Fig. 18 Fidelity decrease for real circuits (**a**) and synthetically generated ones (**b**). In this figure, we included only the benchmarks whose fidelity was higher than 10% to begin with

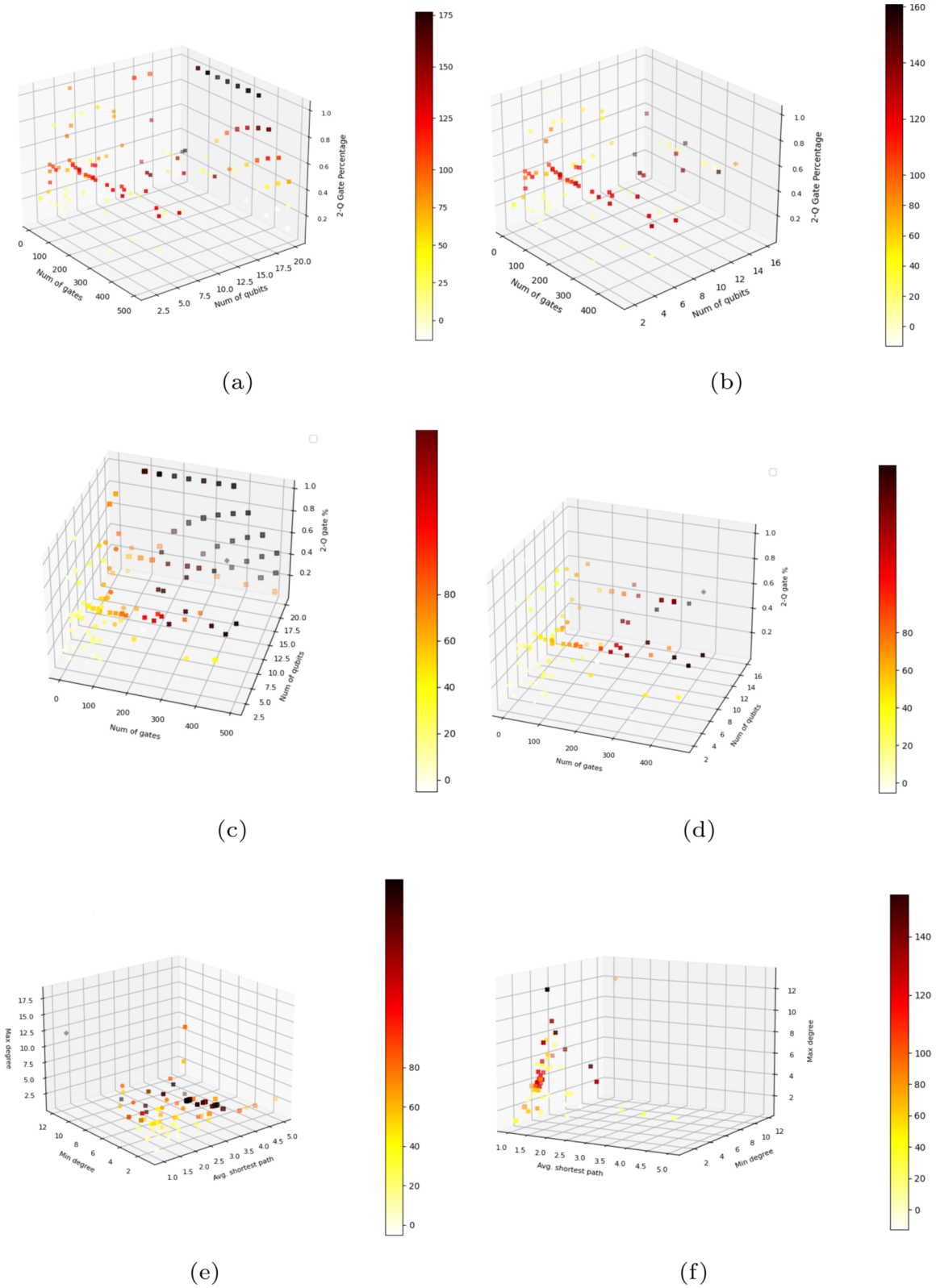

Fig. 19 Results of the circuit compilation when mapping different quantum circuits (Random, QUEKO, Reversible arithmetic circuits, Quantum-algorithm based circuits) to the IBM Rochester (left) and Aspen 16q (right) device topologies using the MinExtend mapper in terms of **a** and **b** gate overhead and size parameters; **c** and **d** fidelity decrease and size parameters; **e** fidelity decrease and IG parameters; and **f** gate overhead and IG parameters

Table 2 Mapping overhead results containing a sample of used quantum circuits with their properties for Surface-97 device

