Limits of quantum run-time advantage

J. Tuziemski,[1,2,*] J. Pawłowski,[3,2] P. Tarasiuk,[2] Ł. Pawela,[4,2] and B. Gardas[4]

[1] *Institute of Informatics, National Quantum Information Centre, Faculty of Mathematics, Physics and Informatics, University of Gdańsk, Wita Stwosza 57, 80-308 Gdańsk, Poland*

[2] *Quantumz.io Sp. z o.o., Puławska 12/3, 02-566 Warsaw, Poland*

[3] *Institute of Theoretical Physics, Faculty of Fundamental Problems of Technology, Wrocław University of Science and Technology, 50-370 Wrocław, Poland*

[4] *Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Bałtycka 5, 44-100 Gliwice, Poland*

(Received 9 October 2025; revised 9 February 2026; accepted 17 March 2026; published 29 April 2026)

A robust definition of quantum run time is essential for assessing the performance of quantum algorithms and claims of *quantum advantage*. While for most classical hardware the total run time is well approximated by computation plus a weakly varying constant, on current quantum hardware a clean experimental separation between "pure computation" and "overhead" is often not justified. Consequently, conventional quantum run-time analyses that exclude substantial system-level overheads (e.g., readout, transpilation, thermalization) can lead to biased performance assessments. In this work we introduce experimentally grounded, end-to-end definitions of quantum run-time for both digital and analog quantum computers, together with a methodology for selecting strong classical baselines for quantum-classical run-time comparisons. Within this framework, we evaluate recent claims of *quantum advantage* in annealing- and gate-based algorithms. We examine three representative case studies. First, we revisit quantum annealing for approximate quadratic unconstrained binary optimization problems [Phys. Rev. Lett. **134**, 160601 (2025)], which employs a well-motivated time-to-epsilon metric but effectively uses annealing time as a proxy for run time. Second, we analyze a restricted implementation of Simon's problem [Phys. Rev. X **15**, 021082 (2025)], where the favorable scaling in oracle calls is undisputed; however, we show that the estimated wall-clock run time of the quantum demonstration is approximately two orders of magnitude slower than a tuned classical baseline at the tested sizes. Finally, we find that the recently reported run-time advantage of the bias-field digitized counterdiabatic quantum optimization hybrid algorithm (arXiv:2505.08663) is not observed under more comprehensive benchmarking. Therefore, on current NISQ hardware, run-time-based quantum advantage has not yet been demonstrated under experimentally grounded performance metrics, and credible claims require careful time accounting, appropriate performance measures, and properly chosen classical reference implementations, as discussed in this work.

DOI: 10.1103/gpsf-pn1x

I. INTRODUCTION

The main motivation for the development of quantum computers, as noted in pioneering works on the subject [1,2], is the possibility of efficiently solving problems that are currently intractable for classical computation. Initially a theoretical field, quantum computing has seen rapid technological progress, and major roadmaps now predict error-corrected machines by around 2030 [3–5]. Quantum annealing technology is also advancing [6]. Despite this progress, experimentally demonstrating quantum advantage for both paradigms remains a central challenge, driving ongoing research [7].

Quantum advantage can be defined in various ways, including advantages in computation or metrology [8]. One prominent approach relies on complexity-theoretic arguments that establish more favorable scaling than classical algorithms, as exemplified by Simon's algorithm [9], which exhibits an exponential separation in query complexity, or Shor's algorithm [10], based on the presumed hardness of factoring [11]. While generic speedups for NP-hard problems are widely believed to be unlikely, restricted instances may admit improvements [12], motivating studies of heuristic quantum algorithms [13]. Such approaches may provide either complexity-theoretic guarantees for specific problem classes or empirical performance improvements on suitable hardware. Other notions of advantage, such as energy efficiency, have also been discussed [14].

As quantum devices continue to improve, increasing attention has turned toward experimental verification of quantum advantage claims [15]. Early demonstrations

*Contact author: jan.tuziemski@ug.edu.pl

based on random circuit sampling [16] were subsequently challenged by classical simulation techniques [17,18], although more recent results appear to lie beyond the reach of known classical methods [19]. Additional experimental claims include favorable time-to-solution scaling for quadratic unconstrained binary optimization (QUBO) on D-Wave annealers [20], query-complexity advantages in restricted implementations of Simon's problem [21], and reported run-time benefits for hybrid classical-quantum algorithms [22]. These studies reflect the growing technological maturity of quantum hardware and are therefore of considerable interest. At the same time, they raise an important question: do quantum performance assessment methodologies currently in use—particularly those based on scaling arguments or proxy metrics—result in robust quantum advantage with actual end-to-end run-time improvements critical for practical adoption?

Specifically, in Sec. II, we discuss challenges associated with defining total run time in annealing- and gate-based demonstrations, and propose consistent and operationally meaningful quantum run-time definitions for both quantum computing paradigms. Using those definitions, we revisit two recent results regarding quantum advantage for analog and digital quantum algorithms.

Firstly, we address Ref. [20], which reported favorable scaling of a quantum-annealing-based approximate QUBO solver relative to a classical reference algorithm [parallel tempering with isoenergetic cluster moves (PT-ICM) [23]], and show that, when a rigorous end-to-end run-time definition is employed, the reported advantage is no longer observed.

Next, we analyze the results of Ref. [22], where a digitized counterdiabatic quantum optimization algorithm—a gate-based approach—was reported to achieve a run-time advantage over classical simulated annealing (SA) [24] and CPLEX [25]. We find that also in this case the use of the comprehensive run-time definition does not allow one to conclude that the algorithm considered provides a run-time advantage.

In Sec. III, we address the reported query-complexity advantage for a restricted implementation of Simon's problem [21] on noisy gate-based IBM quantum processors. We examine whether reduced oracle-call complexity leads to shorter wall-clock run times and find that, while the classical algorithm exhibits exponentially worse scaling in oracle calls, its actual run time in the regime explored in Ref. [21] remains approximately two orders of magnitude shorter than that of the quantum implementation.

Finally, in Sec. IV, we discuss the importance of selecting appropriate classical reference algorithms when evaluating run-time performance. Focusing on run time as the primary metric, we outline key criteria for classical baseline selection and revisit the choices made in Ref. [22]. We show that adopting an alternative, well-motivated classical reference is sufficient to remove the reported run-time advantage. This situation parallels recent discussions in Ref. [26], where the use of stronger classical baselines was shown to alter conclusions regarding quantum advantage in discrete approximate optimization problems (cf. Ref. [20]) under operationally meaningful conditions. For completeness, we also evaluate the performance of a D-Wave quantum annealer on the instances studied in Ref. [22] and find that, in this setting, adiabatic quantum computing does not yield a run-time advantage. Instead, these instances prove particularly challenging for the annealer, as the required embedding significantly degrades solution quality.

II. QUANTUM ADVANTAGE IN ALGORITHM RUN TIME

In this section, we address the problem of defining run time in a manner suitable for quantum computation and investigate how sensitive claims of run-time advantage are to changes in this definition. Our focus is on approximate QUBO, where we assume that the problem has been mapped to an instance of the Ising model, whose ground state encodes the solution [27]. The Ising model, originating from statistical physics, describes the energy associated with a configuration of discrete variables (spins) $s \in \{-1, 1\}^N$,

$$H(s) = \sum_{i<j} J_{ij} s_i s_j + \sum_i h_i s_i, \tag{1}$$

where the J_{ij} are coupling strengths between spins and h_i are local magnetic fields [28].

In the case of approximate optimization, a standard figure of merit is the *time-to-epsilon*, TTε (sometimes used interchangeably with the *time-to-approximation ratio*, TT$_\mathcal{R}$ [29]), which quantifies the time required to obtain a solution whose energy lies within a fraction ε of the ground-state energy. This quantity is defined as

$$\mathrm{TT}\varepsilon \doteq t_f \cdot \frac{\log(1-0.99)}{\log(1-p_{E\le E_0+\varepsilon|E_0|})}, \tag{2}$$

where t_f denotes the time taken to generate a single solution and $p_{E\le E_0+\varepsilon|E_0|}$ is the probability of obtaining a solution with energy E within an ε optimality gap of the true ground-state energy E_0, when this value is known, or relative to a reference energy otherwise (e.g., the best energy found by a reference solver). In most practical settings, the probability distribution $p_{E\le E_0+\varepsilon|E_0|}$ is unknown and must be estimated. This typically involves nontrivial optimization of solver parameters, followed by multiple independent runs using the optimized settings, from which the success probability is inferred. Its functional dependence then determines the expected number of repetitions required to obtain a solution within the desired optimality threshold.

Investigations of TTε for specific classes of optimization problems, as well as studies of its scaling with problem size, are commonly used as comparison tools between different heuristic algorithms, including quantum approaches [20]. However, it follows directly from the above definition that a careful and consistent definition of the run time t_f is essential for fair and unbiased comparisons.

In most cases, defining and measuring run time for classical algorithms is relatively straightforward, and a variety of mature profiling tools are available for this purpose [30–32]. In contrast, defining and measuring run time for quantum algorithms presents two key challenges. The first concerns the operational definition of quantum run time, namely which stages of the quantum computation pipeline should be included in the total run time. The second concerns the practical measurement of the durations of these stages in an experimental setting. In the following subsections, we address these issues separately for quantum annealing and gate-based quantum devices.

A. Run time for quantum annealing devices

Solution of an optimization problem on a quantum annealer consists of several stages. First, the optimization problem must be loaded onto the annealing device. This involves two steps: embedding, which maps a given instance of the Ising model onto the device topology; and programming, which configures the physical parameters of the device to implement the embedded problem. Subsequently, the quantum annealing protocol is executed. It is important to note that the annealing duration is an input parameter to the protocol and, under standard cloud access, cannot be independently measured by the user [33]. Due to the probabilistic nature of the annealing process, multiple runs are typically required to increase the probability of obtaining a high-quality solution. These runs are executed sequentially and are separated by two additional processes: readout, which retrieves information about the final state of each run; and thermalization, which restores the device to its initial state. Both processes contribute non-negligible time overheads.

The comprehensive total run-time definition of a quantum annealer consists of the sum of all annealing runs together with the associated readout, thermalization, and programming times. This definition is not universally adopted; for example, in Ref. [20], run time was defined solely in terms of the annealing time, with other contributions excluded. Our goal here is to examine how robust this choice of run-time definition is in the context of TTε, and to what extent the scaling is affected when additional stages of the annealing protocol are incorporated.

To this end, we repeated the demonstrations using the same problem instances as in Refs. [20,34]. When t_f is identified with the annealing time per sample, we reproduce the results reported in Ref. [20], confirming that the methodology was implemented consistently. We then examine how the scaling of TTε changes when t_f is defined as the total QPU access time, including programming, annealing, readout, and thermalization, as reported by the cloud interface [35]. To further validate these results, we also measure the total cloud-access time, which includes additional access-related overheads.

Under these definitions, the results indicate that the effective run time remains nearly constant over the range of problem sizes considered. An analysis of the scaling exponent supports this observation, as the uncertainties in the fitted exponents are too large to reliably distinguish them from zero. Examining the individual run-time components reveals that readout constitutes the dominant contribution to the total run time, of the order of 200 μs per annealing run. By comparison, the annealing time in these demonstrations varies between 0.5 μs and 27 μs, indicating that measurement of the final quantum state requires significantly more time than the quantum evolution used to prepare it. This disparity complicates the investigation of TTε scaling and provides context for the exclusion of readout time in Ref. [20]. However, measurement is an intrinsic component of quantum computation—both analog and digital—and the information encoded in a quantum state must ultimately be extracted through readout in order to obtain a solution to the computational problem.

