Profiling quantum circuits for their efficient execution on single- and multi-core architectures

Bandic, M.; le Henaff, P.; Ovide, Anabel; Escofet, Pau ; Ben Rached, Sahar ; Rodrigo, Santiago; van Someren, J.; Abadal, Sergi; Feld, S.; More Authors

DOI
10.1088/2058-9565/ada180

Publication date
2025

Document Version
Final published version

Published in
Quantum Science and Technology

Profiling quantum circuits for their efficient execution on single- and multi-core architectures

IOP Publishing Quantum Sci. Technol. 10 (2025) 015060 https://doi.org/10.1088/2058-9565/ada180

Quantum Science and Technology

OPEN ACCESS

RECEIVED
17 July 2024

REVISED
7 October 2024

ACCEPTED FOR PUBLICATION
19 December 2024

PUBLISHED
6 January 2025

PAPER

Profiling quantum circuits for their efficient execution on single- and multi-core architectures

Medina Bandic[1,3,*], Pablo le Henaff[1,3], Anabel Ovide[2], Pau Escofet[4], Sahar Ben Rached[4], Santiago Rodrigo[4], Hans van Someren[1], Sergi Abadal[4], Eduard Alarcón[4], Carmen G Almudever[2] and Sebastian Feld[1,3]

[1] Delft University of Technology, Delft, The Netherlands
[2] Universitat Politécnica de Valéncia, Valencia, Spain
[3] QuTech, Delft, The Netherlands
[4] Universitat Politècnica de Catalunya, Barcelona, Spain
[*] Author to whom any correspondence should be addressed.

E-mail: m.bandic@tudelft.nl

Keywords: quantum circuit mapping, multi-core quantum computers, modular architectures, quantum communication, interaction graphs, quantum benchmarks, gate-dependency graphs

Abstract

Application-specific quantum computers offer the most efficient means to tackle problems intractable by classical computers. Realizing these architectures necessitates a deep understanding of quantum circuit properties and their relationship to execution outcomes on quantum devices. Our study aims to perform for the first time a rigorous examination of quantum circuits by introducing graph theory-based metrics extracted from their qubit interaction graph and gate dependency graph (GDG) alongside conventional parameters describing the circuit itself. This methodology facilitates a comprehensive analysis and clustering of quantum circuits. Furthermore, it uncovers a connection between parameters rooted in both qubit interaction and GDGs, and the performance metrics for quantum circuit mapping, across a range of established quantum device and mapping configurations. Among the various device configurations, we particularly emphasize modular (i.e. multi-core) quantum computing architectures due to their high potential as a viable solution for quantum device scalability. This thorough analysis will help us to: i) identify key attributes of quantum circuits that affect the quantum circuit mapping performance metrics; ii) predict the performance on a specific chip for similar circuit structures; iii) determine preferable combinations of mapping techniques and hardware setups for specific circuits; and iv) define representative benchmark sets by clustering similarly structured circuits.

1. Introduction

In recent decades, the realm of quantum technology has witnessed remarkable progress, holding the potential to tackle problems that were once deemed insurmountable using classical means. Although these advancements are impressive, we are still in the early stages of understanding its full potential. The current generation of quantum devices, referred to as noisy intermediate-scale quantum (NISQ) devices [1], present severe limitations due to their size and susceptibility to noise. As a result, they are currently adept at handling only simple and modestly-sized circuits (i.e. executable descriptions of algorithms). These devices also face other hurdles, including restricted qubit connectivity, a narrow set of supported operations, and challenges pertaining to classical control resources [2, 3]. These collective constraints make the successful execution of a quantum circuit on such processors an intricate endeavor. Furthermore, NISQ devices often adhere to a 'one-size-fits-all' approach, which can lead to architectures ill-suited for certain quantum algorithms, resulting in lower success rates and fidelity. This is already well showcased in classical computing, where devices are often tailored for the purpose of usage (e.g. GPUs for gaming).

Most current quantum computers operate as single-processor devices, containing all qubits on a single chip. Scaling these designs proves challenging due to issues like crosstalk and limitations in control

electronics [4]. An alternative approach, akin to classical computing, involves multi-processor (or multi-core (MC)) architectures, which are proposed by various quantum processor manufacturers [5–10]. These new designs facilitate distributed quantum computing that enables the execution of large algorithms across multiple cores to accommodate more qubits than a single processor can handle, and represent a feasible avenue for achieving scalability in quantum computing.

For both NISQ and MC architectures, the quantum circuit mapping process [11, 12] is necessary to efficiently run the circuits and maximize the usage of hardware resources. Quantum circuit mapping essentially represents adapting quantum circuits to quantum devices to adhere to all hardware constraints, forming a vital component of a full-stack quantum computing system [13].

Several studies emphasize the importance of considering a broader range of circuit features during the process of mapping [14–17]. A comprehensive profiling or characterization of quantum circuits offers several advantages, including gaining a better understanding of why certain algorithms achieve higher fidelity with specific processors and mapping techniques [18]. Additionally, it enables the classification and prediction of the performance of similar circuits based on the circuits' attributes (like in [19]), all without the need for actual hardware execution. Moreover, this approach facilitates the development of mapping techniques and overall quantum systems customized for specific applications, respecting both the requirements (i.e. characteristics) of those particular circuits and the limitations of the hardware [13, 20, 21]; this consequently leads to an improvement in quality and execution time of solving currently intractable problems. It is important to note that this exhaustive characterization of quantum circuits is not only the key for formulating meaningful and representative sets of quantum benchmarks that evaluate quantum circuit mapping techniques and entire quantum computing systems [22, 23], but also for establishing a suite of algorithm-level metrics to measure system performance [24].

The state-of-the-art characterization of quantum circuits proposed in [18] extends beyond conventional metrics such as qubit and gate counts. In addition to these standard attributes, their approach includes an examination of qubit interaction graphs (IG), drawing insights from graph theory and machine learning to clarify the circuit's two-qubit gate connections (qubit interactions). They performed an analysis of how these circuit parameters affect the performance of a specific quantum circuit technique when considering three different single-core (SC) devices.

In this paper, we extend the results of [18] by additionally encompassing metrics extracted from GDGs (which portray inter-dependencies among gates within the circuit), as well as parameters that describe the *density of the circuit* and its *repetitive oracles*. Furthermore, we not only consider parameters that are relevant for SC processor devices but also identify those that are of special interest for the next generation of MC architectures. Within our approach, we also experiment with *diverse quantum circuit mapping configurations* (four for SC and three for MC quantum architectures). This intricate exploration allows us to: i) discern the most influential quantum circuit attributes that impact the performance of circuit mapping; ii) predict the mapping performance of similarly structured quantum circuits on a specific chip; iii) identify the most adequate combination of mapping technique and quantum hardware for a given quantum circuit or set of circuits; and iv) define a representative benchmark set [24], by specifying a finite amount of groups of similarly-structured circuits. This thorough analysis, therefore, holds the potential to contribute to the future co-design of compilation methodologies driven by algorithms and the evolution of quantum hardware.

In summary, the main contributions of this paper are:

(i) performing the most comprehensive profiling of quantum circuits by extracting: (a) standard parameters (i.e. number of qubits and gates, two-qubit gate percentage and depth), metrics from the (b) IG (e.g. average node degree), and (c) GDG (e.g. critical path length), (d) gate density related parameters (e.g. amount of idling) and (e) characteristics related to repetitive sub-circuits. We believe that this list of parameters encompasses all relevant aspects of a quantum circuit. It helps us gain insights into why certain circuits excel or falter on particular architectures, potentially revealing correlations between circuit structure and performance across different quantum setups. Leveraging these parameters, we can adapt existing or craft new full-stack quantum systems with higher precision.

(ii) Identifying, for the first time, circuit parameters that are key for scalable modular quantum computing architectures. Those architectures demand a unique parameter set due to their intricate quantum circuit mapping requirements, as elaborated in detail in section 2.

(iii) Finding a correlation between extracted circuit features and compilation performance across various mapper-device combinations for SC and MC architectures (totaling in seven mappers and six devices). Utilizing the Pearson correlation score [25], we rank parameters from most positively correlated to most negatively correlated for each combination. This analysis shows the significance of selecting suitable device topologies and mapping techniques for quantum circuits with specific structural parameters. It also highlights the key circuit parameters crucial for designing application-aware

quantum systems. Identifying these influential circuit parameters marks the initial stride towards crafting such systems.

(iv) Clustering similarly structured circuits and determining the most effective mapper-device setups for them. We illustrate how these clusters also correspond to circuit origin groups found in qbench (i.e. random, QUEKO, real algorithms) [22]. This discovery of distinct groups of quantum circuits allows us to establish representative sets of quantum benchmarks without the need for an exhaustive list, which also facilitates the design of application-specific quantum systems tailored to each group, rather than a separate system for each benchmark.

The paper is organized as follows: section 2 introduces SC and MC computation and quantum circuit mapping, as well as the previous work on quantum circuit characterization. Section 3 showcases our novel circuit parameters and profiling process in this work, done for SC and MC quantum computation separately. In section 4, we dive into methodology and performance metrics. We show and discuss results in section 5 and finally conclude our work in section 6.

2. Background and previous research

2.1. MC quantum computation

Just like for classical computing, modular architectures are envisioned as a solution for the scalability of quantum devices and they are expected to accommodate thousands to millions of qubits within a single system. For dealing with problems regarding crosstalk, classical control electronics, and wiring complexity, qubits are distributed over multiple cores or processors. This strategy capitalizes on quantum parallelization while addressing qubit control demands and improving qubit isolation [26]. Nevertheless, constructing MC processors introduces new challenges, mainly due to latency-prone quantum communications that introduce inefficiencies. Quantum state transfer across chip-scale networks represents an alternative to conventional communication technologies, which prove impractical for modular architectures. Communication latencies heighten the risk of data loss during qubit transmission due to state decoherence [2]. Several strategies have emerged to create inter-core communication networks for modular quantum computing architectures, accommodating varying technology platforms. These include quantum links for superconducting chips [5], ion-shuttling for ion-trapped quantum computers [27], and photonic networks [28]. Modularity requirements extend beyond the qubit layer, imposing constraints on networking, control, and compilation layers, necessitating a comprehensive software–hardware stack [26].

2.2. The quantum circuit mapping problem

As previously stated, running a quantum circuit on any device is not a straightforward task. Given the constraints of current NISQ devices (e.g. highly error-prone and limited size and qubit connectivity), the execution of quantum circuits often requires making some modifications within the circuit. This process, referred to as quantum circuit mapping, is pivotal in adapting algorithms to quantum devices, and it forms a vital component of a full-stack quantum computing system [29]. Limited connectivity is the main challenge addressed by quantum circuit mapping because, during the execution of circuits, all interacting logical/virtual qubits (of the circuit) must be adjacent. This problem is solved by finding the most suitable allocation of logical qubits to physical qubits on the chip, and repositioning those logical qubits on the chip to make them adjacent when necessary, which is usually done by inserting additional gates (e.g. SWAPs or shuttle operations). The specifics of quantum circuit mapping can differ from one device to another due to technology disparities and qubit connectivity. Ensuring effective utilization of limited hardware resources and minimizing errors during quantum operation execution by reducing additional gates and circuit latency necessitates this process. Yet, solving the quantum circuit mapping problem as the qubit count increases is computationally challenging, even for current monolithic (or SC) devices. To address this, diverse quantum circuit mapping algorithms have been introduced, ranging from heuristic and brute-force strategies to graph-theoretical techniques, dynamic programming algorithms, and machine learning-based approaches [17, 30–53], relying on different performance metrics like gate count, circuit depth (i.e. number of layers of gates where gates can run in parallel), fidelity [38, 39, 54] or the circuit success rate [35, 55].

However, mapping techniques designed for SC-processor NISQ devices do not readily apply to modular MC (or multi-node) quantum computing architectures, which offer a promising path for quantum computing scalability [2]. This architectural approach involves cores (quantum processing units, or QPUs) interconnected via classical and, ultimately, quantum communication channels. Quantum links facilitate the transfer of quantum states among processors or the execution of inter-core quantum gates depending on the technology, while classical links ensure the coordination of quantum communication [56]. The intricate communication channels and traffic patterns in MC architectures add complexity to quantum circuit

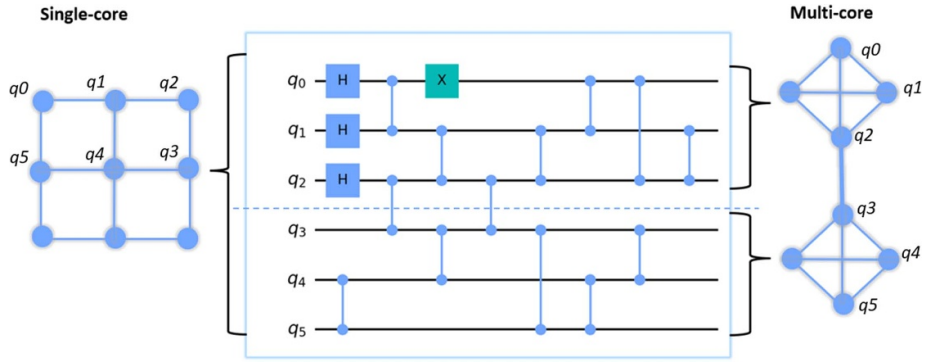

Figure 1. Initial mapping of a quantum circuit (middle) to a SC (left) or MC device (right); For the latter case, the quantum circuit is partitioned into two highly connected circuit slices so that any communication between the two cores is minimized.

mapping compared to SC devices [3, 57, 58]. In response, novel techniques have emerged for modular architectures, aiming to minimize the costly (in terms of time and effort overhead or reduced fidelity) long-distance inter-core operations. To reduce the amount of inter-core operations, it is essential to efficiently allocate logical qubits across the physical qubits of the given cores. While the literature in this emerging domain is limited, some approaches have concentrated on quantum compilation and mapping for modular quantum computing [59–62]. In these approaches, the quantum circuit is divided into smaller partitions, and IGs reflecting operations within a circuit segment are mapped onto cores by grouping qubits with high interaction levels (see figure 1).

All previously stated strategies within both SC and MC quantum computation share a common goal: tailoring quantum circuits to device-specific attributes and constraints, while minimizing the communication overhead. However, they often only focus on a limited set of circuit features, such as gate and qubit counts, and qubit interactions. What is missing is a more comprehensive quantum circuit characterization that goes into deeper aspects. For example, one can explore the characteristics of the qubit IG, such as the frequency of interactions between qubit pairs [18] and the distribution of these interactions among qubits, as well as the quantum instruction dependency graph (representing gate dependencies for scheduling).

Some researchers have already highlighted the significance of incorporating application-specific properties [13, 15–18, 36] to enhance quantum circuit mapping and overall quantum system performance. Even in classical computing, the allocation of computing resources depends on the intended applications and processes. In a similar vein, thorough profiling aids in identifying the essential circuit features for successful execution on a particular device and vice versa. Yet, as we explore the intricacies of running algorithms on modular architectures, it becomes evident that this realm demands a distinct approach when compared to conventional monolithic NISQ devices. Consequently, profiling quantum circuits in the context of modular architectures should adopt a tailored strategy and select parameters that align with the nuances of this scenario.

In conclusion, understanding the structural attributes of quantum circuits sheds light on why and which groups of algorithms perform better on specific processors with designated mapping techniques than on others. This holistic understanding not only guides mapping but also opens avenues for improving the overall performance of quantum circuits on quantum hardware and represents a first step towards application-based quantum computers.