Benchmark	Qubit count	Gate count	2q gate %	Avg. hopcount	Max degree	Min. degree	σ adj. mat	Gate inc	Latency inc	Fidelity dec
15_enc	15	125	0.27	1.73	10	2	0.00	820.00	583.33	99.99
16QBT15CYCTFL2	16	197	0.22	4.23	3	1	1.00	556.35	180.91	100.00
16QBT20CYCTFL3	16	261	0.22	3.70	3	1	1.00	318.77	106.85	99.97
16QBT35CYCTFL6	16	455	0.22	3.47	3	2	2.00	251.87	127.17	100.00
16QBT10CYCTFL1	16	131	0.22	4.97	3	1	0	474.81	216.22	99.77
20QBT45CYC0D11D20	19	135	0.33	2.88	6	1	1.00	208.89	317.39	95.09
20QBT45CYC0D11D25	13	135	0.33	2.69	4	1	1.00	343.70	300.00	99.05
20QBT45CYC0D12D20	20	270	0.33	2.47	6	1	1.00	714.81	658.24	100.00
20QBT45CYC0D12D25	20	270	0.33	2.54	6	1	1.00	581.48	448.35	100.00
20QBT45CYC0D12D29	20	270	0.33	2.78	5	1	1.00	507.78	384.62	100.00
20QBT45CYC0D13D20	20	405	0.33	2.30	6	2	1.41	539.26	384.56	100.00
20QBT45CYC1D12D26	20	360	0.25	2.54	6	1	1.00	435.00	164.09	100.00
20QBT45CYC1D13D27	20	495	0.27	2.34	6	2	1.41	472.53	249.12	100.00
20QBT45CYC2D12D23	20	450	0.20	2.55	6	1	1.00	281.33	76.38	100.00
20QBT45CYC4D11D23	20	495	0.09	2.95	5	1	0.00	134.34	-27.59	99.98
3_17_13	3	102	0.17	1.00	2	2	2.83	128.43	478.38	74.26
4gt10-v1_81	5	424	0.16	1.00	4	4	3.87	78.54	234.46	97.68
4gt11_82	5	79	0.23	1.30	4	2	1.41	116.46	378.57	61.60
4gt13_92	5	190	0.16	1.40	4	2	2.65	39.47	159.70	62.93
4gt5_75	5	239	0.16	1.10	4	3	2.45	64.02	260.24	83.34
4mod5-bdd_287	7	196	0.16	1.62	5	1	1.73	112.76	307.14	90.71
4mod7-v0_94	5	466	0.15	1.10	4	3	4.47	73.39	236.20	98.00
adder_n10	10	328	0.20	2.33	4	1	1.41	229.57	1590.32	99.92
adder_n4	4	63	0.16	1.33	2	2	1.41	84.13	195.65	42.71
aj-e11_165	5	433	0.16	1.00	4	4	4.36	76.91	250.99	97.68
alu-bdd_288	7	240	0.16	1.43	5	2	1.73	153.75	336.90	97.68
alu-v0_26	5	240	0.16	1.20	4	3	3.16	121.25	341.67	95.48
alu-v1_28	5	105	0.17	1.30	4	2	1.41	98.10	224.32	67.00
alu-v3_34	5	148	0.16	1.30	4	2	1.73	124.32	283.02	85.25
alu-v4_36	5	329	0.16	1.10	4	3	3.32	90.88	222.41	96.34
basis_change_n3	3	79	0.13	1.33	2	1	2.65	34.18	74.07	21.84
bell_n4	4	66	0.11	1.67	2	1	1.00	81.82	62.75	38.92
bv_n14	14	94	0.14	1.86	13	1	0.00	593.62	285.37	99.50

Table 2 (continued)

Benchmark	Qubit count	Gate count	2q gate %	Avg. hopcount	Max degree	Min. degree	σ adj. mat	Gate inc	Latency inc	Fidelity dec
cat_state_n4	4	11	0.27	1.67	2	1	0.00	490.91	780.00	38.92
cuccaroAdder_1b	4	83	0.20	1.33	3	1	1.41	130.12	2025.00	65.07
cuccaroMultiplier_1b	6	180	0.18	1.20	4	2	1.00	151.11	958.82	92.57
decod24-v1_41	5	241	0.16	1.20	4	2	2.65	92.95	287.21	91.35
deutsch_n2	2	10	0.10	1.00	1	1	0.00	60.00	180.00	7.39
dnn_n2	2	422	0.10	1.00	1	1	24.25	2.13	85.00	7.89
error_correctiond3_n5	5	278	0.18	1.50	4	1	5.92	0.36	64.91	51.67
ex-1_166	3	53	0.17	1.00	2	2	1.41	45.28	210.53	26.43
fredkin_n3	3	51	0.16	1.00	2	2	1.41	43.14	285.00	26.17
graycode6_47	6	15	0.33	2.33	2	1	0.00	0.00	166.67	0.00
grover_n2	2	30	0.07	1.00	1	1	1.00	-36.67	6.25	4.51
grover_n3	3	102	0.12	1.00	2	2	1.00	45.10	1160.00	49.15
iswap_n2	2	21	0.10	1.00	1	1	1.00	4.76	111.11	6.55
linearsolver_n3	3	43	0.09	1.33	2	1	1.00	4.65	108.70	12.67
lpn_n5	5	24	0.08	1.33	2	1	0.00	216.67	272.73	42.61
miller_11	3	144	0.16	1.00	2	2	3.87	52.08	327.45	60.40
mod5d2_64	5	151	0.17	1.20	4	2	1.73	135.76	358.49	88.35
multiply_n13	11	144	0.18	2.44	4	1	0.00	393.75	2311.11	99.47
one-two–three-v1_99	5	378	0.16	1.10	4	3	3.61	87.83	222.56	97.52
q=10_s=50_2qbf=096_1	10	60	0.80	1.36	8	3	1.00	1125.00	326.23	99.79
q=10_s=90_2qbf=011_1	10	100	0.10	2.67	3	1	0.00	393.00	18.00	97.78
q=11_s=49_2qbf=061_1	11	60	0.52	1.62	7	2	0.00	1466.67	368.85	99.97
q=11_s=89_2qbf=022_1	11	100	0.17	1.84	6	1	0.00	567.00	99.01	99.47
q=17_s=43_2qbf=028_1	17	60	0.20	3.97	3	1	0.00	896.67	120.00	99.27
q=3_s=57_2qbf=011_1	3	60	0.05	1.33	2	1	0.00	38.33	-20.00	21.28
q=3_s=97_2qbf=01_1	3	100	0.10	1.00	2	2	1.41	32.00	-10.00	32.11
q=4_s=56_2qbf=032_1	4	60	0.28	1.00	3	3	1.00	171.67	126.67	62.35
q=4_s=96_2qbf=052_1	4	100	0.54	1.00	3	3	5.20	248.00	156.44	89.90
q=5_s=55_2qbf=087_1	5	60	0.85	1.00	4	4	3.16	268.33	191.80	77.17
q=5_s=95_2qbf=095_1	5	100	0.90	1.00	4	4	4.36	225.00	182.00	87.18
q=6_s=54_2qbf=022_1	6	60	0.23	1.47	3	2	0.00	443.33	125.00	91.43
q=6 s=942qbf=053_1	6	100	0.52	1.00	5	5	2.24	354.00	190.10	96.22
q=7 s=532qbf=0341	7	60	0.33	1.33	6	2	0.00	660.00	200.00	97.48
q=7_s=93_2qbf=054_1	7	100	0.52	1.14	6	4	1.73	522.00	214.85	99.15

Table 2 (continued)