For comparison, we evaluate the scaling behavior of the quantum annealing protocol against a classical simulated bifurcation machine (SBM [26,36,37]); see Appendix A. As a classical algorithm, the SBM allows direct measurement and decomposition of run time into components, including the pure computation time, performed on a GPU (t_f^{GPU}), and overhead time (t_f^{overhead}), dominated by data transfer between CPU and GPU and hyperparameter tuning, as described in Appendix A. We denote the total run time from an end-user perspective by $t_f^{\mathrm{tot}} = t_f^{\mathrm{GPU}} + t_f^{\mathrm{overhead}}$. As shown in Fig. 1, the impact of overhead on the scaling of $[\mathrm{TT}\varepsilon]_{\mathrm{Med}}$ is non-negligible, though substantially less pronounced than in the quantum annealing case.

These results indicate that the approximation

$$\text{run time} \approx \text{compute} + \text{weakly varying constant}$$

does not hold for current quantum annealers, whereas it remains reasonable for classical hardware such as GPUs. This difference helps explain why scaling behavior for classical algorithms is typically more robust with respect to the choice of run-time definition.

To fully address this issue, all stages of the quantum computation pipeline would need to be included in the run-time definition. On the corresponding timescales, however, the total run time becomes nearly constant for the problem sizes accessible in current demonstrations. Obtaining conclusive scaling results for present-day annealing devices would therefore require demonstrations with annealing

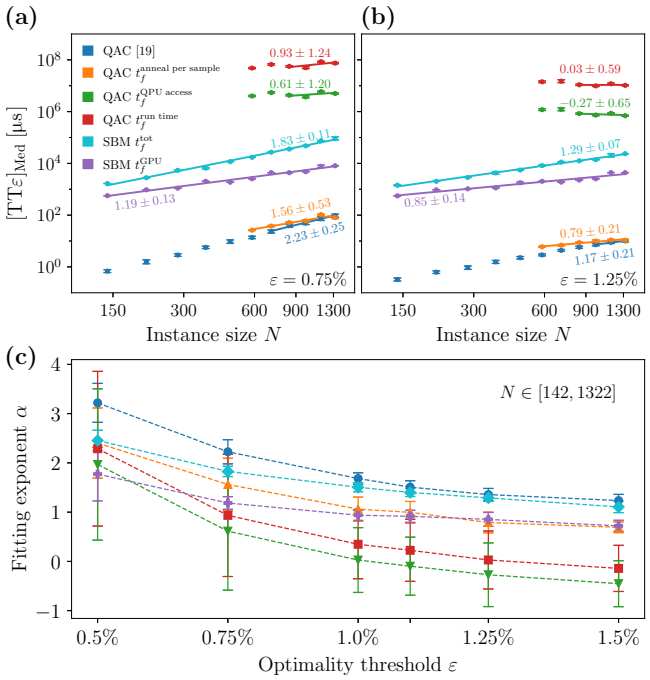

FIG. 1. Time-to-epsilon $[\text{TT}\varepsilon]_{\text{Med}}$ scaling with the size N of the Sidon-28 instances, for values of $\varepsilon = 0.75\%$ in (a), and $\varepsilon = 1.25\%$ in (b). Results for QAC (blue) solver are reproduced from Ref. [20], courtesy of the authors. Remaining implementation with quantum annealing correction (QAC) data concern demonstrations performed by us on the instances from Ref. [20]. With the same definition of run time the results (orange) are consistent with [20], which confirms the correctness of the methodology implementation. In addition, $[\text{TT}\varepsilon]_{\text{Med}}$ defined using the complete QPU access time $t_f^{\text{QPU access}}$ as reported by D-Wave's cloud interface (green), as well as run time measured with cloud access $t_f^{\text{run time}}$ (red), are presented. Solid lines are power-law fits $[\text{TT}\varepsilon]_{\text{Med}} \propto N^{\alpha}$, with the corresponding exponents α shown on the plot. The instance size range spanned by the lines denotes which data points were used for the respective fits. Both plots clearly demonstrate that with the proper run-time definition $[\text{TT}\varepsilon]_{\text{Med}}$ is constant for the problem sizes considered. For comparison, data for the SBM solver [26,36,37,39] with $[\text{TT}\varepsilon]_{\text{Med}}$ computed using one GPU and total run time t_f^{tot} (cyan), as well as pure GPU run time t_f^{GPU} (magenta), are presented. In this case the scaling of $[\text{TT}\varepsilon]_{\text{Med}}$ is more robust with respect to different run-time definitions, which is generally the case for classical algorithms. Panel (c) shows detailed data of the fitting exponent and its uncertainty for values of $\varepsilon \in \{0.5, 0.75, 1.00, 1.10, 1.25, 1.5\}\%$.

times exceeding 200 μs and substantially larger numbers of qubits.

Our results also allow us to identify another possible route toward a scaling quantum annealing advantage. This concerns a hypothetical scenario in which annealing technology progresses such that the number of qubits increases, while other relevant timescales—in particular the annealing and readout times—remain constant or grow only slowly. Under this assumption, and assuming no comparable improvements in the classical reference technology, a rough extrapolation of the data presented in Fig. 1 suggests that a crossover between the SBM and annealing performance could occur for problem sizes of the order of 10^5 variables.

B. Run time for digital quantum devices

For digital quantum computers, one can identify computation stages analogous to those discussed for quantum annealers. The initial stage involves preprocessing, in which a quantum circuit is translated into the topology and native gate set of a given device through transpilation. The transpiled circuit is then executed multiple times (shots), with consecutive runs separated by readout and thermalization steps. A comprehensive definition of run time for digital quantum devices should therefore include all of these processes. This perspective is also adopted in Ref. [38], which outlines procedures for measuring these contributions on IBM quantum devices.

Using this definition of run time, we revisit the results of Ref. [22]. In that work, a digitized counterdiabatic quantum optimization algorithm was reported to achieve a run-time advantage over SA [24] and IBM's proprietary CPLEX solver [25] for a class of higher-order unconstrained binary optimization (HUBO) problems. These problems involve cost functions with multivariable terms of order $N = 3$, generalizing the Ising model in Eq. (1),

$$P(s) = \sum_{\substack{i_1,\ldots,i_N \\ i_1+\cdots+i_N \le 3}} J_{i_1\ldots i_N} s_1^{i_1} \ldots s_N^{i_N}. \tag{3}$$

The choice of classical reference algorithms is discussed in Sec. IV. Here, we focus specifically on the definition and measurement of run time. The algorithm in Ref. [22] is a hybrid quantum-classical procedure consisting of two runs of SA and a single execution of the quantum routine. The first SA run provides an initial guess for the quantum step, and the output of the quantum routine is subsequently refined using a second SA run to mitigate potential readout errors.

The total run time therefore includes both classical and quantum components. In Ref. [22], the classical run time was estimated based on the time required to perform a single pass of SA, rather than being directly measured. In addition, overhead contributions associated with SA were not included, although these overheads are reported by the authors to be of the order of 1.6 s. Including such contributions would affect the resulting run-time comparison. Similarly, the quantum run time was estimated assuming a device throughput of 10^4 circuit executions per second, without explicitly accounting for programming and transpilation overheads. A comprehensive definition of run

time for digital quantum devices should therefore include all of these processes.

Taken together, these considerations indicate that the reported run-time advantage in Ref. [22] is sensitive to the chosen definition and estimate of run time. Under operationally grounded run-time metrics and consistent accounting of all relevant stages, a run-time advantage is not observed at the present stage. Even under the original measurement assumptions, comparison against stronger classical reference implementations alters the conclusions, as discussed in Sec. IV. A detailed analysis is provided in Appendix C.

C. Run time for classical-quantum algorithms

Heuristic algorithms typically depend on a set of hyperparameters that must be tuned in order to achieve optimal performance [40]. Such tuning can be computationally demanding [41,42]. As a result, reported solver run times that exclude the cost of hyperparameter optimization are difficult to interpret as evidence of a run-time advantage, unless the chosen hyperparameters can be shown to be problem-independent or the tuning effort is applied uniformly across all instances. In addition, for hybrid quantum-classical approaches, run-time accounting should include the overhead associated with data transfer between the classical host and the quantum device. This practice is standard in classical heterogeneous computing, such as CPU-GPU workflows, and analogous considerations apply in the quantum-classical context.

Within this framework, we consider the results of Ref. [43], where a quantum-classical solver for large-scale spin-glass problems was proposed and reported to achieve improved solution quality and favorable run-time performance relative to other approaches. The solver operates as a sequential procedure in which each step requires a set of hyperparameters. However, the reported run times do not explicitly include the time required to determine these hyperparameters, and the procedure used for their optimization is not described. Consequently, while the reported solution quality is notable, the available data do not allow for a fully reproducible assessment of end-to-end run-time performance.

From a benchmarking perspective, a more informative comparison would again rely on the time-to-epsilon metric TTε, with all relevant time contributions—including hyperparameter optimization—consistently accounted for, as advocated in Ref. [40].

III. SUPREMACY IN THE ORACLE QUERY COMPLEXITY FRAMEWORK

Oracle query complexity is a well-established framework for analyzing the computational resources required to solve certain classes of problems [44,45]. The central assumption of this framework is that the problem specification is not fully known; instead, partial information can be obtained by querying an external computational routine, known as an oracle. For example, consider the task of minimizing a function $f(x)$ whose explicit form is unknown to the solution algorithm. The algorithm can interact with the oracle to obtain a function value at a specific point x. The query complexity of the algorithm is then characterized by the number of oracle calls it makes as a function of the problem size.

In general, the query complexity framework applies only to certain families of algorithms that access functions in a way that can be modeled as querying an oracle. It is worth noting that the first theoretical demonstrations of quantum speedup were obtained within this framework [9,46,47]. However, once the restriction on limited problem knowledge is lifted (e.g., if the details of an oracle's implementation are made publicly available), the validity of query complexity results no longer holds. In such cases, it becomes possible to design more efficient algorithms that exploit the additional information about the problem specification. See, for example, the recent discussion in [48] regarding Grover's algorithm.

Here, we aim to interpret the query complexity framework from the perspective of a resource theory. To regard the number of oracle calls as a computational resource, one must assume that the oracle performs the most resource-intensive part of the computation. Only under this assumption can the claim that an algorithm requiring fewer oracle queries is more efficient be justified. Furthermore, even when a separation between the query complexities of different algorithms is proven, it is important to examine how this separation translates into run-time performance. This is particularly relevant in comparing quantum and classical computing, where the operations are characterized by different timescales—typically, a single quantum operation is slower than a classical one [49]. Because of these considerations, it has been argued that a practical quantum advantage can be achieved only by algorithms offering superquadratic speedups [49]. Run-time analysis thus plays a crucial role in determining the problem scales at which an oracle query complexity advantage leads to a genuine computational advantage in practice.

