IEEE Transactions on Quantum Engineering

Received February 25, 2021; revised April 26, 2021; accepted May 20, 2021; date of publication June 17, 2021; date of current version August 3, 2021.

Digital Object Identifier 10.1109/TQE.2021.3090207

Benchmarking Quantum Coprocessors in an Application-Centric, Hardware-Agnostic, and Scalable Way

SIMON MARTIEL, THOMAS AYRAL, AND CYRIL ALLOUCHE
Atos Quantum Laboratory, 95877 Les Clayes-sous-Bois, France

Corresponding author: Thomas Ayral (thomas.ayral@atos.net)

ABSTRACT Existing protocols for benchmarking current quantum coprocessors fail to meet the usual standards for assessing the performance of high-performance-computing platforms. After a synthetic review of these protocols—whether at the gate, circuit, or application level—we introduce a new benchmark, dubbed Atos Q-score, which is application-centric, hardware-agnostic, and scalable to quantum advantage processor sizes and beyond. The Q-score measures the maximum number of qubits that can be used *effectively* to solve the MaxCut combinatorial optimization problem with the quantum approximate optimization algorithm. We give a robust definition of the notion of effective performance by introducing an improved approximation ratio based on the scaling of random and optimal algorithms. We illustrate the behavior of Q-score using perfect and noisy simulations of quantum processors. Finally, we provide an open-source implementation of Q-score that makes it easy to compute the Q-score of any quantum hardware.

INDEX TERMS Combinatorial optimization, quantum algorithms, quantum benchmarking.

I. INTRODUCTION

Recent years have witnessed great progress in the field of quantum technologies, whether on the hardware side—with growing computer sizes and quantum operation fidelities—or on the software side—with many algorithmic improvements. This progress has, among other achievements, enabled recent claims that quantum advantage—the capacity for a quantum processor to outperform a classical machine—was attained by some of the most advanced noisy, intermediate scale quantum (NISQ) [34] processors [1], [40]. However, these claims pertain to rather contrived, if not useless computational tasks carefully tailored for specific quantum processors.

In fact, the crucial milestone for the field to truly come of age is to identify hard, real-world computational problems whose solution can be accelerated by quantum computers. Many different hardware platforms with many different algorithmic ideas are vying for this goal today. This diversity of quantum hardware and software candidates for quantum advantage requires a precise metric of success in order to appraise the relative power of each quantum computing stack for outperforming classical computers. This metric will not only provide a much-needed synthetic overview of the current status of the field to end-users such as the high-performance-computing (HPC) community, but it will also help fuel the quantum community's efforts toward real-world applications.

This metric must fulfill a number of criteria to achieve these goals.

1) *Application-centric:* The metric must measure the ability to solve a hard, real-world computational problem that should be, at least to some extent, representative of a wide class of computational problems relevant to industry.
2) *Hardware-agnostic:* The metric must not favor any hardware or software over another.
3) *Scalable:* One must be able to compute the metric for large problem sizes. In particular, the scaling of the classical processing time for computing the metric must be polynomial with the problem size.

In the field of classical HPC, these criteria are typically fulfilled by the LINPACK benchmark [12] that is used to rank the TOP500 supercomputers. In the field of quantum computing, a number of metrics have already been proposed in the literature. As we will explain in more detail in the following section, none of them fulfill all the above requirements.

The purpose of this article is to fill this gap by proposing a metric, dubbed "Q-score," that satisfies these requirements. Essentially, the Q-score measures the maximum number of quantum bits that a quantum computer can use to solve a combinatorial optimization problem—the MaxCut problem—significantly better than a classical random algorithm. In other words, it is an estimate of the largest (MaxCut) combinatorial optimization problem that can be solved better (compared to a random heuristic) on a quantum processor than on a classical computer. Q-score can be run and computed on any gate-based quantum hardware. Q-score is compatible with, although not restricted to, NISQ QPUs. In the version of Q-score presented in this article, we solve the combinatorial problem at hand, MaxCut, with a NISQ-compatible hybrid quantum-classical algorithm, the quantum approximate optimization algorithm (QAOA) [15]. Yet, any quantum algorithm tackling the MaxCut problem can in principle be considered as a suitable candidate. It takes into account the performance of the compilation. An implementation is available under an open-source license.

To define the Q-score, we carefully investigate the size-dependence of the average performance of random and optimal classical algorithms, as well as the QAOA quantum algorithm, for solving the MaxCut problem on classes of random graphs. The metric that we propose, akin to an improved approximation ratio, allows to measure nontrivial performance above the level of random classical algorithms. Finally, we illustrate the behavior of the Q-score using noisy simulations with a depolarizing noise intensity compatible with today's NISQ processors.

This article is organized as follows. We start by spelling out the desirable properties of quantum metrics and by reviewing the main existing quantum metrics (Section I). We then describe the Q-score protocol (Section II) and discuss its properties (Section III). We finally explain how to run this benchmark using an open-source script we provide online (Section IV).

II. CHARACTERIZING QUANTUM PROCESSORS: GOALS AND PRIOR WORK

The careful design of quantum characterization, verification and validation (QCVV) protocols is crucial for assessing the potential of current and future quantum processing units (QPUs). Several such protocols have been proposed in the recent years, with various levels of proximity to applications, scalability, fairness, and practicality.

In this section, we start by laying out the QCVV criteria we deem to be most important from an HPC perspective. We then briefly review the main existing proposals and to what extent they fulfill these criteria.

A. HPC-DRIVEN LIST OF CRITERIA

The first useful applications of quantum processors will likely be demonstrated in setups where quantum coprocessors will be used as accelerators for performing very specific hard computational tasks within an HPC system. The usefulness of the coprocessor will be measured by comparing the performance of such a (possibly hybrid) computation with the performance of its purely classical counterpart. With this in mind, we argue that useful QCVV protocols should fulfill the following three criteria.

1) *Application-centric:* The protocol should yield a single number (or a few) that unequivocally reflects the potential of a given QPU for solving a real-life HPC application. Ideally, the score of the QPU for this given application should be a proxy for how well the processor performs in general, i.e., for other applications. This focus on applications and its "holistic" goal excludes protocols that narrow the characterization down to low-level components only, such as, e.g., gate quality or ability to sample specific classes of circuits (random or square circuits).

2) *Hardware-agnostic:* The protocol should put all the existing or future hardware technologies on an equal footing. In particular, it should not unduly favor a given technology over the others. Focusing on applications (see previous point) already ensures that the benchmark will incentivize hardware makers to make meaningful overall improvements, instead of focussed fine-tunings aimed at spoofing the benchmark as can more easily happen for gate- or circuit-level protocols. The application itself should also not be targeted to a given platform, and the difficulty of solving it should be representative of that of solving other hard problems (one wants to avoid niche applications that are contrived to perform well only in particular circumstances, and whose level of complexity is not easily comparable to other problems; here, we believe that our choice of QAOA, which is quite representative of variational algorithms, and of the MaxCut problem, whose encoding requires a reasonably low number of qubits, meets these demands—while being easily adjustable to better-performing future processors).

3) *Scalable:* The protocol should be scalable to large numbers of qubits. In particular, the classical computational complexity for processing the quantum output and outputting the metric should be reasonably moderate. This constraint excludes protocols that involve classical computations that are exponentially costly in the number of qubits.

B. PRIOR PROPOSALS

Most previously proposed QCVV protocols focus on gate-level and circuit-level characterization. We briefly review these protocols, which give valuable, albeit partial insights into the performance of a given QPU. We then turn to the previous attempts at characterizing QPUs from an application perspective.