Benchmark	Qubit count	Gate count	2q gate %	Avg. hopcount	Max degree	Min. degree	σ adj. mat	Gate inc	Latency inc	Fidelity dec
q=8_s=52_2qbf=104_1	8	60	0.87	1.11	7	5	1.00	825.00	291.80	98.91
q=8_s=92_2qbf=011_1	8	100	0.17	1.57	5	1	0.00	351.00	48.00	96.44
q=9_s=51_2qbf=012_1	9	60	0.12	2.29	3	1	0.00	480.00	75.00	93.24
q=9_s=51_2qbf=059_1	9	60	0.47	1.47	6	2	0.00	943.33	253.33	99.43
q=9_s=91_2qbf=088_1	9	100	0.81	1.08	8	5	1.41	639.00	259.41	99.71
qec_en_n5	5	61	0.16	1.60	4	1	1.00	106.56	138.46	54.00
qec_sm_n5	5	61	0.16	1.60	4	1	1.00	106.56	138.46	54.00
quantum_volume_n5	5	411	0.12	1.20	4	2	6.93	101.46	114.41	97.23
rd32_270	5	236	0.15	1.10	4	3	2.45	98.73	252.94	92.03
seca_n11	11	396	0.21	1.95	6	2	1.73	228.28	536.23	99.99
simon_n6	5	92	0.15	1.70	3	1	0.00	138.04	900.00	72.30
teleportation_n3	3	20	0.10	1.33	2	1	0.00	55.00	125.00	14.07
toffoli_n3	3	48	0.13	1.00	2	2	1.00	62.50	266.67	31.87
variational_n4	4	94	0.17	1.67	2	1	2.83	72.34	150.00	47.80
vbeAdder_1b	4	74	0.19	1.17	3	2	1.00	122.97	2300.00	58.90
vqe_uccsd_n4	4	452	0.19	1.67	2	1	14.70	-29.87	74.55	2.03
wstate_n3	3	68	0.13	1.00	2	2	1.41	66.18	470.59	41.88
xor5_254	6	17	0.29	1.67	5	1	0.00	429.41	537.50	51.57

Table 3 Mapping overhead results containing a sample of used quantum circuits with their properties for Aspen-16 device

Benchmark	Qubit count	Gate count	2q gate %	Avg. hopcount	Max degree	Min. degree	σ adj. mat	Gate inc	Latency inc	Fidelity dec
15_enc	15	53	0.64	1.73	10	2	0.00	133.96	344.64	79.82
16QBT_10CYC_TFL_1	16	73	0.40	4.97	3	1	0.00	10.96	78.95	55.32
16QBT_15CYC_TFL_2	16	109	0.40	4.23	3	1	1.00	24.77	151.79	77.17
16QBT_20CYC_TFL_3	16	145	0.40	3.70	3	1	1.00	9.66	104.73	81.65
16QBT_35CYC_TFL_6	16	253	0.40	3.47	3	2	2.00	9.88	84.38	94.39
3_17_13	3	36	0.47	1.00	2	2	2.83	111.11	341.03	21.96
4gt10-v1_81	5	148	0.45	1.00	4	4	3.87	127.70	384.00	77.81
4gt11_82	5	27	0.67	1.30	4	2	1.41	96.30	356.67	23.40
4gt13_92	5	66	0.45	1.40	4	2	2.65	127.27	360.87	49.28
4gt5_75	5	83	0.46	1.10	4	3	2.45	130.12	388.24	59.26
4mod5-bdd_287	7	70	0.44	1.62	5	1	1.73	117.14	373.61	49.10
4mod7-v0_94	5	162	0.44	1.10	4	3	4.47	124.07	370.30	78.28
adder_n10	10	294	0.22	2.33	4	1	1.41	17.35	1948.48	80.84
adder_n4	4	23	0.43	1.33	2	2	1.41	82.61	124.00	7.52
aj-e11_165	5	151	0.46	1.00	4	4	4.36	126.49	383.55	78.84
alu-bdd_288	7	84	0.45	1.43	5	2	1.73	121.43	354.12	58.81
alu-v0_26	5	84	0.45	1.20	4	3	3.16	115.48	376.47	53.61
alu-v1_28	5	37	0.49	1.30	4	2	1.41	108.11	342.11	25.31
alu-v3_34	5	52	0.46	1.30	4	2	1.73	125.00	358.18	41.42
alu-v4_36	5	115	0.44	1.10	4	3	3.32	117.39	358.47	62.04
basis_change_n3	3	79	0.13	1.33	2	1	2.65	0.00	46.91	0.00
bell_n4	4	49	0.14	1.67	2	1	1.00	8.16	49.02	2.88
bv_n14	14	41	0.32	1.86	13	1	0.00	160.98	266.67	63.94
cat_state_n4	4	4	0.75	1.67	2	1	0.00	25.00	100.00	0.18
cuccaroAdder_1b	4	73	0.23	1.33	3	1	1.41	6.85	1540.00	20.45
cuccaroMultiplier_1b	6	176	0.18	1.20	4	2	1.00	8.52	852.94	47.40
decod24-v1_41	5	85	0.45	1.20	4	2	2.65	124.71	395.45	57.27
deutsch_n2	2	5	0.20	1.00	1	1	0.00	20.00	14.29	0.18
dnn_n2	2	338	0.12	1.00	1	1	24.25	0.00	97.65	0.00
error_correctiond3_n5	5	114	0.43	1.50	4	1	5.92	60.53	139.13	24.28
ex-1_166	3	19	0.47	1.00	2	2	1.41	105.26	319.05	11.66
fredkin_n3	3	19	0.42	1.00	2	2	1.41	94.74	209.09	7.35
graycode6_47	6	5	1.00	2.33	2	1	0.00	0.00	150.00	0.00
grover_n2	2	16	0.13	1.00	1	1	1.00	-43.75	-23.53	-1.27
grover_n3	3	89	0.13	1.00	2	2	1.00	-19.10	945.45	11.59
iswap_n2	2	9	0.22	1.00	1	1	1.00	44.44	27.27	0.72
linearsolver_n3	3	23	0.17	1.33	2	1	1.00	8.70	56.00	0.36
lpn_n5	5	11	0.18	1.33	2	1	0.00	18.18	50.00	2.53
miller_11	3	50	0.46	1.00	2	2	3.87	120.00	373.58	31.05

