Quantum Computer Benchmarking

Table of contents

1. Introduction

This article provides an overview of the state-of-the-art in the field of gate-based quantum computer benchmarking, including a description of some currently available benchmarking techniques and benchmarks[1].

Notes

1. This overview is neither exhaustive nor very detailed in its descriptions of the various techniques and benchmarks, given its extensive range and the fact that this field is rapidly evolving to keep pace with ongoing developments in quantum computing hardware and software.

2. It is assumed that the reader has reasonable knowledge of classical computing, some basic knowledge of Artificial Intelligence (AI) techniques, and also some basic knowledge of quantum computing hardware and software (as for example included in the articles 'Quantum Computing Explained', 'Quantum Annealing Explained' and 'Quantum Software Development Tools' published by the NOREA Taskforce Quantum Computing).

Quantum computer benchmarking is used to evaluate quantum computers for multiple purposes:

- comparing commercially available quantum computers;
- determining suitability for specific problem-solving quantum algorithms;
- comparing quantum computing with classical computing[2], e.g. for assessing potential quantum advantage and for cost-benefit analysis;
- estimating the rate of technological progress over time (e.g. to estimate short/medium term projections);
- etc.

Note

A distinction can made between a *benchmark*, i.e. a procedure used to measure (a/the) specific aspect(s) of a quantum computer (component) and a *metric*, i.e. a number for a particular characteristic of a quantum computer (component). Given the probabilistic nature of quantum computing, metrics measurements are

[1] One could argue that quantum computer benchmarks should also be applicable to quantum annealers. However, the range of problems that can be solved with quantum annealers and with gate-based quantum computers differs significantly and furthermore, the solutions (i.e. quantum algorithms) for solving similar problems are very different. Therefore, only the speed factor (aka time-to-solution) is deemed useful for the purpose of comparing the performance of quantum annealers and gate-based quantum computers.

[2] It should be noted that both are "moving targets"!

typically obtained using some sort of benchmarking because direct measurements of the characteristics are often not feasible. For this reason, the distinction between benchmark and metric is often blurred.

In this article, three different types of quantum computer benchmarking are distinguished:

1. *component-level benchmarking* aka *device benchmarking* or *subsystem benchmarking* (described in Chapter 2);

   Note
   Depending on the scope of testing, some of these benchmarks could also be used for system-level benchmarking.

2. *system-level benchmarking* aka *aggregated benchmarking* (described in Chapter 3);

3. *application-level benchmarking* (described in Chapter 4).

In the context of this article, the goal of quantum computer benchmarking is to somehow measure the (relative) performance of a given (set of) quantum computer(s). Quantum computer performance relates to *scale*, *speed* and *quality*.

For system-level and application-level benchmarking, the *scale factor* determines the maximum size of problems that can be encoded and solved. The number of qubits determines the amount of quantum information that can be encoded, which caps the size of solvable problems. Since fault-tolerant quantum computation requires very large numbers of (stable) qubits, scale is a key metric for quantum computers.

Notes

1. Scale should not be confused with scalability, which is an important characteristic of quantum computer technology (not of a quantum computer itself) but is impossible to measure as such (it can only be determined qualitatively based on the characteristics of this technology and its manufacturing processes). For most quantum computing platforms, increasing the number of qubits relies heavily on the available materials and fabrication technologies, but the real challenge is to develop technologies that make scaling (i.e. increasing the number of qubits) possible while maintaining quantum coherence.

2. Qubits can also be used as a resource to improve speed and quality. For example, extra qubits can be used in multiprogramming of QPUs to increase their quantum circuit processing capability and auxiliary qubits can be used to reduce the depth of quantum circuits and increase their fidelity.

The *speed factor* determines the number of operations that can be executed per unit of time.

The *quality factor* determines how faithfully these operations are executed.

Most quantum computer benchmarking techniques address the quality factor, only a few of them address the speed factor.

Some quantum computer benchmarks provide a single-number score while others provide multiple scores for a range of performance metrics. Single-number scores are easier to compare but have at least two important drawbacks: (1) a single number cannot capture every aspect of a quantum computer's performance, and (2) not everyone is interested in the same type of performance information.

An ideal quantum computer benchmark should have the following properties:

- broadly accepted by the quantum computing community[3];
- universal, i.e. defined at the logical instead of the physical computational level[4];
- platform-independent (aka system-agnostic)[5];
- applicable to both (Noisy Intermediate-Scale Quantum (NISQ) and Fault-Tolerant Quantum Computer (FTQC) systems[6];
- representative of the computational power needed to execute problem-solving quantum algorithms corresponding with real-world applications[7];
- simple enough to be understandable and useful for non-experts[8];
- efficient and scalable verification of the benchmark results[9];
- simple pass/fail determination[10];
- well-defined, i.e. specifying a clear set of rules for the benchmarking process[11].

---

[3] However, only a few quantum computer benchmarks currently enjoy wide acceptance.

[4] This property does of course not apply to physical metrics.

[5] For some quantum computer benchmarks, this property coincides with the previous one.

[6] However, several mainstream quantum computer benchmarks are limited to NISQ quantum computers.

[7] However, several mainstream quantum computer benchmarks are based on random quantum circuits while quantum circuits corresponding with real-world applications are typically highly structured.

[8] This property conflicts with the previous one hence a trade-off is needed. It should also be noted that this property does not apply to physical metrics that are intended for use by specialists.

[9] However, few quantum computer benchmarks are efficiently verifiable because classical verification becomes infeasible as the number of qubits grows. It turns out that quantum computational power is a double-edged sword: quantum computers with a large number of qubits offer the possibility of computational speedups but simultaneously pose real problems for testing and assessing their capability.

[10] However, for some mainstream quantum computer benchmarks fail/pass determination is rather complex and also very resource-hungry.

[11] However, for some mainstream quantum computer benchmarks the rules are not clearly defined, e.g. it is sometimes not explicitly specified whether the benchmark's quantum circuits are defined at the physical or logical level.

A quantum computer system typically consists of multiple subsystems: the Quantum Processor Unit (QPU) subsystem, the qubits control subsystem (based on classical electronics and/or photonics technology), the classical computer subsystem (for the orchestration of quantum tasks and optionally for the classical part of an hybrid problem solving algorithm), and the networking subsystem when Quantum Computing-as-a-Service (QCaaS) access is used). Depending on the particular benchmarking technique, the performance of one, some or all of these subsystems is addressed and included in the benchmark results.

2. Component-level benchmarking

Component-level benchmarking is concerned with metrics related to the performance of quantum computers at the physical level. These low-level metrics address the various characteristics of a quantum computer's qubits.

The qubit characteristics that relate to the performance of quantum computers typically include the following metrics:

- *number of qubits*;
- *qubit fidelity* (aka *qubit stability*) refers to the *qubit coherence time*[12] and the *qubit dephasing time*[13] (see Figure 2.1).

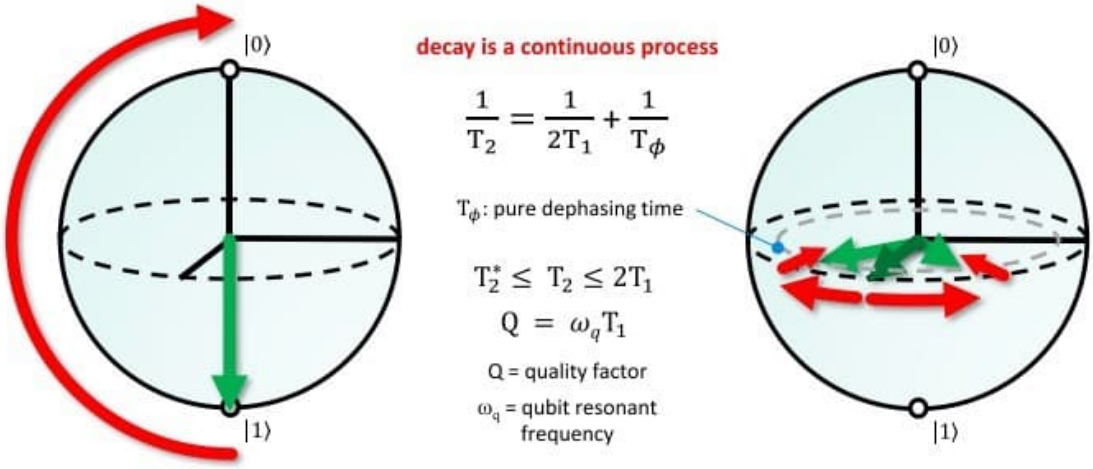

Figure 2.1: Bit-flip error T_1 and phase error T_2/T_2^*

The qubit coherence time T_1 refers to the spontaneous transition (aka relaxation) over time from the |1⟩ state to the |0⟩ state caused by various energy dissipation sources (aka "baths").

The qubit relaxation time is measured with a simple experiment using an X gate and measuring the result n times at different t times. T_1 corresponds to the time when the probability of obtaining a |1⟩ reaches 1/e (e = 2.718281828459045...); see Figure 2.2.

[12] Also known as longitudinal coherence time, longitudinal relaxation time, amplitude damping time or bit-flip error.

[13] Also known a phase coherence time, phase relaxation time, transverse relaxation time, phase damping time or phase-flip error.

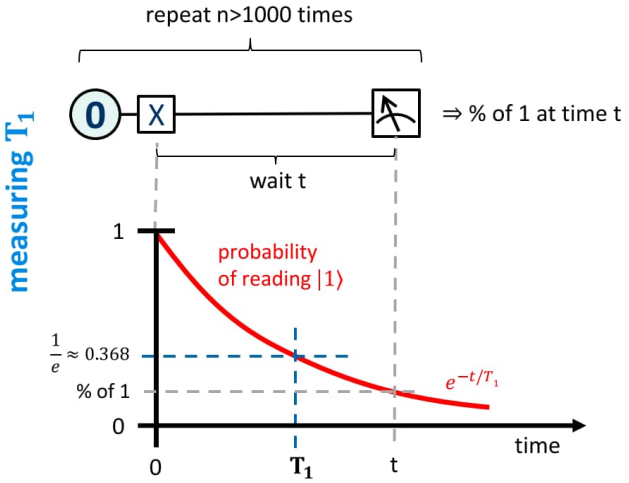

Figure 2.2: T_1 measurement (source: Olivier Ezratty 2023)

T_ϕ is the "pure" qubit dephasing time and refers to the loss of phase information over time in the complex probability amplitudes α and β of the qubit quantum state $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ (destroying the interference between them), when assuming an infinite T_1 value.

T_2 is the actual qubit dephasing time that refers to the loss of phase information under real circumstances. In practice, two variants of qubit dephasing time measurement are being used:

1. The average phase relaxation time T_2 corresponding with the probability that the qubit's state has relaxed to $|0\rangle$ is equal to 1/e for a qubit with Dynamical Decoupling (DD), which consists of applying echo sequences to the qubit to compensate for low frequency decoherence. T_2 (aka T_2^{echo} or T_2^{DD}) is obtained with a so called Hahn echo experiment using H and X gates.

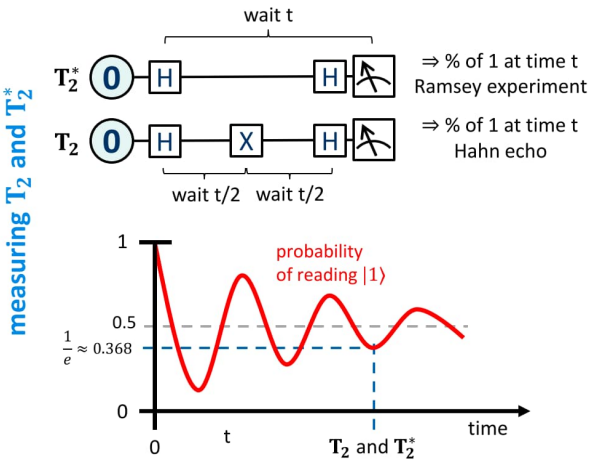

Figure 2.3: T_2 and T_2* measurement (source: Olivier Ezratty 2023)

2. The elapsed time T_2^* corresponding with the probability that the qubit's state has relaxed to $|0\rangle$ is equal to 1/e when the qubit is left to evolve freely. T_2^* is obtained with a so-called Ramsey experiment using H gates.

In general $T_2^* \leq T_2 \leq 2\ T_1$. T1 and T_2/ T_2^* can vary widely between qubit technologies.

The *qubit quality factor* Q is defined as: $Q = \omega_q T_1$

(ω_q is the qubit resonant frequency[14], i.e. the resonant frequency of the qubit drive tone that changes its quantum state from the ground state $|0\rangle$ to the first exited state $|1\rangle$)

The qubit quality factor Q consists of various components:

$1/Q = 1/Q_i + 1/Q_d + 1/Q_c + 1/Q_m$

- Q_i is the qubit internal quality factor (related to the qubit's internal loss mechanisms;
- Q_d is the driving ports (for qubit control signals) quality factor;
- Q_c is the qubit coupling ports (for qubit couplers) quality factor;
- Q_m is the qubit measurement ports (for qubit readout) quality factor.

- *quantum gate fidelity* refers to the error rate that is associated with single-qubit and two-qubit quantum gate operations; the techniques used for measuring intrinsic noise/error behaviour of one- and two-qubit quantum gates are commonly known as *Quantum Characterization, Verification, and Validation* (QCVV);
- *trace distance* refers to differences in phase which can be overlooked by the gate fidelity metric;
- *qubit readout fidelity* (aka *qubit measurement fidelity*) refers to the error rate associated with qubit readout operations;
- *quantum gate execution time* (aka *quantum gate speed*) refers to the time needed to perform quantum gate operations;
- *qubit readout execution time* (aka *qubit readout speed*) refers to the time needed to perform qubit readout operations;

[14] The qubit resonant frequency ω_q is defined as the energy difference between the $|1\rangle$ and $|0\rangle$ states divided by the reduced Planck constant $\hbar$ (h-bar), which is $h/2\pi$ (the value of h is $6.62607015 \times 10^{-34}$ Js).