1) GATE-LEVEL PROTOCOLS

In the past years, several protocols have been proposed to characterize the performance of the main low-level components of QPUs, quantum gates, and sequences of gates—namely, quantum circuits. The corresponding metrics give valuable information to compare different implementations

of similar quantum technologies, such as two different experimental realizations of superconducting transmon processors. They also give indications about the ability of QPUs to run certain classes of quantum circuits.

The most widely used protocol for characterizing the gate-level quality of a QPU is *randomized benchmarking* (RB) [25]. It yields the average fidelity f or average error rate $\epsilon = 1 - f$ of a given gate set [35] while requiring only polynomial classical resources (provided potentially exponential compilation overheads are avoided, such as in direct randomized benchmarking [31]). It is also robust to state preparation and measurement (SPAM) errors. These two aspects are major advantages over direct fidelity estimation protocols. On the flip side, RB is not application-centric: it gives little information as to the performance of circuits, let alone applications. Indeed, structured circuits (as opposed to the random circuits used in RB) are more sensitive to errors than randomized circuits, and thus, one can hardly predict the performance of a structured circuit given the RB metrics of its gate set (see [36] for protocols that use structured circuits). One major reason for this deficiency is that RB gives little information about crosstalk errors, which influence the performance of a QPU at the circuit level (although we note that recent works propose ways of extending RB to crosstalk estimation [23]).

Another widely used protocol that goes beyond the measurement of the mere average fidelity of a gate set is *gateset tomography* (GST) [5], [17], [26]. This quantum process tomography method yields the specific noise model of each quantum operation that a QPU is able to perform, including gates, state preparation, and measurement. Once the so-called GST gauge has been properly fixed (see, e.g., [9]), average fidelities can be extracted from the noise models. More interestingly, the noise models can be used as inputs to circuit-level simulations. These simulations are generically exponentially costly in the number of qubits, but can yield precise information about the behavior of a given circuit executed on a given processor. However, if GST is realized at the one-qubit and two-qubit level only, crosstalk effects beyond two-qubit crosstalk, which are suspected to play an important role in NISQ devices, will be neglected. Going beyond one-qubit and two-qubit errors to capture those effects is possible, but requires a cost in terms of classical processing *and* amount of data to be collected from the QPU that scales exponentially with the number of qubits. Thus, GST is hardly scalable for real-world applications.

Recently, a protocol called *cycle benchmarking* (CB) [14] has been proposed to go beyond the limitation of RB and GST to the characterization of few-qubit error processes. While RB and GST require a number of experiments that scales exponentially with the number of qubits involved in the quantum process to be characterized (whether an error or a gate), thus limiting them to very few qubits, CB allows to characterize processes acting on much larger registers. This applies to crosstalk errors (see previous paragraph), but also to multiqubit gates like the Mølmer–Sorensen gate of trapped-ion processors that act on multiple (even all) qubits in a register. In additional, CB is robust to SPAM errors [14]. However, as a gate-level protocol, it cannot be used to characterize the potential of a QPU at an application level.

2) CIRCUIT-LEVEL PROTOCOLS

A number of protocols has been proposed to measure the ability of QPUs to run certain classes of circuits.

One such protocol is the *quantum volume* (QV) [7] metric, and its generalization to nonsquare circuits, *volumetric benchmarks* (VB) [6]. They measure the ability of a QPU to prepare a random state given a certain number of qubits (circuit width) and a certain gate count (circuit depth). While QV looks only at square circuits (with equal width and depth), which fails to capture algorithms that do not involve square circuits (like Shor's algorithm), VB lifts this limitation. However, the core metric of QV/VB, namely the heavy output generation probability (HOV) [3], requires the exponentially costly computation of probability amplitudes (to compute the set of heavy outputs). These approaches are thus not scalable. Furthermore, they focus on classes of random circuits, making them hardly suitable for assessing the performance of a QPU on a real application.

Related protocols, dubbed *cross-entropy benchmarking* (XEB) and *cross-entropy fidelity* [1], [30], have been recently proposed and used to compare the ability of a QPU to generate random states with that of a classical computer. Like QV and VB, these protocols require classical resources that are exponential in the number of qubits, thereby limiting their scalability. Second, the task they seek to optimize is the sampling of bitstrings measured after executing families of random circuits. The performance of a given QPU in solving such a specific problem hardly qualifies as application-centric in the absence of a straightforward extrapolation of the corresponding metric to real-world applications.

Recently, the authors in [28] proposed a series of benchmarks that comprise the QV, VB, and XEB metrics together with the l_1 norm to compare probability distributions. Similar limitations in terms of the classical complexity to compute the metric and difficulty to use it as a proxy for an actual application also apply to this article.

The authors in [36] recently proposed a protocol based on the "mirroring" concept (also used in RB) that allows to get rid of the exponential classical effort that plagues the previous circuit-level protocols. Yet, the ability to use this other circuit-level metric to reliably predict the behavior of a given QPU for a real application remains to be investigated.

3) APPLICATION-LEVEL PROTOCOLS

One of the most promising applications of quantum processors is the field of quantum many-body physics, since quantum processors are by construction quantum many-body

systems with a large number of quantum bits interacting with one another in a controlled fashion.

The authors in [8] recently proposed a metric dubbed *fermionic depth* (FD) to quantify the ability of a QPU to tackle a quantum many-body problem. The prototypical many-body problem chosen in this article is the 1-D Fermi–Hubbard model, whose ground-state energy in the infinite-size limit, E_∞^{exact}, can be computed exactly in polynomial time on a classical computer via the so-called Bethe Ansatz method [22]. The protocol consists in computing, with a QPU, the approximate ground-state energy of this model E_L for different (linear) sizes L, and then returning the deviation to the exact energy at infinite size, $\Delta E_L = E_L - E_\infty^{\text{exact}}$. In practice, due to the limited coherence of current (NISQ) processors, E_L is computed via a hybrid quantum-classical method, the variational quantum Eigensolver (VQE, [32]) method, as opposed to fully coherent algorithms like the quantum phase estimation algorithm, that are not suitable for nonerror-corrected QPUs. Due to decoherence effects, the corresponding ΔE_L curve is going to display a minimum at a given size L^*, dubbed the *fermionic length* of the QPU under investigation. This fermionic length, thus, gives an indication about the maximum size of a fermionic problem that a given QPU can handle.

The predictive power of this metric for problems outside the 1-D Fermi–Hubbard model remains to be investigated: whether the fermionic length estimated for a 1-D problem is related to the fermionic length that can be achieved for 2-D quantum many-body problems is an open question. Indeed, those 2-D problems, which are among the hardest to tackle with the most advanced classical algorithms, display phenomena (high-temperature superconductivity, pseudogap phase,...) that are radically different from 1-D problems.

Quantum chemistry problems, on the other hand, usually feature interactions between many orbitals, whereas the Hubbard model has only local interactions, raising the question of the relevance of the fermionic length for chemistry problems. We note that [27] proposed a chemistry-based benchmark of quantum processors, albeit with a focus on small molecules only and therefore no clear path towards scalability yet.

Finally, the authors in [11] proposed an extension of the LINPACK benchmark (that is used to rank classical supercomputers) to a quantum setting. The protocol consists in solving a linear system of equations $Ax = b$, with A a random dense matrix, by outputting an approximate solution $g(A)|b\rangle$ with $g(x)$ a polynomial approximation of x^{-1}. While this protocol avoids the usual read-in problem (it does not require the use of a QRAM to load A from classical data) through a block-encoding method (random circuits U_A are used such that one of the blocks of this unitary is A, with A a random dense matrix), its measure of success consists in comparing the output vector $g(A)|b\rangle$ to the actual solution (in addition to a measure of the wall-clock time). This entails an exponential classical cost (through e.g., a cross-entropy test), which limits the scalability of the method.