Table 3 (continued)

Benchmark	Qubit count	Gate count	2q gate %	Avg. hopcount	Max degree	Min. degree	σ adj. mat	Gate inc	Latency inc	Fidelity dec
mod5d2_64	5	53	0.47	1.20	4	2	1.73	116.98	383.64	44.86
multiply_n13	11	140	0.19	2.44	4	1	0.00	15.71	1263.64	51.37
one-two–three-v1_99	5	132	0.45	1.10	4	3	3.61	115.15	371.11	67.01
q=10_s=50_2qbf=096_1	10	60	0.80	1.36	8	3	1.00	53.33	159.02	53.28
q=10_s=90_2qbf=011_1	10	100	0.10	2.67	3	1	0.00	-11.00	-2.00	32.82
q=11_s=49_2qbf=061_1	11	60	0.52	1.62	7	2	0.00	60.00	186.89	59.35
q=11_s=89_2qbf=022_1	11	100	0.17	1.84	6	1	0.00	28.00	47.52	57.84
q=3_s=97_2qbf=01_1	3	100	0.10	1.00	2	2	1.41	-10.00	-5.00	4.68
q=4_s=56_2qbf=032_1	4	60	0.28	1.00	3	3	1.00	3.33	65.00	14.57
q=4_s=96_2qbf=052_1	4	100	0.54	1.00	3	3	5.20	12.00	133.66	31.15
q=5_s=55_2qbf=087_1	5	60	0.85	1.00	4	4	3.16	18.33	132.79	24.69
q=5_s=95_2qbf=095_1	5	100	0.90	1.00	4	4	4.36	15.00	97.00	30.00
q=6_s=54_2qbf=022_1	6	60	0.23	1.47	3	2	0.00	25.00	63.33	35.89
q=6_s=94_2qbf=053_1	6	100	0.52	1.00	5	5	2.24	42.00	151.49	67.72
q=7_s=53_2qbf=034_1	7	60	0.33	1.33	6	2	0.00	20.00	71.67	38.32
q=7_s=93_2qbf=054_1	7	100	0.52	1.14	6	4	1.73	48.00	181.19	68.06
q=8_s=52_2qbf=104_1	8	60	0.87	1.11	7	5	1.00	38.33	131.15	42.13
q=8_s=92_2qbf=011_1	8	100	0.17	1.57	5	1	0.00	1.00	17.00	34.26
q=9_s=51_2qbf=012_1	9	60	0.12	2.29	3	1	0.00	-5.00	-40.00	11.88
q=9_s=51_2qbf=059_1	9	60	0.47	1.47	6	2	0.00	43.33	106.67	47.29
q=9_s=91_2qbf=088_1	9	100	0.81	1.08	8	5	1.41	40.00	120.79	61.37
qaoa_n16	16	372	0.52	1.20	12	12	0.00	84.68	159.79	99.84
qaoa_n6	6	414	0.13	1.40	3	3	3.00	6.28	49.28	38.52
qec_en_n5	5	25	0.40	1.60	4	1	1.00	64.00	167.86	11.02
qec_sm_n5	5	25	0.40	1.60	4	1	1.00	64.00	167.86	11.02
qft_n15	15	540	0.39	1.00	14	14	0.00	35.74	123.11	98.59
quantum_volume_n5	5	338	0.15	1.20	4	2	6.93	7.10	66.76	42.23
rd32_270	5	84	0.43	1.10	4	3	2.45	117.86	385.06	57.67
seca_n11	11	333	0.25	1.95	6	2	1.73	7.81	510.00	75.03
shor_15	11	4792	0.37	1.38	8	4	36.61	122.58	215.33	100.00
shor_35	15	16529	0.37	1.41	12	5	79.04	120.47	216.06	100.00
simon_n6	5	82	0.17	1.70	3	1	0.00	-9.76	635.29	14.91
teleportation_n3	3	8	0.25	1.33	2	1	0.00	62.50	50.00	0.90
toffoli_n3	3	18	0.33	1.00	2	2	1.00	116.67	245.00	9.86
variational_n4	4	54	0.30	1.67	2	1	2.83	14.81	103.57	1.43
vbeAdder_1b	4	70	0.20	1.17	3	2	1.00	5.71	1928.57	20.31
vqe_uccsd_n4	4	220	0.40	1.67	2	1	14.70	-12.73	110.86	-5.17
wstate_n3	3	49	0.18	1.00	2	2	1.41	12.24	363.16	9.40
xor5_254	6	7	0.71	1.67	5	1	0.00	85.71	400.00	13.30

Table 4 Mapping overhead-based results containing a sample of used quantum circuits with their properties for IBM Rochester device