2.3. On the importance of qubit interaction and GDGs

Quantum circuits so far have mostly been characterized in terms of size, i.e. the number of qubits, gates, two-qubit gates, and depth, which are blind to the circuit's structure. Considering that the main quantum hardware constraints are low fidelity and limited qubit connectivity, it is important to extract more information about the qubit interaction distribution as well as gate dependencies as they directly relate to gate and depth overhead that results from the compilation process. Previous works have emphasized the significance of deriving additional circuit parameters based on qubit interaction and quantum instruction dependency graphs for the development of mapping techniques [12, 31, 32, 61]. While GDGs have been utilized for operation scheduling optimization and look-ahead techniques, IGs have typically been employed for the initial qubit placement and routing procedure. In this work, we utilize these two graphs and their graph-theory-based attributes for characterizing circuits targeting both SC and MC architectures. In the next paragraphs, we will introduce these two graph representations of the quantum circuit.

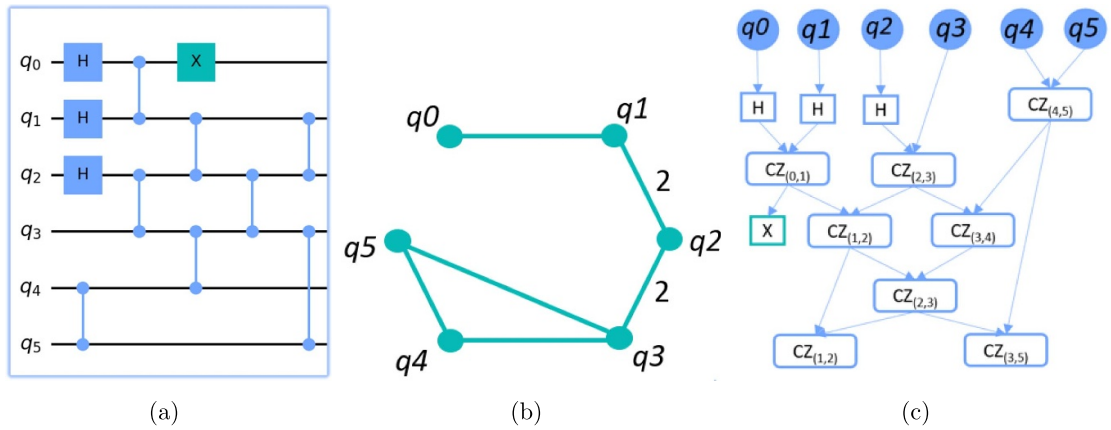

Figure 2. An exemplary quantum circuit (a) and its corresponding qubit interaction graph (b) and GDG (c).

The IG $G_i(V_i, E_i)$ offers a visual representation of the spatial distribution of two-qubit gates for a given quantum circuit. Typically, this is an undirected and connected graph (see figure 2(b)), with edges denoted as $e(v_i, v_j) \in E_i$ representing the two-qubit gates and nodes labeled as $v_i \in V_i$ representing the qubits engaged in these gates [18, 61]. Previous research [18] has demonstrated that different categories of circuits (e.g. real algorithms and random circuits) exhibit varying compilation performance in terms of gate and depth overhead as well as fidelity decrease when executed on different physical-qubit topologies (see figures 5 and 14 of [18]). To understand the underlying reasons for these differences, prior studies have explored the structural disparities between various quantum circuit groups based on their IGs. The quantum circuit mapping problem can be conceptualized as a graph problem, in which the comparison between the IGs of a circuit and the coupling graph of the device is crucial [53]. Therefore, the parameters of the IG can significantly influence the outcome. Studies such as [18, 24] have illustrated how the IG and size parameters of quantum circuits directly correlate to their performance on different chips when employing the same mapping technique. For example, circuits with fully connected IGs are expected to incur higher overhead, unless the processor's coupling graph is fully or almost fully connected. Moreover, research such as [18] has further clustered the circuits based on these extracted parameters and identified performance discrepancies among the groups based on the aforementioned mapping metrics.

IGs play an even more significant role in the compilation of circuits for MC systems [63]. Mapping to MC devices involves minimizing inter-core communication by partitioning the IGs and effectively allocating the logical qubits (IG nodes) onto the physical qubits of different cores. This is achieved by aiming to map highly interacting qubits (with high IG weights) onto the same core. Given the importance of IG partitioning in modular compilation and its distinct approach compared to SC systems, it is essential to use different IG metrics. These metrics should focus on identifying local clusters and cliques to effectively describe how 'partitionable' a circuit is.

Besides that, the GDG $G_{\mathrm{gd}}(V_{\mathrm{GD}}, E_{\mathrm{GD}})$ is a directed and connected graph illustrating the interrelations between the gates in the circuit. Each gate in the circuit is represented by a node $v_j \in V_{\mathrm{GD}}$, and the dependency between gates is depicted as a directed edge $e(v_j, v_k) \in E_{\mathrm{GD}}$. An illustrative example of this concept is shown in figure 2, where a quantum circuit consisting of 6 qubits and 8 two-qubit gates is displayed alongside its corresponding IG and GDG [42]. It is worth noting that both graphs can also be represented as weighted graphs: the IG becomes weighted when multiple two-qubit operations occur between the same pairs of qubits, while the GDG becomes weighted when considering, for instance, gate duration (i.e. time). For the purpose of this paper, we focus on the simplified, unweighted version of GDG, where all gates take one time step.

In this paper, we aim to examine quantum circuits by analyzing parameters extracted from IGs that are relevant for SC and multi-processor architectures, from GDGs (e.g. path distribution and critical path length), parameters related to repeating sub-circuits/oracles within the circuit, and circuit density-related parameters, followed by a clustering of the circuits. Our focus is to identify and quantify the relationships between circuit parameters, clusters, and various compilation techniques across different quantum topologies. That is only possible if we observe the circuit as a whole and utilize all its complementary properties like those of IG (spatial) and GDG (temporal). Gaining a comprehensive understanding of circuit structures can aid in designing and evaluating quantum systems optimized for the highest success rates for specific circuit groups and help define a finite set of representative benchmarks.

In the subsequent sections, we will provide a detailed overview of the entire quantum circuit profiling process.

3. Characterizing quantum circuits—definition and profiling

3.1. Parameter selection

Table 1 illustrates the five groups of circuit parameters we consider in this work. The process selection of the **parameters of interest** involved two steps: (1) selecting parameters within these groups relevant to SC and MC architectures considering their main constraints. For instance, the clustering coefficient [64] is important for the execution of circuits in MC devices but not in SC devices, as it requires graph partitioning. Conversely, node degree is relevant to both scenarios: a higher average degree in the IG makes mapping more challenging; and (2) reducing the number of parameters by identifying highly correlated ones using the Pearson correlation matrix [25], like in [18].

The resulting reduced set of parameters is as follows [18, 64]:

3.1.1. Size

Standard parameters used for circuits description in previous works:

- *Number of qubits*: n_{q}
- *Number of gates*: n_{g}
- *Two-qubit gate percentage*: $\frac{n_{2\mathrm{qg}}}{n_{\mathrm{g}}}$, where $n_{2\mathrm{qg}}$ is the number of two-qubit gates.
- *Decomposed circuit depth*: d

3.1.2. IG (definitions taken from [18])

- *Average shortest path length*: average hopcount between all nodes [64]. The larger the average hopcount between the nodes, the less connected the graph is. It also means that a simpler IG is easier to map.
- *Standard deviation of adjacency matrix* $\sigma(A)$: an adjacency matrix A is a square matrix used for representing a graph whose elements are $a_{ij} = a_{ji}$ represent the number of connections between nodes n_i and n_j. It shows which nodes are connected and with how many edges. A large σ value means some specific pairs of qubits interact much more than others and that there is less additional routing required.
- *Diameter*: longest shortest path in the circuit,

$$dm = \max_{n_i \in \mathcal{N}} (\epsilon_i),$$

  where ϵ_i is the longest hopcount between node n_i and any other node among the total of N nodes. The larger the diameter, the simpler the IG and, therefore, easier to map onto a device.
- *Central point of dominance*: maximal betweenness of any node in the graph, where betweenness is the number of shortest paths between nodes that traverse some node or edge [64]. A value of 0 results for complete graphs, and 1 for star-shaped graphs. Values approaching 0 or 1 are undesirable from the perspective of quantum circuit mapping, as 0 reports a graph that is too much connected, and 1 indicates that one qubit is involved in all gates, making the circuit hard to parallelize.
- *Average degree*: average degree of neighbor nodes, where the degree is the number of nodes to which one node is connected and defined as

$$\deg_i = \sum_{j=1}^{N} a_{ij}.$$

  The lower the value of the average degree, the less connected the IG is, and the easier it is to map.
- *Number of maximal cliques*: the total number of the largest all-to-all connected subgraphs. This metric also depends on the size of the maximal clique. The smaller the largest clique, the less connected the graph and, therefore, easier to map.

  Another cliques-related graph metric is *clustering coefficient* which measures the cliquishness of a neighborhood. The values range between 0 and 1, where 1 represents a fully connected graph, which is always the worst-case scenario for the quantum circuit mapping:

$$c_i = \frac{y_i}{\binom{\deg_i}{2}},$$

  where y_i is the number of links between neighbors of node n_i. These two metrics are of high importance for MC computation as they show the presence of highly connected clusters of nodes within the graph, which is related to the IG partitioning part of MC quantum circuit mapping.

Table 1. Selected metrics for the characterization of quantum circuits.

Metric	Metric type	SC or MC
Num. of qubits	Size	Both
Num. of gates	Size	Both
Two-qubit gate %	Size	Both
Circuit depth	Size	Both
Avg. shortest path	IG	SC
Standard deviation of adjacency matrix	IG	SC
Diameter	IG	MC
Central point of dominance	IG	MC
Maximal cliques and num. of maximal cliques	IG	MC
Clustering coefficient	IG	MC
Avg. degree	IG	Both
Vertex/edge reliability	IG	MC
Coreness	IG	MC
Pagerank	IG	MC
Critical path length	GDG	Both
Num. of critical paths	GDG	Both
GDG path length distribution metrics	GDG	Both
% of gates in critical path	GDG	Both
Density score	Circuit density	Both
Idling score	Circuit density	Both
Num. of largest rep. sub-circuit	Sub-circuit repetitions	Both
Size of largest rep. sub-circuit	Sub-circuit repetitions	Both

- *Vertex/edge reliability*: the minimal number of nodes/edges whose removal can disconnect the graph. The lower the reliability, the easier it is to partition the graph for MC mapping.
- *Coreness*: maximal k for specific node i such that i is present in k-core graph but removed from $(k+1)$-core (k-core is a subgraph of some graph made by removing all the nodes of degree $<= k$). Coreness as the local metrics also relates to IG partitioning and modular computing.
- *Pagerank*: ranking of the importance of each node in the graph [65] based on the number and weights of the links with other nodes and the rank of those nodes. This graph metric emphasizes which nodes should be mapped to the most connected part of the chip.

3.1.3. GDG

One of the main disadvantages of current QPUs is the short lifetime of qubits, i.e. their decoherence. That makes the circuit *scheduling* one of the crucial segments of the quantum circuit compilation. The goal during scheduling is to make the gates run as parallel as possible. However, not every circuit is parallelizable. GDG and especially its *critical path* (longest path in GDG) showcases the minimal necessary duration of the circuit and represents its longest inter-depending gate sequence. Therefore, this sequence of gates should be scheduled to run as soon as possible in order to shorten the circuit duration and prevent qubit decoherence. Metrics related to critical and other paths of GDG can also tell us how sequential the circuit is and, therefore, how easy it is to map to a single or to multiple cores. Qubits participating in the inter-dependent gates (gates of the same common paths) should be placed nearby on a chip or within the same core during the mapping process. Furthermore, the higher the percentage of the circuit gates included in the critical path, the more sequential the circuit is and, therefore, less parallelizable. Metrics that describe the GDG paths include:

- *critical path length* or number of gates in the critical path;
- *number of critical paths*;
- *path length distribution* and its mean and standard deviation; and
- *percentage of gates included in the critical path.*

These metrics' implementation and detailed definitions are shown in appendix B.

3.1.4. Circuit density

This set of metrics evaluates the degree of parallelization of the circuit gates before any optimizations. It indicates the number of gates executed in each layer (time-step) of the circuit relative to the maximum number of gates that could be executed if the circuit were fully parallelized (similar to quantum volume circuits [55]). A denser circuit implies greater difficulty in further optimization and execution. This paper utilizes two metrics to describe this behavior:

- Density score: parallelization level of the circuit;

$$D = \frac{\frac{2*n_{2qg}+n_{1qg}}{d} - 1}{n_q - 1},$$

where n_{2qg} and n_{1qg} are number of two- and SC-qubit gates, respectively. This is an extended version of the parallelism metric from [24]: we made a distinction between the SC and two-qubit gates for better preciseness, where D can actually reach all the values in the range between 0 and 1, (1 is maximal density); and
- Idling score: average amount of qubit idling in the circuit;

$$I = \frac{\sum_{i=1}^{n_q} d - q_i}{n_q * d},$$

where q_i signifies the number of layers of the circuit in which the qubit is used; range between 0 and 1, with 0 meaning no idling and 1 meaning no scheduled gates.

3.1.5. Longest sub-circuit repetitions

In circuits that are based on real algorithms, there are always patterns in terms of gate order and repetitions. In contrast, completely random circuits show, on average, no such patterns. In order to express the randomness of the gates and groups of gates in circuits, we have defined the following metrics

- the number of occurrences of the largest repetitive sub-circuit; and
- the size of this largest repetitive sub-circuit.

The extraction of these two metrics relies on existing algorithms used for strings of characters. The problem of finding the longest repeating sub-string in a text and the length thereof is efficiently solved by filling a data structure called a *suffix tree*[66]. We used the same implementation by identifying characters and quantum gates. The significance of gate randomness on circuit performance is closely tied to the type of mapper employed. Some mappers are expected to exhibit a high correlation, where the routing algorithm is designed to identify gate patterns in advance. In contrast, more stochastic approaches do not benefit from recognizing these patterns.

The IG parameters reveal patterns necessary for initial placement algorithms, whereas GDG parameters indicate the length and sequential nature of the circuit, crucial for mitigating decoherence and enhancing scheduling and routing. Parameters related to circuit density illustrate the extent of parallelization in the circuit before optimization, influencing the complexity of all stages of quantum circuit mapping. Additionally, identifying recurring gate patterns in the circuits assists in refining scheduling and look-ahead routing techniques. For the complete list of metrics, please refer to [11] (IG-based) and appendices D and C.

3.2. Quantum circuit clustering

As previously stated, one of our objectives is to identify structural similarities among quantum circuits and establish 'circuit families', wherein the constituent elements (i.e. the quantum circuits) exhibit similar compilation behavior and require comparable hardware resources. For this purpose, we employed a two-step clustering approach: an initial clustering of circuits based on size parameters (Group 1), followed by clustering based on the remaining parameters. This strategy was implemented to prevent size parameters from exerting undue influence on the clustering algorithm. Consequently, we initially categorized the set of 341 selected benchmarks (see section 4) into five clusters using the K-means clustering algorithm [67]. Subsequently, each of the five size-related clusters could be further subdivided into sub-clusters based on the previously described structural circuit parameters (groups 2-5). In this regard, we once again opted for the K-means algorithm after evaluating various methods and parameter configurations using the silhouette coefficient method [68]. The specific settings for clustering SC and MC devices, as well as the clusters themselves, are detailed or referred to in appendices C and D, respectively. It is expected that circuits assigned to the same sub-cluster will exhibit similar fidelity and gate overhead outcomes. The correlation between our circuit groups and mapping performance metrics, as well as potential explanations for these relationships, will be discussed in the subsequent sections.