III. PROTOCOL

In this section, we describe our benchmark metric proposal. Similarly to other benchmark proposals, Q-score works by iteratively testing a quantum coprocessor using a scalable test T_n indexed by a *problem size* n. Naturally, the score will be the largest problem size $n^\star$ such that $T_{n^\star}$ holds.

Informally the test consists of the following:

1) picking a collection of random graphs of size n;
2) running a QAOA-MaxCut algorithm on these graphs and computing $C(n)$, the average of the expected cut cost for each instance;
3) computing a score $\beta(n)$ that depends on $C(n)$ and testing $T_n : \beta(n) > \beta^\star$ for some constant $\beta^\star$.

The next section is dedicated to the description of this test T_n. The detailed explanation of the various choices described in this section can be found in Section III.

A. DESCRIPTION OF THE TEST

Our test T_n consists in running a quantum approximate optimization algorithm (QAOA) for a MaxCut instance of size n. We now describe the settings in which the algorithm is run, and how its performance is assessed for a given instance size.

1) CIRCUIT IMPLEMENTATION

We assume that we tackle instances using the standard QAOA Ansatz as described in [15]. Given a graph $G = (V, E)$ (with V and E the vertex and edge set, respectively) and a depth parameter p, we implement the parameterized circuit

$$U(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \prod_{1 \le q \le p} e^{-i\frac{\beta_q}{2}H_0} e^{-i\frac{\gamma_q}{2}H_G} \tag{1}$$

where $H_0 = -\sum_{1 \le i \le n} \sigma_x^{(i)}$ and

$$H_G = \sum_{i,j \in E} \sigma_z^{(i)} \sigma_z^{(j)} - \frac{|E|}{2}. \tag{2}$$

Here, σ_x and σ_z denote the Pauli X and Z operators, and $|E|$ is the number of edges in the graph.

In practice, each rotation $e^{-i\frac{\gamma_q}{2}\sigma_z^{(i)}\sigma_z^{(j)}}$ is decomposed using a subcircuit of 2 CNOT gates and a single R_Z rotation. The propagator $e^{-i\frac{\beta_q}{2}H_0}$ is implemented using a wall of R_X gates.

2) CLASSICAL OPTIMIZER

The classical optimization routine used to minimize the Ansatz energy is COBYLA [33]. This optimizer behaves well in perfect settings and for shallow circuits (i.e., circuits with a low number of parameters). Since we expect that increasing the depth p of the Ansatz will probably only degrade performances, this choice seems reasonable. Indeed, although increasing the depth leads to a larger variational search space and thus a potentially lower variational energy, on NISQ QPUs, larger depths also lead to an increased sensitivity to noise, and thus, usually degraded performances.

IEEE Transactions on Quantum Engineering

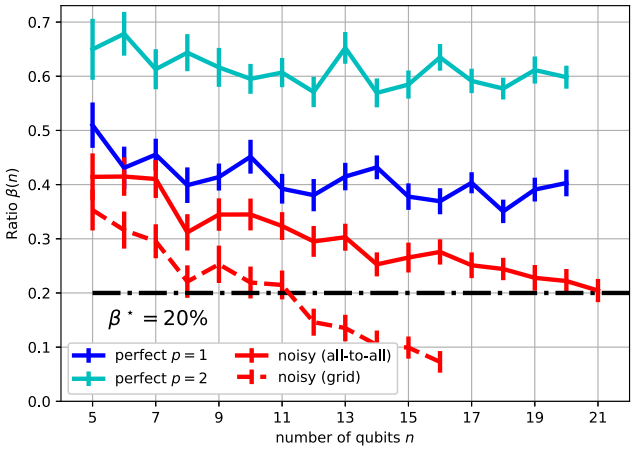

FIGURE. 1. Evolution of $\beta(n)$ for different simulated QPUs: perfect QPU with $p = 1$ (blue), $p = 2$ (cyan), and noisy QPU with a depolarizing noise model (see text), with $p = 1$, all-to-call connectivity (solid red lines), and grid connectivity (dashed red lines). The dash-dotted black line shows the 20% threshold above which the Q-score test is passed. The error bar is the standard error of the mean score over 100 graphs.

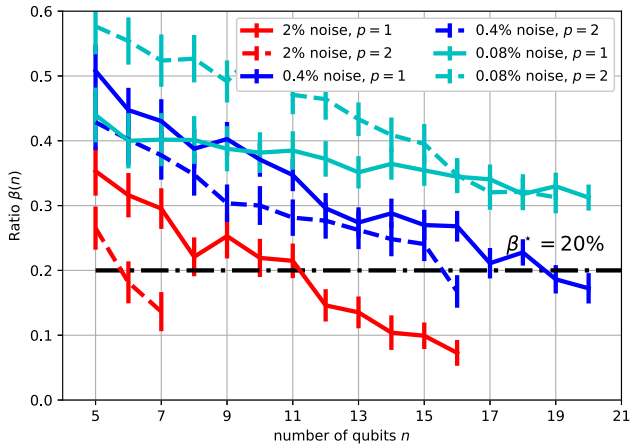

FIGURE. 2. Evolution of $\beta(n)$ for different levels of depolarizing noise and number of QAOA layers p, with grid connectivity: $p = 1$ (solid lines), $p = 2$ (dashed lines). The dash-dotted black line shows the 20% threshold above which the Q-score test is passed. The error bar is the standard error of the mean score over 100 graphs.

This is what we observe in Fig. 2 except for noise levels that are very low compared to the noise levels reported for NISQ processors (this phenomenon was also observed in [2]). COBYLA is thus a sensible choice for current QPUs.

3) COMPUTING THE SCORE

For a given size n, we run a QAOA-MaxCut on 100 random graphs in $\mathcal{G}(n, p = \frac{1}{2})$, the distribution of Erdös–Renyi graphs obtained by taking an empty graph and connecting each pair of vertices with probability $\frac{1}{2}$. These graphs are relatively dense and constitute a standard class used for benchmarks. Given $C(n)$, the average of the energies (multiplied by -1) produced by QAOA over these 100 graphs, we compute the following ratio:

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

We say that the quantum processor passes the test for this size n if $\beta(n) > \beta^\star$. Here, the threshold $\beta^\star \in]0, 1[$ dictates how demanding is the test: a test with $\beta^\star = 0$ can be passed by a simple coin toss, while a test with $\beta^\star = 1$ can only be passed by an exact solver. Hence, $\beta^\star$ can be seen as fraction of performance between a naive randomized algorithm and an exact solver. In practice, the threshold $\beta^\star$ is arbitrarily set to 0.2. We take $\lambda = 0.178$ (see discussion below, Section III). We also fix the number of shots (repetitions) to be used to get the estimate of the QAOA energy for a given graph to 2048 per $(\boldsymbol{\beta}, \boldsymbol{\gamma})$.

The final Q-score is the largest n such that this test succeeds, i.e.,

$$n^\star \equiv \max\{n \in \mathbb{N}, \beta(n) > \beta^\star\}. \tag{4}$$

4) REMARKS

The choice of $\beta^\star$ is somewhat arbitrary. $\beta^\star$ was set so that a QAOA of depth $p = 1$ running on a perfect quantum processor will pass the test and will have an infinite Q-score. (As will be seen later see (Fig. 3), for $p = 1$, $\beta^{\mathrm{Q}}(n) \approx 40\%$ for a perfect QPU). In practice, the Q-score implementation we provide is parameterized by this $\beta^\star$. Moreover, it is usually not necessary to iteratively try each instance size until the test fails, since $\beta(n)$ is expected to be a monotonically decreasing function of n. This implies that one can employ a dichotomic search in order to find $n^\star$, the largest n such that $\beta(n^\star) > \beta^\star$. Our implementation supports both iterative evaluation and dichotomic search.