Recently, it has been experimentally demonstrated that, for certain problem sizes and even in the presence of uncorrected noise, quantum computers can exhibit an exponential advantage in a variant of Simon's problem [21]. In the original problem, the task is to find a hidden period (a bitstring) encoded in an unknown two-to-one function. In contrast, the authors of [21] study N-bit periods with a restricted Hamming weight w (i.e., the number of nonzero bits). In their formulation, the algorithm's figure of merit is defined in terms of a score function, whose primary purpose is to penalize random period-guessing strategies. The authors study scaling of this figure of merit, as a

function of the total number of possible periods denoted by $N_w \equiv \sum_j \binom{N}{j}$, where N is the total number of bits, and w the maximal allowed Hamming weight. Theoretical considerations are presented that suggest the possibility of a quantum advantage for such defined problem even for noisy, uncorrected quantum computers. The exponential advantage manifests itself in a polylogarithmic scaling of the score function as a function of the total number of possible periods (the classical model scales polynomially with this parameter). The authors then verify these claims by cloud quantum computing demonstrations. An exponential speedup is observed for functions with 29-bit inputs and restricted Hamming weights in the range $w \in [2, 7]$. For larger problem sizes, however, the advantage is destroyed by noise. This raises the question, how does this favorable scaling relate to the actual run-time of the algorithm?

To compare the run times of quantum and classical algorithms for Simon's problem, we implemented a classical brute-force algorithm running on a single GPU. In Ref. [20], the notion of a *compiler* was introduced to construct an oracle feasible for NISQ devices. The compiler's role is to split the computations performed by Simon's oracle into classical and quantum parts. Its objective is to generate the shallowest possible oracle circuit that can be embedded into the topology of a given NISQ device. A detailed discussion of this compiler can be found in Ref. [20]. For a fair comparison, we implemented a classical oracle corresponding to the quantum oracle described in Ref. [20], and we included only the oracle operation in the run time. Other computations performed by the compiler were excluded. We assume that any additional classical computations required for the oracle would take the same run-time in both the classical and quantum cases. Furthermore, the comparison does not account for the time needed to construct a shallow quantum circuit. Therefore, in principle, it should also apply to the next generation of quantum devices that will support deeper quantum circuits.

The quantum oracle computes the following classical n-bit function $f_b : \{0,1\}^n \mapsto \{0,1\}^n$:

$$f_b(x) = (x_0, \ldots, 0, x_{n-i+1} \oplus x_{n-i}, \ldots, x_{n-i}), \qquad (4)$$

where $b = 0^{n-i}1^i$ is the function period of Hamming weight i [20]. The additional steps transforming this function into a generic two-to-one function are performed by the classical compiler [21]. In our implementation we relied on the same function form.

The quantum algorithm for Simon's problem was implemented following the approach of Ref. [21], with its run-time estimated using Qiskit functionalities [50]. For each problem size, the quantum circuit was generated and transpiled with the Qiskit compiler (optimization level 3) to account for the connectivity and native gate set of IBM Brisbane [51]. The number of shots was chosen according to the oracle call bound given in Eq. (16) of Ref. [21].

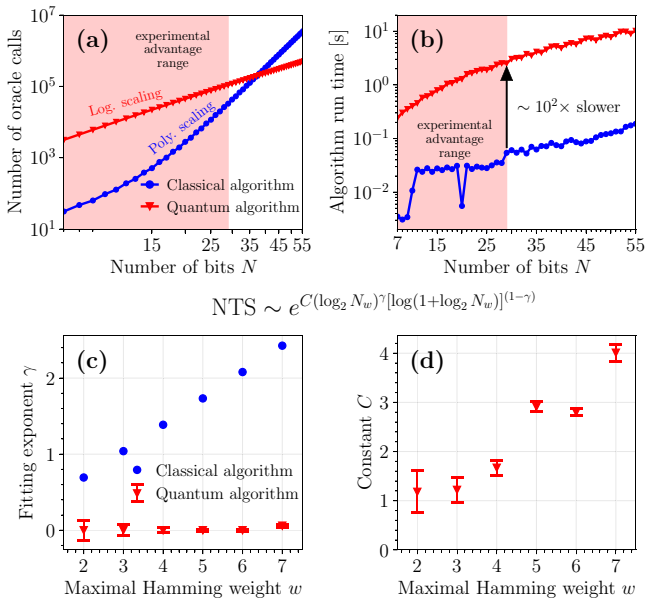

FIG. 2. For the restricted Simon's problem the quantum advantage manifests itself in a favorable polylogarithmic scaling of the protocol score function with the total number of periods $N_w = \sum_j \binom{N}{j}$ as compared to the exponential scaling for the classical algorithm. The score function depends on number of oracle queries and a probability of protocol success. (a) Total number of oracle queries for the protocol and (b) run time as a function of total number of bits N, as obtained via execution of the classical and quantum algorithm solving restricted Simon's problem with $w = 7$. The shaded area indicates sizes of problems, for which the advantage was verified by cloud quantum computing demonstrations in [21]. For the problem sizes and oracle periods considered, the classical algorithm run-time is shorter than the quantum one; we predict that the run-time crossover occurs for problem sizes $N = 60$. For example, in the case of $N = 29$ bits, which corresponds to the implementation presented in Ref. [21], the run-time of the classical algorithm is two orders of magnitude shorter than the quantum algorithm. The quantum implementation is based on the oracle constructed in Ref. [21], and the number of shots required for the algorithm was established using Eq. (16) of Ref. [21]. In this case the quantum circuit was transpiled to take into account the connectivity and native gate set of IBM Brisbane, and run-time was estimated using Qiskit functionalities. The classical algorithm was implemented on a single GPU. The fitting parameters of the score function for quantum and classical algorithm are presented in (c),(d). The parameter $\gamma = 0$ indicates that the score scales polylogarithmically, which implies quantum scaling advantage. The data for the quantum algorithm correspond to the results obtained for IBM Brisbane with dynamical decoupling (Table XX in [21]).

The results are summarized in Fig. 2. The run-time analysis clearly demonstrates that the exponential query complexity speedup reported in [21] does not translate into a faster solution time. For instance, with a problem size of $N = 29$ bits and Hamming weight $w = 7$, the classical brute-force algorithm requires approximately 0.035 s to find the

solution, while the estimated run-time for the quantum algorithm is of the order of 2 s, assuming $s = 10^5$ shots as in the demonstration of Ref. [21]. The projected problem size at which the quantum algorithm would achieve a run-time advantage is $N = 60$. However, as already discussed in Ref. [21], for such problem sizes the impact of noise is too severe to obtain a correct solution to Simon's problem with the quantum algorithm. Let us also stress that the classical algorithm was run on a single general-purpose GPU unit. Implementation of this algorithm on specifically tailored hardware, such as a field programmable gate array (FPGA), would result in even larger separation between classical and quantum run-times [52].

We emphasize once again that results derived from complexity theory, and query complexity in particular, are typically understood in an asymptotic sense. In contrast, currently available quantum devices restrict us to very small problem sizes. Combined with the fact that quantum and classical computers operate on different timescales for their basic operations, this leads to an important conclusion: a query complexity separation in favor of a quantum algorithm does not necessarily imply a run-time advantage. For certain ranges of problem sizes, algorithms with worse asymptotic scaling may in practice run much faster than those requiring, even exponentially, fewer operations or oracle calls. Similarly, the fastest known algorithm for matrix multiplication, which scales as $O(n^{2.371\,339})$ [53], does not provide any practical advantage over optimized implementations of the standard method with $O(n^3)$ scaling.

IV. APPROPRIATE CHOICE OF A CLASSICAL REFERENCE ALGORITHM

A promising approach to identifying quantum advantage for restricted classes of NP-hard instances, where no complexity-theoretic guarantees can be made, is the investigation of quantum heuristic methods. In such cases, demonstrating quantum advantage necessarily requires a direct comparison with classical algorithms solving the same problems. Because classical algorithms are continuously improving in both design and performance, most announcements of quantum supremacy should be understood as claims relative to the state-of-the-art classical methods (see also the related discussion in [12]). This was, for example, the case with the first random-circuit-sampling-based supremacy results [16–18]. Even though some of these quantum supremacy claims may later be invalidated, they nonetheless represent important milestones on the path toward establishing an unambiguous quantum advantage. At the same time, the choice of the classical reference algorithm must be made with great care, to avoid situations in which the observed quantum speedup arises merely from comparing against a suboptimal classical baseline. This naturally raises the question, how should the reference algorithm be chosen?

For heuristic algorithms, there is no universal guideline for selecting the classical reference, and the decision must be made on a case-by-case basis. Nevertheless, several criteria can reasonably guide this choice. First, the selection depends on the quality metric, such as run-time or energy consumption, as well as on the specific problem under consideration. In the context of optimization problems, which form the main focus of this work, arguably the most relevant practical quantity is TTε, which measures the expected time required to obtain an approximate solution within a given precision [cf. Eq. (2)].

For this figure of merit, an optimal reference classical algorithm should ideally combine solution quality with short run-time. Thus, the most reasonable choices are parallelizable algorithms, whose execution times are significantly shorter than those of sequential approaches, thanks to their ability to fully exploit modern computing resources such as GPUs [54] or FPGAs [55]. In this regard, a particularly interesting class of classical heuristics is based on the dynamics of Hamiltonian nonlinear systems [26,36,37,39,56–58], as these methods combine good performance with short execution times. Recently, it was shown that one algorithm from this class, the SBM, achieves scaling of TTε comparable to, or better than, that of quantum annealing [26]. This result effectively closes the previously reported quantum-classical gap in approximate optimization announced in [20], where the chosen classical reference was PT-ICM. The PT-ICM algorithm belongs to the class of physics-inspired methods based on temperature annealing [59]. However, unlike algorithms rooted in nonlinear Hamiltonian dynamics, PT-ICM is not straightforwardly parallelizable, and consequently exhibits longer execution times. This makes it a suboptimal reference for comparison with quantum annealing, a conclusion supported by the detailed analysis in [26].

Secondly, a classical reference algorithm should be designed to solve the same class of problems. Otherwise, the comparison is biased as it favors the quantum algorithm. The potentially worse performance of a chosen classical algorithm is not evidence of a quantum algorithm advantage, but merely reflects the fact that the problem cannot be solved natively by the classical reference. For example, in [22], the counterdiabatic quantum optimization algorithm investigated is designed to solve HUBO problems, whereas the reference CPLEX solver requires reformulation into a mixed-integer program. As the authors note, this reformulation is responsible for the longer run-time of CPLEX compared to their quantum algorithm. Setting aside the fact that the overheads of counterdiabatic quantum optimization were not accounted for (see the discussion in Sec. II), the relevant question is how the performance of the counterdiabatic quantum optimization algorithm would compare against a classical solver capable of solving HUBO with significantly lower overhead.

To address this issue, we consider the same class of HUBO problems as in Ref. [22], described by the Hamiltonian of the form

$$H = \sum_{(m,n)\in G_2} J_{mn}\sigma_m^z\sigma_n^z + \sum_{(p,q,r)\in G_3} K_{pqr}\sigma_p^z\sigma_q^z\sigma_r^z, \quad (5)$$

where G_2 and G_3 are the sets of two- and three-body couplings, respectively, and J_{mn} and K_{pqr} are the corresponding coupling strengths. The topology of instances considered is constructed iteratively, starting from the coupling graph C_0 of the heavy-hexagonal lattice of IBM's Heron architecture. Each step starts by using graph coloring to identify sets of independent two- and three-body interactions, then S_{2q} (S_{3q}) of them get included in the G_2 (G_3) set, and finally the coupling graph is modified by performing SWAP operation on pairs of qubits as defined by one of the two-body interactions sets. This procedure creates rather challenging instances, which are, however, particularly well suited for the bias-field digitized counterdiabatic quantum optimization (BF-DCQO) algorithm, since by changing the number of SWAP iterations, one can directly control the depth of the quantum circuit necessary for the quantum part of the algorithm. The authors of Ref. [22] thus restricted their analysis to only one SWAP iteration, which results in shallow circuits that can be executed on an IBM device in a regime where the results are not dominated by noise. The instances used in Ref. [22] were not made publicly available [60]. Consequently, we generated our own instances, closely following the description in Ref. [22] (cf. Appendix C), and we make both these instances and the generation routine available on GitHub [61].