- *qubit reset execution time* (aka *qubit reset speed*) refers to the time needed to set the qubit's quantum state to its ground state (|0⟩) or a chosen basis state;
- *qubit connectivity* refers to the way in which qubits can be linked together;
- *qubit entanglement scope* refers to entanglement not being limited to immediately neighbouring qubits.

Note
Individual metrics are sometimes combined to provide an aggregated metric. For example, qubit stability, quantum gate fidelity and qubit readout fidelity are often combined to denote *qubit fidelity*. Another example of a combined metric is *State Preparation And Measurement* (SPAM) fidelity.

There are many techniques for determining the physical performance characteristics of quantum computers. Quantum computing vendors use these techniques to calibrate[15] and test their systems and they often publish the performance metrics extracted from the results of these calibrations.

The performance metrics that providers report to users of their systems typically include one- and two-qubit gate fidelity, SPAM fidelity and T_1 and T_2(*) coherence times. For example, Figure 2.4 shows the contents of the text file that the Amazon Braket cloud service presents to users of the IonQ QPU. Other quantum computing vendors provide similar types of information about their QPUs.


```
{
  "braketSchemaHeader": {
    "name": "braket.device_schema.ionq.ionq_provider_properties",
    "version": "1"
  },
  "fidelity": {
    "1Q": {
      "mean": 0.99717
    },
    "2Q": {
      "mean": 0.9696
    },
    "spam": {
      "mean": 0.9961
    }
  },
  "timing": {
    "T1": 10000,
    "T2": 0.2,
    "1Q": 0.000011,
    "2Q": 0.00021,
    "readout": 0.000175,
    "reset": 0.000035
  }
}
```


[15] Calibration is a technique to reduce systematic errors in quantum circuits.

Figure 2.4: Example of published metrics (source: Amazon)

Component-level performance metrics may provide a lot of information about a quantum computer but they have at least two important limitations:

1. Such metrics are typically not sufficient to accurately predict a quantum computer's performance running a quantum algorithm. This is because these metrics cannot capture the full complexity of errors in quantum computer components (e.g. individual quantum gates and qubit readouts), and because not all sources of error (e.g. crosstalk) are even apparent when testing a single component.

2. Component-level metrics are challenging for non-specialists to interpret. It is very difficult for prospective quantum computer users to extrapolate from component-level metrics to predict how their quantum application will perform.

2.1. Randomized Benchmarking (RB)

Randomized Benchmarking (RB) is a technique used to determine fidelity metrics. RB is based on the principle that a subset of quantum circuit "subroutines" is chosen randomly from some large set. Long periods of these random quantum circuit "subroutines" are then chosen to amplify certain types of noise, thus allowing for the measurement of the average effect of such noise.

RB estimates the average error rates by implementing long sequences of uniformly randomly sampled quantum Clifford gate[16] operations.

RB scales efficiently, requiring only polynomial classical resources[17]. It is the industry-standard technique used by quantum hardware manufacturers such as Google Quantum AI and IBM Q.

2.2. Tomographic benchmarks[18]

Quantum State Tomography (QST) is the process by which a quantum state or (set of) quantum process(es) is reconstructed using repeated measurements on an ensemble of identical quantum states. The source of these states may be any device or system which prepares quantum states, either consistently into quantum pure states or otherwise into general mixed quantum states. To

[16] A Clifford gate is a quantum gate that can be decomposed into quantum gates of the Clifford group. The Clifford group includes the Pauli gates (X, Y and Z), the H (Hadamard) gate, the CNOT (Controlled NOT) gate, the SWAP gate and many more. Clifford gates are sometimes called "digital" gates while non-Clifford gates are called "analogue" gates.

[17] According to the Gottesman-Knill theorem quantum circuits using only quantum gates from the Clifford group can be emulated in polynomial time on classical computers.

[18] Tomography is the reconstruction of a comprehensive model (of something) from many partial cross-sections or slices, each of which provides only a limited view that may be useless by itself.

be able to uniquely identify the quantum state(s), the measurements must be tomographically complete, i.e. the measured operators must form an operator basis on the Hilbert space[19] of the system, providing all the information about the quantum state.

In Quantum Process Tomography (QPT) on the other hand, known quantum states are used to probe a quantum (set of) process(es) to find out how the process(es) can be described. QPT is a procedure to completely determine the evolution of a quantum system.

QST and QPT become systematically unreliable when the "reference frame" operations on which they rely (pre-calibrated measurements for QST, pre-calibrated state preparations and measurements for QPT) are either unknown or misidentified. This limits their practical utility and makes them unsuitable for rigorous characterisation.

Gate Set Tomography (GST), aka long-sequence Gate Set Tomography, is a technique for detailed predictive characterisation of quantum gates. GST differs from QST and QPT in two fundamental ways. Firstly, it is almost entirely calibration-free ("self-calibrating"): when GST reconstructs a model of a quantum system, it does not depend on a prior description of the measurements used (as does GST) or the quantum states that can be prepared (as does GPT). Secondly, GST does not reconstruct or estimate a single logic operation (e.g. state preparation or logic gate), but an entire set of logic operations – a gate set including one or more state preparations, measurements and logic gates.

Tomographic benchmarking does not scale well and its use is therefore limited to small components or subsystems.

2.3. Compressed/adaptive learning-based benchmarks

Rather than performing an exhaustive deterministic set of tests as in tomographic benchmarking, one can choose a random subset of tests (compressive tomography benchmarking) or an adaptive subset of tests (learning-based benchmarking).

The efficiency of these methods is much better than the tomographic benchmarking methods and can therefore be used for testing larger components or subsystems, but the quality and completeness of the information obtained strongly depends on the quality of the model assumptions.

2.4. Quantum fidelity estimation/witnessing

[19] Hilbert spaces (named after the German mathematician David Hilbert) allow the methods of linear algebra and calculus to be generalised from finite-dimensional Euclidean vector spaces to spaces that may be infinite-dimensional. Formally, an Hilbert space is a vector space equipped with an inner product that induces a distance function for which the space is a complete metric space. A qubit state is a vector in a 2-dimensional Hilbert space.

Fidelity estimation measures how close a quantum state or process is to the ideal state or process, rather than performing a full tomographic characterisation.

For certain special types of desired quantum states (certain highly entangled systems in particular), a much more efficient fidelity witness (entanglement witness[20] in particular) can be used to estimate if the quantum system is close to being in such a state.

Fidelity estimation is more efficient than tomography but typically still scales exponentially.

2.5. Entropic sampling benchmarks

Entropic sampling benchmarking is a type of fidelity estimation which measures the entropy in the output of a series of randomly applied quantum circuits.

Cross-Entropy Benchmarking (XEB) uses the properties of random quantum circuits to determine the fidelity of a wide variety of quantum circuits. When applied to deep two-qubit quantum circuits, it can be used to accurately characterise a two-qubit interaction potentially leading to better calibration. When applied to quantum circuits with many qubits, XEB can potentially characterise the performance of a large device.

Entropic sampling benchmarking is easy to test but exponentially hard to verify[21].

[20] An entanglement witness is an Hermitian operator which helps to decide whether a quantum state is entangled or not. The basic idea is that the expectation value of the witness will be different for separable and entangled quantum states (an Hermitian operator has only real eigenvalues).

[21] This technique was used by Google in 2019 to claim quantum supremacy, even though the measured circuits responsible for these claims showed very low fidelity.

3. System-level benchmarking

Most system-level benchmarks are based on determining the performance metrics of a quantum computer by measuring its behaviour while executing a set of carefully selected quantum circuits, which is believed to be representative for the load that quantum computers will be subjected to in practice.

Many problem-solving quantum algorithms for NISQ quantum computers, for example Variational Quantum Algorithms (VQAs), translate into quantum circuits whose outlines are known but whose parameters are runtime-dependent. Therefore, the compilation of quantum circuits that arise from them is a key part of executing quantum programs and should be taken into account by the benchmarking technique and when comparing benchmark outcomes. For example, quantum circuits can be compiled and optimised to a great degree offline (Figure 3.1 shows an example of compilation phases and how they interact with the runtime system).

Furthermore, variational quantum algorithms require repeated interactions between quantum and classical subsystems and therefore, their speed is considered critical and should also be taken into account by the benchmarking technique.

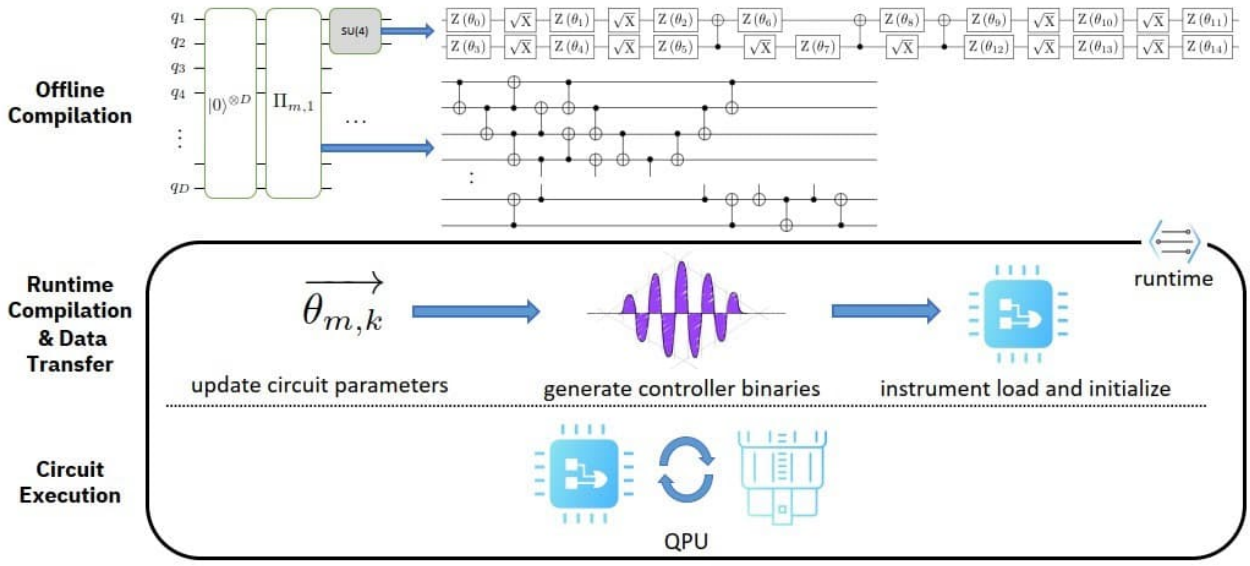

Figure 3.1: Runtime architecture and phases of compilation (source: IBM Q)

The quantum circuit *width* is the number of qubits; some relevant quantum circuits used in real-world applications are either *narrow* or *wide*. The quantum circuit *depth* is the number of quantum gates; some relevant quantum circuits used in real-world applications are either *shallow* or *deep*[22].

[22] This is a bit confusing because in most graphical representations of quantum circuits, the qubits are shown from top to bottom and the quantum gates are shown from left to right.

Furthermore, some relevant quantum circuits used in real-world applications are *wide shallow circuits* or *narrow deep circuits*. Many benchmarks do not include these "special" kinds of quantum circuits and their results are therefore not valid for the corresponding problem-solving quantum algorithms. For example, the Quantum Volume (QV) benchmark (see § 3.1) only comprises *square quantum circuits* (where the width is equal to the depth).

The following quantum circuit types are distinguished for the sake of benchmarking:

- *random circuits*: these are randomly selected small quantum circuits that build up larger quantum circuits[23];
- *periodic circuits* (aka *cyclic circuits* or *repetitive circuits*): these quantum circuits are structured periodic rather than random;
- *structured circuits* (aka *application circuits*): not clearly defined; this applies to a whole range of quantum circuits used in real-world applications (i.e. specific problem-solving quantum algorithms).

The quantum circuits specified in quantum computer benchmarks can be specified at various levels of abstraction:

- *pseudocode circuits*

  Quantum circuits can be specified at a very high level in terms of a set of layers that are actually quantum circuit subroutines. Example subroutines include Clifford gates, Controlled-U gates, n-qubit Toffoli gates, Grover iterations, QFTs, and adders. This kind of representation is roughly analogous to classical pseudocode: it is relatively human-readable, and generally requires each layer to be extensively and creatively compiled to run on a quantum computer.

- *canonical-gate circuits*

  Quantum circuits can be specified in terms of layers formed by parallel combinations of "canonical" one- and two-qubit gates. A typical example are arbitrary SU(2)[24] rotations on

[23] The Quantum Performance Laboratory of Sandia National Laboratories (SNL) has determined that some quantum computers show massive variance in the performance of random circuits of the same depth and width (caused by the occurrence of structured errors that are catastrophically amplified by quantum circuits that have the "correct" structure), while other quantum computers don't. SNL's finding has a significant impact on the accuracy of the results of quantum computer benchmarks that are based on the use of random circuits.