B. ILLUSTRATION: PERFECT AND NOISY SIMULATIONS

To illustrate the meaning of the Q-score, we simulated the behavior of QAOA-MaxCut on various QPUs using the Atos Quantum Learning Machine (QLM).

We started by running QAOA-MaxCut on a perfect (noiseless) QPU for two values of the number p of QAOA layers. As expected, we see, in Fig. 1, that the score increases with an increasing p due to an increased expressivity of the QAOA ansatz. We also observe that the ratio $\beta(n)$ achieved by this perfect QPU is roughly constant as n increases, with $\beta(n) \approx 40\%$ for $p = 1$, and $\beta(n) \approx 60\%$ for $p = 2$. This means that QAOA executed on a perfect quantum processor achieves scalings within 40 % (resp. 60%) of the optimal scaling $\lambda n^{3/2}$ (after subtraction of the leading $n^2/8$ term).

We compare this behavior to the scores obtained with simulations of noisy QPUs. We choose a simple depolarizing noise model with a level of noise that is consistent with today's NISQ processors. More specifically, we add depolarizing noise after each gate, with an average error rate of

$$\epsilon_2 = 2\%$$

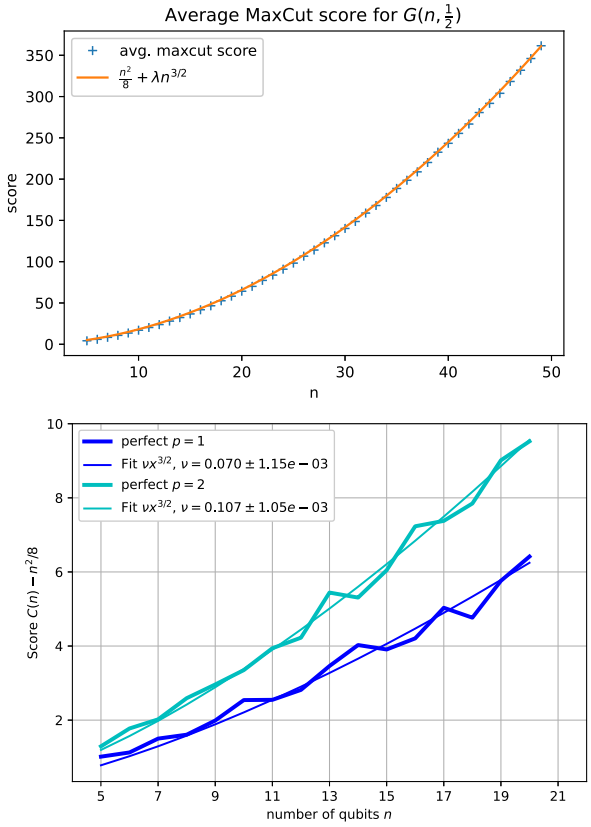

FIGURE. 3. Top: Scaling of the expected maximum cut size for Erdös–Renyi graphs of increasing size. Each data point (in blue) is computed by solving 200 MaxCut instances (using the AKMaxSAT solver [4]). In orange is a fit of shape $y = \frac{n^2}{8} + \lambda n^{\frac{3}{2}}$ with $\lambda \approx 0.178$ obtained by a standard least-squares method (r-value $> 1 - 10^{-3}$). Bottom: Fit of QAOA scores $C(n) - n^2/8$ to $\nu n^{3/2}$ for $p = 1$ (blue) and $p = 2$ (cyan). The obtained values for ν correspond to $\beta = \nu/\lambda = 40\%$ and 60%, respectively.

for two-qubit gates (in comparison, the two-qubit error rates reported for IBM Johannesburg [19], Google Sycamore [1, Fig. 2, Table II], Rigetti Aspen 7 [37], and ionQ [39], are, respectively, 0.2%, 0.62%, 4.8% and 2.5%), and

$$\epsilon_1 = 0.4\%$$

for one-qubit gates (this factor of 5 between the one- and two-qubit error rates is observed in typical superconducting and trapped-ion architectures, with reported one-qubit error rates of 0.041%, 0.16%, 0.77%, and 0.5% for the four aforementioned platforms). For the sake of simplicity, we assume perfect initialization and readout, and neglect noise during idling periods.

We observe that the ratio $\beta(n)$ achieved with a noisy QPU is, as expected, lower than with a perfect QPU. More importantly, it decreases with the problem size n (i.e., the number of qubits): larger problems require longer circuits and hence lead to an increased sensitivity to noise. Moreover, a limited connectivity (e.g., a grid connectivity) leads to a decreased ratio, since these connectivity constraints require the original QAOA circuit to be optimized to comply with the constraints. This optimization, carried out following a method described in [18] and [24] using one of Atos QLM's compilation plugins, leads to longer circuits and hence degraded performance in the presence of noise.

From these simulations, we can infer that the Q-score for a noisy QPU with a grid connectivity is $n^\star = 11$. For the noisy QPU with an all-to-all connectivity, we can infer that $n^\star = 21$. For perfect QPUs, QAOA achieves an infinite Q-score. In Fig. 2, we exemplify the tradeoff between increasing the expressivity of QAOA's ansatz (by increasing the number of layers p) and curbing the impact of noise. As expected, we observe that for the higher noise levels ($\epsilon_2 = 2\%$ and 0.4%), a larger p leads to a decreased Q-score (for $\epsilon_2 = 2\%$, the Q-score is 5 for $p = 2$ while it is 11 for $p = 1$) because the detrimental effect of noise outweighs the expressivity gain. At the lowest noise level ($\epsilon_2 = 0.08\%$), the situation is more constrasted: while, for small graph sizes, a larger p leads to a larger $\beta(n)$ (as is the case for the noiseless case, as shown in Fig. 1), for larger graph sizes (or numbers of qubits), noise penalizes longer circuits ($p = 2$) over shorter ones ($p = 1$), counterbalancing the increase in representativity due to a larger p.

Let us stress that this example also shows that beyond assessing the quality of the hardware for solving QAOA-MaxCut, the Q-score also assesses the performance of the software stack: for instance, a better compiler to optimize for connectivity constraints will lead to an increased $\beta(n)$ and hence to an increased Q-score. This is a major advantage of Q-score over lower-level metrics as improving *both* the software stack and the hardware is crucial in the overall advancement of the field, whether at the algorithmic level, at the compilation stage, or via noise-mitigation techniques. While the risk of some users deliberately fine-tuning their software to spoof the Q-score benchmark exists, we believe that it is outweighed by the overall benefit that the community will draw from a healthy competition to increase Q-score.

IV. DISCUSSION

In this section, we discuss the various choices made in this proposal. First of all, let us recall briefly what we need to achieve.

A. ALGORITHM CHOICE

We are not looking at finding a discerning metric for quantum supremacy. Our goal is simply to consider an application that is both representative of practical needs from the industry and challenging for current hardware platforms.

1) CHOICE OF QAOA-MAXCUT

Most, if not all, proposed algorithms compatible with the NISQ era, are variational algorithms. It thus seems natural, in an application-centric benchmark, to focus on this type of algorithms. Among all these propositions, we need one that fits a particular set of requirements. First, the algorithm should be scalable, in the sense that one should be able to rather smoothly increase the problem size in order to isolate

IEEE Transactions on Quantum Engineering

the precise threshold were the quantum coprocessor fails. Combinatorial optimization problems usually fit this criterion quite easily. Moreover, we also need the test to be efficiently computable. By averaging over a simple class of random instances, we can deduce asymptotic values for usually intractable quantities (see next section). This might be hard to do efficiently for other classes of problems. Hence, the QAOA seems to be a good candidate that fits these needs. We chose the MaxCut problem for the simple reason that it is both simple to implement and simple to analyze. For instance, it is possible to know the average number of entangling gates required in the Ansatz, even after compilation and optimization. This would not be the case were we to consider problems that involved clauses over more that two variables (mainly due to the variability of the literature in architecture-aware phase polynomial synthesis algorithms [24], [29], [38] or other less competitive SWAP-based routing techniques).