Benchmark	Qubit count	Gate count	2q gate %	Avg. hopcount	Max degree	Min. degree	σ adj. mat	Gate inc	Latency inc	Fidelity dec
15_enc	15.00	53.00	0.64	1.73	10.00	2.00	0.00	145.28	448.21	82.50
16QBT_10CYC_TFL_1	16.00	73.00	0.40	4.97	3.00	1.00	0.00	13.70	132.89	57.40
16QBT_15CYC_TFL_2	16.00	109.00	0.40	4.23	3.00	1.00	1.00	22.94	127.68	76.06
16QBT_20CYC_TFL_3	16.00	145.00	0.40	3.70	3.00	1.00	1.00	11.72	97.30	82.91
16QBT_35CYC_TFL_6	16.00	253.00	0.40	3.47	3.00	2.00	2.00	20.95	119.14	97.12
20QBT_45CYC_0D1_1D2_0	15.00	45.00	1.00	2.88	6.00	1.00	1.00	102.22	516.67	66.51
20QBT_45CYC_0D1_1D2_5	13.00	45.00	1.00	2.69	4.00	1.00	1.00	113.33	533.33	70.26
20QBT_45CYC_0D1_2D2_0	20.00	90.00	1.00	2.47	6.00	1.00	1.00	173.33	536.56	97.55
20QBT_45CYC_0D1_2D2_5	20.00	90.00	1.00	2.54	6.00	1.00	1.00	175.56	293.55	97.67
20QBT_45CYC_0D1_2D2_9	20.00	90.00	1.00	2.78	5.00	1.00	1.00	163.33	411.83	96.97
20QBT_45CYC_0D1_3D2_0	20.00	135.00	1.00	2.30	6.00	2.00	1.41	202.22	523.19	99.85
20QBT_45CYC_0D1_3D2_5	20.00	135.00	1.00	2.34	6.00	2.00	1.41	220.74	578.26	99.92
20QBT_45CYC_0D1_3D2_9	20.00	135.00	1.00	2.33	6.00	2.00	1.41	180.00	420.29	99.69
20QBT_45CYC_1D1_2D2_6	20.00	180.00	0.50	2.54	6.00	1.00	1.00	63.89	183.61	98.19
20QBT_45CYC_1D1_3D2_7	20.00	225.00	0.60	2.34	6.00	2.00	1.41	105.78	252.63	99.88
20QBT_45CYC_2D1_2D2_3	20.00	270.00	0.33	2.55	6.00	1.00	1.00	11.48	109.16	97.89
20QBT_45CYC_3D1_1D2_8	20.00	315.00	0.14	2.88	5.00	1.00	0.00	-50.79	-2.20	78.94
20QBT_45CYC_4D1_1D2_3	20.00	405.00	0.11	2.95	5.00	1.00	0.00	-59.26	-31.86	80.48
3_17_13	3.00	36.00	0.47	1.00	2.00	2.00	2.83	111.11	341.03	21.96
4gt10-v1_81	5.00	148.00	0.45	1.00	4.00	4.00	3.87	126.35	403.33	76.73
4gt11_82	5.00	27.00	0.67	1.30	4.00	2.00	1.41	96.30	356.67	23.40
4gt13_92	5.00	66.00	0.45	1.40	4.00	2.00	2.65	130.30	401.45	51.63
4gt5_75	5.00	83.00	0.46	1.10	4.00	3.00	2.45	136.14	445.88	63.83
4mod5-bdd_287	7.00	70.00	0.44	1.62	5.00	1.00	1.73	120.00	391.67	51.46
4mod7-v0_94	5.00	162.00	0.44	1.10	4.00	3.00	4.47	126.54	419.39	80.25
adder_n10	10.00	294.00	0.22	2.33	4.00	1.00	1.41	21.09	2027.27	85.25
adder_n4	4.00	23.00	0.43	1.33	2.00	2.00	1.41	82.61	124.00	7.52
aj-e11_165	5.00	151.00	0.46	1.00	4.00	4.00	4.36	123.84	407.24	76.73
alu-bdd_288	7.00	84.00	0.45	1.43	5.00	2.00	1.73	125.00	356.47	61.65
alu-v0_26	5.00	84.00	0.45	1.20	4.00	3.00	3.16	115.48	376.47	53.61
alu-v1_28	5.00	37.00	0.49	1.30	4.00	2.00	1.41	108.11	342.11	25.31
alu-v3_34	5.00	52.00	0.46	1.30	4.00	2.00	1.73	125.00	394.55	41.42
alu-v4_36	5.00	115.00	0.44	1.10	4.00	3.00	3.32	117.39	358.47	62.04
basis_change_n3	3.00	79.00	0.13	1.33	2.00	1.00	2.65	0.00	46.91	0.00
bell_n4	4.00	49.00	0.14	1.67	2.00	1.00	1.00	8.16	49.02	2.88
bv_n14	14.00	41.00	0.32	1.86	13.00	1.00	0.00	146.34	280.95	58.41

Table 4 (continued)

Benchmark	Qubit count	Gate count	2q gate %	Avg. hopcount	Max degree	Min. degree	σ adj. mat	Gate inc	Latency inc	Fidelity dec
bv_n19	19.00	56.00	0.32	1.89	18.00	1.00	0.00	178.57	349.12	79.99
cat_state_n4	4.00	4.00	0.75	1.67	2.00	1.00	0.00	25.00	100.00	0.18
cuccaroAdder_1b	4.00	73.00	0.23	1.33	3.00	1.00	1.41	6.85	1540.00	20.45
cuccaroMultiplier_1b	6.00	176.00	0.18	1.20	4.00	2.00	1.00	8.52	944.12	47.40
decod24-v1_41	5.00	85.00	0.45	1.20	4.00	2.00	2.65	124.71	409.09	57.27
deutsch_n2	2.00	5.00	0.20	1.00	1.00	1.00	0.00	20.00	14.29	0.18
dnn_n2	2.00	338.00	0.12	1.00	1.00	1.00	24.25	0.00	97.65	0.00
error_correctiond3_n5	5.00	114.00	0.43	1.50	4.00	1.00	5.92	60.53	139.13	24.28
ex-1_166	3.00	19.00	0.47	1.00	2.00	2.00	1.41	105.26	319.05	11.66
fredkin_n3	3.00	19.00	0.42	1.00	2.00	2.00	1.41	94.74	209.09	7.35
graycode6_47	6.00	5.00	1.00	2.33	2.00	1.00	0.00	80.00	375.00	9.07
grover_n2	2.00	16.00	0.13	1.00	1.00	1.00	1.00	-43.75	-23.53	-1.27
grover_n3	3.00	89.00	0.13	1.00	2.00	2.00	1.00	-19.10	945.45	11.59
iswap_n2	2.00	9.00	0.22	1.00	1.00	1.00	1.00	44.44	27.27	0.72
linearsolver_n3	3.00	23.00	0.17	1.33	2.00	1.00	1.00	8.70	56.00	0.36
lpn_n5	5.00	11.00	0.18	1.33	2.00	1.00	0.00	18.18	50.00	2.53
miller_11	3.00	50.00	0.46	1.00	2.00	2.00	3.87	120.00	373.58	31.05
mod5d2_64	5.00	53.00	0.47	1.20	4.00	2.00	1.73	115.09	420.00	43.53
multiply_n13	11.00	140.00	0.19	2.44	4.00	1.00	0.00	24.29	1854.55	63.44
one-two–three-v1_99	5.00	132.00	0.45	1.10	4.00	3.00	3.61	115.15	371.11	67.01
q=10_s=50_2qbf=096_1	10.00	60.00	0.80	1.36	8.00	3.00	1.00	76.67	242.62	66.51
q=10_s=90_2qbf=011_1	10.00	100.00	0.10	2.67	3.00	1.00	0.00	-1.00	14.00	47.04
q=11_s=49_2qbf=061_1	11.00	60.00	0.52	1.62	7.00	2.00	0.00	86.67	234.43	72.21
q=11_s=89_2qbf=022_1	11.00	100.00	0.17	1.84	6.00	1.00	0.00	39.00	68.32	67.54
q=17_s=43_2qbf=028_1	17.00	60.00	0.20	3.97	3.00	1.00	0.00	53.33	90.00	55.29
q=3_s=57_2qbf=011_1	3.00	60.00	0.05	1.33	2.00	1.00	0.00	-6.67	-58.33	-0.72
q=3_s=97_2qbf=01_1	3.00	100.00	0.10	1.00	2.00	2.00	1.41	-10.00	-5.00	4.68
q=4_s=56_2qbf=032_1	4.00	60.00	0.28	1.00	3.00	3.00	1.00	3.33	65.00	14.57
q=4_s=96_2qbf=052_1	4.00	100.00	0.54	1.00	3.00	3.00	5.20	12.00	133.66	31.15
q=5_s=55_2qbf=087_1	5.00	60.00	0.85	1.00	4.00	4.00	3.16	15.00	139.34	21.02
q=5_s=95_2qbf=095_1	5.00	100.00	0.90	1.00	4.00	4.00	4.36	13.00	114.00	26.59
q=6_s=54_2qbf=022_1	6.00	60.00	0.23	1.47	3.00	2.00	0.00	26.67	150.00	37.40
q=6_s=94_2qbf=053_1	6.00	100.00	0.52	1.00	5.00	5.00	2.24	31.00	221.78	58.07
q=7_s=53_2qbf=034_1	7.00	60.00	0.33	1.33	6.00	2.00	0.00	23.33	101.67	41.18
q=7_s=93_2qbf=054_1	7.00	100.00	0.52	1.14	6.00	4.00	1.73	78.00	244.55	84.35
q=8_s=52_2qbf=104_1	8.00	60.00	0.87	1.11	7.00	5.00	1.00	51.67	213.11	52.15