To make a connection with the supremacy claims of Ref. [22], we employ the same figure of merit, which is the time-to-approximation ratio $\mathrm{TT}_{\mathcal{R}}$ [29], defined as the time required to find a solution with an energy lower than $\mathcal{R}E_{\mathrm{GS}}$, where E_{GS} is the ground-state energy of the problem instance. While this metric is conceptually similar to $[\mathrm{TT}\varepsilon]_{\mathrm{Med}}$, its execution in Ref. [22] is questionable, since it does not reflect the stochastic nature of the solvers investigated. In Fig. 3, we compare SBM with the results of BF-DCQO, as well as SA and CPLEX taken from Ref. [22]. The authors choose to present their results only for either the best-performing instance or as an average over just a few instances. In Appendix C, we argue in detail why this approach is insufficient for reproducible end-to-end comparison, especially for these particular instances, and furthermore, we show that even an optimized version of HUBO-native SA is enough to disprove their supremacy claims. Here, for the sake of the argument, we proceed in a similar fashion, showing SBM data for the best-performing instance in Fig. 3(a), and the average over five instances in Fig. 3(b). In both cases, SBM outperforms all other solvers, including BF-DCQO. Nevertheless, we stress that the purpose of this comparison is not to claim any kind of supremacy, but only to highlight how easily such claims can arise from selective reporting.

Nevertheless, an important point is to be made by looking at particular results for $N = 100$ in Fig. 3(a). There, it seems like BF-DCQO retains its advantage, since it achieves a better approximation ratio. However,

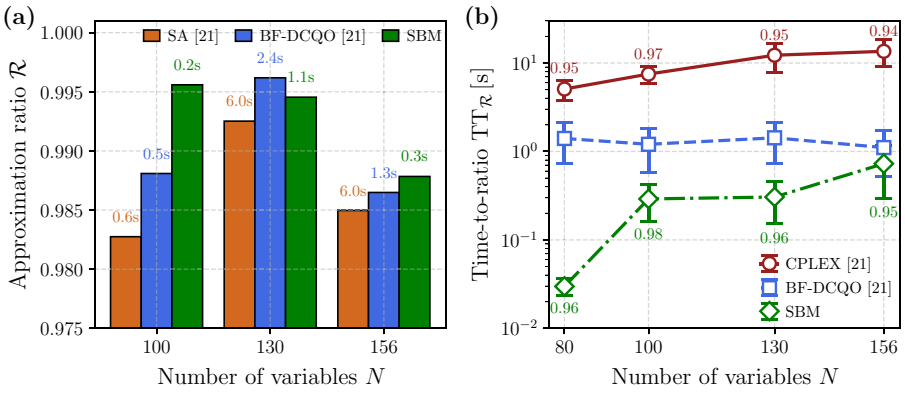

FIG. 3. Partial reproduction of Fig. 5 from Ref. [22], with additional results from the SBM solver. (a) The approximation ratio $\mathcal{R}$ achieved in a time annotated on top of the bars, for instance of type $S_{2q} = 1$, $S_{3q} = 4$ and couplings, from the Cauchy distribution. The SA and BF-DCQO results from Ref. [22] correspond to a single best-performing instance. Since the exact instances used by the authors of Ref. [22] were not disclosed, we generated our own instances and show SBM results for the best-performing instance. This underscores that reporting only the best-performing instance can bias comparisons and lead to overstated advantage claims. (b) The value of $\mathrm{TT}_{\mathcal{R}}$ for instances of type $S_{2q} = 1$, $S_{3q} = 6$ and couplings from a symmetrized Pareto distribution, while the annotations indicate the target ratio $\mathcal{R}$. The results for CPLEX and BF-DCQO are again taken from Ref. [22], where they were obtained as a result of averaging over five random instances. Similarly, we constructed our own instances and show SBM results averaged over five of them. In both cases SBM outperforms other solvers, casting doubt on the supremacy claim of Ref. [22]. Moreover, in Appendix C we present a more thorough analysis of these instances, and show how to optimize HUBO SA to outperform BF-DCQO.

the difference between SBM and BF-DCQO is minuscule ($\Delta\mathcal{R} \simeq 0.002$), while the corresponding $\mathrm{TT}_{\mathcal{R}}$ is smaller by a factor of 2 for SBM. A proper metric for comparing stochastic solvers, such as the time-to-epsilon $[\mathrm{TT}\varepsilon]_{\mathrm{Med}}$ defined in Eq. (2), should take into account both solution quality, and the expected run-time to achieve it.

This analysis illustrates how important it is to select a suitable classical reference algorithm when assessing quantum advantage claims. Let us also mention that the results of [22] were recently also disputed in [62], where it was claimed that the superior performance of BF-DCQO is not due to the quantum routine, as a modified algorithm with the quantum routine replaced by a classical solver exhibits a similar performance. Finally, for the sake of completeness, we tested the performance of D-Wave's newest quantum annealer, the Advantage2 1.6 based on the Zephyr topology. The results are presented in Fig. 4. Since the instances considered are not native to the topology of the QPU, an embedding procedure must be carried out. Figure 4(c) shows the distribution of chain lengths in the embedding, with a tail extending to very long chains. This, unsurprisingly, results in very poor performance, even after postselecting the best result out of 35 random instances, and five independent shots of 2^{10} samples each. We stress that this is just a single manifestation of a more general issue regarding claims of quantum advantage—the necessity of embedding problem instances onto the hardware graph of a quantum device often significantly reduces performance, and thus one should be very careful when making such claims on the basis of experiments or cloud quantum computing demonstrations with hardware-native problems only.

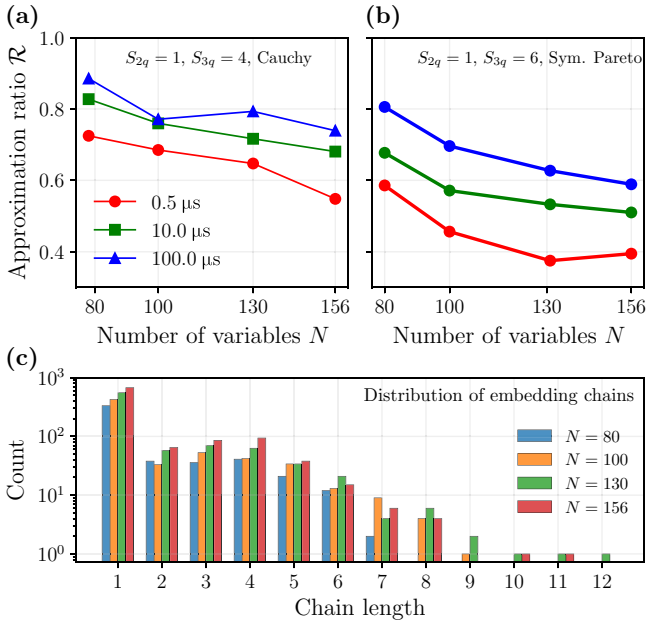

FIG. 4. (a),(b) Approximation ratio $\mathcal{R}$ achieved by D-Wave's Advantage2 1.6 quantum annealer on the same HUBO instances as in Fig. 3, computed for each instance type and size, by selecting the best result out of five shots with 2^{10} samples each. A forward annealing scheme was used, with annealing times of 0.5 µs (red), 10 µs (green), and 100 µs (blue). The instances were first reduced to QUBO form in the same way as in the case of SBM results (see Appendix C for details), and then embedded into a Advantage2 1.6 working graph. The results are, unsurprisingly, much worse than all other solvers considered. They can be explained by investigating the distribution of chain lengths in the necessary embedding, shown in (c). Since very long chains are needed to fit the problem instances onto the QPU, the performance of the quantum annealer is severely degraded by possible chain breaks. This highlights an issue that is often overlooked, yet crucial in the context of benchmarking solvers with hardware-imposed constraints on problem topology.

V. SCALING BEHAVIOR IN SMALL-INSTANCES REGIME

Quantum advantage is usually demonstrated by showing that a quantum algorithm exhibits more favorable scaling with respect to the problem size than a reference classical algorithm. Results derived from theoretical considerations, such as those in the oracle query complexity paradigm, are unambiguous. However, when the scaling of heuristic methods is investigated through demonstrations on quantum hardware, one must account for the fact that, due to the limited number of qubits available in both digital and analog quantum devices, the accessible instance sizes are severely restricted and do not fully meet the requirements for asymptotic scaling. In particular, the scaling behavior of classical algorithms in the problem-size range accessible to current quantum devices—of the order of hundreds of variables for digital hardware and thousands for annealers—may be strongly affected by finite size effects. Classical algorithms implemented on hardware with timescales

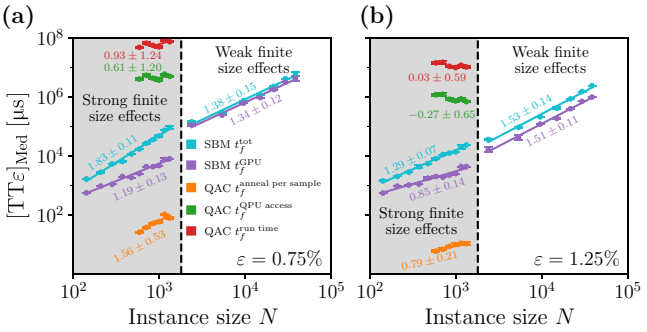

FIG. 5. Illustration of impact of solver-related overhead on scaling behavior of $[\mathrm{TT}\varepsilon]_{\mathrm{Med}}$, as defined in Eq. (2), using a certain type of Ising instances, relevant for near-term quantum devices (see Ref. [26] for details). Note that the finite size effects diminish significantly beyond the gray-colored region ($N \gtrsim 2000$), as demonstrated with SBM results. However, such system sizes are currently beyond the capabilities of correct quantum annealing devices.

comparable to quantum devices, such as GPU clusters or FPGAs, are especially susceptible to overheads that dominate at small scales but become negligible for larger problem sizes. Therefore, despite the limited capabilities of today's quantum hardware, it is essential to investigate sufficiently large problem sizes in order to establish robust scaling properties. Such a situation was identified in [26] for the SBM, a heuristic inspired by nonlinear Hamiltonian dynamics [36,37,39]. The study demonstrated that the scaling properties of this method—specifically its time to solution—cannot be reliably inferred when restricted to problem sizes corresponding only to the qubit counts of current quantum annealers. See Fig. 5 for a summary of relevant results.

VI. SUMMARY

Quantum computing has the potential to become a disruptive technology. Realizing this potential, however, requires not only continued technological progress but also the identification of use cases—potentially relevant for industry—in which quantum methods provide a clear advantage over existing classical approaches. In this work, we adopt a pragmatic perspective in which such an advantage should manifest as a *shorter end-to-end time to solution* for a quantum algorithm.

From this perspective, we reexamined several recent claims of quantum advantage and find that, when run time is defined operationally and compared against appropriately chosen classical references, none of the demonstrations considered establish a practical run-time advantage on current NISQ hardware. We focus in particular on three representative case studies.