2) CHOICE OF THE CLASS $\mathcal{G}(n, \frac{1}{2})$

This class of graphs is quite standard in random graph literature and has a predictable behavior with regard to the MaxCut problem. Moreover, they constitute a class of dense graphs, with half of their possible edges present (on average). QAOA-MaxCut are often run using k-regular graphs for the simple reason that these graphs are very sparse. In fact their edge density decrease with their size. We suggest that most real-world applications will not have this property. Hence the choice of $\mathcal{G}(n, \frac{1}{2})$. One could relax a bit the test by picking a class $\mathcal{G}(n, f(n))$ with $f(n) = o(\frac{1}{n})$, that is a class of graph where edges are picked uniformly with a probability that decreases with n, but such that the average number of edges $f(n)\frac{n^2}{2}$ still grows faster that n. Another potential choice would be to consider bipartite graphs. These graphs can be perfectly cut and thus hard instances for QAOA. They are however trivial to deal with classically, and as such, do not constitute an interesting benchmark target.

B. TEST DEFINITION AND APPROXIMATION RATIO

In this section, we detail the reasoning behind the definition of the score [see (3)] and the corresponding success criterion.

1) USUAL APPROXIMATION RATIO AND ITS LOWER BOUND

We recall that the algorithm is run on Erdös-Renyi graphs G of fixed size n and with edge probability $\frac{1}{2}$, denoted $\mathcal{G}(n, \frac{1}{2})$. A standard way to evaluate the performance of an approximation heuristic such as QAOA is to consider the *approximation ratio* $\alpha(G) = \frac{C(G)}{C_{\max}(G)}$, where $C(G)$ is the score of the worst solution that can be produced by the heuristic and $C_{\max}(G)$ is the cost of the optimal solution for the given graph G. Since we are dealing with a randomized algorithm, this quantity translates into $\alpha^{\mathrm{Q}}(G) = \frac{\mathbb{E}^{\mathrm{Q}}[C(G)]}{C_{\max}}$ where $\mathbb{E}^{\mathrm{Q}}[C]$ would be the expected score of a solution produced by QAOA. [With an infinite number of shots, $\mathbb{E}^{\mathrm{Q}}[C(G)] = -\langle\Psi(\boldsymbol{\gamma}, \boldsymbol{\beta})|H_G|\Psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle$, with $|\Psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = U(\boldsymbol{\gamma}, \boldsymbol{\beta})|0\rangle^{\otimes n}$, see (1) and (2).]

Since we are interested in a typical behavior over a class of random graphs, we want to average this quantity, giving us an expected approximation ratio over instances of a given size

$$\overline{\alpha}^{\mathrm{Q}}(n) = \mathbb{E}_{G\sim\mathcal{G}(n,\frac{1}{2})}\left[\alpha^{\mathrm{Q}}(G)\right].$$

The behavior of this quantity is hard to derive, but it is easy to derive the behavior of the closely related quantity

$$\alpha^{\mathrm{Q}}(n) \equiv \frac{\mathbb{E}_{G\sim\mathcal{G}(n,\frac{1}{2})}\left[\mathbb{E}^{\mathrm{Q}}[C(G)]\right]}{\mathbb{E}_{G\sim\mathcal{G}(n,\frac{1}{2})}[C_{\max}(G)]} \equiv \frac{C^{\mathrm{Q}}(n)}{C_{\max}(n)}.$$

$\alpha^{\mathrm{Q}}(n)$ can be seen as a first-order approximation to $\overline{\alpha}^{\mathrm{Q}}(n)$. Since QAOA produces score distributions that are *at least* as good as straightforward random sampling, we get

$$\alpha^{\mathrm{Q}}(n) \geq \frac{C^{\mathrm{R}}(n)}{C_{\max}(n)}. \tag{5}$$

We now turn to the behavior of $C^{\mathrm{R}}(n)$ and $C_{\max}(n)$. Erdös–Renyi graphs of $\mathcal{G}(n, \frac{1}{2})$ have, on average, $\frac{n^2}{4}$ edges. On average over the complete family, their cuts have an expected cost

$$C^{\mathrm{R}}(n) \equiv \mathbb{E}_{G\sim\mathcal{G}(n,\frac{1}{2})}\left[\mathbb{E}^{\mathrm{R}}[C(G)]\right] = \frac{\mathbb{E}[|E|]}{2} = \frac{n^2}{8}.$$

Recent results [13], [16] show that their typical maximum cut size grows as

$$C_{\max}(n) \equiv \mathbb{E}_{G\sim\mathcal{G}(n,\frac{1}{2})}(C_{\max}(G)) = \frac{n^2}{8} + \lambda n^{\frac{3}{2}} + o(n^{3/2}) \tag{6}$$

with $\lambda \geq \frac{1}{2\sqrt{\pi}} \approx 0.159$. In practice, a numerical fit in the range $n \in [5, 40]$ yields a value of $\lambda \approx 0.178$ (see Fig. 3). Plugging these results into (5), we obtain

$$\alpha^{\mathrm{Q}}(n) \geq \frac{\frac{n^2}{8}}{\frac{n^2}{8} + \lambda n^{\frac{3}{2}}} \tag{7}$$

which approaches 1 when n diverges.

2) IMPROVED APPROXIMATION RATIO

This lower bound suggests that $\alpha^{\mathrm{Q}}(n)$ is not the appropriate quantity to consider to assess the quality of a heuristic for MaxCut on this class of graphs. Because the expected approximation ratio of random sampling grows with n, requiring a quantum processor to achieve a fixed approximation ratio is not an interesting test (for this class of graphs): this ratio will get easier and easier to reach as n grows. For instance, the previous inequality tells us that over random graphs of size 1500, random sampling will produce cuts with an average score that is 99.5% of the average score of the maximal cuts. This means that the most ineffective quantum processor, as long as it has 1500 qubits, will achieve at least the same ratio of expected cost. This phenomenon was for instance observed in [10], where both random and quantum approaches seemed to behave increasingly well for larger instances. This behavior is not a particularity of the $\mathcal{G}(n, \frac{1}{2})$