Table 4 (continued)

Benchmark	Qubit count	Gate count	2q gate %	Avg. hopcount	Max degree	Min. degree	σ adj. mat	Gate inc	Latency inc	Fidelity dec
q=8_s=92_2qbf=011_1	8.00	100.00	0.17	1.57	5.00	1.00	0.00	20.00	50.00	58.16
q=9_s=51_2qbf=012_1	9.00	60.00	0.12	2.29	3.00	1.00	0.00	10.00	25.00	28.86
q=9_s=51_2qbf=059_1	9.00	60.00	0.47	1.47	6.00	2.00	0.00	68.33	170.00	63.10
q=9_s=91_2qbf=088_1	9.00	100.00	0.81	1.08	8.00	5.00	1.41	69.00	248.51	80.62
qaoa_n16	16.00	372.00	0.52	1.20	12.00	12.00	0.00	90.59	192.23	99.91
qaoa_n6	6.00	414.00	0.13	1.40	3.00	3.00	3.00	6.52	75.48	39.96
qec_en_n5	5.00	25.00	0.40	1.60	4.00	1.00	1.00	64.00	167.86	11.02
qec_sm_n5	5.00	25.00	0.40	1.60	4.00	1.00	1.00	64.00	167.86	11.02
qft_n15	15.00	540.00	0.39	1.00	14.00	14.00	0.00	30.37	112.75	97.19
qft_n20	20.00	970.00	0.39	1.00	19.00	19.00	0.00	42.16	154.79	99.99
quantum_volume_n5	5.00	338.00	0.15	1.20	4.00	2.00	6.93	6.80	91.18	40.84
rd32_270	5.00	84.00	0.43	1.10	4.00	3.00	2.45	117.86	408.05	57.67
seca_n11	11.00	333.00	0.25	1.95	6.00	2.00	1.73	11.71	590.00	81.67
shor_15	11.00	4792.00	0.37	1.38	8.00	4.00	36.61	130.74	267.26	100.00
shor_35	15.00	16,529.00	0.37	1.41	12.00	5.00	79.04	130.82	256.91	100.00
simon_n6	5.00	82.00	0.17	1.70	3.00	1.00	0.00	-9.76	635.29	14.91
teleportation_n3	3.00	8.00	0.25	1.33	2.00	1.00	0.00	62.50	50.00	0.90
toffoli_n3	3.00	18.00	0.33	1.00	2.00	2.00	1.00	116.67	245.00	9.86
variational_n4	4.00	54.00	0.30	1.67	2.00	1.00	2.83	14.81	103.57	1.43
vbeAdder_1b	4.00	70.00	0.20	1.17	3.00	2.00	1.00	5.71	1928.57	20.31
vqe_uccsd_n4	4.00	220.00	0.40	1.67	2.00	1.00	14.70	-12.73	110.86	-5.17
wstate_n3	3.00	49.00	0.18	1.00	2.00	2.00	1.41	12.24	363.16	9.40
xor5_254	6.00	7.00	0.71	1.67	5.00	1.00	0.00	85.71	400.00	13.30

Acknowledgements The authors sincerely appreciate the contribution of Nikiforos Paraskevopoulos in creating the benchmark collection and its documentation, as well as scientific discussions with Prof. Eduard Alarcon (UPC). MB and SF would also like to acknowledge funding from Intel Corporation. This work has been partially supported by the Spanish Ministerio de Ciencia e Innovación and European ERDF under grant PID2021-123627OB-C51 and by the QuantERA grant EQUIP, by the Ministerio de Ciencia e Innovación and Agencia Estatal de Investigación, MCIN/AEI/10.13039/501100011033, and by the European Union "NextGenerationEU/PRTR" (CGA).

Author contribution Author M.B. wrote the main manuscript which was reviewed and approved by authors C.G.A and S.F.

Data availability Once the final reviews are completed, the data will be accessible on (qbench benchmark suite 2021).

Declarations

Competing interests The authors declare no competing interests.