A. Quantum annealing (approximate QUBO)

In Ref. [20], run time was proxied by the annealing time. Using end-to-end run-time measurements, we find that the median TTε remains approximately constant over the problem sizes tested. This behavior is dominated by the readout time per shot (approximately 200 μs), which exceeds the annealing duration (0.5–27 μs) by one to two orders of magnitude. As a result, statistical uncertainties are too large to support reliable scaling conclusions over the explored regime (cf. Fig. 1). Moreover, comparison with a stronger classical baseline, such as the SBM , indicates that no run-time advantage is observed even under optimistic assumptions regarding the annealing time, consistent with Ref. [26]. The SBM exhibits robust scaling behavior at system sizes well beyond those accessible to current quantum annealers (cf. Fig. 5).

B. Gate-based oracle query (restricted Simon's problem)

While an exponential separation in query complexity is present, wall-clock performance in the regime accessible for cloud quantum computing demonstrations favors a tuned classical GPU implementation. For $N = 29$ and restriction parameter $w = 7$ [21], the classical run time is approximately 0.05 s, whereas the quantum implementation requires roughly 2 s, corresponding to a roughly $100\times$ difference. Extrapolations suggest a crossover only at $N \approx 60$, which lies beyond the capabilities of current noise-limited devices (cf. Fig. 2).

C. BF-DCQO hybrid algorithm

Finally, we analyze reported performance gains for solving HUBO instances [22]. We find that these conclusions rely on run-time estimates and limited statistical sampling, which complicate a reproducible assessment of end-to-end performance. In particular, the problem instances considered were designed to align with specific hardware characteristics [63] and require substantially richer statistics to support claims of systematic advantage. Our analysis shows that, under consistent timing protocols and broader sampling, the reported performance differences are not robust (cf. Fig. 3 and Appendix C).

In summary, across all examined cases, a practical run-time advantage on current NISQ hardware is not observed under experimentally grounded performance metrics. Credible claims of run-time quantum advantage require careful accounting of dominant overheads (including readout, thermalization, and compilation), the use of well-defined metrics such as TTε with appropriate optimization, and benchmarking against state-of-the-art classical solvers selected impartially for the problem class under consideration.

VII. OUTLOOK

No single, universal procedure exists for evaluating quantum advantage, since different problem classes and computational paradigms exhibit distinct characteristics. Nevertheless, a set of general principles can be distilled from our analysis. In the context of run-time-based quantum advantage, we summarize these principles as follows.

(a) *Primary run-time measurement*. Run-time measurements should be obtained from direct timing of executions, rather than inferred from nominal device parameters or documentation.

(b) *Comprehensive overhead accounting*. Run-time should be reported as a direct measurement of the total execution time, including all relevant system-level overheads such as readout, compilation, data transfer, and hyperparameter tuning.

(c) *Standardized performance metrics*. Performance should be evaluated using well-defined and widely accepted metrics, such as the time-to-ε for sufficiently small values of ε, with results assessed across diverse problem instances and structures.

(d) *Scaling integrity*. Scaling analyses should span a sufficiently broad range of problem sizes to capture asymptotic behavior, and extrapolations beyond the measured regime should be treated with appropriate caution.

(e) *High-performance classical baselines*. Classical reference algorithms should be selected with care, taking into account the current state of algorithmic development and implementation-level optimizations.

(f) *The hybrid null hypothesis*. For hybrid quantum-classical algorithms, a suitable null hypothesis should be tested, namely whether replacing the quantum subroutine with a purely classical counterpart yields comparable or improved performance.

We stress that these principles are intended as general guidance rather than problem-specific prescriptions, and are not meant to constitute an exhaustive set of criteria.

We emphasize that while energy efficiency is a potentially relevant dimension of quantum advantage [14], the present work deliberately focuses on run-time-based performance metrics. At the current level of user access to quantum hardware, energy consumption cannot be independently verified or consistently defined, as it depends on system-level factors such as cryogenics and control electronics that are typically not transparent to the user. At the same time, contemporary high-performance classical accelerators, such as modern GPUs, typically operate at subkilowatt power levels, underscoring the importance of clearly defined system boundaries when comparing different computational platforms. The development of verifiable and reproducible energy-efficiency metrics therefore remains an important open challenge for future benchmarking efforts.

As a final note, while the present work does not focus on system-level capability metrics such as quantum volume [64] and CLOPS [65], we note that such metrics provide a valuable and complementary means of tracking technological progress in quantum hardware, particularly at the level of device capability and system throughput.

ACKNOWLEDGMENTS

This project was supported by the National Science Center (NCN), Poland, under Projects: Sonata Bis 10, No. 2020/38/E/ST3/00269 (BG). Quantumz.io Sp. z o.o acknowledges support received from The National Centre for Research and Development (NCBR), Poland, under Project No. POIR.01.01.01-00-0061/22.

DATA AVAILABILITY

The data that support the findings of this article are openly available [61].

APPENDIX A: SIMULATED BIFURCATION MACHINE

To demonstrate that the results for classical algorithms are less sensitive to run-time definition changes we consider a discretized version of the Simulated Bifurcation Machine algorithm [26,36,37,39], first introduced as dSB in Ref. [37]. The algorithm is based on a nonlinear Hamiltonian system, whose motion is governed by the following equations:

$$\begin{aligned} \dot{q}_i &= a_0 p_i, \\ \dot{p}_i &= -\left[a_0 - a(t)\right] q_i + c_0 \left(\sum_{j=1}^{N} J_{ij} f(q_j) + h_i \right), \end{aligned} \tag{A1}$$

where $f(x) = \mathrm{sign}(x)$ is the signum function. The original formulation of dSB is modified to include a ternary discretization scheme [66], replacing sign with

$$f(x) = \begin{cases} 0 & |x| \le \Delta(t), \\ \mathrm{sign}(x) & |x| > \Delta(t), \end{cases} \tag{A2}$$

where $\Delta(t) = 0.7(t/T)$ is a time-dependent threshold and T is the total time of the evolution.

The nonlinearity predominantly stems from perfectly inelastic walls that are inserted at $|q_i| = 1$, and after hitting a wall ($|q_i| > 1$), the particle position q_i is set to $\mathrm{sign}(q_i)$ and its momentum to $p_i = 0$. The system's evolution depends on a set of hyperparameters. Hyperparameters a_0 and c_0 are typically set to $a_0 = 1$ and $c_0 = 0.7a_0/\sigma\sqrt{N}$, where σ is the standard deviation of off-diagonal matrix elements of J. The system undergoes bifurcations, which are caused by a change of the linear time-dependent function $a(t) = t/T$. Bifurcations change the system's energy landscape to one approximately encoding the local minima of the Ising term. This mechanism allows one to find low-energy solutions of the binary optimization problem by binarizing the final system's state, taking $s_i = \mathrm{sign}(q_i)$. The SBM is a stochastic system sensitive to initial conditions; a large number of independent replicas can be integrated simultaneously from different starting points, enabling massive parallelization. The remaining hyperparameters are the timestep Δt and the number of steps N_s, which jointly determine the total evolution time $T = N_s \Delta t$. Since the SBM is a stochastic algorithm it needs to be executed a number of times (which can be done in parallel); we denote this number of replicas by N_r. For each replica the timestep δt is chosen randomly; for details see [26].

APPENDIX B: BENCHMARKING CLASSICAL ALGORITHM ON PSEUDORANDOM SIMULATED ORACLES

Simon's problem for a function that simulates the behavior of an oracle can be solved using a classical

ALGORITHM 1. Simulated Bifurcation Machine.

Input:

J, h – coupling matrix and local fields vector specifying an instance of the Ising model,
Δt – time step,
N_s – number of steps,
$a(t)$ – pump function,
$f(x)$ – discretization function,
a_0, c_0 – hyperparameters.

Output:

E, q – energy and its corresponding state of the found minimum.

$n = \text{length}(h)$
$q_1 \leftarrow \{\text{rand}(-1,1)\}^n$
$p_1 \leftarrow \{\text{rand}(-1,1)\}^n$
for $i = 1, \ldots, N_s$ **do**
  $p_{i+1} \leftarrow \left\{[a_0 - a(i\Delta t)]\, q_i + c_0 \left(\sum_{i=1}^{N} J_{ij} f(q_j) + h_i\right)\right\} \Delta t$
  $q_{i+1} \leftarrow q_i + a_0 \Delta t p_{i+1}$
  for $j = 1, \ldots, n$ **do**
    if $|q_{i+1,j}| \geq \sqrt{2}$ **then**
      $q_{i+1,j} \leftarrow \sqrt{2}\text{sign}(q_{i+1,j})$
      $p_{i+1,j} \leftarrow 0$
    end if
  end for
end for
$q = \text{sign}(q_{N_s+1})$
$E = \frac{1}{2} q^T J q + h^T q$
return E, q

algorithm. We have used a solver that not only confirms the existence of a nonzero period, but also finds the period. The series of tests and benchmarks we have performed require simulation of oraclelike two-to-one functions. Given the number of bits n, a function $f : \{0,1\}^n \mapsto \{0,1\}^n$ with period $p \in \{0,1\}^n \setminus \{0\}^n$ needs to match the following property:

$$\forall x, y \in \{0,1\}^n, \quad f(x) = f(y) \iff y = x \oplus p. \tag{B1}$$

Given a one-to-one function $g : \{0,1\}^n \mapsto \{0,1\}^n$ and ordering $\prec$ over $\{0,1\}^n$, one can define a function

$$f(x) = g(\min_{\prec}(x, x \oplus p)) \tag{B2}$$

that meets the requirements. The $\prec$ used in our benchmarks was the lexicographic ordering of bit tuples. The other possible orderings can be achieved by comparing $h(x)$ and $h(x \oplus p)$ for one-to-one functions $h : \{0,1\}^n \mapsto \{0,1\}^n$ instead.

Since we can build the right function for any given period, properties of the period, such as number of set bits, can be assigned arbitrarily. The last step to consider is construction of one-to-one functions such as g. To include all f_b functions from (4), we have implemented affine functions over the field GF(2):

$$g(x) = Ax \oplus b \quad \text{for } A \in \{0,1\}^{n \times n}, \\ \text{rank}_{\text{GF(2)}} A = n, \quad b \in \{0,1\}^n. \tag{B3}$$

Notably, $A\,x$ is also intended to be computed in GF(2). The requirement that $\text{rank}_{\text{GF(2)}} A = n$ is equivalent to A being invertible in GF(2), which is crucial to g being a bijection.

Pseudorandom generation of A and b is straightforward to implement. A constraint on A needs to be considered—the probability that a pseudorandom $\{0,1\}^{n \times n}$ matrix will be invertible in GF(2) decreases with n. However, as shown in [67], it converges to a constant that exceeds $1/4$, which makes it practically viable to generate matrices in a loop until finding a matrix with rank n.

A wider selection of g functions (and order-altering h) can be achieved by applying complex strategies, including cryptographic schemes such as AES with a fixed key unknown to the solver. The proposed choice of skipping h and using affine g is a compromise that ensures the oracle is nontrivial, and includes all cases described in (4).

The classical Algorithm 2 is oblivious to the inner implementation of the oracle. The oraclelike function is

applied for all the x vectors such that either $(x_1, \ldots, x_{\lfloor n/2 \rfloor})$ or $(x_{\lfloor n/2 \rfloor+1}, \ldots, x_n)$ are all zeros. For any period p, both higher $\lceil n/2 \rceil$ and lower $\lfloor n/2 \rfloor$ bits of p will be tested—so there is exactly one pair x_i, x_j such that $f(x_i) = f(x_j)$, yielding $p = x_i \oplus x_j$. The total number of vectors to evaluate is $2^{\lceil n/2 \rceil} + 2^{\lfloor n/2 \rfloor} - 1$. In our scenario, the complexity is in $\Theta(n^2 2^{n/2})$. That cost is reached both by evaluating of the linear functions, and by sorting evaluation results, which involves comparisons of n-bit vectors.