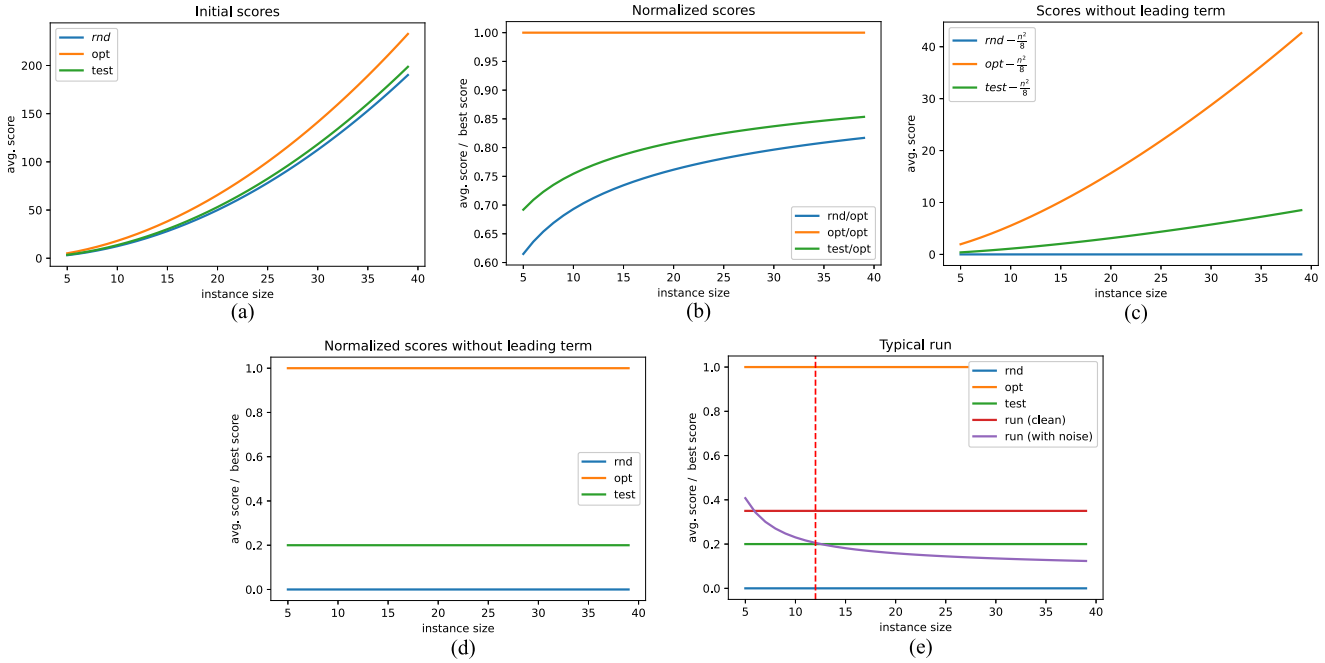

FIGURE. 4. **(a) Typical scaling of the expected costs $C(n)$ for three cases: expected maximum cut size $C_{\max}(n)$ (orange), expected random cut size $C^{\mathrm{R}}(n)$ (blue), and cost corresponding to our threshold $\beta^\star = 20\%$ (green). (b) Scaling of the average expectation ratios $\alpha(n)$ (namely cost normalized by the expected maximum cut size $C_{\max}(n)$). (c) Scaling of the cost with the leading $n^2/8$ term subtracted. (d) Scaling of the improved expectation ratios $\beta(n)$ (namely with the leading term subtracted and normalized by the maximum cut scaling). (e) Evolution of $\beta(n)$ for a typical Q-score run: in red, the scaling of a QAOA running on a perfect quantum processor. In purple, the scaling of a QAOA running on a imperfect processor. In this last setting, the dashed red line will give the returned Q-score.**

class. In fact, this result holds for any class of random graphs such that edges are picked uniformly at random [13], [16] and such that the number of edges grows faster than $O(n)$. If the number of edges is a $O(n)$, then the standard average approximation ratio definition will be upper bounded by a constant that can be analytically derived. This is for instance the case for k-regular graphs (see Section III-D).

In order to avoid this issue, we consider instead the same quantities after subtracting the leading $\frac{n^2}{8}$ term

$$\beta(n) \equiv \frac{C(n) - \frac{n^2}{8}}{C_{\max}(n) - \frac{n^2}{8}} = \frac{C(n) - \frac{n^2}{8}}{\lambda n^{3/2}}. \tag{8}$$

We use this definition to specify the conditions to pass the Q-score: we require the quantum algorithm achieve a ratio that exceeds a constant value $\beta^\star \in]0, 1[$:

$$\beta^{\mathrm{Q}}(n) \geq \beta^\star. \tag{9}$$

Based on numerical simulations with NISQ-compatible noise levels (see Section II-B above), we fix $\beta^\star$ to $\beta^\star = 20\%$.

This requirement implies that the quantum heuristic must fulfill satisfactory scalability properties: indeed, achieving a ratio $\beta_n^{\mathrm{Q}} \geq \beta^\star$ implies that the quantity $C^{\mathrm{Q}}(n) - \frac{n^2}{8}$ grows at least as $\nu n^{3/2}$, with $\nu = \beta^\star\lambda$ and λ the scaling of the optimal solution [see (6)]. In other words, we require the scaling rate of the quantum heuristic to be at least within a fraction $\beta^\star = 20\%$ of the scaling of the optimal solution.

For instance, random sampling, which always produces a vanishing ratio $\beta^{\mathrm{R}}(n) = 0$, cannot fulfill the Q-score for any $\beta^\star > 0$. Conversely, requiring $\beta^\star = 100\%$ would mean requiring to achieve the optimal solution.

Fig. 4 gives a qualitative graphical summary of the different quantities discussed here.

3) REMARK

In [2], the authors use a similar definition for the approximation ratio, with different motivations. Their definition comes from the fact that they are interested in minimizing the energy of Ising Hamiltonians of shape $H'_G = \sum_{i,j\in E} \sigma_z^{(i)}\sigma_z^{(j)}$, i.e., without the constant energy offset of $\frac{|E|}{2}$ [compare to H_G in (2)]. The spectra of these Hamiltonians do not coincide with the usual cut size functions, but exhibit the same feature as the cost metric described in (8).

4) CONTINUOUS SCORE

Even though the proposed protocol outputs a single number, it is possible to extract far more information from a run of Q-score. For instance, a good benchmark metric would be to track the largest ν constant accessible for each problem size n. This scaling would allow a manufacturer to track the performances of its processors when scaling up the number of qubits/problem size. Moreover, this ν factor provides a comparison tool with various behaviors whether it is random

IEEE Transactions on Quantum Engineering

sampling ($\nu = 0$), perfect solving ($\nu = \lambda \approx 0.178$), perfect QAOA ($\nu \approx 0.07$ for $p = 1$, ≈ 0.107 for $p = 2$, see Fig. 3).

C. NOTE ON THE EXPERIMENTAL PARAMETERS

When defining the protocol, we set the value of the number of shots as well as the optimization procedure. While these choices are somewhat arbitrary, they arguably do not significantly impact the final value of the Q-score.

The number of shots (2048) is representative of the typical numbers of shots used on experimental processors. It gives reasonable statistical errors on the estimate of the cost function. Given the wide range of clock speeds of existing QPUs, this fixed number of shots does not yield equivalent run times. These clock speeds, or equivalently, the time-to-solution, could easily be taken into account by Q-score by setting a maximum time budget to compute $\beta^{\mathrm{Q}}(n)$ for a given graph size n. For the time being, we did not specify such a time limit, since in the current state of QPUs, increasing the processor's fidelities probably comes before increasing their speed. But Q-score should be reported together with the absolute time required to compute $\beta(n^\star)$.

As for the choice of the classical optimization procedure: here, we took the COBYLA optimizer that is very commonplace and usually provides a good tradeoff between convergence speed and quality of the attained minimum [20], [21] (although, as a local optimizer, it may have issues in the presence of flat surfaces—issues which we did not observe for the parameter ranges we investigated). Beyond this particular choice, we stress that the same optimizer should be used across all platforms in order to ensure a consistency in the obtained scores.

D. CHANGING THE GRAPH CLASS

In this protocol, and the discussion of Section III-B, we focused on a particular class of random graphs, namely Erdös–Renyi random graphs with edge probability $\frac{1}{2}$. These graphs have the nice property of being dense, and thus any (positive) result for this class of graph has a good chance to transpose to any application. However, running QAOA-MaxCut for these graphs can be quite demanding, since a typical circuit would have around $k\frac{n^2}{2}$ CNOT gates for an Ansatz of depth k over a graph of size n. This quadratic scaling can be quite demanding for a real hardware platform.

In this section, we show how a similar score/test can be derived for other classes of random graphs that would define less demanding tests, as in running circuits with a lower entangling gate count. All the results presented below can be derived from the scaling proven in [13]. In this article, the authors state that the scaling of the average maximum cut size for random graphs with γn edges picked uniformly can be expressed as

$$C_{\max}(n) = \frac{n\gamma}{2} + P_\star\sqrt{\frac{\gamma}{2}}n + o(n\sqrt{\gamma}) \tag{10}$$

where $P_\star \leq \sqrt{2/\pi}$. Numerical estimate of this constant gives $P_\star = 0.76321 \pm 0.00003$. This result gives us quite naturally the difference in scaling between the cut sizes produced by random sampling, $\frac{n\gamma}{2}$, and the cut sizes produced by an exact solver.