References

Anis MS et al. (2021) Qiskit: an open-source framework for quantum computing. https://doi.org/10.5281/zenodo.2573505

A quadratic unconstrained binary optimization approach for qubit mapping (2023) unpublished, Master Thesis

Bahreini T, Mohammadzadeh N (2015) An MINLP model for scheduling and placement of quantum circuits with a heuristic solution approach. J Emerhing Technol Comput 12(3):29

Baker JM, Duckering C, Hoover A, Chong FT (2020) Time-sliced quantum circuit partitioning for modular architectures. In: Proceedings of the 17th ACM International Conference on Computing Frontiers, pp 98–107

Bandic M, Feld S, Almudever CG (2022) Full-stack quantum computing systems in the nisq era: algorithm-driven and hardware-aware compilation techniques. In: 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, pp 1–6

Bandic M, Prielinger L, Nüßlein J, Ovide A, Rodrigo S, Abadal S, van Someren H, Vardoyan G, Alarcon E, Almudever CG, et al (2023) Mapping quantum circuits to modular architectures with QUBO. arXiv preprint arXiv:2305.06687

Bandic M, Zarein H, Alarcon E, Almudever CG (2020) On structured design space exploration for mapping of quantum algorithms. In: 2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS), IEEE, pp 1–6

Bharti K, Cervera-Lierta A, Kyaw TH, Haug T, Alperin-Lea S, Anand A, Degroote M, Heimonen H, Kottmann JS, Menke T, Mok W-K, Sim S, Kwek L-C, Aspuru-Guzik A (2022) Noisy intermediate-scale quantum algorithms. Rev Mod Phys 94(1):015004. https://doi.org/10.1103/revmodphys.94.015004

Blume-Kohout R, Young KC (2020) A volumetric framework for quantum computer benchmarks. Quantum 4:362. https://doi.org/10.22331/q-2020-11-15-362

Cross A (2018) The IBM Q experience and QISKit open-source quantum computing software. In: APS March Meeting Abstracts, vol. 2018, pp 58–003

Cross AW, Bishop LS, Sheldon S, Nation PD, Gambetta JM (2019) Validating quantum computers using randomized model circuits. Phys Rev A 100(3):032328

Developers, C (n.d.) Cirq. See full list of authors on Github: https://github.com/quantumlib/Cirq/graphs/contributors. https://doi.org/10.5281/zenodo.6599601

Dousti MJ, Pedram M (2012) Minimizing the latency of quantum circuits during mapping to the ion-trap circuit fabric. In: Design Automation and Test in Europe

Freedman D, Pisani R, Purves R (2007) Statistics (international student edition). Pisani, R. Purves, 4th edn. WW Norton & Company, New York

Guerreschi GG (2019) Scheduler of quantum circuits based on dynamical pattern improvement and its application to hardware design. arXiv:1912.00035

Guerreschi GG, Park J (2018) Two-step approach to scheduling quantum circuits. Quantum Sci Technol 3(4):045003

Herbert S, Sengupta A (2018) Using reinforcement learning to find efficient qubit routing policies for deployment in near-term quantum computers. arXiv:1812.11619

Hernández JM, Van Mieghem P (2011) Classification of graph metrics. Delft University of Technology: Mekelweg, The Netherlands, pp 1–20

Hillmich S, Zulehner A, Wille R (2021) Exploiting quantum teleportation in quantum circuit mapping. In: 2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC), IEEE, pp 792–797

IBM (n.d.) https://www.ibm.com/. Accessed 2022–11

Itoko T, Raymond R, Imamichi T, Matsuo A (2020) Optimization of quantum circuit mapping using gate transformation and commutation. Integration 70:43–50

Jiang H, Deng Y, Xu M (2021) Quantum circuit transformation based on subgraph isomorphism and tabu search.arXiv preprint arXiv:2104.05214

JKU: Quantum Circuit Test Set (Zulehner) (2018) JKU

Khammassi N, Ashraf I, Someren J, Nane R, Krol A, Rol MA, Lao L, Bertels K, Almudever CG (2021) Openql: a portable quantum programming framework for quantum accelerators. ACM J Emerg Technol Comput Syst (JETC) 18(1):1–24

Lao L, Browne DE (2021a) 2QAN: A quantum compiler for 2-local qubit Hamiltonian simulation algorithms. arXiv. https://doi.org/10.48550/ARXIV.2108.02099. https://arxiv.org/abs/2108.02099

Lao L, Browne D (2021b) 2qan: a quantum compiler for 2-local qubit hamiltonian simulation algorithms. arXiv preprint arXiv:2108.02099

Lao L, Manzano DM, van Someren H, Ashraf I, Almudever CG (2019b) Mapping of quantum circuits onto NISQ superconducting processors. arXiv preprint arXiv:1908.04226

Lao L, van Someren H, Ashraf I, Almudever CG (2022) Timing and resource-aware mapping of quantum circuits to superconducting processors. IEEE Trans Comput Aided Des Integr Circuits Syst 41(2):359–371. https://doi.org/10.1109/TCAD.2021.3057583

Lao L, van Wee B, Ashraf I, van Someren J, Khammassi N, Bertels K, Almudever C (2019a) Mapping of lattice surgery-based quantum circuits on surface code architectures. Quantum Sci Technol 4:015005

Last T, Samkharadze N, Eendebak P, Versluis R, Xue X, Sammak A, Brousse D, Loh K, Polinder H, Scappucci G, Veldhorst M,