1. Polynomial scaling for restricted Simon's problem with fixed w

The proposed classical algorithm can be further optimized for solving the restricted Simon's problem with $w < \lceil n/2 \rceil$. A period p with at most w set bits can always be decomposed into $p = x_i \oplus x_j$ such that x_i and x_j have at most w set bits in total. Retaining the core concept from the general approach, this can be achieved by x_i with $\lceil n/2 \rceil$ lowest bits of p, and x_j with the remaining $\lfloor n/2 \rfloor$ highest bits.

In order to find the decomposition in a single pass, we can constrain both x_i and x_j to have at most w set bits, rather than w bits set in total. This approach remains highly efficient for parallel computations, as demonstrated by the run-time measurements from Fig. 2. Constrained vectors can be generated with Algorithm 3. The required precomputation of $S(m,k) = \sum_{v=0}^{k} \binom{m}{v}$ values involves $\Theta(n^2)$ computational and memory cost, and is calculated once per n, and reusable across multiple solver calls.

The adjusted solver for the restricted Simon's problem described in Algorithm 4 is a specialized version of Algorithm 2 where only vectors x_i with up to w set bits are constructed and used in duplicate search. For small fixed w, specifically $w \le n/4$, the total number of vectors processed is

$$\begin{aligned} \mathrm{v}(n,w) &= S(\lfloor n/2 \rfloor, w) + S(\lceil n/2 \rceil, w) \\ &\le 2S(\lceil n/2 \rceil, w) \le 2\lceil n/2 \rceil \binom{\lceil n/2 \rceil}{w} \\ &= 2\lceil n/2 \rceil \prod_{i=1}^{w} \frac{\lceil n/2 \rceil - i + 1}{i} \le 2\lceil n/2 \rceil^{w+1}. \end{aligned} \tag{B4}$$

The most dominant component to Algorithm 4 complexity is sorting an array of $\mathrm{v}(n,w)$ bit vectors of length n. This operation is $\Theta(\mathrm{v}(n,w) \cdot \log(\mathrm{v}(n,w)) \cdot n)$. For small or fixed w, the bound from Eq. (B4) applies, so the complexity is in the class $O(wn^{w+2} \log n)$. This shows that the proposed classical solver for the restricted Simon's problem has polynomial complexity for fixed w. The polynomial behavior can be observed in Fig. 2.

ALGORITHM 2. Classical solver for Simon's problem.

Input:
n – number of bits,
$f - \{0,1\}^n \mapsto \{0,1\}^n$ function that evaluates oracle.
Output:
$p \in \{0,1\}^n$ – oracle period, zeros when there is none.
$x_1 \leftarrow \{0\}^n$
for $i = 1, \ldots, 2^{\lceil n/2 \rceil} - 1$ **do**
  $x_{i+1} \leftarrow (\lceil n/2 \rceil$ lowest bits of $i, 0 \ldots)$
end for
for $i = 1, \ldots, 2^{\lfloor n/2 \rfloor} - 1$ **do**
  $x_{i+2^{\lceil n/2 \rceil}} \leftarrow (0 \ldots, \lfloor n/2 \rfloor$ lowest bits of $i)$
end for
for $i = 1, \ldots, 2^{\lceil n/2 \rceil} + 2^{\lfloor n/2 \rfloor} - 1$ **do**
  $y_i \leftarrow f(x_i)$
end for
$(j_1, \ldots, j_{2^{\lceil n/2 \rceil}+2^{\lfloor n/2 \rfloor}-1}) \leftarrow$ **SortingPermutation**$(y_1, \ldots, y_{2^{\lceil n/2 \rceil}+2^{\lfloor n/2 \rfloor}-1})$ ▷ done by sorting indices with y as key
for $i = 1, \ldots, 2^{\lceil n/2 \rceil} + 2^{\lfloor n/2 \rfloor} - 2$ **do**
  if $y_{j_i} = y_{j_{i+1}}$ **then**
    return $x_{j_i} \oplus x_{j_{i+1}}$
  end if
end for
return $\{0\}^n$

ALGORITHM 3. Computing the ith vector within limit of w set bits.

Input:

n – number of bits,

$w \leq n$ – maximum number of set bits,

$i \in \left\{0, \ldots, \left(\sum_{v=0}^{w} \binom{n}{v}\right) - 1\right\}$ – index of vector.

$S(m, k) = \sum_{v=0}^{k} \binom{m}{v}$ – precomputed values for $m = \{0, \ldots, n\}$ and $k = \{0, \ldots m\}$.

Output:

$x \in \{0, 1\}^n$ – the i-th vector meeting the constraint in lexicographic order.

$x \leftarrow \{0\}^n$
$v \leftarrow w$
for $b = 1, \ldots, n$ **do**
  if $i \geq S(n - b, v)$ **then**
    $x_b \leftarrow 1$
    $i \leftarrow i - S(n - b, v)$
    $v \leftarrow v - 1$
  end if
end for
return x

ALGORITHM 4. Classical solver for the restricted Simon's problem.

Input:

n – number of bits,

$f - \{0, 1\}^n \mapsto \{0, 1\}^n$ function that evaluates oracle,

$w \leq n$ – maximum number of set bits in oracle period,

$S(m, k) = \sum_{v=0}^{k} \binom{m}{v}$ – precomputed values for $m = \{0, \ldots, n\}$ and $k = \{0, \ldots m\}$.

Output:

$p \in \{0, 1\}^n$ – oracle period, zeros when there is none.

$x_1 \leftarrow \{0\}^n$
for $i = 1, \ldots, S\left(\lceil n/2 \rceil, \min\left\{w, \lceil n/2 \rceil\right\}\right)$ **do**
  $t \leftarrow$ **ALG3**$\left(\lceil n/2 \rceil, \min\left\{w, \lceil n/2 \rceil\right\}, i, S\right)$
  $x_{i+1} \leftarrow (\lceil n/2 \rceil$ lowest bits of $t, 0 \ldots)$
end for
for $i = 1, \ldots, S\left(\lfloor n/2 \rfloor, \min\left\{w, \lfloor n/2 \rfloor\right\}\right)$ **do**
  $t \leftarrow$ **ALG3**$\left(\lceil n/2 \rceil, \min\left\{w, \lfloor n/2 \rfloor\right\}, i, S\right)$
  $x_{i+2^{\lceil n/2 \rceil}} \leftarrow (0 \ldots, \lfloor n/2 \rfloor$ lowest bits of $t)$
end for
for $i = 1, \ldots, 2^{\lceil n/2 \rceil} + 2^{\lfloor n/2 \rfloor} - 1$ **do**
  $y_i \leftarrow f(x_i)$
end for
$(j_1, \ldots, j_{2^{\lceil n/2 \rceil}+2^{\lfloor n/2 \rfloor}-1}) \leftarrow$ **SortingPermutation**$(y_1, \ldots, y_{2^{\lceil n/2 \rceil}+2^{\lfloor n/2 \rfloor}-1})$ ▷ done by sorting indices with y as key
for $i = 1, \ldots, 2^{\lceil n/2 \rceil} + 2^{\lfloor n/2 \rfloor} - 2$ **do**
  if $y_{j_i} = y_{j_{i+1}}$ **then**
    return $x_{j_i} \oplus x_{j_{i+1}}$
  end if
end for
return $\{0\}^n$

ALGORITHM 5. Building conflict graph for two-body interactions.

Input:
$C = (V, E)$ — connectivity graph
Output:
$C^{(2)}$ – conflict graph where vertices represent 2-body interactions.
$C^{(2)} \leftarrow (\{1, \ldots, |E|\}, \emptyset)$ ▷ Initialize conflict graph
for $i = 1, \ldots, |E| - 1$ **do**
  for $j = i + 1, \ldots, |E|$ **do**
    if $E[i] \cap E[j] \neq \emptyset$ **then**
      Add edge (i, j) to $C^{(2)}$ ▷ Interactions share a qubit
    end if
  end for
end for
return $C^{(2)}$

APPENDIX C: DETAILED ANALYSIS OF HUBO INSTANCES FROM REF. [22]

1. HUBO instance construction

We begin by providing a more detailed description of the HUBO instances used in Sec. IV of the main text. These instances are designed to be challenging, yet well suited for IBM quantum computers, thus the starting point in their construction is the heavy-hex graph C_0 of the IBM Heron architecture [68]. Two conflict graphs are then constructed, $C_0^{(2)}$ and $C_0^{(3)}$, with vertices corresponding to the edges of C_0 in the case of $C_0^{(2)}$, and to the triangles and three-qubit paths in the case of $C_0^{(3)}$. Edges in the conflict graphs connect vertices that share a vertex in C_0, which means that the operations on qubits encoding these interactions cannot be executed simultaneously. By applying graph coloring to the conflict graphs, one obtains partitions of the edges of C_0 into sets of non-conflicting edges, $P_{2q} = \{P_{2q}^{(1)}, P_{2q}^{(2)}, \ldots, P_{2q}^{(M_2)}\}$ and $P_{3q} = \{P_{3q}^{(1)}, P_{3q}^{(2)}, \ldots, P_{3q}^{(M_3)}\}$, where M_2 and M_3 are the number of colors used in the coloring of $C_0^{(2)}$ and $C_0^{(3)}$, respectively. Each set $P_{2q}^{(i)}$ and $P_{3q}^{(i)}$ contains two- and three-body interactions that can be executed in parallel on the quantum hardware.

Interaction sets are then included in the HUBO Hamiltonian (5) by updating the G_2 and G_3 sets: $G_2 \leftarrow G_2 \cup \{P_{2q}^{(i)}\}_{i=1,\ldots,S_{2q}}$ and $G_3 \leftarrow G_3 \cup \{P_{3q}^{(i)}\}_{i=1,\ldots,S_{3q}}$. The parameters $S_{2q} \leq M_2$ and $S_{3q} \leq M_3$ control the number of sets of two- and three-body interactions included in the

ALGORITHM 6. Building conflict graph for three-body interactions.

Input:
$C = (V, E)$ — connectivity graph
Output:
$C^{(3)}$ – conflict graph where vertices represent 3-body interactions.
$\mathcal{I} \leftarrow \emptyset$
for $v \in V$ **do**
  $N_v \leftarrow \{u \in V : (v, u) \in E\}$ ▷ Neighbors of v
  for $(u, w) \in \{(u, w) : u, w \in N_v \wedge (u, w) \in E\}$ **do** ▷ All pairs of neighbors
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{(v, u, w)\}$
  end for
end for
$C \leftarrow (\{1, \ldots, |\mathcal{I}|\}, \emptyset)$ ▷ Initialize conflict graph
for $i < j \leq |\mathcal{I}|$ **do**
  if $\mathcal{I}[i] \cap \mathcal{I}[j] \neq \emptyset$ **then** ▷ Interactions share a qubit
    Add edge (i, j) to C
  end if
end for
return C

Hamiltonian, and thus directly affect the complexity of its topology. The final step is the SWAP operation, carried out using the first two-body interaction set $P_{2q}^{(1)}$, which maps $C_0 \to C_1$ by permuting all qubits pairs $(q_1, q_2) \in P_{2q}^{(1)}$. This iteration can then be repeated, successively increasing the number of interactions. After the interaction topology is prepared, the couplings are sampled from one of two distributions: (i) Cauchy, with density function $f(x) = 1/[\pi(1+x^2)]$; and (ii) symmetrized Pareto, which starts from density function $f(x) = \alpha/x^{\alpha+1}$, but the samples are symmetrized by multiplying them by a Bernoulli distributed random sign with probability $1/2$.