4. Methodology

In our research, the quantum circuit profiling process encompasses the following four steps:

(i) **Benchmark collection**—we gather benchmarks (i.e. quantum circuits) from diverse sources, translate them into the same quantum language (in our case cQASM [69]), and extract their interaction and GDGs [18].
(ii) **Parameter selection and extraction**—we select and extract graph-theory-based parameters from the qubit IG and GDG, focusing on parameters relevant to mapping quantum circuits to SC and MC devices. Additionally, we extract supplementary circuit parameters to enhance the characterization related to circuit density.
(iii) **Benchmark clustering**—we cluster benchmarks based on their size-related (number of qubits and gates, two-qubit gate percentage, and circuit depth) and structure-related parameters.
(iv) **Compilation**—we compile the quantum circuits using Qiskit [70], OpenQl [71], and additional MC compilation solutions [61, 62], and analyze the relationship between their performance and the extracted parameters, as well as their cluster affiliation (see section 5).

4.1. Quantum benchmarks selection

In this paper, we employed the qbench benchmark set [18, 22], which offers a comprehensive collection of quantum circuits sourced from various platforms and written in different programming languages. This set encompasses a range of circuit types categorized based on their origin, including circuits derived from real quantum algorithms (e.g. QFT), simpler algorithm-based circuits (such as arithmetic circuits), QUEKO circuits optimized for specific devices [72], and randomly generated circuits produced by randomly selecting SC and two-qubit gates from a predefined set and applying them to arbitrarily chosen qubits or qubit pairs [73]. This category also includes highly parallelized quantum volume circuits used for device benchmarking [74].

Given the inclusion of MC architectures in our study, we also incorporated benchmarks used so far for MC computations [3, 12, 58, 61]. These benchmarks comprise a variety of circuit instances, ranging from 16 to 128 qubits, such as the Cuccaro Adder [75], Grover's main routine, GHZ state preparation [76], QFT, QAOA, and VQE [77]. To accommodate this aspect, we expanded our synthetic circuit set to include instances with up to 128 qubits. The complete list of 341 quantum circuits utilized in our study is detailed in appendix A, with accompanying code available at [22].

4.2. Hardware configuration

To investigate the relationship between the previously identified circuit clusters (see section 3) and the mapping outcomes, we compiled the selected quantum circuits using various target SC and MC quantum architectures. For SC architectures, we utilized the Rigetti Aspen-1, Surface-17 [32], IBM Rochester, and Google Bristlecone devices, as depicted in [32, 78–80]. Regarding modular architectures, we adopted the same configuration as detailed in [61]: all-to-all connected qubits within the cores, while the cores themselves are interconnected either in an all-to-all manner or in a grid-like fashion, as illustrated in figure 3. These device configurations were selected because they are commonly employed in quantum circuit mapping research and offer realistic and diverse connectivity patterns in their coupling graphs. Note, that we did not execute the quantum circuits on actual devices; instead, the hardware constraints of the devices were considered during the compilation process alone.

4.3. Quantum compilers

In this study, we investigate not only the relationship between different quantum circuit groups, parameters, and quantum devices but also explore the connection between quantum circuits and mapping techniques. The goal is to gain insights into developing and selecting mapping techniques tailored to circuits with specific common properties.

For SC devices, we utilize the widely used Qiskit compiler and test four combinations of compilation settings by pairing two routers with two optimization levels. Specifically, we employ the Stochastic and Sabre routers, along with optimization levels 1 and 2. The key distinction between the Stochastic and Sabre approaches lies in their routing path selection: Sabre utilizes a more deterministic approach, whereas Stochastic employs a more randomized strategy. Optimization level 1 involves simplistic mapping with minimal circuit optimization but short runtime, while optimization level 2 employs more aggressive gate

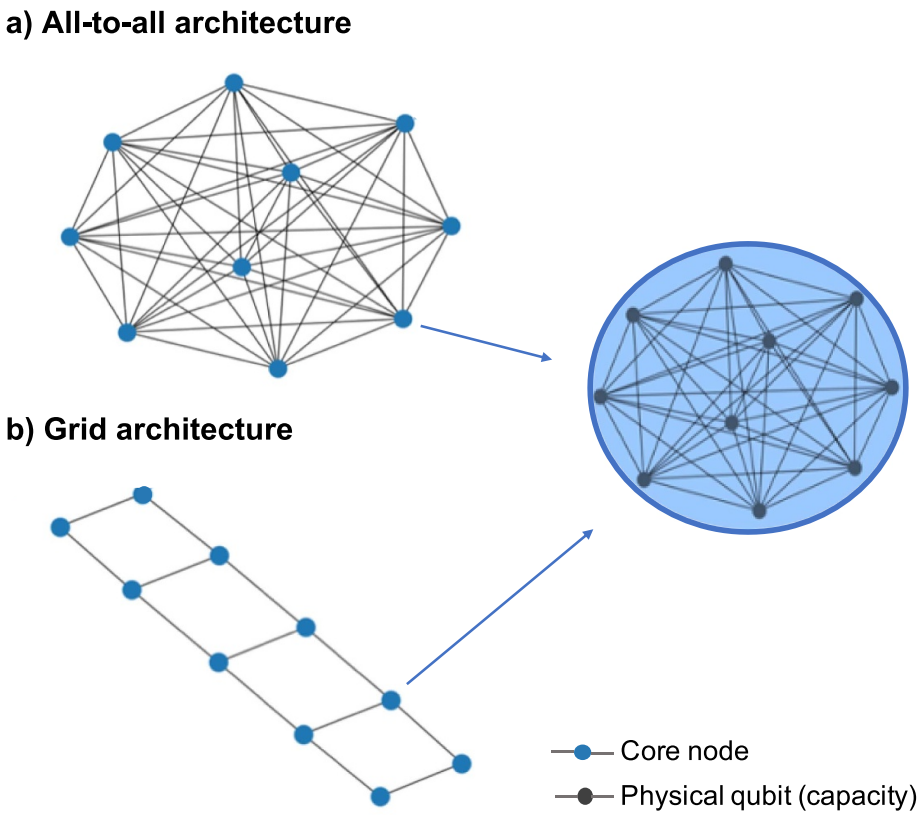

Figure 3. In our experiments, we examined two distinct multi-core architectures: (a) all-to-all connected cores and (b) 2D Grid core connectivity. Each node in the two graphs on the left represents a core, while the edges correspond to communication links between the cores. On the right, the intra-core qubit topology is displayed, comprising 10 all-to-all connected qubits. We chose these topologies due to their ability to offer a simplified representation of architectures anticipated to emerge in the near future. This approach enables us to promptly address the immediate need for mapping solutions based on minimizing inter-core communication before going into more complex architectures [61]. © 2023 IEEE. Reprinted, with permission, from [61].

cancellation and commutation techniques to minimize qubit crosstalk and increase parallelism. Level 2 is suitable for complex circuits or cases in which circuit optimization is crucial despite requiring more computational resources and time during compilation. For simplicity, we will refer to these four method combinations as *Stochastic1*, *Stochastic2*, *Sabre1*, and *Sabre2*.

In the modular regime, we explore three techniques:

- The *time-sliced circuit partitioning rOEE* method [12], which leverages qubit clustering to create tractable partitioning heuristics for mapping quantum circuits to modular physical machines one slice at a time. This method also uses a tunable lookahead scheme to reduce the cost of moving to future time slices. Note that even though this method is mainly focused on reducing the number of inter-core operations, the gates in the time-slices are scheduled to run as concurrently as possible in order to reduce the circuit depth as well.
- The Hungarian qubit assignment (HQA) algorithm [62], which employs the Hungarian algorithm [81] to enhance qubit-to-core assignment by considering interactions between qubits across the entire circuit, and enabling fine-grained partitioning and enhanced qubit utilization.
- A *QUBO*-based approach [82], which encodes qubit allocation and inter-core communication costs in binary decision variables [61]. This method splits the quantum circuit into slices and formulates the qubit assignment as a graph partitioning problem for each slice, reducing costly inter-core communication by penalizing that. The final solution minimizes the overall cost across all circuit slices. It is important to note that unlike the rOEE method, this method includes different type of time-slicing, where gates are scheduled within the same time-layer as long as they can be run without imposing new inter-core operations, sometimes resulting in multiple gates run on the same qubits.

4.4. Performance metrics

Quantum circuit mapping performance metrics are defined as follows:

(i) **Gate overhead** is calculated using the formula: $G_{\text{overhead}} = \frac{(G_{\text{after}} - G_{\text{before}})}{G_{\text{before}}}$. Here, G_{before} and G_{after} denote the number of gates before and after compilation, respectively.

(ii) **Depth overhead** is determined by: $D_{\text{overhead}} = \frac{(D_{\text{after}} - D_{\text{before}})}{D_{\text{before}}}$, where D_{before} and D_{after} represent the circuit depth before and after compilation. Depth refers to the number of cycles or layers of simultaneously running gates in the circuit.

(iii) **Fidelity decrease** is computed as: $F_{\text{decrease}} = \frac{(F_{\text{before}} - F_{\text{after}})}{F_{\text{before}}}$. Here, F_{before} and F_{after} represent the circuit fidelity before and after compilation, respectively. *Circuit fidelity* is a product of the error rates of the gates in the circuit. The primary objective during circuit mapping is to maximize this metric. We assumed uniform error rates for all one-qubit and two-qubit gates, using average values from the Starmon-5 chip [69, 83, 84].

(iv) **Number of non-local (inter-core) communications** is the number of qubit moves from one core to another within modular quantum architectures. Entanglement-based quantum communication protocols are employed that require bell-pair generations that enable the teleportation of quantum states. However, generating entangled pairs results in a resource overhead, and the process itself is non-deterministic, adding complexity to scheduling tasks [3].

In the following section, we will discuss the relation of the structural parameters of circuits from section 4.1 with the above-stated obtained metrics after mapping them into different combinations of quantum system setups mentioned in sections 4.2 and 4.3.

5. Results and discussion

In this section, we assess and contrast the mapping results of our chosen circuits while examining the influence of circuit parameters on the outcomes. Furthermore, we juxtapose the performance of various circuit clusters using different mapping techniques and processor designs.

5.1. Mapping to SC devices

We begin our analysis by examining the correlation between the circuit parameters introduced in section 3 and the three quantum circuit mapping performance metrics across four different SC architectures: Surface 17, IBM Rochester, Rigetti Aspen, and Google Bristlecone (see [32, 78–80]). The correlations are visualized in figure 4(a) using a Pearson correlation map, where the correlation values range from dark red for the highest positive correlation to dark blue for the highest negative correlation. A positive correlation means that if values of one parameter increase, then so do of the other, and a negative is vice versa. White color, and correlation factors around zero indicate no significant correlation. The parameters are sorted to facilitate the identification of the most correlated ones. Notably, while some parameters are consistently important across all devices and metrics, others are not significant (e.g. the number of gates), and some vary in importance depending on the metrics and devices (e.g. the number of qubits has a high negative correlation factor only for the fidelity decrease in the IBM Rochester device). In addition to the analysis of the four devices using the same mapping technique (Sabre2), figure 4(a) also includes results for two different mapping configurations, namely Stochastic1 and Sabre2 (see section 4.3), for the Google Bristlecone device. The configurations Stochastic2 and Sabre1 are not shown, as their correlation results are similar to Stochastic1 and Sabre2, respectively. This suggests that the routing technique is more significant than other optimization passes concerning different circuit parameters. For the same reason, we opted to use Sabre2 and Stochastic1 for the rest of the experiments explained in this section.

To investigate the meaning of these correlations, we plotted 3D graphs of some of the highest-correlated parameters. As examples, we selected: (1) the gate overhead metric for the Surface 17 device with parameters: average shortest path length, density score, and idling score, shown in figure 4(b), and (2) the depth overhead metric for the Rigetti Aspen device with circuit parameters: number of qubits, average IG degree, and number of critical paths, shown in figure 4(d). In figure 4(b), we observe that the average shortest path length and density score metrics are negatively correlated with the gate overhead, meaning that higher values of these metrics correspond to a lower gate overhead. Conversely, the idling score shows a positive correlation. This implies that a higher average shortest path length simplifies running the circuit due to a simpler IG, confirming our hypotheses in section 3.1.2. In figure 4(d), the average degree of the IG is shown

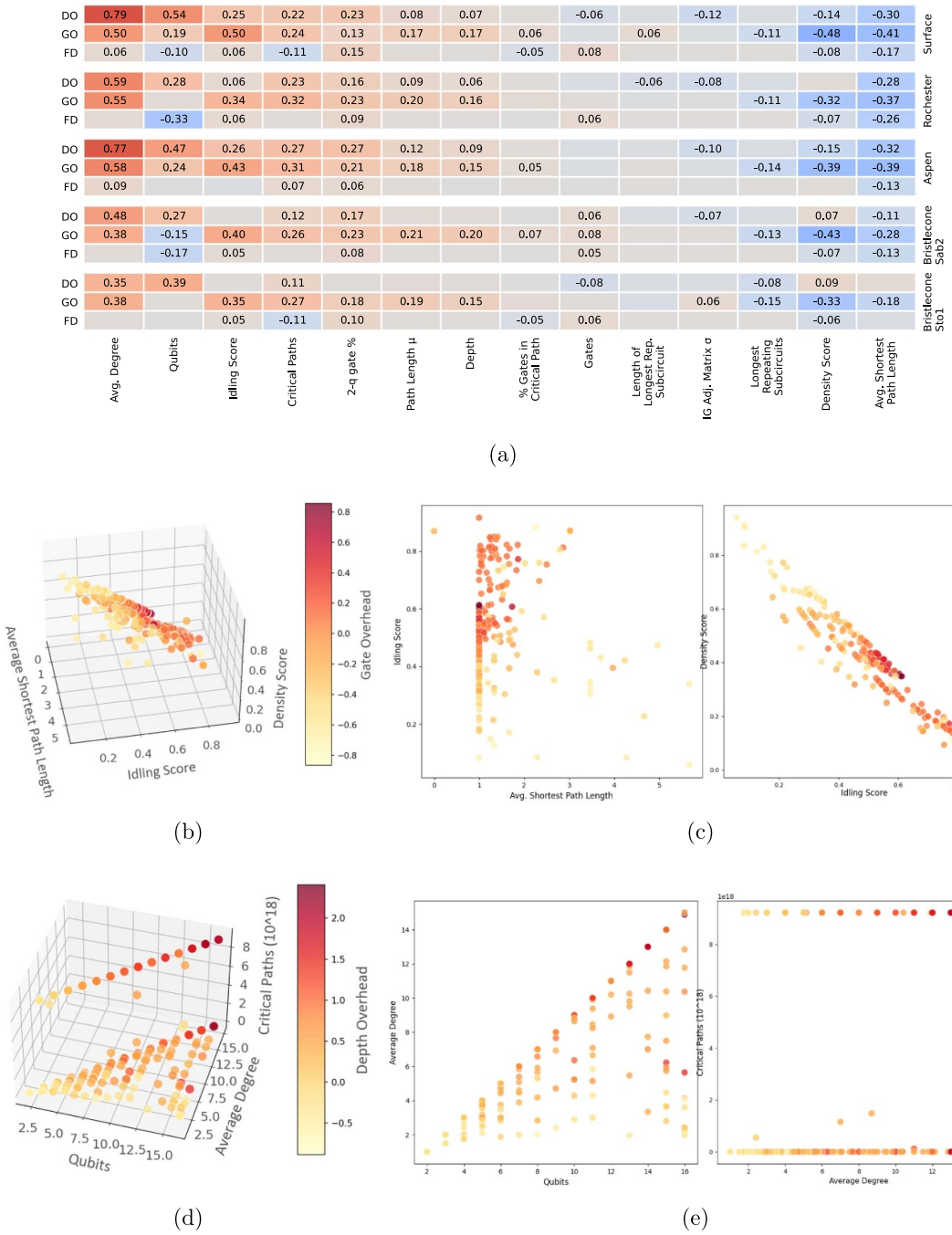

Figure 4. (a) Correlation of the gate overhead, depth overhead and fidelity decrease compilation performance metrics with circuit parameters while using 4 different device configurations: Surface 17, IBM Rochester, Rigetti Aspen and Google Bristlecone for Sabre2 mapper, and Bristlecone device for Stochastic1 mapper. Metrics are ordered from the most positively correlated to the most negatively correlated one (red to blue). (b) and (c) Three parameters with high performance correlation for the surface 17 topology: idling score, density score and avg. shortest path length. (d) and (e) three parameters with high performance correlation for the Aspen-16 topology: average degree, number of critical paths and number of qubits and their depth overhead.

to be highly influential for the depth overhead: a higher degree indicates a denser IG, leading to a larger depth overhead. The number of qubits and the number of critical paths also exhibit positive correlations.