[24] The Special Unitary group SU (2^n) is the space of unitary transformations applicable on n qubits. It covers all the unitary transformations that can be performed on n qubits (quantum gates are unitary, because they are implemented via the action of a Hamiltonian for a specific time, which gives a unitary time evolution according to the Schrödinger equation). Hence SU(2) denotes the unitary transformations applicable to a single qubit, SU(4) denotes the unitary transformations applicable to 2 qubits, and so on.

each qubit and CNOTs between each pair of qubits. Most quantum computers do not implement all these gates natively. Canonical-gate circuits therefore require compilation to run on a quantum computer.

Each canonical gate can typically be compiled individually via a standardised technique or a lookup table. Compiling a single canonical gate may produce a fairly complicated quantum subcircuit (e.g. CNOT gates between widely separated qubits in a local architecture or arbitrary rotations in an architecture with only discrete gates), but this complexity is usually controllable and predictable.

- *native-gate circuits*

  Quantum circuits can be specified using layers formed by parallel combination of actual native gates available on a specific quantum computer, e.g. specific one-qubit gates and entangling gates between physically adjacent qubits.

  Compilation at this level is generally unnecessary, though there may still be some freedom to schedule those gates in time by inserting idle operations as desired and by "sliding" gates forward and backward in time without exchanging the order of any non-idle gates.

3.1. Volumetric Benchmarks (VBs)

The Quantum Volume (QV) benchmark (introduced by IBM Q) aims to indicate the maximum size of square quantum circuits that can be faithfully executed on a particular quantum computer.

QV works as follows. A QV layer is defined as one layer of permutation among qubits and one layer of pairwise random SU(4) 2-qubit unitary gates (Figure 3.2). The QV is defined by the width or number of QV layers of the largest random square circuit (with width equal to the number of layers) that a quantum processor can successfully run.

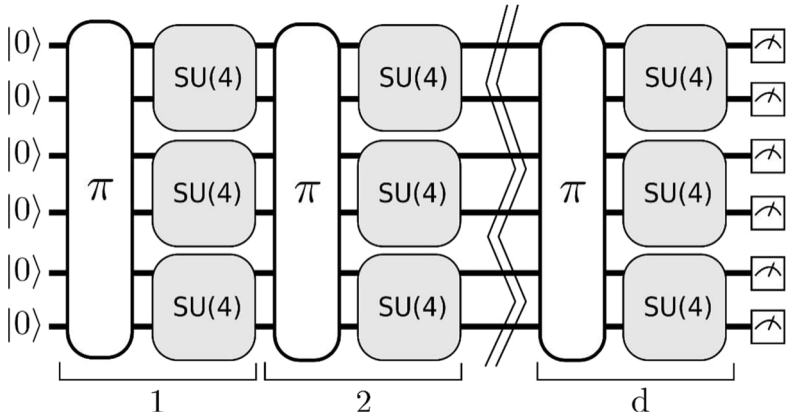

Figure 3.2: Quantum Volume (QV) determination (source: IBM Q)

Note
When a QV circuit is compiled to the native gate set of a particular QPU, the circuit depth of the compiled circuit will typically be much larger than the number of QV layers as the abstract permutations and SU(4) unitaries may be each decomposed into multiple native gates.

QV measurement executes a series of square circuits multiple times and then compares the measurement results from the "heavy output" states (i.e. the states with probabilities higher than the median probabilities of all output states) with the ideal results obtained by emulation. The largest n-qubit square circuit that can run successfully to produce more than 2/3 of heavy outputs determines the QV, which is given by 2^n.

QV is sensitive to coherence, gate fidelity and measurement fidelity and is also influenced by qubit connectivity and by quantum compilers which can make quantum circuits more efficient to minimise the effect of decoherence, i.e. Quantum Error Mitigation (QEM)[25]. QV is a holistic metric because it cannot be improved by just improving one aspect of the system, but rather requires all parts of the system to be improved in a synergistic manner.

QV has been adopted by research and industry and is routinely reported for quantum computers manufactured by IBM, IonQ and Quantinuum. QV is capable of providing a reasonable basis for the comparison of gate-based quantum computers but there are nevertheless some issues with this benchmarking technique, including:

- The QV benchmark does not specify how Quantum Error Correction (QEC) should be handled, i.e. whether a QV test circuit is defined at the physical or logical level. Therefore, QV is only applicable to NISQ quantum computers as they do not implement QEC.

- The quantum circuits for problem-solving quantum algorithms corresponding with real-world applications are not random but rather highly and nontrivially structured. On NISQ quantum computers, structured errors affect structured circuits very differently than they affect random circuits. It is known that performance is typically much worse for structured circuits than for random circuits. Consequently, there is no reason to assume that random circuits will be a reliable proxy for the vast majority of quantum algorithms running on NISQ quantum computers (random circuits only contain structure by chance).

- The quantum circuits for many problem-solving quantum algorithms corresponding with real-world applications are not square. For example, many relevant quantum algorithms translate to quantum circuits with significantly greater depth than width.

- Not all qubits and qubit connections in NISQ quantum computers are of the same quality and therefore, even if a QV test is successfully passed, such success may rely on selecting a good qubit subset, which is not trivial to identify for quantum circuit developers. This is one of the

[25] Quantum Error Mitigation (QEM) relates to NISQ quantum computers and should not be confused with Quantum Error Correction (QEC) which only applies to FTQC quantum computers.

reasons why the result of QV benchmarks performed by customers and independent parties lag the QV benchmark results reported by quantum computer vendors.

- The preparation of QV circuits is very time intensive and not well standardised for different types of quantum computing systems.
- Vendors typically make extensive use of the quantum compiler's optimisation capabilities when performing QV benchmarks in order to obtain the best possible results. However, most quantum circuit developers will use the standard compiler methods of the vendor's Quantum Software Development Kit (QSDK). This is another reason why it will be difficult for them to achieve the performance corresponding with QV benchmark results reported by vendors.
- A quantum computer's noise profile is not constant over time and therefore, obtained QV results may no longer be accurate.
- The probability outputs needed for the "heavy output" states (to determine pass/failure of individual QV quantum circuit tests) must be precomputed, which is computational hard classicly.
- Relatively small improvements can lead to drastically larger QV numbers due to the exponentiation in the QV definition, thereby overemphasising differences in capability.

Generalised volumetric benchmarks have been proposed that are a generalisation of the standard QV benchmark to a more general framework, in terms of both width and depth of the quantum circuits that a quantum computer can faithfully execute. At the same time, these proposals also address some of the shortcomings of the standard QV benchmark that are listed above.

Note
The QED-C benchmark (see § 4.1) defines volumetric benchmarking as follows: "Volumetric benchmarking is a framework for constructing and visualising the results of large, flexible families of benchmarks". The QED-C benchmark is however an application-level benchmark based on a suite of quantum algorithms selected from different application domains. It should therefore not be confused with the system-level volumetric benchmarks described here.

A major problem is that generalised volumetric benchmarks entail much greater complexity as the quantum computer performance is no longer represented by a single quantitative value but rather a set of values for families of successfully tested quantum circuit shapes. To address this issue, MITRE made a proposal for a volumetric benchmark with a restricted subset of 5 quantum circuit shapes, named Quantum Volumetric (QV) classes (Table 3.1).

The counts in this table are based on a survey of NISQ and FTQC quantum algorithms with known resource estimates, which makes it possible to approximate the scaling of quantum circuit depth in relation to the quantum circuit width (n qubits). MITRE identified 58 of such quantum algorithms from the following application areas: Quantum Machine Learning (QML), many-body physics and chemistry, numerical solvers, optimisation, quantum data hiding and other.

Class	Circuit Depth	Counts
QV-1	$\mathcal{O}(n)$	16 (28%)
	$\mathcal{O}(\log(n))$	8 (14%)
	$\mathcal{O}(\log^x(n))$	6 (10%)
	$\mathcal{O}(1)$	2 (3%)
	$\mathcal{O}(\sqrt{n}\,\mathrm{polylog}(n))$	1 (2%)
QV-2	$\mathcal{O}(n^2)$	10 (17%)
	$\mathcal{O}(n\,\mathrm{polylog}(n))$	2 (3%)
QV-3	$\mathcal{O}(n^3)$	8 (14%)
	$\mathcal{O}(n^2\,\mathrm{polylog}(n))$	1 (2%)
QV-4	$\mathcal{O}(n^3\log(n))$	2 (3%)
QV-5	$\mathcal{O}(n^5)$	2 (3%)

Table 3.1: QV classes - quantum circuit depths and quantum algorithm counts (source: MITRE)

3.2. Mirroring benchmarks

In mirroring benchmarks the quantum circuits are concatenated with their mirror copy. Optionally, quantum gate layers can be inserted at the start, at the end and between the two halves of the mirrored quantum circuit to adjust its sensitivity to certain types of errors.

The advantages of mirroring benchmarking are its efficient classical compilation, simple to quantify outcome, use of quantum gates that respect a QPU's qubit connectivity and applicability to quantum circuits of different structure (random circuits, periodic circuits and structured circuits) and shape (with respect to depth and width). See Figure 3.3 for an example of a periodic mirror quantum circuit.

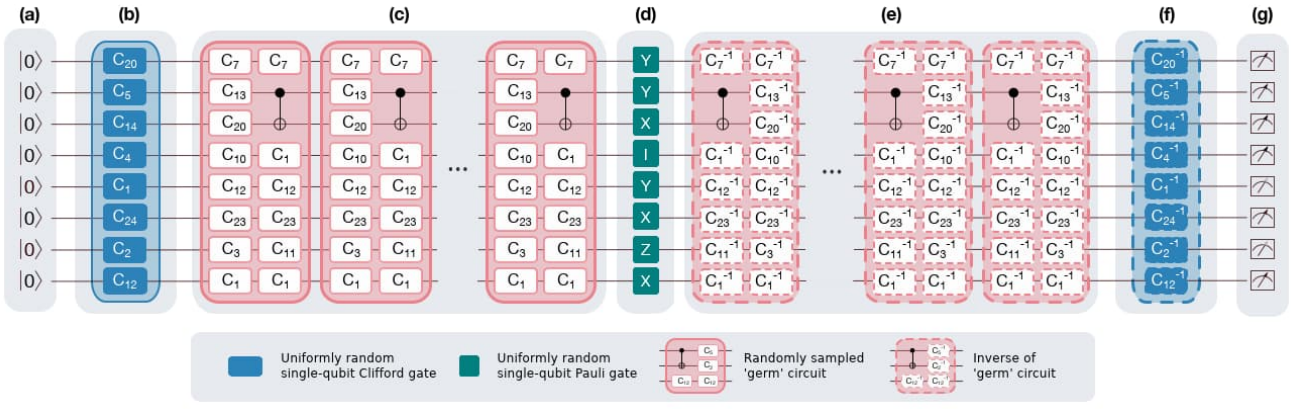

Figure 3.3: Example periodic mirror quantum circuit (source: Sandia National Laboratories)

In Mirror QV, half of the standard Quantum Volume (QV) circuit is replaced with the inverse of the other half. Instead of using the "heavy output" criterion for determining a successful execution of the quantum circuit, mirror QV calculates the success probability of the output bitstring being the same as the ideal bitstring.

Mirror QV is significantly faster than standard QV because there is no need to classically emulate the quantum circuits, but the potential downside is that due to its symmetric nature and simple output, mirror QV may be insensitive to errors that do not impact the all-zero output state and systematic errors that would cancel out between the first and inverse halves of the quantum circuit (which are all errors that standard QV might be able to catch).

In November 2023, IBM introduced a new benchmark named Layer Fidelity (LF) that "provides a more granular understanding of quantum systems while accurately capturing the system's ability to run the kinds of circuits that users are running today". The LF benchmark encapsulates the entire quantum device's ability to run quantum circuits, while also revealing information about individual qubits, quantum gates and crosstalk.

LF expands on Randomized Benchmarking (RB, see § 2.1). With RB, a set of randomised Clifford quantum gates (X, Y, Z, H, SX, CNOT, etc.) is added to the quantum circuit, then operations are performed that represent the inverse of the sequence of operations that precede it. If any of the qubits do not return to their original state by the inverse operations upon measurement, an error must have occurred. A number is extracted by running this test multiple times while adding more and more random quantum gates, plotting on a graph how the errors increase with more quantum gates, fitting an exponential decay to the plot, and using that line to calculate a number between 0 and 1, which is called the fidelity.

In order to extract the LF, the benchmarking test is started with a connected set of qubits, like a chain of qubits where each one is entangled to their neighbour. Then this connected set of qubits is split up into multiple layers so that each qubit only has at most one two-qubit quantum gate acting on it: if a quantum gate is needed to entangle qubit one and qubit two, and another quantum gate is needed to entangle qubit two and qubit three, then these are split out into two "disjoint layers". RB benchmarking tests are then performed on each of these disjoint layers to calculate the fidelity of each one of them. Finally, the fidelity of all layers is multiplied together into a final number, the LF.

IBM's LF benchmark qualifies the whole quantum device, while also providing quantum gate-level information, such as the average error for each quantum gate in the layered circuits, the Error-Per-Layered Gate (EPLG) is calculated as

$$EPLG = 1 - LF^{1/n_{2Q}}$$

where n_{2Q} is the number of two-qubit gates. For a linear chain of N entangled qubits, n_{2Q} is typically equal to N-1 hence

$$EPLG = 1 - LF^{1/N-1}.$$

3.3. Circuit Layer Operations Per Second (CLOPS)

The Circuit Layer Operations Per Second (CLOPS) benchmark (introduced by IBM Q) is used to determine how many Quantum Volume (QV) quantum circuits a QPU can execute per unit of time.