We now detail this scaling for two classes of graphs: generic $\mathcal{G}(n, p)$ random graphs and random k-regular graphs.

1) $\mathcal{G}(n, p)$ GRAPHS

We can run the same calculation as the one for $\mathcal{G}(n, \frac{1}{2})$ for any edge probability p. In this setting, we have $\gamma = \frac{p\bar{n}}{2}$ and similarly to the $\mathcal{G}(n, \frac{1}{2})$ case, the average maximum cut size grows as

$$C_{\max}(n) = p\frac{n^2}{4} + \lambda_p n^{\frac{3}{2}}$$

for some constant λ_p. Analytically, we expect $\lambda_p = \sqrt{2p}\lambda_{\frac{1}{2}}$, with $\lambda_{\frac{1}{2}}$ the scaling of the $p = \frac{1}{2}$ case. The direct consequence is that we can use a similar test as for $p = \frac{1}{2}$ and pose

$$\beta(n) = \frac{C(n) - p\frac{n^2}{4}}{C_{\max}(n) - p\frac{n^2}{4}} = \frac{C(n) - p\frac{n^2}{4}}{\lambda_p n^{3/2}}$$

where λ_p can be either fitted numerically or taken as $\lambda_{\frac{1}{2}}\sqrt{2p}$. Overall, this boils down to comparing the QAOA performance $C(n) - p\frac{n^2}{4}$ against a $n^{3/2}$ scaling.

Here, we derived an expression for $\beta(n)$ where p is constant, but the derivation hold for any size dependent probability $p = f(n)$. Hence, we can define the same benchmark with increasingly dense (and thus difficult to implement) instances.

2) k-REGULAR GRAPHS

Regular graphs have the convenient property of being very sparse, with a number of edges of $\frac{kn}{2}$ for a k-regular graph of size n. For this class of graph the scaling of the average maximum cut is in fact proven, and not only known within an interval. Applying (10) with $\gamma = k/2$ gives us

$$C_{\max}(n) = \frac{nk}{4} + \frac{P_\star\sqrt{k}}{2}n + o(n\sqrt{k})$$

hence a natural choice of β is

$$\beta(n) = \frac{C(n) - \frac{nk}{4}}{C_{\max}(n) - \frac{nk}{4}} = \frac{C(n) - \frac{nk}{4}}{\lambda n}$$

for some constant $\lambda = P_\star\sqrt{k}/2$. Once again, we can either use the analytical value for λ or fit it numerically for small instances. That is, if we fix k, we are looking to compare the QAOA performances over k-regular graphs $C(n) - \frac{nk}{4}$, to a linear scaling in n.

Listing 1: Python Script to Run Q-Score.

```
from qat.qscore.benchmark import QScore
from qat.plugins import ScipyMinimizePlugin
from qat.qpus import get_default_qpu

# Our QPU is composed of:
# - a variational optimizer plugin
# - a QLM/myQLM default qpu (either LinAlg or
    pyLinalg)

QPU = ScipyMinimizePlugin(
    method="COBYLA",
    tol=1e-4,
    options={"maxiter": 300}
) | get_default_qpu()

benchmark = QScore(
    QPU,
    size_limit=20,  # limiting the instace sizes
    to 20
    depth=1,        # using an Ansatz depth of 1
    output="perfect.csv",
    rawdata="perfect.raw"
)
benchmark.run()
```

Listing 2: Python Script to Make Your Own QPU Compatible With myQLM.

```
from qat.core.qpu import QPUHandler
from qat.core import Result


class MyQPU(QPUHandler):
    def submit_job(self, job):
        # Evaluate the job using your QPU
        # A job constains:
        # a circuit:
        circuit = job.circuit
        # possibly an observable
        observable = job.observable
        # or a list of qubits to sample:
        qubits = job.qubits

        # Results are returned in a 'Result'
    object
        return result
```

V. RUNNING Q-SCORE YOURSELF: AN OPEN-SOURCE REPOSITORY

We provide a Python package, `qscore` (https://www.github.com/myQLM/qscore), to compute the Q-score for any QPU that has been interfaced with the open-source `myqlm` library. Once the `qscore` package is installed, Listing 1 shows the typical script that needs to be run.

Here, the QPU is a perfect circuit simulator provided by myQLM. In order to use a true hardware QPU, one simply needs to interface one's QPU with the myQLM API. This thin layer typically looks as shown in Listing 2.

VI. CONCLUSION

In this article, we have introduced the Atos Q-score, an application-centric, hardware-agnostic, and scalable metric that measures the ability of a full quantum stack—hardware and software—to solve a prototypical combinatorial optimization problem, MaxCut, using the QAOA, a widespread variational quantum heuristic compatible with NISQ coprocessors. Instead of focusing on how well the basic building blocks of a quantum processor work, like most existing metrics, Q-score provides information as to the capacity of the processor to solve an actual problem. It does so without favoring any hardware technology or software paradigm, and will be applicable to very large problems due to its scalability.

Like the classical LINPACK benchmark, the Q-score focuses on a given problem as a proxy for most other hard computational problems. Here, MaxCut was chosen as a representative hard problem, because it appears to be quite simple and universal. In the search for the "killer application" for quantum co-processors, other more relevant problems may appear and supersede MaxCut, but the same strategy as the one we describe in this note will likely be applicable. Likewise, the choices of optimizer (COBYLA) and other parameters (number of shots, number of graphs, etc.) we set the value of for the sake of standardization have a degree of arbitrariness. In a similar vein, the current protocol is geared to digital quantum coprocessors. An extension to analog processors is rather straightforward, and will be the topic of future work.

All these variations on the protocol proposed in this note should not influence the overall outcome of the procedure, and thus, the usefulness of the benchmark.

ACKNOWLEDGMENT

The authors would like to thank the members of the Atos Quantum Advisory Board, A, Aspect, D. DiVincenzo, A. Ekert, D. Estève, and S. Haroche. The computations have been performed on the Atos Quantum Learning Machine.

REFERENCES

[1] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019, doi: 10.1038/s41586-019-1666-5.

[2] M. P. Harrigan *et al.*, "Quantum approximate optimization of non-planar graph problems on a planar superconducting processor," *Nat. Phys.*, vol. 17, no. 3, pp. 332–336, 2020, doi: 10.1038/s41567-020-01105-y.

[3] S. Aaronson and L. Chen, *Complexity-Theoretic Foundations of Quantum Supremacy Experiments*. Dec. 2016, https://arxiv.org/abs/1612.05903

[4] T. Alsinet, F. Many, and J. Planes, "Improved exact solvers for weighted max-sat," in *Theory and Applications of Satisfiability Testing* F. Bacchus and T. Walsh, Eds. Berlin, Germany: Springer, 2005, pp. 371–377, doi: 10.1007/11499107_27.

[5] R. Blume-Kohout, J. K. Gamble, E. Nielsen, J. Mizrahi, J. D Sterk, and P. Maunz, "Robust, Self-Consistent, Closed-Form Tomography of Quantum Logic Gates on a Trapped Ion Qubit. 87185(Ml)," Oct. 2013, arXiv:1310.4492.

[6] R. Blume-Kohout and K. C. Young, "A Volumetric Framework for Quantum Computer Benchmarks," *Quantum*, vol. 4, p. 362, 2019, doi: 10.22331/q-2020-11-15-362.

[7] A. W Cross, L. S Bishop, S. Sheldon, P. D Nation, and J. M Gambetta, "Validating quantum computers using randomized model circuits,"*Phys. Rev. A*, vol. 100, no. 3, Sep. 2019, Art. no. 032328, doi: 10.1103/PhysRevA.100.032328.