The reasoning behind these results can be summarized as follows:

- **Average shortest path length (hop count) and gate overhead**: a negative correlation is observed between the average shortest path length (or hop count) and the gate overhead, particularly in the case of the surface-17 device. This stems from the fact that a higher number of hops in the IG indicates poor connectivity. A poorly connected IG is easier to map onto grid-like architectures such as the Surface-17, as discussed in [18], resulting in fewer additional SWAP gates required to run the algorithm and, consequently, lower gate overhead.

- **Idling score and gate overhead**: a positive correlation is found between the idling score and gate overhead across all devices, especially with Surface-17. As described in section 3, the idling score ranges from 0 (no idling) to 1 (maximum average qubit idling). This correlation arises from both the limited connectivity of the device and the mapping techniques that employ lookahead strategies to minimize the number of SWAP gates for the immediate portion of the circuit. In circuits with higher idling, gates are spread across more time slices, reducing the effectiveness of such optimizations.
- **Density score and gate overhead**: the density score, which measures the degree of parallelization in the circuit, is negatively correlated with gate overhead. This is essentially the inverse of the idling score: circuits with higher parallelization tend to have lower gate overhead due to better optimization opportunities.
- **Average degree of the IG and overhead**: the average degree of the IG is positively correlated with both gate and depth overhead. As outlined in section 3, the average degree reflects the connectivity of the IG. Highly connected qubits in the IG make it difficult to map onto weakly connected architectures like Aspen-16, naturally leading to increased overhead for circuits with more complex topologies.
- **Number of qubits and depth overhead**: a positive correlation exists between the number of qubits and the depth overhead. This is due to both the device topology and the challenges of the mapping technique itself. With higher qubit counts, routing becomes more complex, as fewer ancillary qubits are available for routing, and the options for optimal initial placement decrease. These factors lead to more qubit idling during routing, which increases the circuit depth.
- **Number of critical paths and depth overhead**: the number of critical paths also shows a positive correlation with depth overhead. In all experiments, circuits with more critical paths (where many paths in the GDG are of equal largest length) create scheduling and gate prioritization challenges for the compilation algorithms, reducing their ability to optimize the circuit's overall latency.

In the next part of our work, we created 'circuit families' containing circuits of similar structure based on extracted features. The two-level clustering approach used to achieve this is explained in section 3.2. Figure 5 presents examples of these created families and their results concerning the three compilation metrics. Figures 5(a)–(c) illustrate performance differences when targeting Surface 17, Bristlecone using Sabre2 mapper, and Bristlecone using the Stochastic1 mapper, respectively. Notably, the group of circuits marked in purple (Group 0) performed better on average with Bristlecone than on Surface 17, exhibiting approximately 20% fewer gates. Conversely, the group noted in grey (Group 5) performed better on Surface 17, particularly in terms of fidelity decrease, which remained below 40% except for one outlier. In contrast, for the Bristlecone device, the majority of benchmarks for this group ranged between 60% and 100%. The Stochastic1 mapper generally performed worse than Sabre2, as expected due to its simplicity. For example, Group 3 (yellow) reached 150% in gate overhead, and Group 5 reached 200% in depth overhead, whereas these metrics for Sabre2 were 75% and 150%, respectively. However, Group 0 did not show significant performance differences between the two mappers, with benchmarks performing within the same ranges (mostly between ∼-50% and ∼50%) regarding gate and depth overhead. In this case, it would be preferable to use the simpler and faster Stochastic1 method. It is important to note that the majority of circuits in Group 0 are real-algorithm-based circuits, including many instances of Grover's algorithm [77], Cuccaro adder [75] and RevLib [85] algorithms. On the other hand, most circuits in Group 5 are QUEKO circuits [72]. This analysis provides guidance on which types of circuits work better with specific device/mapper configurations.

The Aspen and Rochester device configurations exhibited patterns very similar to Surface 17 and Bristlecone, respectively, for these particular circuit families. Therefore, they were not showcased in this paper. For the rest of the results, refer to appendix D.

5.2. From SC to MC architectures

Our analysis continues by examining the correlations between the number of inter-core communications and circuit parameters using three different mapping configurations for all-to-all connected MC devices: QUBO, HQA, and rOEE, as introduced in section 4. (see figure 6). Notably, the importance of circuit parameters differs among the mapping techniques. For instance, the number of qubits and maximal cliques is positively correlated with the performance when using HQA, but not with other techniques. Circuit depth and GDG path length mean exhibit a high correlation score (approximately 0.75) with QUBO, while the scores are only 0.2 and 0.4 for the other two mappers. Edge connectivity is not influential when using rOEE, but it is for the other two mappers. Additionally, we explored two different core topologies with the QUBO mapper: all-to-all and grid. Generally, the parameter correlations between these topologies are similar, though some differences exist. For example, the average degree circuit parameter is significantly more important for grid connectivity due to the less connected device connectivity graph, emphasizing the importance of IG connectivity of the circuit, as expected in section 3. We anticipate a larger difference when also reducing the connectivity within the cores, an experiment we leave for future work.

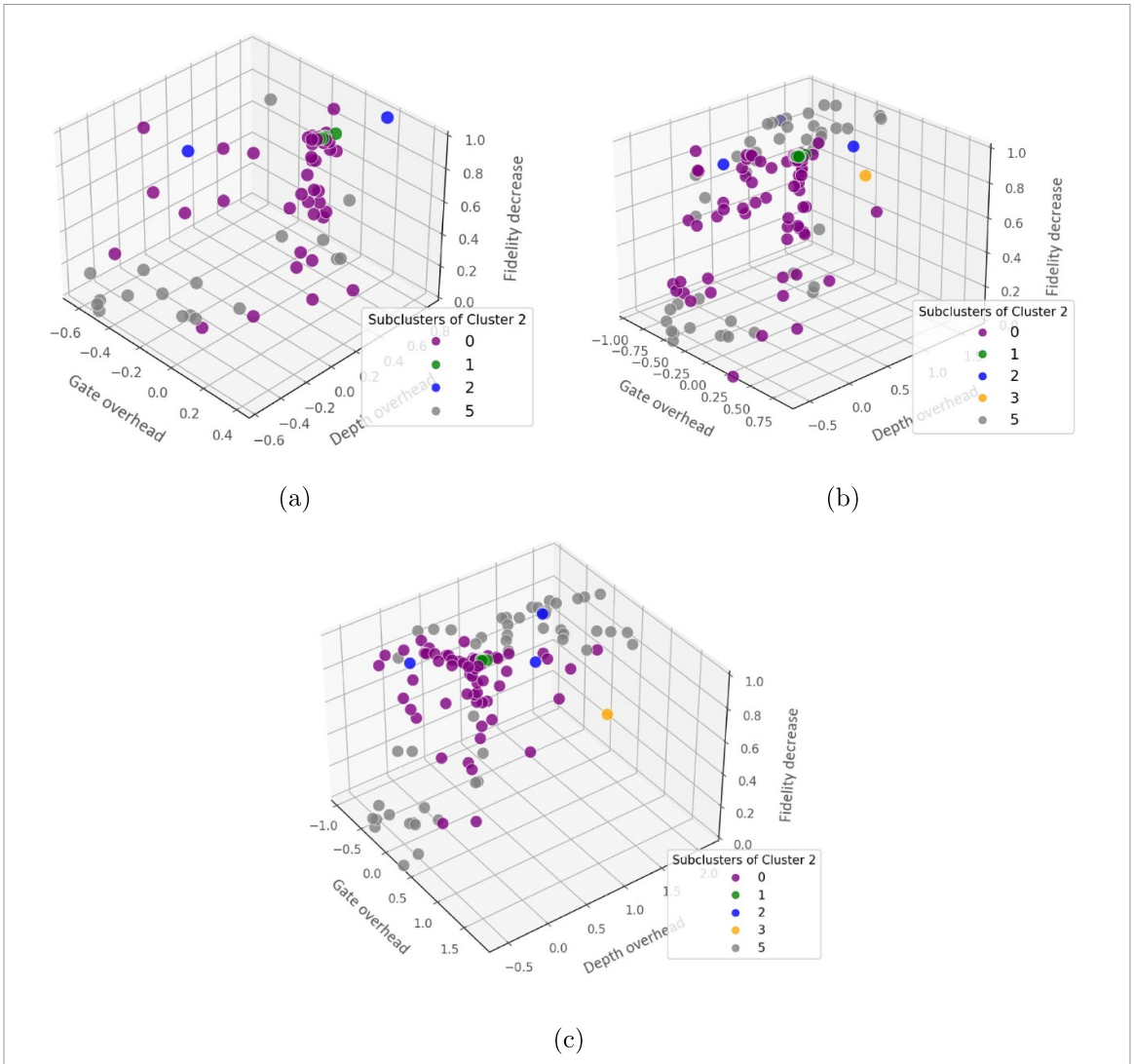

Figure 5. Relation of circuit families (sub-clusters) of cluster 2 with their compilation metrics when targeting: (a) Surface 17, (b) bristlecone for Sabre2 mapper, and (c) bristlecone for Stochastic1 mapper.

Figure 6(b) shows an example of the correlation between parameters: edge connectivity, GDG path length mean, and circuit depth, with the number of inter-core communications when using the QUBO mapper. A trend is observed where overhead increases with higher metrics values, confirming the high positive correlation shown by their correlation factors in figure 6(a).

The reasoning behind this can be summarized as follows:

- **Initial circuit depth and inter-core communications**: there is a positive correlation between the initial circuit depth and the number of inter-core moves (communications). For methods like QUBO, which aim to minimize inter-core moves by optimizing each time-slice globally, mapping a circuit with significant depth onto a MC architecture is highly challenging and requires substantial computational resources. This difficulty leads to increased overhead. Leveraging quantum devices like D-Wave could potentially alleviate this issue [86].
- **Average GDG path length and inter-core moves**: a positive correlation is observed between the average GDG path length and the number of inter-core moves. Longer GDG paths result in greater circuit depth, which, as mentioned earlier, is correlated with higher overhead. Additionally, longer GDG paths complicate scheduling and partitioning for mapping techniques like QUBO, further contributing to the difficulty of optimizing these circuits.
- **IG edge connectivity and inter-core moves**: The number of inter-core moves also positively correlates with the edge connectivity of the IG . High edge connectivity makes it challenging to divide the graph into independent subgraphs, making it difficult to map the IG across multiple cores in a MC architecture. As a result, more inter-core communications are required to facilitate interactions between qubits, increasing the overhead.

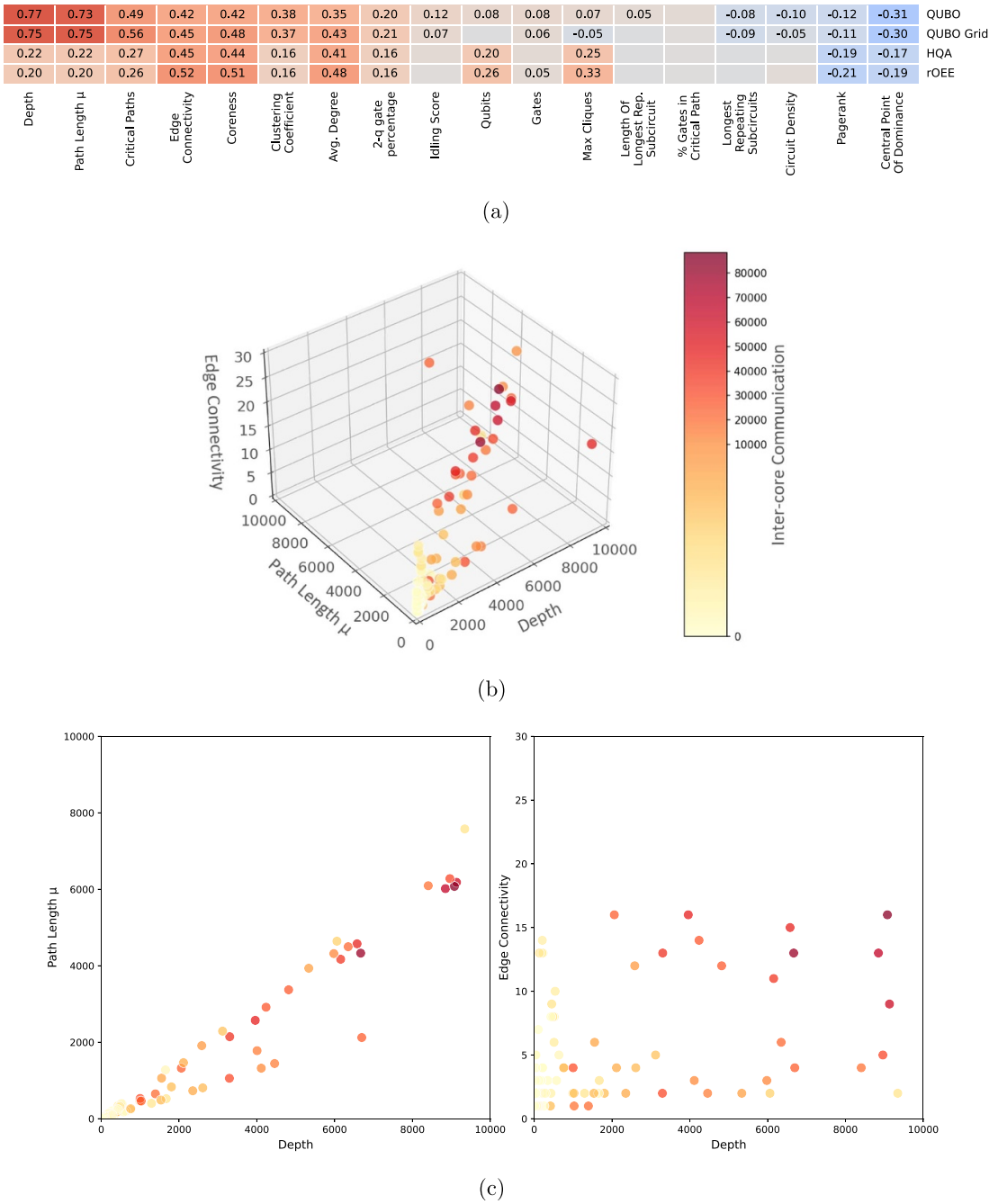

Figure 6. (a) Correlation of the number of inter-core communications performance metric with circuit parameters while using 3 different mappers: QUBO, HQA, and rOEE for all-to-all connected cores, and QUBO mapper for grid-like core connectivity. Metrics are ordered from the most positively correlated to the most negatively correlated one. (b) Three highly correlated parameters for the QUBO mapper and the all-to-all connected topology: edge connectivity, mean of GDG path length and circuit depth, and their number of inter-core moves. (c) Two-dimensional version of the figure in (b).