In order to faithfully model real-world use, it is essential to capture interaction time with the run-time environment that invokes the quantum circuits. This avoids the pitfall seen in some synthetic benchmarks that characterise classical systems by their instruction clock rate without considering the effects of data transfers between CPU, cache and main memory. It is not possible to persist quantum data across multiple invocations, so data transfer plays an even more prominent role for quantum computers.

To capture the interaction with the runtime environment, multiple executions of parameterised quantum circuits are measured, where the choice of parameters is deferred until run time. This mimics the scenario found in variational quantum algorithms such as the Variational Quantum Eigensolver (VQE), where the ability of a quantum system to efficiently handle parameterised circuits is key to the performance of the quantum algorithm.

Measuring execution speed of typical applications requires either a corpus of representative quantum circuits or a choice of a quantum circuit family that somehow captures an “average” practically useful quantum circuit. While the difficulty of the latter is recognised, QV quantum circuits are at least representative of random quantum circuits while simultaneously allowing for a rigorous notion of quality. This ensures that the benchmarked quantum circuits operate in a regime where the QPU is producing meaningful results.

CLOPS is formally defined as the number QV layers executed per second using a set of parameterised QV quantum circuits, where each of the QV circuits has $D = log_2 QV$ layers.

The CLOPS benchmark (Figure 3.4) consists of 100 parameterised template circuits of the same type as the model circuits used when measuring the QV of the system (see § 3.1), except that the CLOPS SU(4) random unitaries are parameterised.

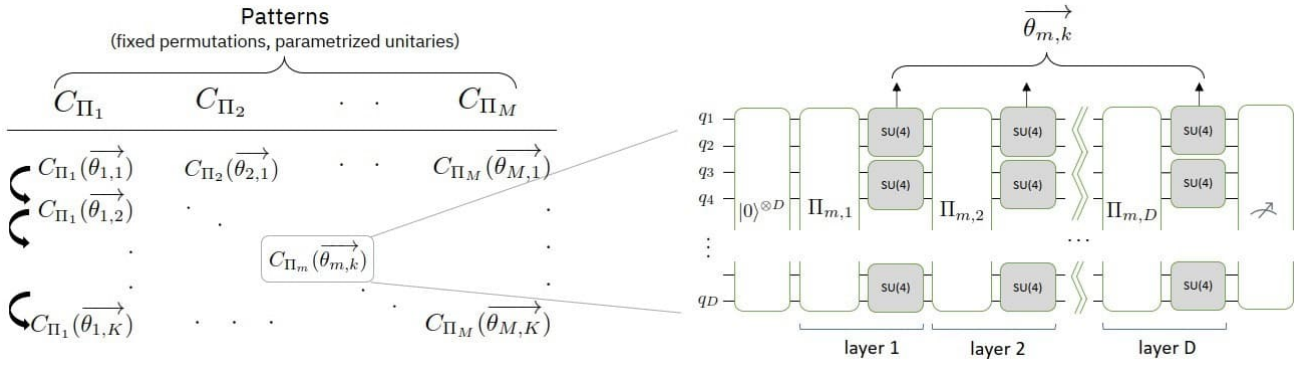

Figure 3.4: Matrix of quantum circuits used for CLOPS benchmark (source IBM Q)

Each quantum circuit template is executed 10 times with 10 choices of random parameters. These parameters are applied to the quantum circuit template to generate the final quantum circuit that is then run on the system without any further parameter updates.

Each of these quantum circuits is executed with 100 shots, which is an attempt to balance the benchmark between just measuring setup time for execution against the number of shots typically required to estimate an observable with reasonable variance from the output.

CLOPS is calculated as the total number of QV layers executed:

$M \times K \times S \times D$

where

M is the number of quantum circuit templates (100);

K is the number of parameter updates (10);

S is the number of shots (100);

D is the number of QV layers ($log_2 QV$).

This value is then divided by the total execution time to calculate the CLOPS value.

The CLOPS benchmark is designed to allow the system to leverage all of the quantum resources and optimisation features on a quantum computing system to run a collection of quantum circuits as fast as possible, as well as stress all parts of the execution pipeline. This includes data transfer of quantum circuits and execution results, runtime compilation (lowering basis-quantum gate level quantum circuits to qubit control subsystem instructions), latencies in loading qubit control electronics, initialisation of qubit control subsystem, quantum gate times, qubit measurement (aka qubit readout) times, qubit reset times, delays between quantum circuits, processing of results and parameter updates. Including all of these parameters in the CLOPS benchmark ensures that all aspects of the quantum computing system are included to allow for a meaningful comparison between systems.

In November 2023, IBM updated CLOPS (which was renamed $CLOPS_v$) to $CLOPS_h$ to better reflect how quantum computer hardware executes quantum circuits. $CLOPS_h$ recognises that the QV layer concept, on which $CLOPS_v$ is based, is a bit idealised: after compilation, quantum gate operations such as single-qubit rotations and random two-qubit gates typically require more physical quantum gates to run on a quantum processor. In particular, qubit connectivity implies that what is considered a “layer” in QV may in fact require implementing multiple layers on the quantum computer.

$CLOPS_h$ therefore defines a "layer" differently (Figure 3.5). In $CLOPS_h$, rather than representing a set of two-qubit gates acting across all random pairs of qubits at once, a $CLOPS_h$ layer only includes the two-qubit gates that can be run in parallel on the quantum processor; if a QV layer has two quantum gates that cannot be run in parallel, it is split into two or more layers.

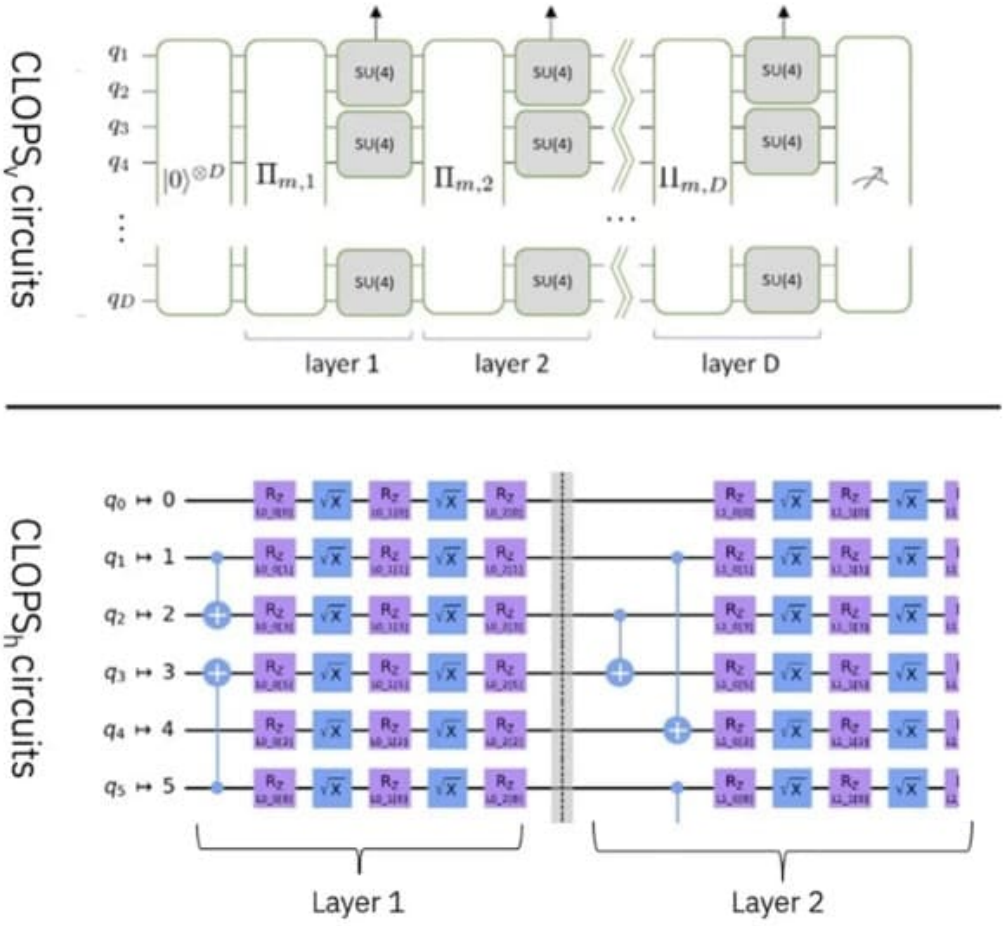

Figure 3.5: Layer splitting in $CLOPS_h$ benchmark (source IBM Q)

3.4. Generative modelling benchmarks

The qBAS score is a metric designed for benchmarking hybrid quantum-classical systems. It was developed to enable comparison of the performance of shallow circuits for different quantum computers.

The qBAS score is based on the generative modelling performance on a canonical synthetic data set which is easy to generate, validate and visualise for sizes up to hundreds of qubits. Yet, implementing a shallow circuit that can uniformly sample such data is difficult and some candidate solutions require large amounts of entanglement. Hence, miscalibration or environmental noise will affect its performance.

Data-Driven Quantum Circuit Learning (DDQCL, see Figure 3.6) is used to learn a quantum circuit that encodes all the BAS patterns[26] in the wave function of the quantum state of an ensemble of n qubits. The 2^n amplitudes of this wave function are used to capture the correlations observed in the data.

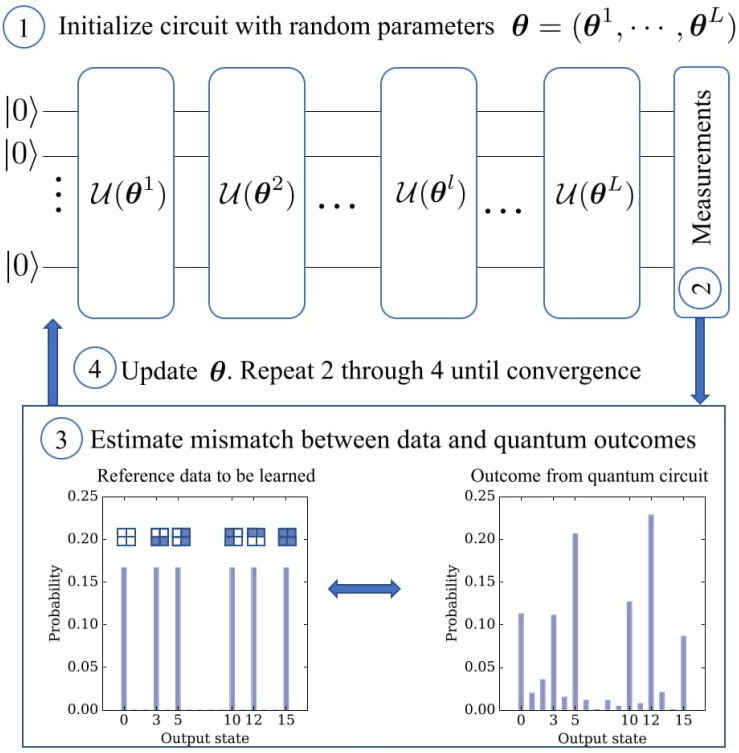

Figure 3.6: General framework for DDQCL (source: npj Quantum Information)

Training of the quantum circuit is achieved by successive updates of the parameters θ, corresponding to specifications of single qubit operations and entangling quantum gates. At each iteration, measurements from the quantum circuit are collected and contrasted with the data through evaluation of a cost function which tracks the learning progress.

This allows determination of the qBAS score, which is a figure of merit to assess the performance of shallow quantum circuits. In a single number, it captures the model capacity of the quantum circuit layout and intrinsic quantum computer hardware strengths and limitations in solving a complex sampling task that requires a fair amount of entanglement. It takes into account the model quantum circuit depth, quantum gate fidelities and architectural design aspects (such as the quantum computer's qubit-to-qubit connectivity and native set of single- and two-qubit gates). It also takes classical resources such as the choice of cost function, optimiser and

[26] BAS (Bars and Stripes) is a synthetic data set of images that is widely used to study generative models for unsupervised machine learning.

hyperparameters into account. Therefore, it can be used to benchmark the performance of the whole hybrid quantum-classical system.

3.5. Other system-level benchmarks

Various other system-level benchmarks have been developed, including:

- Cycle benchmarking which assesses the low-level quality of qubit entanglement (developed by an international team from Canada, Denmark and Austria).
- DARPA launched an RFP for the creation of application-specific hardware-agnostic benchmarks for quantum computing. The project was awarded to Raytheon BBN and USC. Their benchmark targets all sorts of quantum technologies, both computing and sensing, and is named Size, Weight, Application and Power (SWAP). DARPA also awarded a contract to Rigetti, the University of Technology Sydney, Aalto University and USC to create benchmarks for large-scale quantum computers.
- A similar approach was proposed by researchers from Brookhaven and the DoE Pacific Northwest National Laboratory (PNNL).
- Another DoE lab, Sandia National Laboratories (SNL), proposed a variation of Randomized Benchmarking (RB) that works in the quantum advantage regime.
- Still another DoE lab, Oak Ridge National Laboratory (ORNL), together with several US universities proposed a Volumetric Benchmark (VB) qualifying the quality of qubit entanglement.
- A proposal to measure the performance of FTQC quantum computers in an hardware-agnostic way with six structured circuits tests (Bell test, Schrödinger's microscope, Mandelbrot, line drawing, matrix inversion and platonic fractals[27]), was made by a team from QuSoft (the Netherlands), the University of Cambridge (UK) and Caltech (USA).
- A team from Berkeley, HRL Labs and the University of Chicago (all USA) devised a randomised benchmark measuring noise in non-Clifford quantum gates.
- Alibaba USA developed the Universal Randomized Benchmarking (URB) benchmark. It scales better than Google Quantum AI's cross-entropy benchmark and it also supports a universal quantum gate set.