[8] P.-L. Dallaire-Demers, M. Stęchły, J. F. Gonthier, N. T. Bashige, J. Romero, and Y. Cao, "An Application Benchmark for Fermionic Quantum Simulations," 2020, arXiv:2003.01862.

[9] O. Di Matteo, J. Gamble, C. Granade, K. Rudinger, and N. Wiebe, "Operational, gauge-free quantum tomography," *Quantum*, vol. 4, Nov. 2020, Art. no. 364, doi: 10.22331/q-2020-11-17-364.

[10] C. Dalyac *et al.*, "Qualifying quantum approaches for hard industrial optimization problems," *Case Study Field Smart-Charging Elect. Veh.*, 2020, doi: 10.1140/epjqt/s40507-021-00100-3.

[11] Y. Dong and L. Lin, "Random circuit block-encoded matrix and a proposal of quantum LINPACK benchmark," *Phys. Rev. A*, vol. 103, no. 6, 2021, Art. no. 062412, doi: 10.1103/PhysRevA.103.062412.

[12] J. J. Dongarra, P. Luszczek, and A. Petite, "The LINPACK benchmark: Past, present and future," *Concurrency Comput. Pract. Experience*, vol. 15, no. 9, pp. 803–820, 2003, doi: 10.1002/cpe.728.

[13] A. Dembo, A. Montanari, and S. Sen, "Extremal cuts of sparse random graphs," *Ann. Probab.*, vol. 45, no. 2, pp. 1190–1217, 2017, doi: 10.1214/15-AOP1084.

[14] A. Erhard *et al.*, "Characterizing large-scale quantum computers via cycle benchmarking," *Nat. Commun.*, vol. 10, no. 1, pp. 1–7, 2019, doi: 10.1038/s41467-019-13068-7.

[15] E. Farhi, J. Goldstone, and S. Gutmann, "A Quantum Approximate Optimization Algorithm," Nov. 2014, arXiv:1411.4028.

[16] D. Gamarnik and Q. Li, "On the max-cut of sparse random graphs," *Random Struct. Alg.*, vol. 52, no. 2, pp. 219–262, 2018, , doi: 10.1002/rsa.20738.

[17] D. Greenbaum, "Introduction to Quantum Gate Set Tomography," 2015, arXiv:1509.02921.

[18] Y. Hirata, M. Nakanishi, S. Yamashita, and Y. Nakashima, "An efficient method to convert arbitrary quantum circuits to ones on a linear nearest neighbor architecture," in *Proc. IEEE 3rd Int. Conf. Quantum, Nano Micro Technol.*, 2009, pp. 26–33, doi: 10.1109/ICQNM.2009.25.

[19] "IBM quantum experience website," Accessed: Mar. 5, 2020. [Online]. Available: https://quantum-computing.ibm.com/

[20] S. Khairy, R. Shaydulin, L. Cincio, Y. Alexeev, and P. Balaprakash, "Learning to optimize variational quantum circuits to solve combinatorial problems," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 3, Apr. 2020, pp. 2367–2375, doi: 10.1609/aaai.v34i03.5616.

[21] W. Lavrijsen, A. Tudor, J. Muller, C. Iancu, and W. deJong, "Classical optimizers for noisy intermediate-scale quantum devices," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, 2020, pp. 267–277, doi: 10.1109/QCE49297.2020.00041.

[22] E. H.Lieb and F. Y. Wu, "Absence of mott transition in an exact solution of the short-range, one-band model in one dimension," *Phys. Rev. Lett.*, vol. 20, no. 25, pp. 1445–1448, 1968, doi: 10.1103/PhysRevLett.20.1445.

[23] C. David, Andrew W. McKay, Christopher J. Cross, Wood, and Jay M. Gambetta, "Correlated Randomized Benchmarking. Arxiv," 2020, arXiv:2003.02354.

[24] S. Martiel and Timothée Goubault de Brugiére, "Architecture Aware Compilation of Quantum Circuits Via Lazy Synthesis," 2020, arXiv:2012.09663.

[25] E. Magesan, Jay M. Gambetta, and J. Emerson, "Characterizing quantum gates via randomized benchmarking," *Phys. Rev. A*, vol. 85, no. 4, Apr. 2012, Art. no. 042311, doi: 10.1103/PhysRevA.85.042311.

[26] S. T. Merkel *et al.*, "Self-consistent quantum process tomography," *Phys. Rev. A*, vol. 87, no. 6,Jun. 2013, Art. no. 062119, doi: 10.1103/PhysRevA.87.062119.

[27] A. J. McCaskey *et al.*, "Quantum chemistry as a benchmark for near-term quantum computers," *NPJ Quantum Inf.*, vol. 5, no. 1, pp. 1–10, 2019, doi: 10.1038/s41534-019-0209-0.

[28] D. Mills, S. Sivarajah, Travis L. Scholten, and R. Duncan, "Application-Motivated, Holistic Benchmarking of a Full Quantum Computing Stack,", 2020, arXiv:2006.01273.

[29] B. Nash, V. Gheorghiu, and M. Mosca, "Quantum circuit optimizations for NISQ architectures," *Quantum Sci. Technol.*, vol. 5, no. 2,, Mar. 2020, Art. no. 025010, doi: 10.1088/2058-9565/ab79b1.

[30] C. Neill *et al.*, "A blueprint for demonstrating quantum supremacy with superconducting qubits," *Science*, vol. 360, no. 6385, pp. 195–199, 2018, doi: 10.1126/science.aao4309.

[31] T. J. Proctor, A. Carignan-Dugas, K. Rudinger, E. Nielsen, R. Blume-Kohout, and K. Young, "Direct randomized benchmarking for multiqubit devices," *Phys. Rev. Lett.*, vol. 123, no. 3, pp. 1–13, 2019, doi: 10.1103/PhysRevLett.123.030503.

[32] A. Peruzzo *et al.*, "A variational eigenvalue solver on a photonic quantum processor," *Nat. Commun.*, vol. 5, no. 1, Sep. 2014, Art. no. 4213, doi: 10.1038/ncomms5213.

[33] M. J. D. Powell, "A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation," *Advances in Optimization and Numerical Analysis*, S. Gomez and J. P. Hennart, Eds. Dordrecht, Germany: Springer, pp. 51–67. Netherlands, Dordrecht: Springer, 1994, doi: 10.1007/978-94-015-8330-5_4.

[34] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, Jan. 2018, doi: 10.22331/q-2018-08-06-79.

[35] T. Proctor, K. Rudinger, K. Young, M. Sarovar, and R. Blume-Kohout, "What randomized benchmarking actually measures," *Phys. Rev. Lett.*, vol. 119, no. 13, Sep. 2017, Art. no. 130502, doi: 10.1103/PhysRevLett.119.130502.

[36] T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, "Measuring the Capabilities of Quantum Computers," pp. 1–30, Aug. 2020, arXiv:2008.11294.

[37] "Rigetti computing website," Accessed: Nov. 23, 2020. [Online]. Available: https://www.rigetti.com/what

[38] A. M. van de Griend and R. Duncan, "Architecture-Aware Synthesis of Phase Polynomials for NISQ Devices," 2020, arXiv:2004.06052.

[39] K. Wright *et al.*, "Benchmarking an 11-qubit quantum computer," *Nature Commun.*, vol. 10, no. 1, pp. 1–6, 2019, doi: 10.1038/s41467-019-13534-2.

[40] H.-S. Zhong *et al.*, "Quantum computational advantage using photons," *Science*, vol. 370, no. 6523, pp. 1460–1463, Dec. 2020, doi: 10.1126/science.abe8770.