Similar to SC devices, we also analyze circuit families and their performance patterns for modular architecture. Figure 7 shows the groups of circuits that are part of size cluster 0 and their number of moves (inter-core communications) for the two mappers, HQA (a) and rOEE (b). Overall, the HQA mapper performed better, scaling up to 10 000 moves (with only one outlier with 25 000) compared to 12 000 for rOEE. rOEE reached that number only for group 4 (pink), making HQA a better choice for this group. Group 0 performed particularly well with both mappers, with all benchmarks having up to 100 moves. Group 4 showcased the worst performance for HQA, reaching up to 10 000 moves, but still significantly better than rOEE. Groups 1 and 5 (blue and grey, respectively), on the other hand, showed, on average, a better performance with rOEE, where group 5 outperformed most other groups, despite the higher qubit count. Note that here we do not showcase QUBO results due to lower amount of successfully compiled instances, which would lead to unfair comparison. The results are however available at location shown in appendix D.

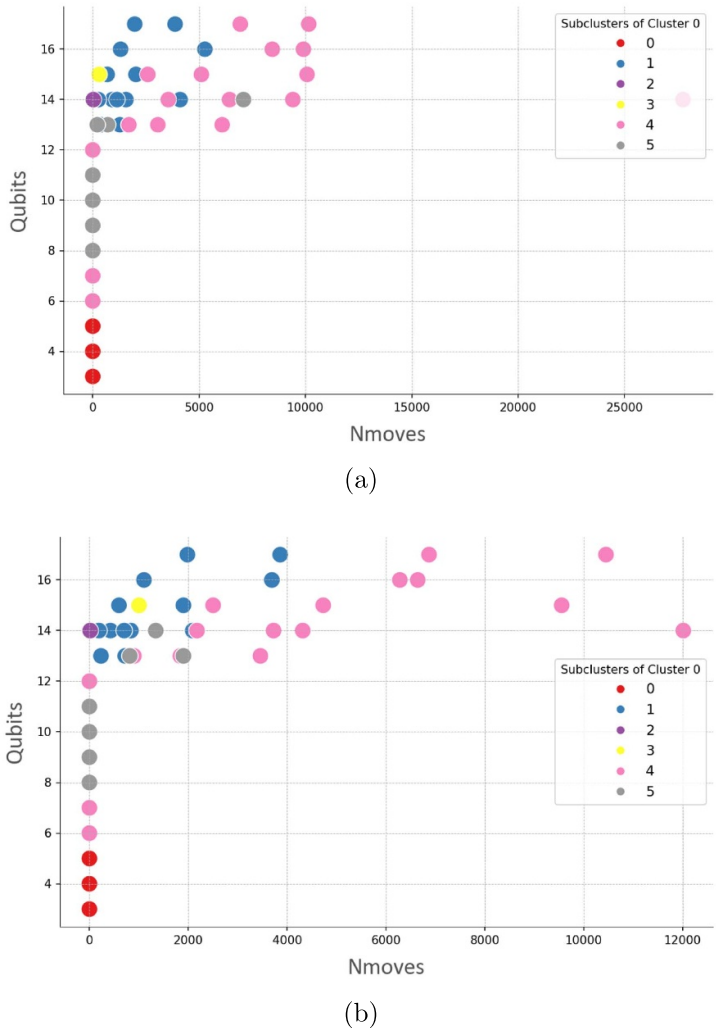

Figure 7. Relation of circuit families (sub-clusters) of cluster 0 with the number of inter-core communications when using all-to-all core connectivity and (a) HQA mapper and (b) rOEE mapper.

The main insights are as follows:

- there is an evident correlation between the structural parameters of the circuits and their performance, which varies with different compilation and device configurations. This confirms that adapting the quantum system layers to the circuit properties and employing hardware-software co-design is key to successful quantum circuit execution.
- It is possible to make appropriate compilation/device choices per circuit family instead of per single circuit, as there are preferred combinations of device and compilation for each group of similar circuits.
- IG parameters are most important for less connected device topologies.
- Idling and density scores are always of high importance for SC devices.
- GDG-related parameters and circuit depth are the most relevant metrics for modular architectures, and they are more significant than for SC architectures.
- Contrary to our expectations, standard parameters like the number of gates, qubits, and the percentage of two-qubit gates, as well as repetitive sub-circuit metrics, are not that significant for circuit success rates.

5.3. Evaluating clusters based on circuit origin

The final part of this section showcases the types of clusters created based on the origin of the circuits. We aim to validate that circuits within the same families exhibit similarities not only in their extracted structural parameters but also in their origin. For instance, it stands to reason that randomly generated circuits would share similar structural parameters. This validation would also support the comprehensiveness of the parameter set we used to describe the circuits.

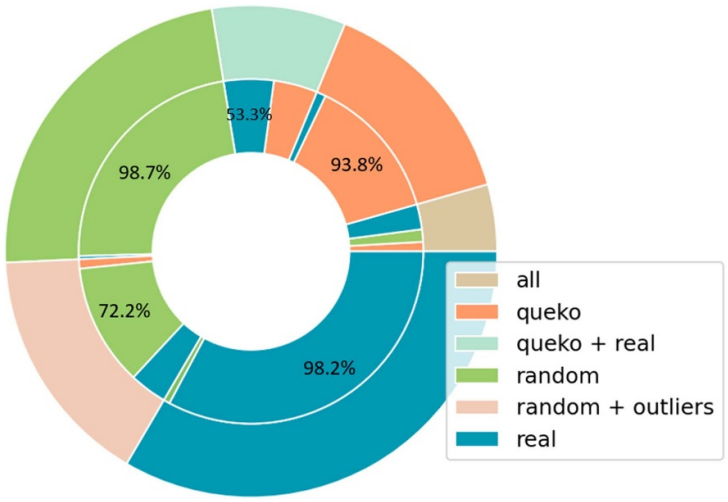

Figure 8. The distribution of the extracted circuit families based on MC-architectures circuit parameters per circuit type: each segment in the outer circle represents groups characterized by the same types of circuits (e.g. the blue segment encompasses circuits based on real algorithms). The inner circle displays the quantity of each circuit type within the respective outer circle segment.

For this purpose, we labeled three groups: real algorithms (based on actual quantum algorithms like Grover's or arithmetic circuits), randomly generated circuits, and QUEKO circuits (synthetic circuits with predefined depth and gate count)[72]. The circuits were sourced from [22]. Figure 8 shows the distribution of the existing sub-circuits after a two-level clustering for the modular-architecture-related metrics. We observe that circuits with the same origin generally belong to the same group, with a few outliers. The exceptions are a very small percentage of mixed benchmarks (noted in light brown) and sub-clusters containing both QUEKO and real benchmarks (in mint-green). Since QUEKO circuits aim to mimic realistic behavior more closely than classical random circuits [72], the existence of these mixed groups is expected. The mixed benchmarks group comprises those circuits that are structurally unique compared to other circuits of the same type. This figure confirms that our defined circuit parameters effectively identify structural similarities between circuits, especially highlighting the distinction of randomly generated circuits (green) compared to the other two types that contain logical patterns and oracles.

The complete set of results (for all benchmarks, clusters, compilers and devices) can be found at https://github.com/QML-Group/QuantumCircuitProfiling.

6. Conclusion

To advance the development of future quantum computers and increase the probability of successfully executing algorithms that are currently unfeasible on classical computers, it is crucial to understand their structure. Quantum circuits (i.e. executable versions of quantum algorithms) will yield the highest success rates when run on a quantum system tailored for it. In classical computing, we see two trends that could also shape the future of quantum computers: a) application-specific devices optimized for particular tasks, and b) MC computation as a solution for current scalability problems.

In this paper, we explored these aspects within the quantum computing field by introducing the most comprehensive characterization of quantum circuit structures to date. This characterization addresses all facets of quantum circuits: circuit size, qubit interactions, gate density and dependencies, and repetitive patterns. For this purpose, we defined 15 and 19 circuit features of importance for SC and MC quantum devices, respectively. We made special emphasis on modular (i.e. MC) architectures, which represent the future of quantum computing and a solution to the scalability challenge of current NISQ devices. MC architecture and related compilation techniques function differently than current quantum systems, because the focus is primarily on reducing the expensive communication between cores. Consequently, extra circuit parameters should be considered for MC architectures.

The extracted parameters were used to create families of similarly structured circuits by utilizing the k-means clustering technique. Identifying circuit families simplifies the development of high-performance architectures (by optimizing for families instead of individual circuits), helps to create a representative set of circuits (using cluster representatives), and aids in approximating the performance of future circuits without running them. We also analyzed the types of circuits within the clusters and found that similarly created

circuits (real-algorithms, QUEKO, uniformly random) are mostly grouped together. This confirms that our characterization can comprehensively describe a circuit and identify structural similarities between different ones.

Furthermore, we conducted experiments with six different devices and seven mapping configurations to showcase varying correlation levels and performance patterns in terms of gate overhead, depth overhead, fidelity decrease, and number of inter-core communications metrics. These experiments revealed which parameters influence circuit success rates the most, and how these correlations change with different devices or compilers. We could also see that some mappers scale better with the same circuit families than others (e.g. HQA vs. rOEE mapper). Therefore, the patterns observed in clustered circuits can help identify the best existing mapper and device to use, particularly for higher numbers of qubits.

In our future work we plan to:

- **expand the set of benchmarks used in the clustering procedure**: the current set of benchmarks is limited in terms of the number of instances per algorithm. We aim to achieve more precisely defined clusters of algorithms by using a more balanced number of instances for each algorithm, covering a wider range of qubit counts. Additionally, we intend to incorporate synthetically-generated real algorithm-like circuits, described in [87].
- **broaden the range of technologies and mapping techniques**: we plan to explore how different circuit properties impact results across a broader set of quantum technologies and corresponding mapping techniques. This will provide a comparative analysis to the findings of our current research;
- **incorporate inter-core traffic parameters for modular architectures**: we aim to integrate inter-core traffic parameters for modular quantum architectures, as introduced in [58], to investigate how these parameters are influenced by circuit characteristics;
- **conduct design space exploration of full-stack quantum computing systems**: we will perform an exploration of the full design space, including circuit properties, compiler options, and a variety of quantum devices. This will allow us to develop architectural design guidelines for future quantum systems;
- **leverage circuit parameters for performance prediction**: finally, we aim to utilize circuit parameters to predict the performance of different mapper-device setups, providing insights into optimizing performance across various quantum computing configurations.

Overall, the proposed method and current findings can aid in the development of quantum systems by incorporating information about circuit structure and providing deeper insights into the variability of outcomes when executing different quantum circuits. Structural parameters of circuits can also be used to predict fidelity decrease, gate overhead, and depth overhead for specific processors and compilation techniques without executing them on actual devices. This can facilitate the co-design of quantum compilers, processors, and applications, ultimately contributing to the development of high-performance application-specific quantum systems. An in-depth analysis of quantum circuits used for benchmarking quantum computing systems is a systematic approach that directly contributes to assessing current devices and lays the foundation for designing future scalable architectures.

Data availability statement

The data that support the findings of this study is openly available following an embargo at the following URL/DOI: https://github.com/QML-Group/QuantumCircuitProfiling [88].

Acknowledgment

S F and M B acknowledge funding of Intel Corporation. CGA acknowledges funding from the Spanish Ministry of Science, Innovation and Universities through the Beatriz Galindo program 2020 (BG20-00023), from project PCI2022-133004 funded by MCIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR, and from the Ministry for Digital Transformation and of Civil Service of the Spanish Government through the QUANTUM ENIA project call—Quantum Spain project, and by the European Union through the Recovery, Transformation and Resilience Plan—NextGenerationEU within the framework of the Digital Spain 2026 Agenda. C G A, E A and S A also acknowledge support from the E U, grant HORIZON-EIC-2022-PATHFINDEROPEN-01-101099697. Finally, S A acknowledges support from the E U, Grant ERC-StG-2021-WINC-101042080.

Appendix A. Selected quantum benchmarks

Table A1. Benchmarks used for the experiments.

Benchmark	Qubits	Gates	Two-qubit gate percentage
0410 184_169	14	211	0.49 289 0995
15_enc	15	264	0.52 272 7273
16QBT_100CYC_QSE_1	16	1400	0.327 142 857
16QBT_10CYC_TFL_1	16	1473	0.330 617 787
16QBT_15CYC_TFL_2	16	1582	0.335 651 075
16QBT_20CYC_TFL_3	16	1727	0.341 053 851
16QBT_35CYC_TFL_6	16	1980	0.348 484 848
16QBT_40CYC_TFL_7	16	2269	0.355 222 565
16QBT_500CYC_QSE_3	16	7949	0.302 679 582
16QBT_700CYC_QSE_5	16	15 901	0.292 182 882
16QBT_900CYC_QSE_7	16	26 125	0.288 076 555
20QBT_100CYC_QSE_8	20	27 545	0.287 747 323
20QBT_400CYC_QSE_8	20	33 225	0.286 711 813
20QBT_45CYC_0D1_1D2_0	15	33 270	0.287 676 586
20QBT_45CYC_0D1_1D2_5	13	33 315	0.288 638 751
20QBT_45CYC_0D1_2D2_0	20	33 405	0.290 555 306
20QBT_45CYC_0D1_2D2_5	20	33 495	0.292 461 561
20QBT_45CYC_0D1_2D2_9	20	33 585	0.294 3576
20QBT_45CYC_0D1_3D2_0	20	33 720	0.297 182 681
20QBT_45CYC_0D1_3D2_5	20	33 855	0.299 985 231
20QBT_45CYC_0D1_3D2_9	20	33 990	0.302 765 519
20QBT_45CYC_0D1_4D2_0	20	34 170	0.306 438 396
20QBT_45CYC_0D1_5D2_1	20	34 395	0.310 975 432
20QBT_45CYC_0D1_6D2_2	20	34 665	0.316 342 132
20QBT_45CYC_0D1_7D2_3	20	34 980	0.322 498 571
20QBT_45CYC_0D1_8D2_4	20	35 340	0.329 400 113
20QBT_45CYC_1D1_1D2_5	20	35 475	0.329 415 081
20QBT_45CYC_1D1_2D2_6	20	35 655	0.330 276 259
20QBT_45CYC_1D1_3D2_7	20	35 880	0.331 967 67
20QBT_45CYC_1D1_4D2_8	20	36 150	0.334 467 497
20QBT_45CYC_1D1_5D2_9	20	36 465	0.337 748 526
20QBT_45CYC_1D1_6D2_0	20	36 825	0.341 778 683
20QBT_45CYC_1D1_7D2_1	20	37 230	0.346 521 622
20QBT_45CYC_2D1_2D2_3	20	37 500	0.346 426 667
20QBT_45CYC_2D1_3D2_4	20	37 815	0.347 110 935
20QBT_45CYC_2D1_4D2_5	20	38 175	0.348 552 718
20QBT_45CYC_2D1_5D2_6	20	38 580	0.350 725 765
20QBT_45CYC_2D1_6D2_7	20	39 030	0.353 599 795
20QBT_45CYC_3D1_1D2_8	17	39 345	0.351 912 568
20QBT_45CYC_3D1_2D2_9	20	39 705	0.350 988 54
20QBT_45CYC_3D1_3D2_0	20	40 110	0.350 810 272
20QBT_45CYC_3D1_4D2_1	20	40 560	0.351 356 016
20QBT_45CYC_3D1_5D2_2	20	41 055	0.352 600 171
20QBT_45CYC_4D1_1D2_3	19	41 460	0.350 241 196
20QBT_45CYC_4D1_2D2_4	20	41 910	0.348 628 012
20QBT_45CYC_4D1_3D2_5	20	42 405	0.347 742 012
20QBT_45CYC_4D1_4D2_6	20	42 945	0.347 560 834
20QBT_45CYC_5D1_1D2_7	20	43 440	0.344 636 28
20QBT_45CYC_5D1_2D2_8	20	43 980	0.342 451 114
20QBT_45CYC_5D1_3D2_9	20	44 565	0.340 985 078
20QBT_45CYC_6D1_2D2_1	20	45 195	0.338 223 255
20QBT_500CYC_QSE_7	20	52 295	0.330 547 854
20QBT_800CYC_QSE_4	20	63 655	0.321 828 607
20QBT_900CYC_QSE_3	20	76 435	0.315 117 42
3_17_13	3	76 471	0.315 191 38

(Continued.)