[27] Platonic solids, named after the Greek philosopher Plato, are symmetrical geometric structures bounded by regular polygons all of the same size and shape. Moreover, all edges of each polygon are the same length and all angles are equal and he same number of faces meets at every vertex (corner or point). Each Platonic solid has its own fractal structure, the same repetitive patterns that fit within each other.

- PyQBench is a NISQ benchmarking tool proposed by the Institute of Theoretical and Applied Informatics from the Polish Academy of Sciences.
- V-score is a benchmark for many-body problem simulation which was proposed by a broad international research team.

4. Application-level benchmarking

Due to the complexity of errors in quantum hardware, neither a quantum computer's Quantum Volume (QV) nor any other single metric is likely to accurately predict its performance for all problem-solving quantum algorithms corresponding with real-world applications. There is thus a need for application-centric metrics and benchmarks that test the performance of quantum computers on practically relevant tasks. Application-level benchmark suites fulfil this need.

To be useful, application-level benchmark suites should comply with the following guiding principles:

- scalable

  The applications included in a benchmark suite should be scalable from just a few qubits to hundreds, thousands and beyond. It is also important that the performance metrics scale efficiently. Resources needed for classical emulation of quantum circuits scale exponentially with the number of qubits, so simply emulating the benchmarks and comparing with the experimental results is not a scalable solution. Therefore, a scalable suite must be composed of applications whose size is parameterisable and the performance of which is efficiently verifiable.

- meaningful and diverse

  Benchmark applications should reflect different use cases that are relevant in real-world environments. Quantum algorithms pulled from these different use cases present wildly varying structures and require vastly different amounts of resources from the quantum computer. A benchmark suite should provide good coverage over these potential use cases to better understand system performance under a variety of circumstances.

- full system evaluation

  The overall performance of a quantum computer relies on the proper functioning and interplay between the hardware and software stacks. For the NISQ era, the role played by the compiler (i.e. effectively cancelling gates, mapping between standard and native quantum gates, etc.) can make or break the execution of a quantum program. In addition, many of the unique properties offered by different quantum computer implementations (e.g. native multi-qubit or parametersable gates) are exploited at the compiler level when the quantum program is transpiled to a hardware supported (aka native) quantum gate set.

  Mandating a single compilation tool flow is ineffective because certain capabilities available only to a particular quantum hardware platform may be overlooked. A benchmark suite should

therefore specify benchmarks at a shared level of abstraction, such as for example defined by OpenQASM[28], and allow the compiler to play a role in overall system performance.

- adaptable

  Quantum computing as a whole, encompassing both hardware and software, is undergoing a period of rapid advancement. This poses a challenge for benchmarking since benchmark suites must keep pace with the development of quantum algorithms, compilation optimisations and quantum hardware advances. The applications included in the benchmark suite should reflect this by adapting to the current state-of-the-art.

Quantum computer benchmark suites that comply with these principles enable quantum computer manufacturers to quantify their progress in commercially relevant ways and make it possible for customers to more accurately predict how a particular quantum computer will perform on their applications.

4.1. QED-C benchmark

The Quantum Economic Development Consortium (QED-C) developed a suite of quantum computer benchmarks designed to measure the effectiveness of quantum computers at executing quantum applications. The QED-C benchmark suite is available as an open-source repository with extensive documentation.

The QED-C benchmarks probe a quantum computer's performance on various algorithms and small applications as the problem size is varied, by mapping out the fidelity of the results as a function of quantum circuit width and depth. Each benchmark in the suite is derived from an algorithm or application and specifies a scalable family of quantum circuits. In addition to square circuits like those tested in the Quantum Volume (QV) benchmark (see § 3.1), QED-C benchmarking can be used to estimate and report the performance of other quantum circuit shapes (including wide shallow quantum circuits and narrow deep quantum circuits).

In addition to estimating the fidelity of results generated by quantum circuit execution, the QED-C benchmark suite is designed to include certain aspects of the execution pipeline in order to provide customers with a practical measure of both the quality of and the time-to-solution.

Each individual benchmark in the QED-C suite has three important adjustable parameters:

1. the values of *n* for which performance is tested (corresponding to the input problem size, which often also corresponds to the number of qubits in the quantum circuits);

[28] OpenQASM is a low-level Intermediate Representation (IR) of quantum instructions. It stands at the conjunction between hardware platform-agnostic quantum software and platform-specific quantum hardware.

2. the number of quantum circuits that are selected from the benchmarking circuit set;

3. the number of times each quantum circuit is run.

Users can choose the values of *n* to test, but if two or more quantum computers are to be tested and the results compared, either the same values of *n* should be used for all of them, or a principled methodology should be chosen for selecting the values of *n* to use (and this methodology should be stated). For example, one reasonable strategy would be to increase *n* until either the result fidelity drops below a threshold value or until the maximum number of qubits available is reached.

QED-C determines the success of a quantum application by comparing the normalised fidelity of the output distribution P_{output} of a quantum computer with the output distribution P_{ideal} of an ideal quantum computer, as defined by:

$$F(P_{\text{ideal}}, P_{\text{output}}) = \frac{F_s(P_{\text{ideal}}, P_{\text{output}}) - F_s(P_{\text{ideal}}, P_{\text{uni}})}{1 - F_s(P_{\text{ideal}}, P_{\text{uni}})}$$

where P_{uni} is the uniform distribution and

$$F_s(P_{\text{ideal}}, P_{\text{output}}) = \left(\sum_x \sqrt{P_{\text{output}}(x) P_{\text{ideal}}(x)} \right)^2$$

where *x* is the bitstring that encodes the state.

An application is then considered successful if the value of *F* exceeds 0.5.

In addition to this primary metric, the benchmark also provides insight into runtime and the ratio between the programmed and transpiled circuit depth. However, QED-C makes the remark that runtime metrics are currently rudimentary, as quantum computing providers can have different definitions of quantum execution time.

QED-C benchmarking permits broad implementation freedom. One particularly important degree of freedom is that the benchmarked quantum circuits can be compiled into other quantum circuits that are logically equivalent (i.e. implement the same unitary evolution). This is necessary if a benchmark's results are to reflect the performance one can expect to achieve using a quantum computing provider's end-to-end stack. However, for benchmarks whose quantum circuits can be efficiently classically emulated, arbitrary compilations cannot be permitted because all processing could be offloaded to a classical co-processor. The current QED-C benchmark does not address this challenge because solving it without forbidding all compilation is rather difficult.

The QED-C benchmark suite in the current QED-C repository comprises the following set of quantum algorithms:

- shallow oracle-based algorithms: Deutsch Josza, Bernstein-Vazirani and Hidden Shift Problem;
- Quantum Fourier Transform (QFT);
- Grover's search algorithm;
- Quantum Phase Estimation (QPE) and Quantum Amplitude Estimation (QAE);
- Hamiltonian simulation;
- Monte Carlo sampling;
- Variational Quantum Eigensolver (VQE);
- Shor's order finding algorithm.

QED-C benchmarking allows the users to choose which of these quantum algorithms to benchmark. This is important because benchmarks for different quantum algorithms test performance on different tasks, and a user might not consider all the tasks to be relevant. Note however, that when the aim is to compare multiple quantum computers, the same set of quantum algorithms should be used for benchmarking all quantum computers whenever feasible.

Execution of the entire suite of benchmark programs results in a data file that stores all of the metrics collected for the tested quantum computer. Plotting all of the accumulated average result fidelity data produces charts like the ones shown in Figure 4.1.

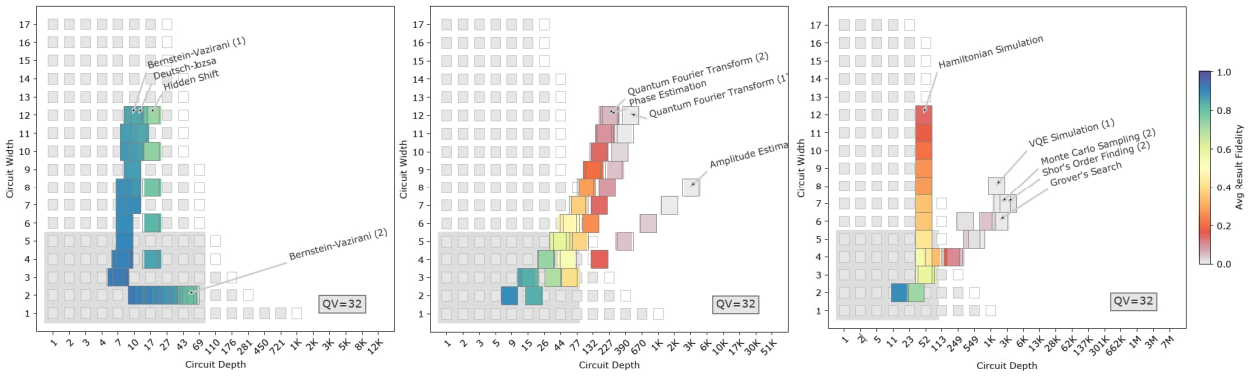

Figure 4.1: Examples of accumulated average result fidelity (source: QED-C)

Results for quantum circuits of the same width and depth are averaged together and shown as coloured squares (where the colours indicate the average result fidelity) on top of a "volumetric background" (the grey and white squares).

The volumetric background is a heuristic extrapolation of the quantum computer's Quantum Volume (QV), to predict the region in which a quantum circuits fidelity will be above 50% (grey squares) or below 50% (white squares). The heuristic extrapolation works as follows: a quantum circuit of width w and depth d is predicted to have a result fidelity above 50% if $w \times d < \log_2 (QV)^2$.

No volumetric background is likely to accurately predict the performance of all quantum circuits, including those corresponding with the quantum algorithms included in the QED-C benchmark suite. This is because a quantum circuit's performance is not only a function of its shape (i.e. its volumetric position), as it also depends on the type of errors present and whether those errors change with time or with number of qubits, and also depends on other characteristics (e.g. qubit connectivity). A volumetric background inferred from a QV may therefore involve particularly large extrapolations. For this reason, a quantum volume region (the large grey square) is included, alongside any volumetric background extrapolated from a QV. This quantum volume region encompasses all quantum circuit shapes that are smaller than the largest successful QV quantum circuit and therefore, quantum circuits within this region will have high result fidelity under weaker assumptions.

Note

The heuristic extrapolation of the QV is not expected to always be accurate but it is nevertheless useful. This is because any deviations between the performance of the algorithmic benchmarks and the prediction of the volumetric background signify that, for the quantum computer in question, the performance of these algorithms is difficult to predict from the QV alone.

QED-C claims that the benchmarking suite is designed to be readily accessible to a broad audience of users and provides benchmarks that correspond to many well-known quantum computing algorithms.

QED-C also predicts that this benchmarking suite is constructed to anticipate advances in quantum computing hardware that are likely to emerge in the next five years. The benchmarking suite is intended to be an evolving code base, accepting contributions from the quantum computing research community.

Combinatorial optimisation is anticipated to be one of the primary use cases for quantum computation in the coming years. Quantum computer algorithms for combinatorial optimisation have the potential to demonstrate significant runtime performance benefits over current classical state-of-the-art solutions.

Quantum annealers and gate-based (aka circuit-based) quantum computers solve combinatorial optimisation problems using fundamentally different strategies: Quantum Annealing (QA) versus Quantum Approximate Optimization Algorithm (QAOA). Both generic quantum algorithms operate on a target quantum system (backend) to solve a problem (with input arguments) and share an outer loop over a range of problem sizes, which encloses a second loop over a selectable number of restarts. The primary difference between both strategies lies within the restart loop, where the QA and QAOA solvers are applied to the input, and how the solution quality is evaluated over

increasing execution times. With QA, convergence to a solution is performed entirely within the Quantum Annealer.

QED-C is currently working on an enhancement of the QED-C benchmark suite for combinatorial optimisation problem solving by means of QA and QAOA algorithms.

4.2. Algorithmic Qubits (#AQ)

The Algorithmic Qubits (#AQ) benchmark (introduced by IonQ) was inspired by the benchmarking technique developed by QED-C (see § 4.1).

The #AQ benchmark is based on testing the performance of structured quantum circuits corresponding to a suite of currently popular quantum algorithms. Given the current "embryonic" state of quantum computing, it is expected that this suite will change in the future, as quantum computing matures. Updated suites of quantum algorithms will be identified with an #AQ version number. For #AQ version 1, six quantum algorithms from the QED-C repository have been chosen: QFT, QPE, QAE, Hamiltonian simulation, Monte Carlo sampling and VQE.

The formal #AQ definition embraces the "foundational notion" of quantum circuits running "approximately" n^2 entangling quantum gates over n qubits. IonQ claims that such quantum circuits are representative of "practical quantum algorithms that are of interest to the industry" but offers no proof to substantiate this claim.

The #AQ benchmark relates to the entire quantum computing stack, including such things as compiler optimisations that translate standard quantum gates into the native quantum gate set of the tested quantum computer and error mitigation (the latter has to be reported, if used). Optimisations are allowed as long as the quantum circuit executed on the quantum computer implements the same unitary operations as submitted and optimisation is not explicitly tuned for any specific benchmark quantum circuit.

The success of each quantum circuit run is measured by computing a classical fidelity function F_c, as follows:

$$F_c\left(P_{ideal}, P_{output}\right) = \left(\sum_x \sqrt{P_{output}(x) P_{ideal}(x)}\right)^2$$

where P_{ideal} is the ideal output probability distribution expected from the quantum circuit without any errors, P_{output} is the measured probability from the quantum computer and x represents each output result.