Like the authors of Ref. [22], we select two types of instances: (i) $S_{2q} = 1$, $S_{3q} = 4$, single SWAP and Cauchy-distributed couplings; (ii) $S_{2q} = 1$, $S_{3q} = 6$, single SWAP and symmetrized Pareto-distributed couplings. We also considered reduced initial couplings graphs, to obtain instances smaller than the full heavy-hex IBM Heron QPU. In the end, we generated 50 random instances per type and size $N \in [80, 100, 130, 156]$. To ensure transparency and reproducibility, we make publicly available a GitHub repository containing all generated instances together with the Python code used for their generation [61].

2. Performance of simulated bifurcation and simulated annealing on HUBO instances

In its usual formulation, both SBM and SA are designed to solve problems with up to second-order interactions. To apply them to HUBO instances, we first need to use some reduction technique to convert higher-order terms into quadratic ones. We choose a standard reduction method for third-order Ising interactions [69]:

$$\begin{aligned} \pm s_i s_j s_k \to\ & 3 \pm (s_1 + s_2 + s_3 + 2s_{\text{aux}}) \\ & + 2s_{\text{aux}}\left(s_i + s_j + s_k\right) + s_i s_j + s_i s_k + s_j s_k, \end{aligned} \tag{C1}$$

which introduces one auxiliary spin s_{aux} per third-order term, and preserves the spectrum of the original problem.

ALGORITHM 7. HUBO instance generation.

Input:
C_0 – starting topology graph
S_{2q} – number of two-body interaction sets to include,
S_{3q} – number of three-body interaction sets to include,
n_{swap} – number of SWAP iterations,
dist – coupling distribution
Output:
J, K – dictionaries with two-body and three-body interactions defining the HUBO instance
$G_2 \leftarrow \emptyset$, $G_3 \leftarrow \emptyset$
$C \leftarrow C_0$
for n $= 0, 1, \ldots, n_{\text{swap}}$ **do**
  $C^{(2)} \leftarrow$ **BuildConflictGraph**$(C, 2\text{-body})$, $C^{(3)} \leftarrow$ **BuildConflictGraph**$(C, 3\text{-body})$
  $P_{2q} \leftarrow$ **GraphColoring**$(C^{(2)})$, $P_{3q} \leftarrow$ **GraphColoring**$(C^{(3)})$
  for $i = 1, \ldots, S_{2q}$ **do**
    $G_2 \leftarrow G_2 \cup P_{2q}^{(i)}$
  end for
  for $i = 1, \ldots, S_{3q}$ **do**
    $G_3 \leftarrow G_3 \cup P_{3q}^{(i)}$
  end for
  if swap $< n_{\text{swap}}$ **then**
    $C \leftarrow$ **ApplySWAP**$(C, P_{2q}^{(1)})$ ▷ Permute qubit pairs from first 2-body set
  end if
end for
for $(m, n) \in G_2$ **do**
  $J_{mn} \leftarrow$ **SampleCoupling**(dist)
end for
for $(p, q, r) \in G_3$ **do**
  $K_{pqr} \leftarrow$ **SampleCoupling**(dist)
end for
return J, K

In the GitHub repository [61], we provide both HUBO instances and their QUBO counterparts after the reduction. We also note that SA can be extended to directly handle higher-order interactions fairly easily, and while in principle SBM can be too [70], it is much more involved. Thus, we only consider the HUBO-native version of SA, which we discuss in greater detail in Appendix D.

We start by analyzing the success fraction, defined as the fraction of instances for which a given algorithm found solutions better than or equal to a certain threshold. The results are shown in Fig. 6 for three selected thresholds, $\mathcal{R} \in \{0.90, 0.95, 0.99\}$, and as a function of instance size N. We see a first hint that these instances are indeed challenging, since the success fraction decreases rapidly with both instance size and the threshold approximation ratio. Interestingly, SBM seems to struggle much more than SA, at least when it comes to pure solution quality. However, when discussing approximate optimization algorithms, one should always be mindful of the time-quality tradeoff.

Let us now consider the run-time of the algorithms, to see a different picture emerge. In Fig. 7, we show the approximation ratio $\mathcal{R}$, averaged over 50 coupling realizations and 10 (5) independent runs of SBM (SA), as a function of run-time, for each considered instance type and size. We immediately notice that while QUBO SA reaches better solutions eventually, SBM is able to reach good solutions close to an order of magnitude faster. Similar conclusions can be drawn when comparing SBM to HUBO SA in the case of Cauchy instances, while the Pareto instances seem to be "easy" for HUBO SA. Nevertheless, even in this case, the performance of SBM is still respectable, especially when considering the potential for further improvement via extending it to natively handle higher-order interactions [70].

Crucially, this is a result of a statistically sound benchmarking procedure, being aware of and taking into account the variability of instances due to random coupling selection. The impact of selective reporting, which was discussed in the main text, here is clearly visible in the insets of Fig. 7, which show the rather broad distribution of best approximation ratios obtained for each instance type and size. To stress this again here, in Fig. 8 we reproduce Fig. 3 from the main text, but with the addition of results from an optimized, GPU-based implementation of HUBO SA. This shows that not only the choice of baseline algorithm matters, but also the specific implementation can influence the results, and one should always strive to use the best available version of a given algorithm.

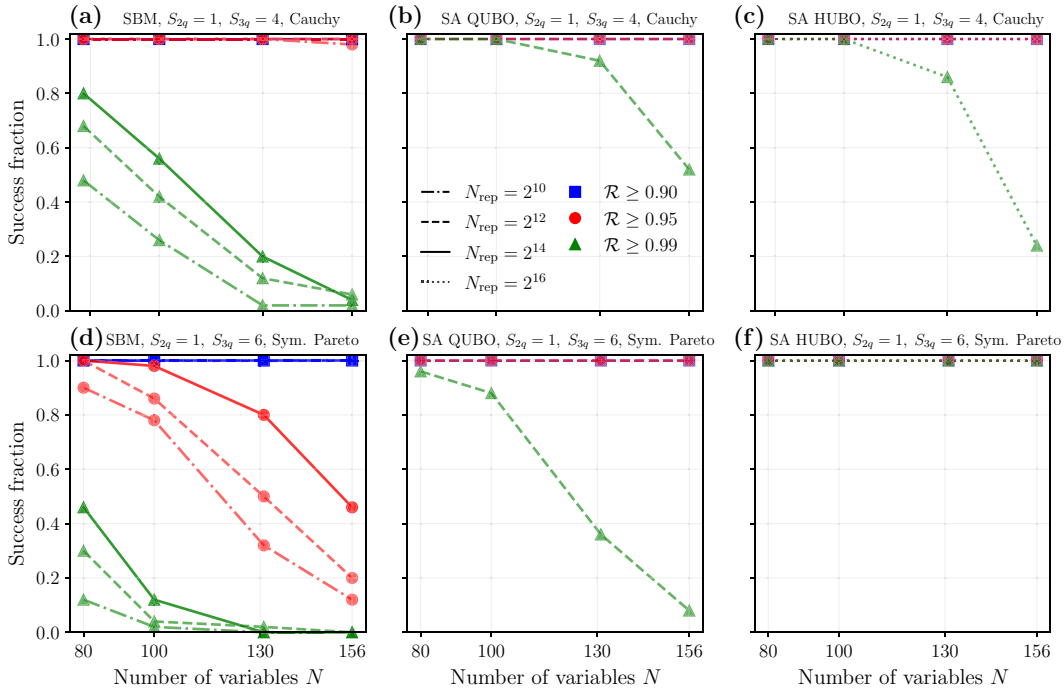

FIG. 6. Success fraction as a function of instance size, for 50 considered instances of each type. Panels (a),(d) show results for SBM, (b),(e) for SA in QUBO form, and (c),(f) for SA in HUBO form. The top row (a)–(c) corresponds to instances with $S_{2q} = 1$, $S_{3q} = 4$, Cauchy-distributed couplings, while the bottom row (d)–(f) to instances with $S_{2q} = 1$, $S_{3q} = 6$, symmetrized Pareto-distributed couplings. These results show that, when it comes to pure solution quality, these instances are challenging for simulated bifurcation. However, the success fraction can be significantly improved by increasing the number of replicas evolved in parallel, for which SBM is particularly well suited. SA, both in QUBO and HUBO form, performs significantly better, with the QUBO-native version being the best for Cauchy instances, and HUBO-native for Pareto instances.

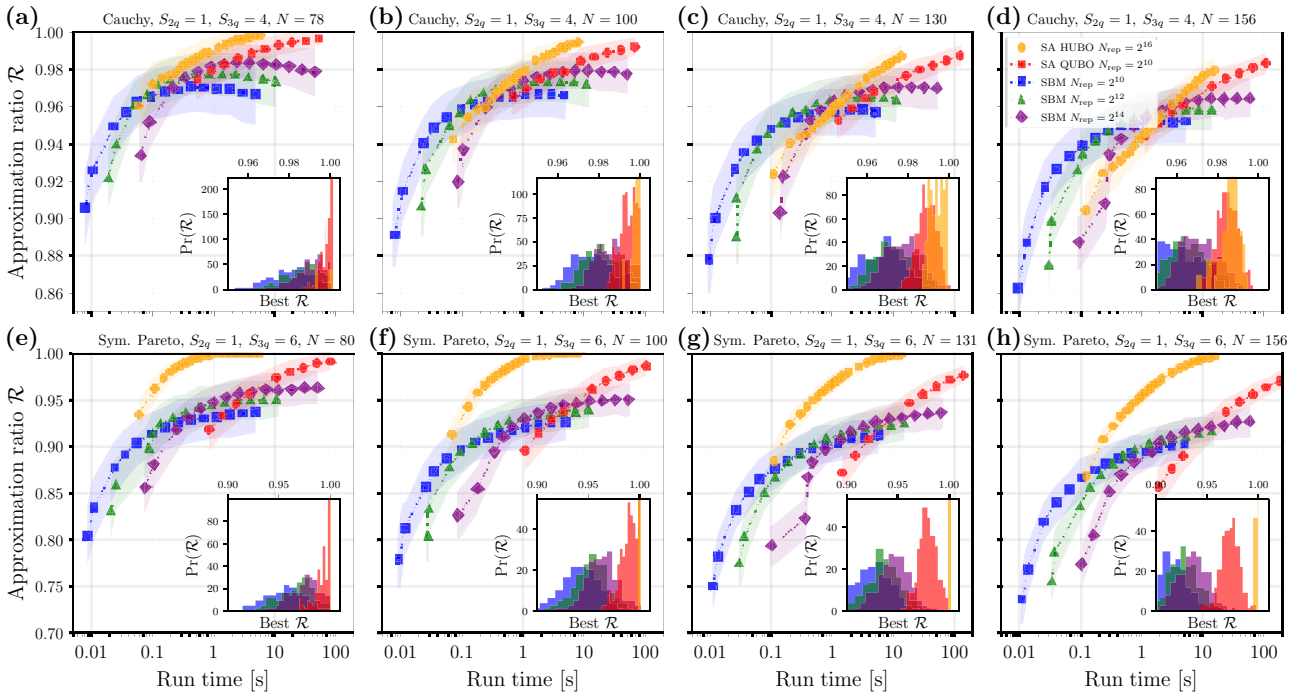

FIG. 7. Approximation ratio $\mathcal{R}$ as a function of run time, averaged over 50 instances and 10 (5) independent runs of SBM (SA), for each instance type and size. Shaded areas correspond to one standard deviation, and insets show the distribution of $\mathcal{R}$ obtained by aggregating best values for a given instance type and size. We can clearly see the biggest advantage of SBM over SA—its ability to efficiently evolve many replicas simultaneously, while keeping the run time low. In terms of the quality-run time tradeoff, it holds its ground even against the HUBO SA variant, which does not suffer overheads from rank reduction. In particular, the SBM is able, on average, to reach solutions of reasonable quality, $\mathcal{R} \approx 0.95$ for Cauchy instances and $\mathcal{R} \approx 0.90$ for Pareto instances, close to an order of magnitude faster than SA, except for large Pareto instances, which seem to be "easy" for the HUBO SA. This is promising, given that it is possible to extend the SBM so that it can handle higher-order interactions natively [70]. Finally, insets reveal a broad distribution of approximation ratios over random coupling selection for a fixed topology, which further highlights the importance of conducting a proper statistical analysis before drawing any conclusions.