Table A1. (Continued.)

Benchmark	Qubits	Gates	Two-qubit gate percentage
4gt10-v1_81	5	76 619	0.315 443 95
4gt11_82	5	76 646	0.315 567 675
4gt12-v0_87	6	76 893	0.316 010 56
4gt13_92	5	76 959	0.316 129 368
4gt4-v0_72	6	77 217	0.316 536 514
4gt5_75	5	77 300	0.316 688 228
4mod5-bdd_287	7	77 370	0.316 802 378
4mod7-v0_94	5	77 532	0.317 069 081
4_49_16	5	77 749	0.317 457 459
53QBT_700CYC_QSE_3	53	104 091	0.308 412 831
54QBT_10CYC_QSE_9	54	104 475	0.308 312 994
54QBT_15CYC_QSE_8	54	105 051	0.308 164 606
54QBT_200CYC_QSE_7	54	112 719	0.306 363 612
54QBT_25CYC_QSE_6	54	113 678	0.306 154 225
54QBT_35CYC_QSE_4	54	115 020	0.305 868 545
54QBT_40CYC_QSE_3	54	116 554	0.305 549 359
54QBT_500CYC_QSE_0	54	135 724	0.302 179 423
54QBT_800CYC_QSE_7	54	166 396	0.298 402 606
9symml_195	11	201 277	0.322 366 689
adder_n10	10	201 571	0.322 218 97
adder_n4	4	201 594	0.322 231 812
adr4_197	13	205 033	0.324 133 188
aj-e11_165	5	205 184	0.324 230 934
alu-bdd_288	7	205 268	0.324 283 376
alu-v0_26	5	205 352	0.324 335 775
alu-v1_28	5	205 389	0.324 364 985
alu-v2_30	6	205 893	0.324 654 068
alu-v2_31	5	206 344	0.324 904 044
alu-v3_34	5	206 396	0.324 938 468
alu-v4_36	5	206 511	0.325 004 479
basis_change_n3	3	206 590	0.324 928 603
basis_trotter_n4	4	208 528	0.323 836 607
bell_n4	4	208 577	0.32 379 409
benstein_vazirani_1b_secret_1	2	208 584	0.323 792 812
benstein_vazirani_28b_secret_64	2	208 645	0.323 707 733
bigadder_n18	18	209 231	0.323 422 437
bv_n14	14	209 272	0.323 421 193
bv_n19	19	209 328	0.323 420 66
C17_204	7	209 795	0.323 677 876
cat_state_n4	4	209 799	0.323 686 004
cm82a_208	8	210 449	0.324 031
cnt3-5_179	16	210 624	0.324 165 337
co14_215	15	228 560	0.333 028 526
con1_216	9	229 514	0.333 452 426
cuccaroAdder_10b	22	230 235	0.333 107 477
CuccaroAdder_1b	4	230 308	0.333 075 707
cuccaroadder_q128	127	232 979	0.333 785 448
cuccaroadder_q16	15	233 258	0.333 827 779
cuccaroadder_q32	31	233 897	0.333 937 588
cuccaroadder_q64	63	235 256	0.334 180 637
cuccaroMultiplier_1b	5	235 432	0.334 066 737
cycle10_2_110	12	241 482	0.336 662 774
dc1_220	11	243 396	0.337 437 756
dc2_222	15	252 858	0.341 147 996
decod24-v1_41	5	252 943	0.341 183 587
deutsch_n2	2	252 948	0.341 180 796
dist_223	13	290 994	0.353 701 451
dnn_n16	16	294 034	0.351 350 524
dnn_n2	2	294 372	0.351 089 778
dnn_n8	8	295 892	0.349 935 111
error_correctiond3_n5	5	296 005	0.349 967 061
ex-1_166	3	296 024	0.349 975 002

(Continued.)

Table A1. (Continued.)

ex3_229	6	296 427	0.350 089 567
f2_232	8	297 633	0.350 434 932
fredkin_n3	3	297 652	0.350 439 439
ghz_q128	127	297 779	0.350 713 113
ghz_q16	15	297 810	0.350 723 616
ghz_q32	31	297 841	0.350 787 836
ghz_q64	64	297 905	0.350 923 952
graycode6_47	6	297 910	0.350 934 846
grover_n2	2	297 926	0.350 922 712
grover_n3	3	298 015	0.350 858 178
grover_operator_8	8	298 506	0.351 554 073
grover_q128_1	127	303 497	0.350 675 624
grover_q16_1	15	304 008	0.350 559 854
grover_q32_1	31	305 159	0.350 338 676
grover_q64	63	307 590	0.349 910 595
ham7_104	7	307 910	0.350 030 853
hwb7_59	8	332 289	0.356 493 895
inc_237	16	342 908	0.358 973 836
ising_model_13	13	343 541	0.358 661 703
iswap_n2	2	343 550	0.358 658 128
life_238	11	365 995	0.363 439 391
linearsolver_n3	3	366 018	0.363 427 482
lpn_n5	3	366 029	0.363 422 024
majority_239	7	366 641	0.363 543 63
max46_240	10	393 767	0.368 578 372
miller_11	3	393 817	0.368 589 98
mini-alu_167	5	394 105	0.368 640 337
misex1_241	15	398 918	0.369 456 881
mlp4_245	16	417 770	0.372 489 647
mod10_171	5	418 014	0.372 530 585
mod5adder_127	6	418 569	0.372 607 623
mod5d2_64	5	418 622	0.372 620 168
multipler_n15	15	419 880	0.372 089 645
multiply_n13	11	420 020	0.372 027 522
one-two-three-v1_99	5	420 152	0.372 051 067
plus63mod4096_163	13	548 896	0.387 408 544
pm1_249	14	550 672	0.3875 592
q = 10_s = 19 990_2qbf = 02_1	10	570 672	0.380 798 427
q = 10_s = 19 990_2qbf = 05_1	10	590 672	0.384 963 228
q = 10_s = 2990_2qbf = 03_1	10	593 672	0.384 581 048
q = 10_s = 29 990_2qbf = 05_1	10	623 672	0.390 166 626
q = 10_s = 39 990_2qbf = 08_1	10	663 672	0.414 793 754
q = 10_s = 49 990_2qbf = 09_1	10	713 672	0.448 717 338
q = 10_s = 50_2qbf = 096_1	10	713 732	0.448 746 869
q = 10_s = 90_2qbf = 011_1	9	713 832	0.448 698 013
q = 10_s = 990_2qbf = 091_1	10	714 832	0.449 327 954
q = 11_s = 19 989_2qbf = 01_1	11	734 832	0.439 857 001
q = 11_s = 2989_2qbf = 02_1	11	737 832	0.438 870 908
q = 11_s = 29 989_2qbf = 03_1	11	767 832	0.433 373 446
q = 11_s = 39 989_2qbf = 05_1	11	807 832	0.436 708 375
q = 11_s = 49 989_2qbf = 08_1	11	857 832	0.458 122 336
q = 11_s = 49_2qbf = 061_1	11	857 892	0.458 126 431
q = 11_s = 89_2qbf = 022_1	10	857 992	0.458 092 849
q = 11_s = 989_2qbf = 081_1	11	858 992	0.458 500 196
q = 12_s = 19 988_2qbf = 01_1	12	878 992	0.450 404 554
q = 12_s = 29 988_2qbf = 03_1	12	908 992	0.445 463 766
q = 12_s = 49 988_2qbf = 05_1	12	958 992	0.448 407 286
q = 12_s = 59 988_2qbf = 09_1	12	1018 992	0.475 073 406
q = 12_s = 9988_2qbf = 01_1	12	1028 992	0.471 399 195
q = 13_s = 19 987_2qbf = 01_1	13	1048 992	0.464 347 679
q = 13_s = 29 987_2qbf = 02_1	13	1078 992	0.457 022 851
q = 13_s = 49 987_2qbf = 03_1	13	1128 992	0.449 958 901
q = 13_s = 59 987_2qbf = 05_1	13	1188 992	0.452 428 612
q = 13_s = 9987_2qbf = 08_1	13	1198 992	0.455 379 185

(Continued.)

Table A1. (Continued.)

Benchmark	Qubits	Gates	Two-qubit gate percentage
q = 14_s = 19 986_2qbf = 09_1	14	1218 992	0.462 717 557
q = 14_s = 29 986_2qbf = 01_1	14	1248 992	0.454 013 316
q = 14_s = 39 986_2qbf = 02_1	14	1288 992	0.446 150 17
q = 14_s = 49 986_2qbf = 03_1	14	1338 992	0.440 653 118
q = 14_s = 5986_2qbf = 03_1	14	1344 992	0.440 031 614
q = 14_s = 5986_2qbf = 05_1	14	1350 992	0.440 291 282
q = 14_s = 59 986_2qbf = 08_1	14	1410 992	0.455 618 459
q = 14_s = 986_2qbf = 051_1	14	1411 992	0.455 654 848
q = 14_s = 9986_2qbf = 09_1	14	1421 992	0.458 755 745
q = 15_s = 19 985_2qbf = 01_1	15	1441 992	0.453 747 316
q = 15_s = 29 985_2qbf = 02_1	15	1471 992	0.448 508 552
q = 15_s = 49 985_2qbf = 03_1	15	1521 992	0.443 598 258
q = 15_s = 59 985_2qbf = 05_1	15	1581 992	0.445 789 865
q = 15_s = 985_2qbf = 051_1	15	1582 992	0.445 810 213
q = 15_s = 9985_2qbf = 08_1	15	1592 992	0.448 015 433
q = 16_s = 19 984_2qbf = 09_1	16	1612 992	0.453 625 312
q = 16_s = 29 984_2qbf = 01_1	16	1642 992	0.447 171 38
q = 16_s = 49 984_2qbf = 02_1	16	1692 992	0.439 795 345
q = 16_s = 59 984_2qbf = 03_1	16	1752 992	0.434 988 865
q = 16_s = 984_2qbf = 051_1	16	1753 992	0.435 031 061
q = 17_s = 19 983_2qbf = 05_1	17	1773 992	0.435 784 942
q = 17_s = 2983_2qbf = 08_1	17	1776 992	0.436 393 636
q = 17_s = 29 983_2qbf = 09_1	17	1806 992	0.444 058 413
q = 17_s = 43_2qbf = 028_1	12	1807 052	0.444 050 31
q = 17_s = 49 983_2qbf = 01_1	17	1857 052	0.434 774 578
q = 17_s = 5983_2qbf = 02_1	17	1863 052	0.434 022 239
q = 17_s = 59 983_2qbf = 03_1	17	1923 052	0.429 783 49
q = 17_s = 983_2qbf = 031_1	17	1924 052	0.429 709 28
q = 17_s = 9983_2qbf = 05_1	17	1934 052	0.430 105 292
q = 3_s = 19 997_2qbf = 01_1	3	1954 052	0.426 738 388
q = 3_s = 2997_2qbf = 01_1	3	1957 052	0.426 255 92
q = 3_s = 2997_2qbf = 02_1	3	1960 052	0.425 914 721
q = 3_s = 29 997_2qbf = 03_1	3	1990 052	0.423 986 408
q = 3_s = 39 997_2qbf = 05_1	3	2030 052	0.425 523 09
q = 3_s = 57_2qbf = 011_1	3	2030 112	0.425 511 991
q = 3_s = 5997_2qbf = 08_1	3	2036 112	0.426 616 021
q = 3_s = 59 997_2qbf = 09_1	3	2096 112	0.440 210 256
q = 3_s = 97_2qbf = 01_1	3	2096 212	0.440 194 026
q = 3_s = 997_2qbf = 02_1	3	2097 212	0.440 084 264
q = 3_s = 9997_2qbf = 03_1	3	2107 212	0.439 416 632
q = 4_s = 19 996_2qbf = 02_1	4	2127 212	0.437 199 959
q = 4_s = 19 996_2qbf = 05_1	4	2147 212	0.437 750 441
q = 4_s = 2996_2qbf = 08_1	4	2150 212	0.438 275 389
q = 4_s = 29 996_2qbf = 09_1	4	2180 212	0.444 609 056
q = 4_s = 39 996_2qbf = 01_1	4	2220 212	0.438 430 654
q = 4_s = 49 996_2qbf = 02_1	4	2270 212	0.433 220 774
q = 4_s = 49 996_2qbf = 09_1	4	2320 212	0.443 277 166
q = 4_s = 56_2qbf = 032_1	4	2320 272	0.443 273 03
q = 4_s = 96_2qbf = 052_1	4	2320 372	0.443 277 199
q = 4_s = 996_2qbf = 03_1	4	2321 372	0.443 202 985
q = 4_s = 9996_2qbf = 05_1	4	2331 376	0.443 385 794
q = 5_s = 19 995_2qbf = 03_1	5	2351 376	0.442 204 479
q = 5_s = 19 995_2qbf = 08_1	5	2371 376	0.445 262 582
q = 5_s = 2995_2qbf = 09_1	5	2374 376	0.445 835 874
q = 5_s = 39 995_2qbf = 01_1	5	2414 376	0.440 142 298
q = 5_s = 49 995_2qbf = 02_1	5	2464 376	0.435 259 473
q = 5_s = 55_2qbf = 087_1	5	2464 436	0.435 269 571
q = 5_s = 95_2qbf = 095_1	5	2464 536	0.435 288 428
q = 5_s = 9995_2qbf = 03_1	5	2474 536	0.434 721 903
q = 6_s = 19 994_2qbf = 05_1	6	2494 536	0.435 235 651
q = 6_s = 2994_2qbf = 08_1	6	2497 536	0.435 676 202

(Continued.)