#AQ is defined as a single metric and is computed as follows:

- Let the set of circuits in the benchmark suite be denoted by C.
- Locate each circuit $c \in C$ as a point on the $2\mathrm{D}$ plot by its
  - Width, w_c = Number of qubits, and
  - Depth, d_c = Number of CX gates
- Define success probability for a circuit c, F_c:
  - Circuit passes if $F_c - \epsilon_c > t$, where ϵ_c is the statistical error based on the number of shots, $\epsilon_c = \sqrt{\frac{F_c(1-F_c)}{s_c}}$ where s_c is the number of shots, and $\mathrm{t} = 1/\mathrm{e} = 0.37$ is the threshold.
- Then, define #AQ=N, when

$$N = \max\left\{n : (F_c - \epsilon_c > t) \forall \left((c \in C) \& (w_c \leq n) \& \left(d_c \leq n^2\right)\right)\right\}$$

The #AQ benchmark results are presented as a “volumetric plot” (see Figure 4.2 for an example), similar to the QED-C plot (§ 4.1 Figure 4.1).

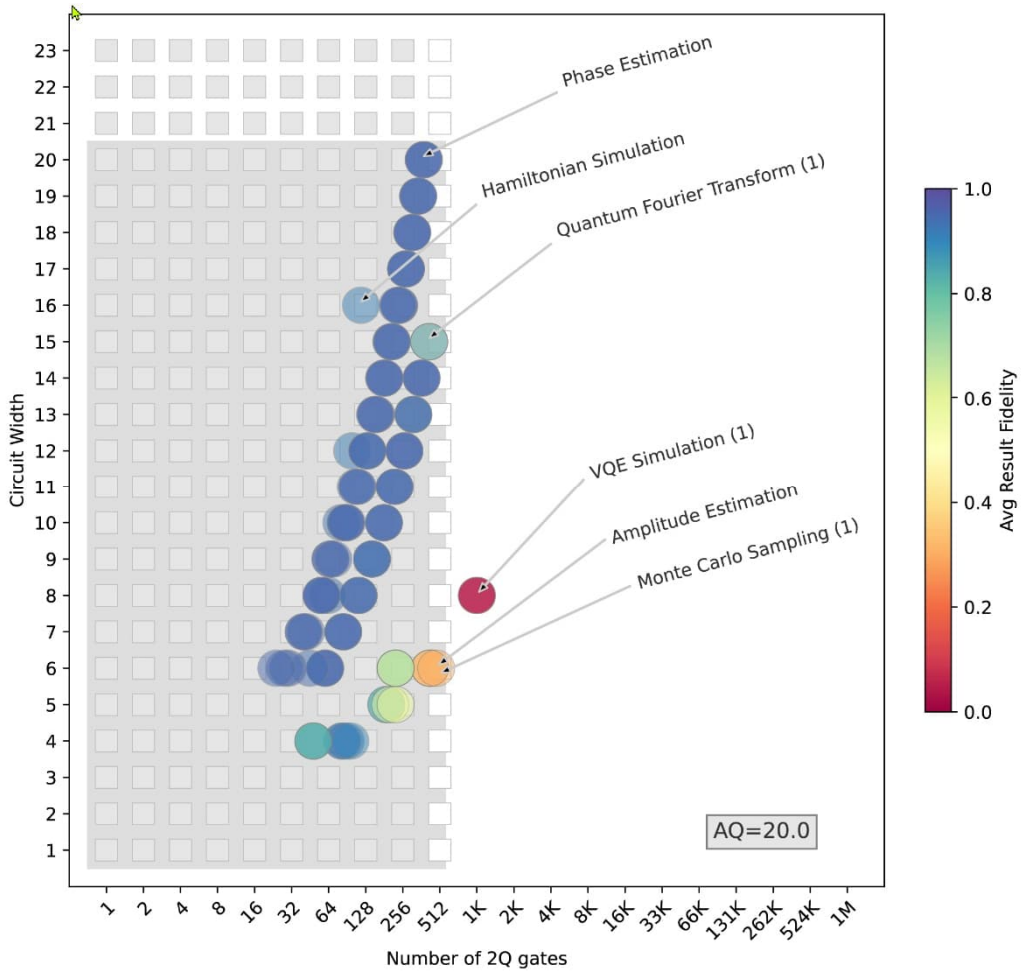

Figure 4.2: Example #AQ plot (source: IonQ)

4.3. Q-score

The Q-score metric (developed by Atos and available under an open-source license) measures the number of quantum bits that a quantum computer can use to solve a combinatorial optimisation problem, the MaxCut problem, significantly better than a classical random algorithm. In other words, it is an estimate of the largest MaxCut combinatorial optimisation problem that presumably can be solved faster on a quantum processor than on a classical computer. The assumption underlying this choice is that MaxCut is considered to be a good proxy for most computationally hard problem-solving quantum algorithms.

The MaxCut problem is solved with a NISQ-compatible hybrid quantum-classical algorithm, the Quantum Approximate Optimization Algorithm (QAOA), but any other quantum algorithm tackling the MaxCut problem can in principle be considered as a suitable candidate.

Q-score takes into account the performance of the entire quantum software/hardware stack, including optimisations at the algorithmic level, at the compilation stage or via noise mitigation techniques.

Note
This means that the risk exists that one could deliberately fine-tune the quantum software to spoof the Q-score benchmark.

The Q-score is computed as follows.

For a given size n, run the QAOA MaxCut algorithm 100 times on random Erdős-Rényi graphs[29] in $G(n, p = 0.5)$, the distribution of graphs obtained by taking an empty graph and connecting each pair of vertices with probability 0.5.[30]

Compute $\beta(n)$ as follows (the rationale for this formula is far too difficult to explain here):

$$\beta(n) = \frac{C(n) - \frac{n^2}{8}}{\lambda n^{3/2}}$$

where $C(n)$ is the average of the energies produced by QAOA over these 100 graphs.

The test for size n succeeds if $\beta(n) > \beta^\star$. The threshold $\beta^\star$, which has a value between 0 and 1, dictates how demanding the test is: a test with $\beta^\star = 0$ can be passed by a simple coin toss, a test

[29] The Erdős-Rényi model (named after the Hungarian mathematicians Paul Erdős and Alfréd Rényi) is used for generating random graphs where all graphs on a fixed vertex set with a fixed number of edges are equally likely.

[30] These graphs are relatively dense and constitute a standard class used for benchmarks.

with $\beta^\star$ = 1 can only be passed by a an exact solver. Atos decided for $\beta^\star$ = 0.2 (a somewhat arbitrary choice) and for λ = 0.178 (the rationale for this choice is far too difficult to explain here).

The Q-score $n^\star$ is the largest size n for which the test succeeds:

$$n^\star \equiv \max\{n \in \mathbb{N}, \beta(n) > \beta^\star\}$$

The Q-score benchmark can be run and computed on any gate-based quantum hardware and is compatible with, although not restricted to, NISQ QPUs. An extension to analogue quantum computers will be provided in the future.

4.4. SupermarQ

SupermarQ is quantum computer benchmark suite developed by Super.tech.

The SupermarQ suite includes the following quantum algorithms:

- Greenberger–Horne–Zeilinger (GHZ) for the generation of entanglement;
- Mermin-Bell for Bell inequality tests;
- error correction subroutines[31];
- Quantum Approximate Optimization Algorithm (QAOA);
- Variational Quantum Eigensolver (VQE);
- Hamiltonian simulation.

SupermarQ uses a set of feature vectors to quantify the coverage of the selected benchmark applications. These features indicate how each of the benchmarks will stress the quantum computer and to what degree. The SupermarQ features are:

- *quantum program communication*

  Quantum algorithms vary in the amount of communication needed between qubits. Within a quantum circuit, a qubit's "degree" is the number of other qubits it interacts with via multi-qubit quantum gate operations.

  It is often the case that physical qubit degree is much more uniform and limited than what is required for algorithmic qubits. For quantum hardware with less than all-to-all connectivity, the compiler may need to insert SWAP operations into the program to successfully map

[31] These are measured by means of proxy applications because NISQ quantum computers do not implement error correction.

between the algorithmic and physical qubits. SupermarQ uses the normalised average degree of the program's interaction graph to quantify the communication requirements of quantum circuits.

The program communication feature is computed by taking the average degree of the interaction graph divided by the average degree of a complete graph with an equivalent number of qubits. It is computed as

$$C = \frac{\sum_i^N d(q_i)}{N(N-1)}$$

for an N-qubit circuit, where $d(q_i)$ is the degree of qubit q_i.

The quantum program communication of sparsely connected applications will have values near to zero while densely connected applications will have values near to one.

- *critical-depth*

The lifetime of the information stored across a QPU's qubits (i.e. the coherence time) is limited. Furthermore, quantum gate errors accumulate when quantum circuits are being executed. Thus, it is essential that quantum circuits are of the shortest duration possible. The minimum duration for a quantum circuit is determined by the critical path: the longest span of dependent operations from circuit input to output.

The critical path is a valuable benchmarking metric because quantum hardware performance must exceed specific thresholds to accommodate continuously compounding quantum gate errors. Operations of particular interest are two-qubit interactions because two-qubit operations dominate single-qubit operations in terms of gate error and execution time on NISQ hardware.

The critical-depth feature gives context about how many two-qubit interactions in a program lie along the critical path and contribute to the overall circuit depth.

Critical-depth is calculated as

$$D = n_{e_d}/n_e$$

where n_{ed} is the number of two-qubit interactions on the longest path that sets the circuit depth and n_e is the total number of two-qubit interactions in the circuit.

Circuits that are heavily serialised will have a critical-depth that's close to 1.

- *entanglement-ratio*

  Entanglement is a critical property which gives quantum computing much of its strength as quantum algorithms without entanglement can be efficiently emulated by classical computers. It is a useful metric for quantum machine performance as it can be applied to computing tasks that demonstrate quantumness.

  While it is in general quite difficult to measure the precise amount of entanglement at every point within a quantum circuit (usually requiring access to the full quantum state vector), we can roughly capture this feature by computing the proportion of all gate operations (n_g) which are two-qubit interactions (n_e):

  $$E = n_e / n_g$$

- *parallelism*

  The structure of different quantum algorithms allow for varying degrees of parallelisation. Parallel operations can stress the quantum hardware because of correlated noise events known as crosstalk that degrade program performance. Crosstalk, which is often caused by simultaneous gate execution, is a common source of error in NISQ systems. This motivates the development of a feature that captures how susceptible a benchmark is to degradation via crosstalk. The parallelism feature represents this aspect by comparing the ratios of the number of qubits (n), gates (n_g), and the circuit depth (d):

  $$P = \left(\frac{n_g}{d} - 1\right) \frac{1}{n-1}$$

  Highly parallel applications fit a large number of operations into a relatively small circuit depth and will therefore have a parallelism close to 1.

- *liveness*

  During program execution, a qubit will either be involved in computation or it will be idle (i.e. waiting for its next instruction). In an ideal environment, the qubit's state would stay coherent while idling. In reality, unwanted environmental interactions such as amplitude damping, dephasing and correlated noise cause decoherence. The liveness feature captures aspects of an application's qubit status during its lifetime. It is defined as

  $$L = \frac{\sum_{ij} A_{ij}}{nd}$$

  where A is the liveness matrix defined by taking a quantum circuit and forming a matrix with n rows equal to the number of qubits and a number of columns equal to the circuit depth d. At every time-step of circuit execution (i.e. each column), a qubit may either be involved in an

operation or idle, corresponding to entries of 1 or 0 in the liveness matrix, respectively. In this way, the liveness feature gives a sense of how often the qubits are being acted upon.

The frequency of idling as $1 - L$ provides insight to qubit inactivity over its application lifetime.

- *measurement*

  Qubit-specific measurement is a critical part of quantum computing. It is required to extract information during and after a program's execution. NISQ devices suffer from non-trivial amounts of measurement error. In fault-tolerant quantum computing, error correcting codes use measurement to extract entropy from a noisy quantum system. The measurement feature

  $$M = l_{mcm}/d$$

  focuses specifically on the mid-circuit measurement and reset operations within a quantum program. For a quantum circuit composed of d sequential layers of gate operations (i.e. the circuit depth), l_{mcm} is the number of layers which contain these measurement and reset operations.

SupermarQ is built upon the cross-platform framework SuperstaQ[32], which allows an application written in OpenQASM to be executed on multiple backends (quantum computers and emulators) that do not support OpenQASM natively.

4.5. Quantum LINPACK

In order to mimic the success of the LINPACK benchmark on quantum computers, the problem of solving the Quantum Linear System Problem (QLSP) is considered. Many challenging high-dimensional problems in physics such as the simulation of a quantum many-body system, can be formulated in terms of QLSP.

The quantum LINPACK benchmark targets directly at the performance of quantum computers for scientific computing applications, as in the case of the LINPACK benchmark for classical supercomputers[33]. The quantum LINPACK benchmark can be concisely stated as the problem of using the quantum computer to evaluate the success probability p for a certain random matrix A.

---

[32] SuperstaQ is a hardware-agnostic software platform that connects applications to quantum computers from IBM Q, IonQ, Quantinuum and Rigetti Computing. SuperstaQ delivers performance gains via optimisations that span the entire hardware/software stack.