Vandersypen L, Maturová K, Veltin J, Alberts G (2020) Quantum inspire - qutech's platform for co-development and collaboration in quantum computing. In: Sanchez, M., Panning, E. (eds.) Novel Patterning Technologies for Semiconductors, MEMS/NEMS and MOEMS 2020. Proceedings of SPIE - The International Society for Optical Engineering, vol. 11324. SPIE, United States. https://doi.org/10.1117/12.2551853
Li A (2019) OpenQASM Benchmarks Collection. GitHub
Li A, Krishnamoorthy S (2020) Qasmbench: a low-level qasm benchmark suite for nisq evaluation and simulation. arXiv preprint arXiv:2005.13018
Li G, Ding Y, Xie Y (2019) Tackling the qubit mapping problem for NISQ-era quantum devices. In: International Conference on Architectural Support for Programming Languages and Operating Systems, pp 1001–1014
Li G, Ding Y, Xie Y (2020) Towards efficient superconducting quantum processor architecture design. In: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, pp 1031–1045
Li G, Shi Y, Javadi-Abhari A (2021a) Software-hardware co-optimization for computational chemistry on superconducting quantum processors.arXiv preprint arXiv:2105.07127
Li S, Zhou X, Feng Y (2021b) Qubit mapping based on subgraph isomorphism and filtered depth-limited search. IEEE Trans Comput 70(11):1777–1788. https://doi.org/10.1109/TC.2020.3023247
Lloyd S (1982) Least squares quantization in PCM. IEEE Trans Inf Theory 28(2):129–137. https://doi.org/10.1109/TIT.1982.1056489
Lubinski T, Johri S, Varosy P, Coleman J, Zhao L, Necaise J, Baldwin CH, Mayer K, Proctor T (2021) Application-oriented performance benchmarks for quantum computing. arXiv preprint arXiv:2110.03137
Lye A, Wille R, Drechsler R (2015) Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits. In: Asia and South Pacific Design Automation Conference, pp 178–183
Microsoft: Microsoft QDK (2020) https://docs.microsoft.com/en-us/quantum/
Mills D, Sivarajah S, Scholten TL, Duncan R (2021) Application-motivated, holistic benchmarking of a full quantum computing stack. Quantum 5:415. https://doi.org/10.22331/q-2021-03-22-415
Möller M, Schalkers M (2020) A cross-platform programming framework for quantum-accelerated scientific computing. In: Krzhizhanovskaya VV, Závodszky G, Lees MH, Dongarra JJ, Sloot PMA, Brissos S, Teixeira J (eds) Computational science – ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part VI. Springer International Publishing, Cham, pp 451–464. https://doi.org/10.1007/978-3-030-50433-5_35
Murali P, Baker JM, Javadi-Abhari A, Chong FT, Martonosi M (2019a) Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In: International Conference on Architectural Support for Programming Languages and Operating Systems, pp 1015–1029
Murali P, Linke NM, Martonosi M, Abhari AJ, Nguyen NH, Alderete CH (2019b) Full-stack, real-system quantum computer studies: architectural comparisons and design insights. In: 2019b ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA), IEEE, pp 527–540
Nielsen MA, Chuang I (2002) Quantum computation and quantum information. Am J Phys 70(5):558–559. https://doi.org/10.1145/3386162
Nishio S, Pan Y, Satoh T, Amano H, Meter RV (2020) Extracting success from IBM's 20-qubit machines using error-aware compilation. ACM J Emerg Technol Comput Syst 16(3):1–25. https://doi.org/10.1119/1.1463744
Overwater RW, Babaie M, Sebastiano F (2022) Neural-network decoders for quantum error correction using surface codes: a space exploration of the hardware cost-performance tradeoffs. IEEE Transactions on Quantum Engineering 3:1–19
Pozzi MG, Herbert SJ, Sengupta A, Mullins RD (2020) Using reinforcement learning to perform qubit routing in quantum compilers. arXiv preprint arXiv:2007.15957
qbench benchmark suite (2021) https://github.com/QE-Lab/qbench
QuTech: python quantum inspire benchmarks (n.d.) GitHub
QUTECH: Quantum inspire (2020). https://www.quantum-inspire.com
Resch S, Karpuzcu UR (2019) Quantum computing: an overview across the system stack.arXiv preprint arXiv:1905.07240
Rigetti (n.d.) https://medium.com/rigetti/. Accessed 2022–11
Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20:53–65. https://doi.org/10.1016/0377-0427(87)90125-7
Siraichi MY, Santos VFd, Collange S, Pereira FMQ (2018) Qubit allocation. In: International Symposium on Code Generation and Optimization, pp 113–125
Sivarajah S, Dilkes S, Cowtan A, Simmons W, Edgington A, Duncan R (2020) t : a retargetable compiler for NISQ devices. Quantum Sci Technol 6(1):014003. https://doi.org/10.1088/2058-9565/ab8e92
Smith RS, Curtis MJ, Zeng WJ (2016) A practical quantum instruction set architecture
Steinberg MA, Feld S, Almudever CG, Marthaler M, Reiner J-M (2022) Topological-graph dependencies and scaling properties of a heuristic qubit-assignment algorithm. IEEE Trans Quantum Eng 3:1–14. https://doi.org/10.1109/TQE.2022.3160015
Tan B, Cong J (2021) Optimal qubit mapping with simultaneous gate absorption. arXiv preprint arXiv:2109.06445
Tannu SS, Qureshi MK (2019) Not all qubits are created equal: a case for variability-aware policies for NISQ-era quantum computers. In: International Conference on Architectural Support for Programming Languages and Operating Systems, pp 987–999
Tomesh T, Gokhale P, Omole V, Ravi GS, Smith KN, Viszlai J, Wu X-C, Hardavellas N, Martonosi MR, Chong FT (2022) Supermarq: a scalable quantum benchmark suite. In: 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), IEEE, pp 587–603
UCLA: QUEKO benchmark (2020) UCLA
Valada D (2020) OpenQl Random circuits. GitHub
Venturelli D, Do M, O'Gorman B, Frank J, Rieffel E, Booth KE, Nguyen T, Narayan P, Nanda S (2019) Quantum circuit compilation: an emerging application for automated reasoning. In: Proceedings of the Scheduling and Planning Applications Workshop (SPARK)
Venturelli D, Do M, Rieffel E, Frank J (2018) Compiling quantum circuits to realistic hardware architectures using temporal planners. Quantum Sci Technol 3(2):025004
Wille R, Große D, Teuber L, Dueck GW, Drechsler R (2008) Revlib: an online resource for reversible functions and reversible circuits. In: 38th International Symposium on Multiple Valued Logic (ismvl 2008), IEEE, pp 220–225
Wille R, Keszocze O, Walter M, Rohrs P, Chattopadhyay A, Drechsler R (2016) Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits. In: Asia and South Pacific Design Automation Conference, pp 292–297
Zulehner A, Paler A, Wille R (2018) An efficient methodology for mapping quantum circuits to the IBM QX architectures. IEEE Trans Comput-Aided Des Integr Circ Syst