Table A1. (Continued.)

q = 6_s = 29 994_2qbf = 09_1	6	2527 536	0.441 167 604
q = 6_s = 49 994_2qbf = 01_1	6	2577 536	0.434 544 852
q = 6_s = 54_2qbf = 022_1	6	2577 596	0.434 540 168
q = 6_s = 94_2qbf = 053_1	6	2577 696	0.434 543 484
q = 6_s = 9994_2qbf = 02_1	6	2587 696	0.433 640 196
q = 7_s = 19 993_2qbf = 03_1	7	2607 696	0.432 629 417
q = 7_s = 2993_2qbf = 05_1	7	2610 696	0.432 714 495
q = 7_s = 2993_2qbf = 08_1	7	2613 696	0.433 130 708
q = 7_s = 29 993_2qbf = 08_1	7	2643 696	0.437 292 336
q = 7_s = 39 993_2qbf = 09_1	7	2683 696	0.444 200 088
q = 7_s = 53_2qbf = 034_1	7	2683 756	0.444 197 61
q = 7_s = 59 993_2qbf = 09_1	7	2743 756	0.454 141 695
q = 7_s = 93_2qbf = 054_1	7	2743 856	0.454 144 095
q = 7_s = 993_2qbf = 081_1	7	2744 856	0.454 264 996
q = 8_s = 19 992_2qbf = 02_1	8	2764 856	0.452 427 902
q = 8_s = 2992_2qbf = 01_1	8	2767 856	0.452 036 883
q = 8_s = 2992_2qbf = 03_1	8	2770 856	0.451 846 289
q = 8_s = 29 992_2qbf = 05_1	8	2800 856	0.452 341 356
q = 8_s = 39 992_2qbf = 08_1	8	2840 856	0.457 238 593
q = 8_s = 49 992_2qbf = 09_1	8	2890 856	0.464 903 129
q = 8_s = 52_2qbf = 104_1	8	2890 916	0.464 911 468
q = 8_s = 92_2qbf = 011_1	8	2891 016	0.464 901 267
q = 8_s = 992_2qbf = 081_1	8	2892 016	0.465 010 567
q = 9_s = 19 991_2qbf = 08_1	9	2912 016	0.467 286 924
q = 9_s = 2991_2qbf = 01_1	9	2915 016	0.466 909 959
q = 9_s = 51_2qbf = 012_1	7	2915 076	0.466 902 75
q = 9_s = 51_2qbf = 059_1	9	2915 136	0.466 902 745
q = 9_s = 91_2qbf = 088_1	9	2915 236	0.466 914 514
q = 9_s = 991_2qbf = 091_1	9	2916 236	0.467 064 394
qaoaWS_128	128	2917 388	0.467 055 462
qaoaWS_16	16	2917 532	0.467 054 346
qaoaWS_32	32	2917 820	0.467 052 114
qaoaWS_64	64	2918 396	0.467 047 652
qaoa_128	128	2931 068	0.467 823 333
qaoa_16	15	2931 579	0.467 790 907
qaoa_32	31	2932 730	0.467 721 884
qaoa_64	64	2935 937	0.467 895 599
qaoa_n6	6	2936 351	0.467 848 02
qec_en_n5	5	2936 376	0.467 847 442
qec_sm_n5	5	2936 401	0.467 846 864
qft_8	8	2936 577	0.467 841 981
qft_n15	15	2937 117	0.467 827 465
qft_n20	20	2938 087	0.467 802 349>
qft_q128	128	2970 919	0.468 104 314
qft_q16	16	2971 439	0.468 103 165
qft_q32	32	2973 503	0.468 111 853
qft_q64	64	2981 727	0.468 172 975
QuantumVolume_128	128	3178 463	0.454 658 745
QuantumVolume_16	16	3181 551	0.454 458 847
QuantumVolume_32	32	3193 871	0.453 667 665
QuantumVolume_64	64	3243 087	0.450 571 94
quantum_volume_8	8	3243 951	0.450 481 527
quantum_volume_n5	5	3244 289	0.450 445 691
queko_128	128	3252 482	0.450 318 557
queko_16	16	3252 611	0.450 316 684
queko_32	32	3253 124	0.450 308 688
queko_64	64	3255 173	0.450 277 143
queko_8	8	3255 205	0.450 276 403
radd_250	13	3258 418	0.450 263 594
rd32_270	5	3258 502	0.450 263 035
rd53_311	13	3258 777	0.450 263 089
rd73_140	10	3259 007	0.450 263 224
rd73_252	10	3264 328	0.450 239 682
rd84_142	15	3264 671	0.450 239 549

(Continued.)

Table A1. (Continued.)

root_255	13	3281 830	0.450 168 656
sao2_257	14	3320 407	0.450 017 423
sat_n11	11	3321 884	0.449 893 193
seca_n11	11	3322 217	0.449 873 383
sf_274	6	3322 998	0.449 868 763
shor_15	11	3327 790	0.449 758 248
shor_35	15	3344 319	0.449 359 944
simon_n6	5	3344 401	0.449 353 113
sqn_258	10	3354 624	0.449 312 948
sqrt8_260	12	3357 633	0.449 301 636
squar5_261	13	3359 626	0.449 293 761
square_root_7	15	3367 256	0.449 193 052
sym9_148	10	3388 760	0.449 118 852
sys6-v0_111	10	3388 975	0.449 119 276
teleportation_n3	3	3388 983	0.449 118 806
toffoli_n3	3	3389 001	0.449 118 191
urf5_280	9	3438 830	0.449 520 913
variational_n4	4	3438 884	0.449 518 507
vbeAdder_1b	4	3438 954	0.449 513 428
vbeAdder_5b	16	3439 584	0.449 467 726
VQEHEA1_128	128	3443 414	0.449 336 618
VQEHEA1_16	16	3443 884	0.449 318 85
VQEHEA1_32	32	3444 834	0.449 284 929
VQEHEA1_64	64	3446 744	0.449 218 741
VQEHEA2_128	128	3450 574	0.449 088 181
VQEHEA2_16	16	3451 044	0.449 070 484
VQEHEA2_32	32	3451 994	0.449 036 702
VQEHEA2_64	64	3453 904	0.448 970 788
vqe_uccsd_n4	4	3454 124	0.448 967 669
vqe_uccsd_n6	6	3456 406	0.448 975 612
vqe_uccsd_n8	8	3467 214	0.449 158 892
wim_266	11	3468 200	0.449 154 316
wstate_n3	3	3468 249	0.449 150 566
xor5_254	6	3468 256	0.449 151 101
z4_268	11	3471 329	0.449 140 372

Appendix B. Generating GDG parameters

The construction of the GDG is done by a linear scan of the circuit, adding a new node w for each gate. For each newly added gate node operating on n qubits $(q_i)_{i=1}^{n}$, the algorithm adds incoming edges $(v_1, w), \ldots, (v_n, w)$ such that v_i is the last gate operating on qubit q_i. This last information is stored in and retrieved from an array that maps qubit indices to the last gate using said qubit. When no such gate exists, the edge comes from an extra sentinel node called **source**. After all gates are processed, all children-less nodes are linked to another sentinel node called the **sink**.

Since a quantum gate cannot depend on itself, the resulting directed graph is always *acyclic*, therefore constitutes a directed acyclic graph (DAG). We ignore the trivial case where this DAG is not weakly connected. Any *topological ordering* of the nodes gives a valid sequential order for the gates during circuit execution.

The *critical path length* is easily computed with an existing library, for instance using `networkx.dag_longest_path_length`. Our other metrics require however custom processing of the graph, since brute-force extraction is doomed to fail for large circuits: for instance, in some DAG families, the *number of critical paths* grows exponentially with the number of circuit gates.

Our solution is to *recursively* compute those metrics. The recurrence relations between a node's values and the values of its children are given in table B1. Our Python implementation (see D) is made iterative instead of recursive using a bottom-up traversal of the GDG, where nodes are iterated in *reversed* topological ordering.

Table B1. Recurrence relations for GDG-related metrics computation. w is the parent and the v_i are the children. The mean and variance metrics recurrence relations use formulas for combined (pooled) mean and variance of multiple sample sets.

Metric	Initialisation and recurrence relation
Number of gates in critical paths to **sink** (L)	$L(\textbf{sink}) = 0$ $L(w) = 1 + \max_i L(v_i)$ From that we can define an edge predicate describing whether a given edge (w, v_i) belongs to a critical path: $\textbf{CP}(w, v_i) = \top \text{if } L(w) = 1 + L(v_i), \perp \text{otherwise.}$
Number of paths to **sink** (n)	$n(\textbf{sink}) = 1$ $n(w) = \sum_i n(v_i)$
Number of critical paths to **sink** (N)	$N(\textbf{sink}) = 1$ $N(w) = \sum_{i \mid \textbf{CP}(w,v_i)} N(v_i)$
Max number of two-qubit gates in critical paths to **sink** (M)	$M(\textbf{sink}) = 0$ $M(w) = \begin{cases} 1 + \max_{i \mid \textbf{CP}(w,v_i)} M(v_i) & \text{if } w \text{ is a 2-qubit gate} \\ \max_{i \mid \textbf{CP}(w,v_i)} M(v_i) & \text{otherwise} \end{cases}$
Number of critical paths to **sink** with max two-qubit gates (K)	$K(\textbf{sink}) = 1$ $K(w) = \sum_{i \mid \textbf{CP}(w,v_i) \wedge M(v_i)+1=M(w)} K(v_i)$
Mean length of paths to **sink** (m)	$m(\textbf{sink}) = 0$ $m(w) = \frac{1}{n(w)} \sum_i n(v_i)\,(m(v_i) + 1)$
Variance of length of paths to **sink** (v)	$v(\textbf{sink}) = 0$ $v(w) = \frac{1}{n(w)} \sum_i n(v_i) \left(v(v_i) + (m(v_i) + 1 - m(w))^2 \right)$

Appendix C. Clustering settings and examples

Table C1. K-means settings: single-core architecture.

Cluster	n_clusters	Silhouette coef.
size	5	0.473
0	4	0.312
1	6	0.290
2	7	0.319
3	1	—
4	7	0.352

Table C2. K-means settings: multi-core architecture.

Cluster	n_clusters	Silhouette coef.
size	5	0.473
0	6	0.386
1	6	0.342
2	10	0.349
3	1	—
4	6	0.288

To cluster the benchmarks according to the specified parameters, we utilize K-Means, a centroid-based clustering algorithm [67]. The configuration of this algorithm is detailed in table C2, where the first column indicates the number of size-based clusters, the second column represents the number of structure-based sub-clusters within each cluster, and the third column showcases their respective Silhouette coefficients, guiding our selection process. An illustration of a cluster and its sub-clusters is provided in figures C1 and C2 for reference.

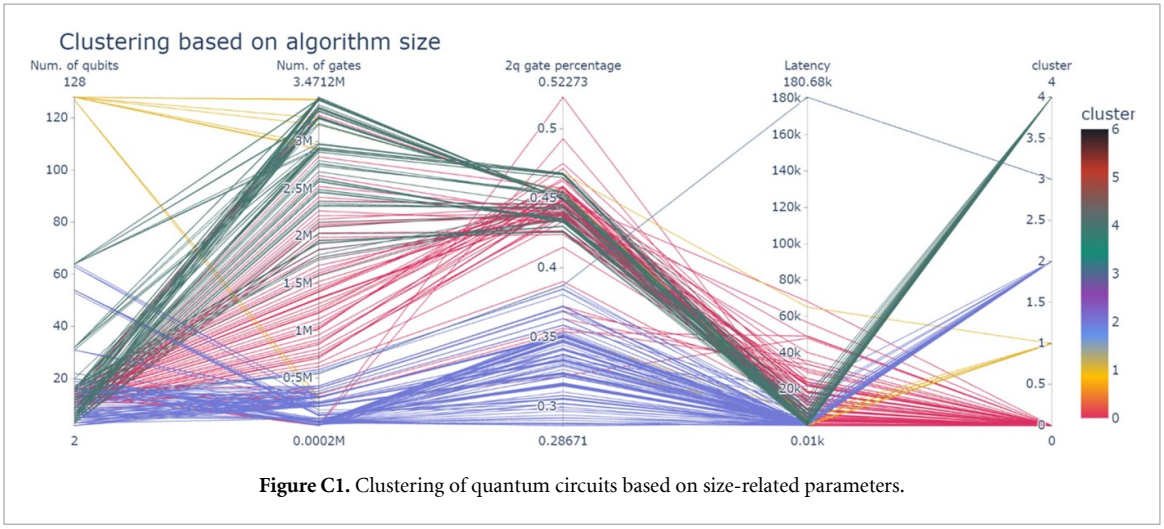

Figure C1. Clustering of quantum circuits based on size-related parameters.

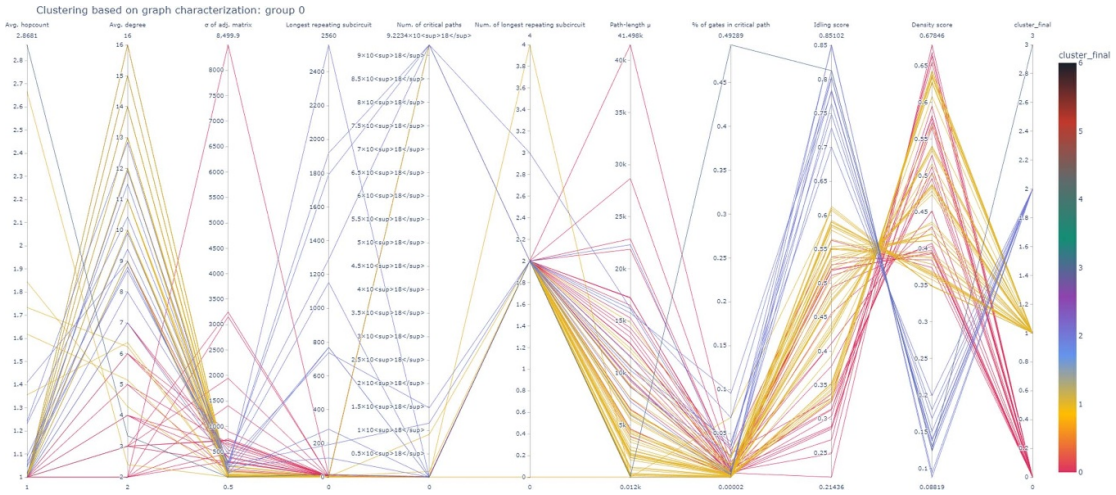

Figure C2. Sub-clustering of quantum circuits of cluster 0 (figure C1) based on other structural parameters.

Appendix D. Software availability

The online GitHub repository [88] contains the following:

(i) benchmarks QASM files;
(ii) code for extracting all circuit parameters and clustering together with prerequisites;
(iii) code for simulations for different compilers and architectures;
(iv) tables containing results: circuit parameters and compilation for all selected benchmarks;
(v) tables containing final clusters of benchmarks; and
(vi) plots of other extracted results.