[33] The LINPACK benchmark, which first appeared in 1979 (the initial versions were based on the FORTRAN programming language), measures the floating-point computing power of a classical computer via its performance for solving linear systems of equations $Ax = b$. The input matrix A is a dense pseudo-random matrix, and there is no immediate application associated with such a matrix. This has led to much controversy over LINPACK's effectiveness in measuring

In order to perform the quantum LINPACK benchmark, it would be highly inefficient to generate a dense pseudo-random matrix A classically and then feed it into the quantum computer using for example QRAM. Matrices are used instead that are inherently easy to generate on quantum computers.

The input model used by quantum LINPACK is RAndom Circuit Block-Encoded Matrix (RACBEM), which is considered a proper generalisation of dense random matrices in the quantum setting, suitable for linear algebra tasks. The RACBEM model, and its Hermitian version H-RACBEM are simple to construct and allow to get access to in principle any n-qubit matrix and n-qubit Hermitian matrix, respectively, (up to a scaling factor) by adding only one ancilla qubit.

Furthermore, the problem of solving the Quantum Linear System Problem (QLSP) is considered. Many challenging high-dimensional problems in physics such as the simulation of a quantum many-body system, can be formulated in terms of QLSP.

With H-RACBEM and QLSP, the quantum circuit used in the quantum LINPACK benchmark can be designed to adapt to the qubit coupling map of almost any given gate-based quantum computer platform. All operations can be carried out with straightforward usage of basic one-qubit gates and CNOT gates, and there are no complex controlled unitaries involved. Due to the use of the basic gate set and the adaptivity to the quantum computer platform's architecture, the quantum LINPACK benchmark does not require the explicit use of a compiler.

Together with the recently developed technique of Quantum Singular Value Transformation (QSVT), a practical algorithm with a shallow quantum circuit is provided for performing the quantum LINPACK benchmark on NISQ quantum computers.

4.6. QASMBench

The QASMBench benchmark developed by DoE's Pacific Northwest National Laboratory (PNNL) is based on the OpenQASM Intermediate Representation (IR). It consolidates a large number of commonly used quantum routines and kernels from a variety of application domains including chemistry, simulation, linear algebra, searching, optimisation, arithmetic, machine learning, fault tolerance and cryptography.

QASMBench is an end-to-end package comprising a diverse variety of benchmark quantum circuits, techniques for evaluating the performance of a quantum circuit, and metrics for interpreting quantum circuit performance characteristics both pre- and post- transpilation.

---

the capability of classical computers in scientific computing applications since the very beginning. Nonetheless, LINPACK is widely used and performance numbers are available for almost all relevant systems. The LINPACK benchmark has also been used as the defining criterion of TOP500 supercomputers since the debut of the list in 1993.

Depending on the number of qubits involved in the benchmark, QASMBench is partitioned into three categories:

1. Small-scale, with the number of qubits ranging from 2 to 5. The purpose is to allow intensive measures such as density matrix tomography, with limited cost.

2. Medium-scale, with the number of qubits ranging from 6 to 15, for general benchmarking usage.

3. Large-scale, contains benchmarks with more than 15 qubits.

QASMBench benchmarking provides the following performance metrics (these metrics serve as useful indicators on how a quantum circuit can stress a NISQ quantum computer):

- *circuit width*

  Circuit width is defined as the number of qubits that enter the superposition state at least once within an application's lifespan (qubits that are measured in the interim of a quantum circuit and re-enter superposition are only counted as one qubit towards the circuit width).

  Circuit width dictates the spatial capacity required for a quantum device in order to run the quantum circuit and is defined as

$$\text{Circuit Width} = n_{q_{\text{active}}}$$

  where $n_{qactive}$ is the number of qubits that are in-use or demanded by the circuit.

- *circuit depth*

  Circuit depth is defined as the minimum time-evolution steps required to complete a quantum application. Time evolution is the process of completing all gates defined at time $t=t_j$, and once these are completed, the circuit moves onto time $t=t_{j+1}$, where the following gates are to be processed. To keep generality and avoid the impact of low-level optimisations, circuit depth is calculated solely using standard OpenQASM gates, i.e. the OpenQASM code is decomposed to standard gates before counting the accurate time evolution steps required.

  Circuit depth can be computed by decomposing OpenQASM code into a n_q x t matrix Q, where $Q_{qi,tj}$ is the time-evolution steps to complete the gate on qubit i at time j. The sum of the maximum time in each column is then equal to the minimum time required for a quantum application:

$$\text{Circuit Depth} = \sum_{j=1}^{t} \max_{0 \le i < \text{width}} (Q_{q_i,t_j})$$

- *gate density*

  Gate density, aka operation density, describes the occupancy of gate slots along the time-evolution steps of a quantum circuit. As certain qubits might need to wait for other qubits in the time evolution (caused by gate dependency), they remain idle by executing the ID gate. Consequently, if a gate slot is empty due to dependency, it implies a lower occupancy for the quantum hardware.

  Gate density is defined as

$$\text{Gate Density} = \frac{G_{1\text{-qubit}} + 2 \times G_{2\text{-qubit}}}{\text{Circuit Depth} \times \text{Circuit Width}}$$

  where $G_{1\text{-}qubit}$ refers to the number of 1-qubit standard gates and $G_{2\text{-}qubit}$ refers to the number of 2-qubit standard gates.

- *retention lifespan*

  Retention lifespan describes the maximum lifespan of a qubit within a system, and is motivated by the T_1 coherence time and the T_2 dephasing time of the quantum device.

  A longer lifespan of a quantum system implies that it is more susceptible to quantum information loss). Therefore, the qubit with the longest lifespan is taken to determine the system's retention lifespan. Using this metric, one can estimate if a particular circuit can be executed in a NISQ device with high fidelity, given these T_1 and T_2 values.

  Retention lifespan, which measures the circuit size and sensitivity to quantum system error, is the lifespan (depth) of the qubit with the longest lifespan, which is defined as

$$\text{Retention Lifespan} = \max_{0 \le i < width} (\log(D_i))$$

  where D_i is the lifespan (depth) of qubit i). As the circuit depth can grow substantially, a logarithmic factor is added to shrink the scale.

- *measurement density*

  Measurement density assesses the importance of measurements in a circuit. A higher measurement count implies that each individual measurement might be of relatively less importance (e.g. periodic measurement in QEC or measurement over ancilla qubits), whereas for application with less measurements, the measurements may be of utmost importance. The importance also increases when a measurement accounts for a wider and/or deeper circuit. A

good example is the swap test[34], where the circuit can be very large but only one measurement is taken to report the similarity. Consequently, this measurement is extremely important to the application.

Measurement density is defined as

$$\text{Measurement Density} = \frac{\log\left(\text{Circuit Depth} \times \text{Circuit Width}\right)}{N_{\text{measurement}}}$$

where $N_{measurement}$ is the number of measurements in an application. Since the circuit depth/width can be large and the importance of measurement decays when circuit depth/width keeps on increasing, a logarithmic factor is added to shrink the scale.

- *entanglement variance*

Entanglement variance measures the balance of entanglement across the qubits of a circuit.

Circuits with a higher entanglement variance indicate that certain qubits are more connected than other qubits (i.e. using more 2-qubit gates such as CX than others). This metric implies that when the circuit is mapped to a NISQ device:

- less SWAP gates are needed if those hotspot qubits are mapped to the central vertices in the NISQ device topology. A higher entanglement variance implies a higher potential benefit from a good logic-physical qubit mapping through quantum transpilation. If the entanglement variance is zero, little benefit should be expected from a better transpilation strategy;
- given that 2-qubit gates are one of the major sources introducing error, a higher entanglement variance implies uneven error introduction among qubits.

Entanglement variance is defined as

$$\text{Entanglement Variance} = \frac{\log\left(\sum_{i=0}^{width}\left(G_{q_i}(\text{2-qubits}) - \overline{G_q}(\text{2-qubits})\right)^2 + 1\right)}{\text{Circuit Width}}$$

where G_{qi}(2-qubits) is the number of 2-qubit gates operating on qubit i, and $\overline{G_q}$ is the average number of 2-qubit gates operating over each qubit. As the variance can be quite large, a logarithmic factor is added to shrink the scale. The inclusion of plus 1 is to eliminate log errors with 0 variance.

[34] The swap test is a procedure that is used to check how much two quantum states differ.

4.7. QPack

QPack (Figure 4.3) is an open source cross-platform benchmarking suite for quantum computers and emulators, developed by TU Delft. It is based on the testing of scalable variants of Quantum Approximate Optimization Algorithm (QAOA) and Variational Quantum Eigensolver (VQE) applications. Using a varied set of benchmark applications, an insight of how well a quantum computer or its emulator performs on a general NISQ-era application can be quantitatively made.

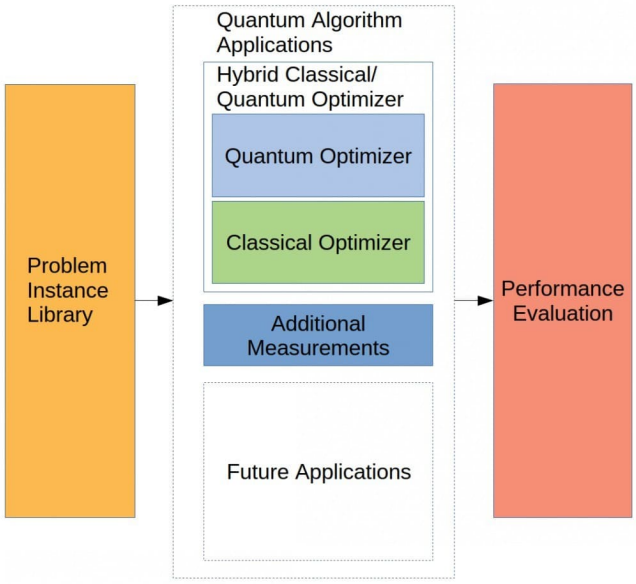

Figure 4.3: QPack benchmark overview (source: TU Delft)

QPack is built atop the cross-platform library LibKet[35], enabling the execution of the benchmark on multiple quantum computers and quantum emulators with a single program.

Performing a QPack benchmark results in a single number benchmark score composed of four sub-scores. These scores are based on the evaluation of the performance characteristics different Variational Quantum Algorithm (VQA) applications. Currently, QPack has implemented 6 VQAs, of which 4 are QAOA-based problems (Maximum Cut, Dominating Set, Maximal Independent Set and Traveling Salesperson Problem) and 2 are VQE-based problems (Random Diagonal Hamiltonian and Ising Chain).

The following design choices were made for the QPack benchmark score:

- the score reflects application-level performance of both quantum computers and quantum emulators;

[35] LibKet is a lightweight expression template library that allows developing quantum algorithms as quantum compiling platform-agnostic generic expressions and execute them on different quantum emulators and quantum computers without changing the program code.

- the score is a composite of measurement data of multiple quantum applications;
- the score is a single number (but may be split up into sub-scores);
- the score is proportional to performance, i.e. a higher score means higher performance;
- the score is scalable, i.e. it has no upper limit;
- the score does not become too abstract from the data it is based on;
- sub-scores are balanced, such that one sub-score does not become dominant in the overall score.

The QPack benchmark score is structured as shown in Figure 4.4.

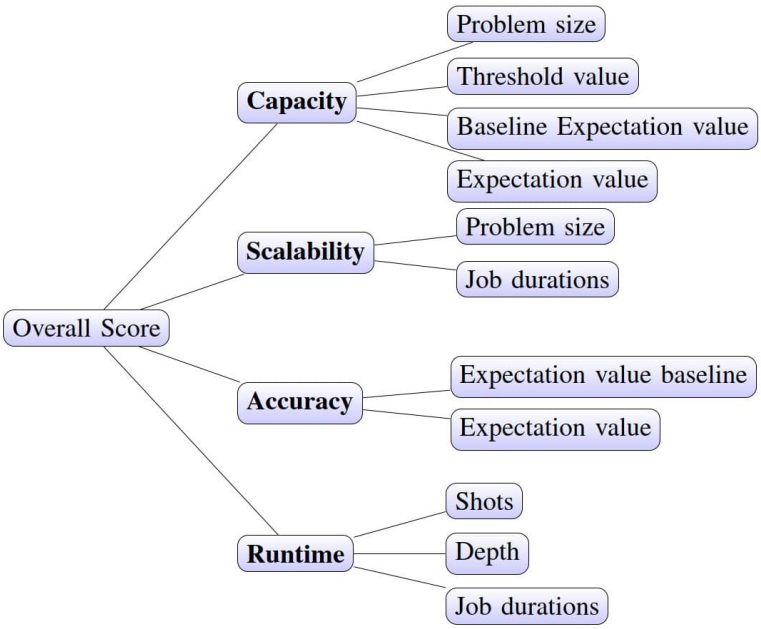

Figure 4.4: QPack benchmark score structure (source: TU Delft)

The four QPack benchmark sub-scores are:

1. *capacity*: the number of qubits that a quantum computer is able to run within a margin of the desired output accuracy;
2. *scalability*: relates to the runtime trend of a quantum computer with growing circuit size (even though a quantum computer has a certain number of qubits to work with, its qubit topology

could make the implementation of larger circuits less efficient as mapping and transpiling optimisation becomes more complex)[36];

3. *accuracy*: output state deficiencies due to gate noise and qubit decoherence and relaxation are compared to the ideal ones (as generated by the QuEST quantum emulator[37]);

4. *runtime*: the time it takes to execute a quantum circuit (for the runtime score, it is assumed that quantum computers can execute quantum gates in parallel where possible).

The runtime sub-score is purely based on the execution time of the quantum circuit and does not include other service overhead like optimisation, transpiling and scheduling. Although this sub-score gives an insight into the performance of the actual quantum hardware, it does not reflect all run time factors of the system. Overhead is a major component in quantum job runtime and is therefore also contributing to the run time. Since quantum providers generally only return the quantum circuit execution time in their response, it was chosen not to include the overhead component in the current version of QPack. However, because the impact of this overhead has such an significant influence on quantum runtime, the next version of QPack should include it.

For the benchmark sub-scores, a distinction is made between pure scores and mapped scores. The pure scores are the values to which the measured data is transformed into a quantitative score metric. The mapped scores take these pure scores and map them to be proportional to performance and be balanced against the other sub-scores.

After a mapped and balanced sub-score has been defined for each problem in the QPack suite, the sub-scores for each problem are combined into an overall score. The sub-score for each performance category is computed as the arithmetic mean over all problems. Figure 4.5 shows an example of the scores obtained in this way.

Computing the sub-scores for a single quantum computer and displaying them like in Figure 4.5 gives a good insight of the performance of a single quantum computer, but makes comparison of performance differences between multiple quantum computers cumbersome. A better way of visualising performance differences between multiple quantum computers can be achieved with radar plots as shown in Figure 4.6. Using this visualisation, the overall score for a given quantum computer is given as a single metric by taking the area of the four-sided region in its radar plot, thus making it possible to easily compare different quantum computers.

[36] This metric relates to the “scale” aspect of a quantum computer’s performance and does not relate to the “scalability” of the quantum computing platform.

[37] The Quantum Exact Simulation Toolkit (QuEST) is a high performance emulator of quantum circuits, quantum state vectors and density matrices. QuEST uses multithreading, GPU acceleration and distribution to run fast on laptops, desktops and networked supercomputers.

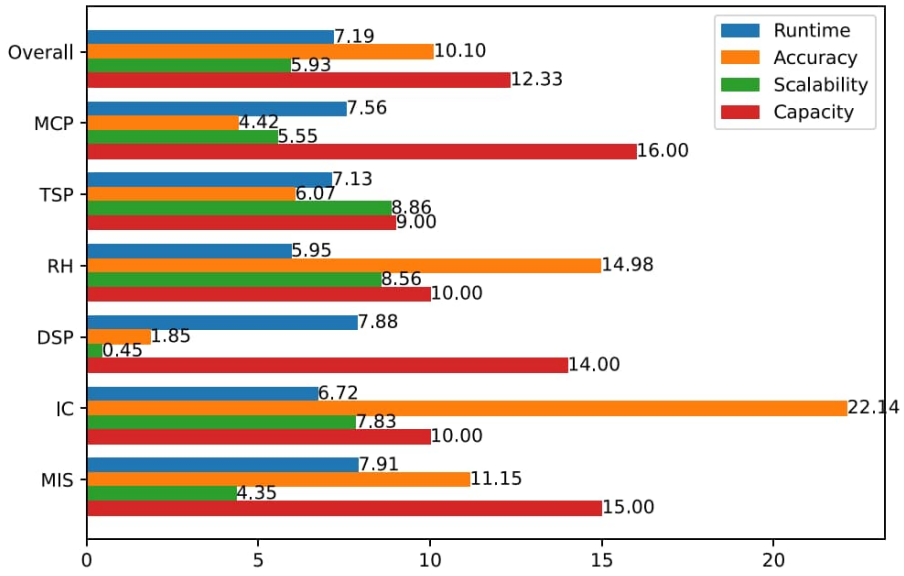

Figure 4.5: Example of QPack benchmark sub-scores for a single system (source: TU Delft)

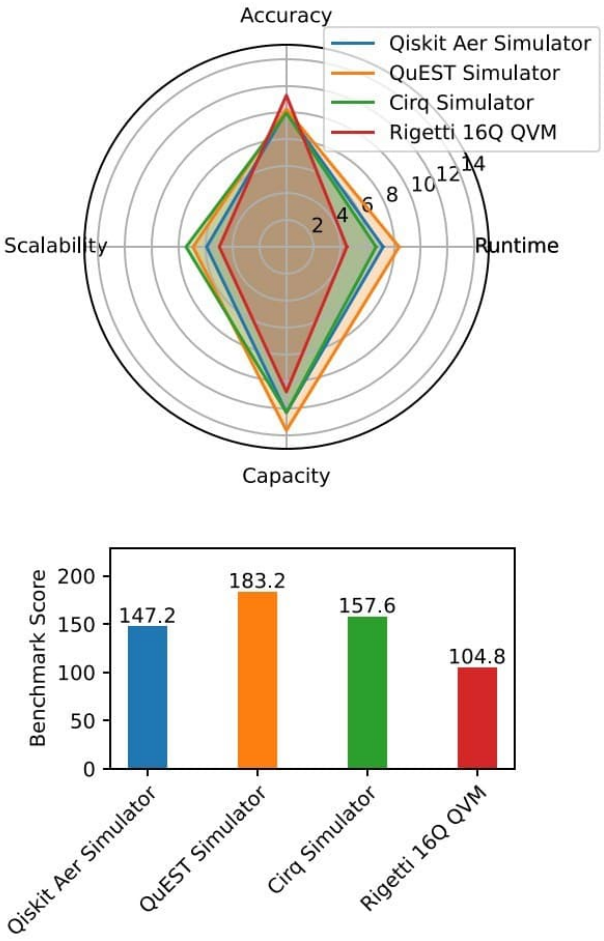

Figure 4.6: Example of QPack benchmark overall scores for multiple systems (source: TU Delft)

4.8. QuantMark

Variational Quantum Eigensolver (VQE) is a promising hybrid quantum/classical algorithm that runs on NISQ quantum computers.

A VQE algorithm has many components that are interchangeable. As such, different implementations of the algorithm are based on different choices for these components. This makes it very difficult to determine whether any performance difference between performance test results is due to an algorithm difference or another type of difference. Thus, it is very difficult to compare reported performance test results.

To address this issue, researchers from the University of Helsinki developed the QuantMark benchmark[38], which:

- can show not only the precision but also the accuracy of a particular variant of the VQE algorithm;
- makes it easy to compare the performance of different variants of the VQE algorithm;
- makes it easy to replicate benchmark results, for example to enable evaluation of expected performance improvements for a particular variant of the VQE algorithm.

4.9. Variational Quantum Factoring (VQF)

Zapata Computing Holding Inc. proposes to benchmark NISQ Quantum computers with Variational Quantum Factoring (VQF) and fermionic simulation quantum algorithms.

VQF solves the integer factoring problem with variational quantum solutions such as VQE and QAOA.

The fermionic simulation includes the 1D Fermi-Hubbard model, which is representative of chemistry and material science problem solutions. Its analytical solution is known and can be easily extended into a 2D structure, which can be converted to a quantum algorithm. The metric for this benchmark is the effective fermionic length of the device, which can reflect the performance of the entire device.

[38] QuantMark is still under active development.

4.10. Other application-level benchmarks

Various other application-level benchmarks have been developed, including:

- Agnostiq created a benchmark dedicated to optimising financial portfolios using the Quantum Approximate Optimization Algorithm (QAOA).
- DoE's Oak Ridge National Laboratory (ORNL) proposed a benchmark for chemical simulation.
- New-York State University scientists created a benchmark related to Grover's search algorithm.

Appendix A - References

[Atos 2021] Benchmarking quantum co-processors in an application-centric, hardware-agnostic and scalable way

[IBM 2021] Scale, Quality, and Speed: three key attributes to measure the performance of near-term quantum computers

[IEEE 2022] QuantMark: A benchmarking API for VQE Algorithms

[IEEE 2022] SupermarQ: A scalable Quantum Benchmark Suite

[IonQ 2023] Algorithmic Qubits: A Better Single-Number Metric

[MITRE 2022] An Improved Volumetric Metric for Quantum Computers via more Representative Quantum Circuit Shapes

[NOREA 2024] Quantum Annealing Explained

[NOREA 2024] Quantum Computing Explained

[NOREA 2024] Quantum Software Development Tools

[npj 2019] A generative modeling approach for benchmarking and training shallow quantum circuits

[PNNL 2022] QASMBench: A Low-Level Quantum Benchmark Suite for NISQ Evaluation and Simulation

[QED-C 2023] Application-Oriented Performance Benchmarks for Quantum Computing

[SNL 2020] A volumetric framework for quantum computer benchmarks

[TUD 2022] QPack: A cross-platform quantum benchmark-suite

[UC 2021] Random circuit-block encoded matrix and a proposal of quantum LINPACK benchmark

[Zapata 2020] An application benchmark for fermionic quantum simulations

Appendix B - Acronyms and abbreviations

1D	1-Dimensional
1Q	1-Qubit quantum gate
2D	2-Dimensional
2Q	2-Qubit quantum gate
#AQ	number of Algorithmic Qubits
AI	Artificial Intelligence
aka	also known as
API	Application Programming Interface
AQ	Algorithmic Qubits
Avg	Average
BAS	Bars *a*nd Stripes
BBN	*Bolt, Beranik and Newman*
bit	binary digit
c	*communication*
C	Clifford gate
Caltech	California *I*nstitute of *T*echnology
CB	Cycle Benchmarking
Cirq	*Circuit*
CLOPS	Circuit Layer Operations Per Second
CNOT	Controlled NOT gate
CPU	Central Processing Unit
CX	Controlled X gate
d	depth
D	Depth
DARPA	Defense Advanced Research Projects Agency
DD	Dynamical Decoupling
DDQCL	Data-Driven Quantum Circuit Learning
DoE	Department of Energy

DSP	Dominating Set Problem
E	Entanglement ratio
e.g.	exempli gratia
EDP	Electronic Data Processing
EPLG	Error-Per-Layered Gate
etc.	et cetera
FD	Fermionic Depth
FORTRAN	FORmula TRANslation
FTQC	Fault-Tolerant Quantum Computer
GBS	Gaussian Boson Sampling
GHZ	Greenberger–Horne–Zeilinger
GPU	Graphics Processing Unit
GST	Gate Set Tomography
H	Hadamard gate
H-RACBEM	Hermitian RACBEM
HOG	Heavy Output Generation
HRL	*Hughes Research Laboratories*
i.e.	id est
IBM	International Business Machines
IC	Ising Chain
ID	I*d*entity gate
IEEE	Institute of Electrical and Electronics Engineers
Inc.	Incorporated
IR	Intermediate Representation
J	Joule
L	Liveness
lab	laboratory
LF	Layer Fidelity

log	logarithm
	logarithmic
M	Measurement
max	maximum
MaxCut	Maximum Cut
MCP	Maximum Cut Problem
MIS	Maximal Independent Set
NISQ	Noisy Intermediate-Scale Quantum
NOREA	Nederlandse Orde van Register EDP-Auditors
npj	*N*ature *P*artner *J*ournals
O	big O notation
OpenQASM	Open Quantum A*ss*se*m*bly language
ORNL	Oak Ridge National Laboratory
P	Parallelism
PNNL	Pacific Northwest National Laboratory
polylog	polylogarithmic
Q	Quality
	Quantum
QA	Quantum Annealing
QAB	Quantum Algorithm Benchmarking
QAE	Quantum Amplitude Estimation
QAOA	Quantum Approximate Optimization Algorithm
QASMBench	Quantum A*ss*se*m*bly Benchmark
qBAS	quantum Bars *a*nd Stripes
QBI	Quantum Benchmarking Integration
QCaaS	Quantum Computing-as-a-Service
QCP	Quantum-Classical Processing
QCVV	Quantum Characterization, Verification, and Validation
QEC	Quantum Error Correction
QED-C	Quantum Economic Development Consortium
QEM	Quantum Error Mitigation

QFT	Quantum Fourier Transform
Qiskit	Quantum *I*nformation *S*oftware *K*it
QLSP	Quantum Linear System Problem
QMI	Quantum Machines Inc.
QML	Quantum Machine Learning
QPE	Quantum Phase Estimation
QPT	Quantum Process Tomography
QPU	Quantum Processor Unit
QRAM	Quantum Random-Access Memory
QSDK	Quantum Software Development Kit
QST	Quantum State Tomography
QSVT	Quantum Singular Value Transformation
qubit	quantum bit
QuEST	Quantum Exact Simulation Toolkit
QV	Quantum Volume
	Quantum Volumetric
QV-*n*	Quantum Volumetric class *n*
QVM	Quantum Virtual Machine
RACBEM	RAndom Circuit Block-Encoded Matrix
RB	Randomized Benchmarking
RCS	Random Circuit Sampling
RFP	Request For Proposal
RH	Random Hamiltonian
RPE	Robust Phase Estimation
s	second
SNL	Sandia National Laboratories
SPAM	State Preparation And Measurement
SU	Special Unitary
SU(2)	Special Unitary transformations applicable to 1 qubit
SU(4)	Special Unitary transformations applicable to 2 qubits
SU(n^2)	Special Unitary transformations applicable to n qubits
SWAP	Size, Weight, Application and Power

TSP	Traveling Salesman Problem
TU	Technische Universiteit
TUD	Technische Universiteit Delft
U	Unitary gate
UC	University of California
UK	United Kingdom
uni	uniform
URB	Universal Randomized Benchmarking
USA	United States of America
USC	University of Southern California
VB	Volumetric Benchmark
VQA	Variational Quantum Algorithm
VQC	Variational Quantum Circuit
VQE	Variational Quantum Eigensolver
VQF	Variational Quantum Factoring
W	Width
X	*P*auli X gate
XEB	*C*ross-Entropy Benchmarking
XRB	*Ex*tended Randomized Benchmarking
Y	*P*auli Y gate
Z	*P*auli Z gate