ORCID iDs

Medina Bandic https://orcid.org/0000-0003-4670-0988
Pau Escofet https://orcid.org/0000-0003-3372-1931
Sahar Ben Rached https://orcid.org/0009-0004-7768-4997
Santiago Rodrigo https://orcid.org/0000-0001-8843-5276
Sebastian Feld https://orcid.org/0000-0003-2782-1469

References

[1] Preskill J 2018 Quantum computing in the nisq era and beyond *Quantum* **2** 79
[2] Rodrigo S, Abadal S, Alarcón E, Bandic M, van Someren H and Garcia Almudever C 2021 On double full-stack communication-enabled architectures for multicore quantum computers *IEEE Micro* **41** 48–56
[3] Ovide A, Rodrigo S, Bandic M, Van Someren H, Feld S, Abadal S, Alarcon E and Almudever C G 2023 Mapping quantum algorithms to multi-core quantum computing architectures *Proc. ISCAS'23*
[4] Sarovar M, Proctor T, Rudinger K, Young K, Nielsen E and Blume-Kohout R 2020 Detecting crosstalk errors in quantum information processors *Quantum* **4** 321
[5] Bravyi S, Dial O, Gambetta J M, Gil D and Nazario Z 2022 The future of quantum computing with superconducting qubits *J. Appl. Phys.* **132** 75
[6] Ang J, et al 2022 Architectures for multinode superconducting quantum computers (arXiv:2212.06167)
[7] Monroe C, Raussendorf R, Ruthven A, Brown K R, Maunz P, Duan L-M and Kim J 2014 Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects *Phys. Rev.* A **89** 17
[8] LaRacuente N, Smith K N, Imany P, Silverman K L and Chong F T 2023 Modeling short-range microwave networks to scale superconducting quantum computation (arXiv:2201.08825)
[9] Jnane H, Undseth B, Cai Z, Benjamin S C, and Koczor B 2022 Multicore quantum computing (arXiv:2201.08861)
[10] Smith K N, Subramanian Ravi G, Baker J M and Chong F T 2022 Scaling superconducting quantum computers with chiplet architectures *55th Int. Symposium on Microarchitecture (MICRO)* (https://doi.org/10.1109/MICRO56248.2022.00078)
[11] Bandic M, Zarein H, Alarcon E and Almudever C G 2020 On structured design space exploration for mapping of quantum algorithms *2020 XXXV Conf. on Design of Circuits and Integrated Systems (DCIS)* (IEEE) pp 1–6
[12] Baker J M, Duckering C, Hoover A and Chong F T 2020 Time-sliced quantum circuit partitioning for modular architectures *Proc. 17th ACM Int. Conf. on Computing Frontiers* pp 98–107
[13] Bandic M, Feld S and Almudever C G 2022 Full-stack quantum computing systems in the nisq era: algorithm-driven and hardware-aware compilation techniques *2022 Design, Automation & Test in Europe Conf. & Exhibition (DATE)* (IEEE) pp 1–6

[14] Lao L and Browne D E 2022 2qan: a quantum compiler for 2-local qubit hamiltonian simulation algorithms *ISCA* 351–65
[15] Lubinski T, Johri S, Varosy P, Coleman J, Zhao L, Necaise J, Baldwin C H, Mayer K and Proctor T 2021 Application-oriented performance benchmarks for quantum computing (arXiv:2110.03137)
[16] Mills D, Sivarajah S, Scholten T L, and Duncan R 2020 Application-motivated, holistic benchmarking of a full quantum computing stack (arXiv:2006.01273)
[17] Li G, Ding Y and Xie Y 2020 Towards efficient superconducting quantum processor architecture design *Proc. Twenty-Fifth Int. Conf. on Architectural Support for Programming Languages and Operating Systems* pp 1031–45
[18] Bandic M, Almudever C G and Feld S 2023 Interaction graph-based characterization of quantum benchmarks for improving quantum circuit mapping techniques *Quantum Mach. Intell.* **5** 40
[19] Quetschlich N, Burgholzer L and Wille R 2023 Predicting good quantum circuit compilation options *2023 IEEE Int. Conf. on Quantum Software (QSW)* (IEEE) pp 43–53
[20] Li G, Shi Y, and Javadi-Abhari A 2021 Software–hardware co-optimization for computational chemistry on superconducting quantum processors (arXiv:2105.07127)
[21] Lao L and Browne D 2021 2qan: a quantum compiler for 2-local qubit hamiltonian simulation algorithms (arXiv:2108.02099)
[22] Medina Bandic and Nikiforos Paraskevopulous 2021 qbench benchmark suite (available at: https://github.com/QML-Group/qbench)
[23] Quetschlich N, Burgholzer L and Wille R 2023 Mqt bench: Benchmarking software and design automation tools for quantum computing *Quantum* **7** 1062
[24] Tomesh T, Gokhale P, Omole V, Subramanian Ravi G, Smith K N, Viszlai J, Xin-Chuan W, Hardavellas N, Martonosi M R and Chong F T 2022 Supermarq: A scalable quantum benchmark suite *2022 IEEE Int. Symp. on High-Performance Computer Architecture (HPCA)* (IEEE) pp 587–603
[25] Freedman D, Pisani R and Purves R 2007 *Statistics (International Student edn)* 4th edn (WW Norton & Company)
[26] Rodrigo S, Abadal S, Alarcon E and Almudever C G 2020 Will quantum computers scale without inter-chip comms? a structured design exploration to the monolithic vs distributed architectures quest *2020 XXXV Conf. on Design of Circuits and Integrated Systems (DCIS)* pp 1–6
[27] Kaushal V, Lekitsch B, Stahl A, Hilder J, Pijn D, Schmiegelow C, Bermudez A, Müller M, Schmidt-Kaler F and Poschinger U 2020 Shuttling-based trapped-ion quantum information processing *AVS Quantum Sci.* **2** 014101
[28] Marinelli B, Luo J, Ren H, Niedzielski B M, Kim D K, Das R, Schwartz M, Santiago D I and Siddiqi I 2023 Dynamically reconfigurable photon exchange in a superconducting quantum processor (https://doi.org/10.48550/arXiv.2303.03507)
[29] Almudever C G, Lao L, Wille R and Guerreschi G G 2020 Realizing quantum algorithms on real quantum computing devices *2020 Design, Automation & Test in Europe Conf. & Exhibition (DATE)* (IEEE) pp 864–72
[30] Zulehner A, Paler A and Wille R 2018 An efficient methodology for mapping quantum circuits to the IBM QX architectures *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **38** 1226–36
[31] Gushu Li, Ding Y and Xie Y 2019 Tackling the qubit mapping problem for NISQ-era quantum devices *Int. Conf. on Architectural Support for Programming Languages and Operating Systems* pp 1001–14
[32] Lao L, van Someren H, Ashraf I and Almudever C G 2021 Timing and resource-aware mapping of quantum circuits to superconducting processors *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **41** 359–71
[33] Itoko T, Raymond R, Imamichi T and Matsuo A 2020 Optimization of quantum circuit mapping using gate transformation and commutation *Integration* **70** 43–50
[34] Pozzi M G, Herbert S J, Sengupta A, and Mullins R D 2020 Using reinforcement learning to perform qubit routing in quantum compilers (arXiv:2007.15957)
[35] Jiang H, Deng Y, and Ming X 2021 Quantum circuit transformation based on subgraph isomorphism and tabu search (arXiv:2104.05214)
[36] Steinberg M A, Feld S, Almudever C G, Marthaler M and Reiner J-M 2022 Topological-graph dependencies and scaling properties of a heuristic qubit-assignment algorithm *IEEE Trans. Quantum Eng.* **3** 1–14
[37] Wagner F, Bärmann A, Liers F and Weissenbäck M 2023 Improving quantum computation by optimized qubit routing *J. Optim. Theory Appl.* **197** 1–34
[38] Murali P, Baker J M, Javadi-Abhari A, Chong F T and Martonosi M 2019 Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers *Int. Conf. on Architectural Support for Programming Languages and Operating Systems* pp 1015–29
[39] Tannu S S and Qureshi M K 2019 Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers *Int. Conf. on Architectural Support for Programming Languages and Operating Systems* pp 987–99
[40] Venturelli D, Do M, O'Gorman B, Frank J, Rieffel E, Booth K E C, Nguyen T, Narayan P and Nanda S 2019 Quantum circuit compilation: an emerging application for automated reasoning *Scheduling and Planning Applications Workshop*
[41] Lao L, van Wee B, Ashraf I, van Someren H, Khammassi N, Bertels K and Almudever C G 2019 Mapping of lattice surgery-based quantum circuits on surface code architectures *Quantum Sci. Technol.* **4** 015005
[42] Lao L, Manzano D M, van Someren H, Ashraf I, and Almudever C G 2019 Mapping of quantum circuits onto nisq superconducting processors (arXiv:1908.04226)
[43] Herbert S and Sengupta A 2018 Using reinforcement learning to find efficient qubit routing policies for deployment in near-term quantum computers (arXiv:1812.11619)
[44] Lye A, Wille R and Drechsler R 2015 Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits *Asia and South Pacific Design Automation Conf.* pp 178–83
[45] Li S, Zhou X and Feng Y 2020 Qubit mapping based on subgraph isomorphism and filtered depth-limited search *IEEE Trans. Comput.* **70** 1777–88
[46] Biuki A, Mohammadzadeh N, Wille R and Sargaran S 2022 Exact mapping of quantum circuit partitions to building blocks of the saqip architecture *2022 IEEE Computer Society Annual Symp. on VLSI (ISVLSI)* (IEEE) pp 402–5
[47] Molavi A, Xu A, Diges M, Pick L, Tannu S and Albarghouthi A 2022 Qubit mapping and routing via maxsat *2022 55th IEEE/ACM Int. Symp. on Microarchitecture (MICRO)* (IEEE) pp 1078–91
[48] Moro L, Paris M G A, Restelli M and Prati E 2021 Quantum compiling by deep reinforcement learning *Commun. Phys.* **4** 178
[49] Devulapalli D, Schoute E, Bapat A, Childs A M and Gorshkov A V 2022 Quantum routing with teleportation *Phys. Rev. Res.* **6** 033313
[50] Upadhyay S, Ash Saki A, Onur Topaloglu R and Ghosh S 2022 A shuttle-efficient qubit mapper for trapped-ion quantum computers *Proc. Great Lakes Symp. on VLSI 2022* pp 305–8

[51] Nottingham N, Perlin M A, White R, Bernien H, Chong F T, and Baker J M 2023 Decomposing and routing quantum circuits under constraints for neutral atom architectures (arXiv:2307.14996)
[52] Paraskevopoulos N, Sebastiano F, Almudever C G, and Feld S 2023 Spinq: compilation strategies for scalable spin-qubit architectures (arXiv:2301.13241)
[53] Steinberg M, Bandic M, Szkudlarek S, Almudever C G, Sarkar A, and Feld S 2024 Resource bounds for quantum circuit mapping via quantum circuit complexity (arXiv:2402.00478)
[54] Tan B and Cong J 2021 Optimal qubit mapping with simultaneous gate absorption (arXiv:2109.06445)
[55] Blume-Kohout R and Young K C 2020 A volumetric framework for quantum computer benchmarks *Quantum* **4** 362
[56] Rodrigo S, Bandic M, Abadal S, van Someren H, Alarcón E and Almudéver C G 2021 Scaling of multi-core quantum architectures: a communications-aware structured gap analysis *Proc. of Computing Frontiers'21* pp 144–51
[57] Rodrigo S, Spanò D, Bandic M, Abadal S, Van Someren H, Ovide A, Feld S, Almudéver C G and Alarcón E 2022 Characterizing the spatio-temporal qubit traffic of a quantum intranet aiming at modular quantum computer architectures *Proc. 9th ACM Int. Conf. on Nanoscale Computing and Communication* pp 1–7
[58] Ben Rached S, Lopez Agudo I, Rodrigo S, Bandic M, Feld S, van Someren H, Alarcón E, Almudéver C G and Abadal S 2023 Characterizing the inter-core qubit traffic in large-scale quantum modular architectures (arXiv:2310.01921)
[59] Cuomo D, Caleffi M, Krsulich K, Tramonto F, Agliardi G, Prati E and Sara Cacciapuoti A 2023 Optimized compiler for distributed quantum computing *ACM Trans. Quantum Comput.* **4** 1–29
[60] Ferrari D, Sara Cacciapuoti A, Amoretti M and Caleffi M 2020 Compiler design for distributed quantum computing (arXiv:2012.09680)
[61] Bandic M *et al* 2023 Mapping quantum circuits to modular architectures with qubo *2023 IEEE Int. Conf. on Quantum Computing and Engineering (QCE)* (IEEE Computer Society) pp 790–801
[62] Escofet P, Ovide A, Almudever C G, Alarcón E and Abadal S 2023 Hungarian qubit assignment for optimized mapping of quantum circuits on multi-core architectures *IEEE Comput. Archit. Lett.* **22** 161–4
[63] Escofet P, Ovide A, Bandic M, Prielinger L, van Someren H, Feld S, Alarcón E, Abadal S and Almudéver C G 2024 Revisiting the mapping of quantum circuits: entering the multi-core era *ACM Trans. Quantum Comput.* (https://doi.org/10.1145/3655029)
[64] Martın Hernández J and Van Mieghem P 2011 *Classification of Graph Metrics* (Delft University of Technology) pp 1–20
[65] Brin S and Page L 1998 The anatomy of a large-scale hypertextual web search engine *Comput. Netw.* **30** 107–17
[66] Wikipedia 2024 Longest repeated substring problem (available at: https://en.wikipedia.org/wiki/Longest_repeated_substring_problem)
[67] Lloyd S 1982 Least squares quantization in pcm *IEEE Trans. Inf. Theory* **28** 129–37
[68] Rousseeuw P J 1987 Silhouettes: a graphical aid to the interpretation and validation of cluster analysis *J. Comput. Appl. Math.* **20** 53–65
[69] QuTech 2020 Quantum inspire
[70] Sajid Anis M D *et al* 2021 IBM Qiskit: an open-source framework for quantum computing
[71] Khammassi N, Imran Ashraf J V S, Razvan Nane A M K, Adriaan Rol M, Lao L, Bertels K and Almudever C G 2021 Openql: a portable quantum programming framework for quantum accelerators *ACM J. Emerg. Technol. Comput. Syst.* **18** 1–24
[72] UCLA 2020 Queko benchmark (available at: https://github.com/UCLA-VAST/QUEKO-benchmark)
[73] Valada D 2020 Openql random circuits (available at: https://github.com/Astlaan/OpenQL/blob/metrics/tools/random_circuit_generator.py)
[74] Cross A W, Bishop L S, Sheldon S, Nation P D and Gambetta J M 2019 Validating quantum computers using randomized model circuits *Phys. Rev.* A **100** 032328
[75] Cuccaro S, Draper T, Kutin S and Moulton D 2004 A new quantum ripple-carry addition circuit (https://doi.org/10.48550/arXiv.quant-ph/0410184)
[76] Greenberger D M, Horne M A and Zeilinger A 2007 Going beyond bell's theorem *Bell's Theorem, Quantum Theory and Conceptions of the Universe* (*Fundamental Theories of Physics* 37) (Springer) 69–72
[77] Nielsen M A and Chuang I 2002 *Quantum computation and quantum information*
[78] IBM (available at: www.ibm.com/quantum)
[79] Rigetti (available at: https://medium.com/rigetti/)
[80] Goodrich T D, Horton E, and Sullivan B D 2018 Practical graph bipartization with applications in near-term quantum computing (arXiv:1805.01041)
[81] Kuhn H W 1955 The hungarian method for the assignment problem *Naval Res. Logist. Q.* **2** 83–97
[82] Punnen A P 2022 *The Quadratic Unconstrained Binary Optimization Problem* (Springer)
[83] Murali P, Matthias Linke N, Martonosi M, Javadi Abhari A, Hong Nguyen N and Huerta Alderete C 2019 Full-stack, real-system quantum computer studies: Architectural comparisons and design insights *2019 ACM/IEEE 46th Annual Int. Symp. on Computer Architecture (ISCA)* (IEEE) pp 527–40
[84] Nishio S, Pan Y, Satoh T, Amano H and Van Meter R 2020 Extracting success from ibm's 20-qubit machines using error-aware compilation *ACM J. Emerg. Technol. Comput. Syst.* **16** 1–25
[85] Wille R, Große D, Teuber L, Dueck G W and Drechsler R 2008 Revlib: an online resource for reversible functions and reversible circuits *38th Int. Symp. on Multiple Valued Logic (Ismvl 2008)* (IEEE) pp 220–5
[86] Pakin S and Steven P R 2019 Programminga d-wave annealing-based quantum computer: tools and techniques *Quantum Inf. Comput.* **19** 721–59
[87] Apak B, Bandic M, Sarkar A and Feld S 2024 Ketgpt–dataset augmentation of quantum circuits using transformers *Int. Conf. on Computational Science* (Springer) pp 235–51
[88] Bandic M 2024 *QuantumCircuitProfiling* (available at: https://github.com/QML-Group/QuantumCircuitProfiling)