APPENDIX D: SIMULATED ANNEALING FOR QUBO AND HUBO

Simulated annealing is one of the most widely used algorithms for finding approximate solutions to combinatorial optimization problems [24]. It relies on thermal fluctuations to probabilistically escape local minima during the search process and has inspired more sophisticated variants: parallel tempering [71], parallel annealing [72], and population annealing [73], among others [58].

In principle, SA can be applied to higher-order interactions of any order k, although it often becomes inefficient once the interaction order exceeds $k = 3$. While the method is guaranteed to converge to a local optimum, in practice convergence is limited by the chosen number of steps or run time. As a result, the gap from the global

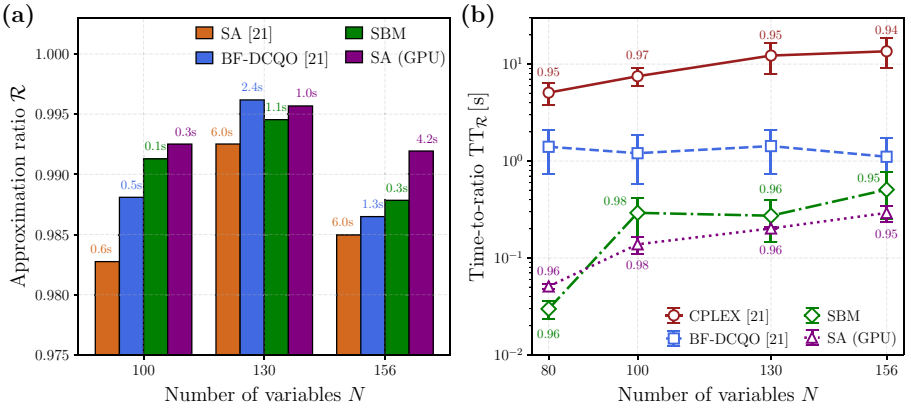

FIG. 8. Reproduction of Fig. 3 from the main text, with additional results from an optimized, GPU-based implementation of SA natively handling third-order interactions (cf. Algorithm 9). This simulation demonstrates that an optimized HUBO SA method is already sufficient to outperform the BF-DCQO hybrid approach of Ref. [22].

ALGORITHM 8. Simulated annealing for quadratic optimization.

Input:
$n \in \mathbb{N}$ – number of variables,
$J \in \mathbb{R}^{n\times n}$ – symmetric matrix of 2-body interactions,
$h \in \mathbb{R}^{n}$ – magnetic field,
$m \in \mathbb{N}$ – number of trajectories (starting from initial states) to process,
$num\_steps \in \mathbb{N}$ – number of steps for consequent temperatures,
$num\_passes \in \mathbb{N}$ – number of passes over variables per temperature,
T_0, T_1 – temperatures in the initial and the final step.
Output:
$s \in \{-1, 1\}^{m\times n}$ – spectrum of near-optimal states.

$s \leftarrow m$ pseudo-random states for n-variable problem ($\{-1, 1\}^{m\times n}$ matrix)
$\beta \leftarrow num\_steps$-long geometric sequence from $1/T_0$ to $1/T_1$
$\delta \leftarrow -2\, s \odot \left(sJ + h^{\top}\right)$ ▷ δ_{rv} is the energy change from flipping variable v in state r
for $step = 1, \ldots, num\_steps$ **do**
  for $pass \in \{1, \ldots, num\_passes\}$ **do**
    for $v \in \{1, \ldots, n\}$ **do**
      for $r \in \{1, \ldots, m\}$ **do** ▷ parallelized with no need for atomic operations
        if $\delta_{rv} < 0$ **or** **Rand**() $< \exp(-\delta_{rv}\, \beta_{step})$ **then** ▷ **Rand** yielding a pseudo-random number from (0, 1) range
          $s_{rv} \leftarrow -s_{rv}$
          $\delta_{rv} \leftarrow -\delta_{rv}$
          for $w \in \{1, \ldots, n\}$ **do**
            $\delta_{rw} \leftarrow \delta_{rw} - 4\, J_{vw}\, s_{rv}\, s_{rw}$ ▷ J_{vv} is assumed to be 0
          end for
        end if
      end for
    end for
  end for
end for
return s

optimum can remain large, especially for dense graphs [74]. Here we briefly outline the algorithm and provide pseudocode for both QUBO and HUBO problems.

The key advantage of SA is the simplicity of its concept. It operates on a batch of trajectories, often initialized as pseudorandom spin configurations. For all the scheduled temperatures T, we perform a fixed number of passes over all variables of all trajectories. For each variable, we compute δ—the energy change that would result from flipping its value to the opposite sign. If $\delta < 0$, the flip is accepted. Otherwise, it is accepted with probability $\exp(-\delta/T)$.

The key performance aspects of SA implementation involve a parallelization strategy, data structures to store matrices, and optimized recomputation of δ values. When the batch of trajectories is sufficiently large, parallelizing over trajectories is effective, and avoids any potential data races, regardless of the specific approach to δ computation. While J can be stored efficiently in either dense or sparse format, the higher-order interaction tensor K is typically very sparse, including in the instances used in this paper. The proposed implementation uses CUSPARSE data structures and the `cusparseSpMM` routine for sparse-dense matrix multiplications in Algorithm 9.

In the quadratic case, described as Algorithm 8, δ values can be updated incrementally after each accepted flip. This makes it feasible to keep the full δ_{rv} matrix in memory—it is computed once, and updated as flips occur. For cubic problems, this optimization is not efficient, since processing slices of K involves higher dimensionality than in the case of J. For that reason, we recompute the δ_r vector for each variable v. This approach guarantees that we always use δ values in sync with states as they are stored in the memory. As shown in Algorithm 9, focusing on one variable at time is more efficient than updating the entire matrix at once. This mitigates the overhead of more frequent δ recomputation.

For technical implementation purposes, we assume that two- and three-body interactions are presented in symmetric arrays: $J_{ij} = J_{ji}$ and $K_{ijk} = K_{jik} = K_{ikj}$ for $i,j,k \in \{1,n\}$, n being the number of variables. We also assume

ALGORITHM 9. Simulated annealing for third order HUBO optimization.

Input:

$n \in \mathbb{N}$ – number of variables,
$K \in \mathbb{R}^{n\times n\times n}$ – tensor of 3-body interactions, symmetric under any permutation of indices,
$J \in \mathbb{R}^{n\times n}$ – symmetric matrix of 2-body interactions,
$h \in \mathbb{R}^{n}$ – magnetic field,
$m \in \mathbb{N}$ – number of trajectories (starting from initial states) to process,
$num\_steps \in \mathbb{N}$ – number of steps for consequent temperatures,
$num\_passes \in \mathbb{N}$ – number of passes over variables per temperature,
T_0, T_1 – temperatures in the initial and the final step.

Output:

$s \in \{-1, 1\}^{m\times n}$ – spectrum of near-optimal states.

$s \leftarrow m$ pseudo-random states for n-variable problem ($\{-1, 1\}^{m\times n}$ matrix)
$\beta \leftarrow num\_steps$-long geometric sequence from $1/T_0$ to $1/T_1$
for $step = 1, \ldots, num\_steps$ **do**
  for $pass \in \{1, \ldots, num\_passes\}$ **do**
    for $v \in \{1, \ldots, n\}$ **do**
      $\delta \leftarrow -2\left(s^\top\right)_v \odot \left(\frac{1}{2}\left(s\,K_v \odot s\right)\,\mathbf{1}_n + s(J_v)^\top + h_v\right)$ ▷ K_v – v-th matrix of K. J_v, $\left(s^\top\right)_v$ – v-th rows
      for $r \in \{1, \ldots, m\}$ **do** ▷ parallelized with no need for atomic operations
        if $\delta_r < 0$ **or** **Rand**() $< \exp(-\delta_r\,\beta_{step})$ **then** ▷ **Rand** yielding a pseudo-random number from (0, 1) range
          $s_{rv} \leftarrow -s_{rv}$
        end if
      end for
    end for
  end for
end for
return s

TABLE I. D-Wave Advantage 4.1 parameters [75].

Property	Value
Model	Advantage, performance update
Graph size	P16
Number of qubits	5627
Number of couplers	40 279
Qubit temperature	15.4 ± 0.1 mK
M_{AFM} : Maximum mutual inductance for qubit pairs	1.647 pH
Quantum critical point for one-dimensional chains	1.391 GHz
L_q : Qubit inductance	372 pH
C_q : Qubit capacitance	119 fF
I_c : Qubit critical current	2.1 μA
Average single-qubit temperature	0.198
Ferromagnetic-problem freezeout	0.064
Single-qubit freezeout	0.612
Φ^i_{CCJJ} : Initial (at $s = 0$) external flux on compound Josephson junctions	$-0.621\Phi_0$
Φ^f_{CCJJ} : Final (at $s = 1$) external flux on compound Josephson junctions	$-0.717\Phi_0$
Readout time range	17.0 to 235.0 μs
Programming time	∼14 100 μs
QPU delay time per sample	20.5 μs
Readout error rate	≤0.001

TABLE II. D-Wave Advantage2 (Z12) parameters.

Property	Value
Model	Advantage2
Graph size	Z12
Number of qubits	4593
Number of couplers	41 796
Qubit temperature	20 ± 1.0 mK
M_{AFM} : Maximum mutual inductance for qubit pairs	2.113 pH
Quantum critical point for one-dimensional chains	2.308 GHz
L_q : Qubit inductance	120 pH
C_q : Qubit capacitance	147 fF
I_c : Qubit critical current	4.75 μA
Average single-qubit temperature	0.112
Ferromagnetic-problem freezeout	0.008
Single-qubit freezeout	0.684
Φ^i_{CCJJ} : Initial (at $s = 0$) external flux on compound Josephson junctions	$-0.694\Phi_0$
Φ^f_{CCJJ} : Final (at $s = 1$) external flux on compound Josephson junctions	$-0.760\Phi_0$
Readout time range	17.0 to 101.0 μs
Programming time	∼33 600 μs
QPU delay time per sample	60.6 μs
Readout error rate	≤0.001

$J_{ii} = 0$ and $K_{iik} = 0$, since such interactions can be represented by lower-order terms. This makes it possible to rewrite the cost function from Eq. (3) as

$$P(s) = \left(\frac{1}{6}\left(\sum_{i=1}^{n}(K_i s)s_i\right) + \frac{1}{2}Js + h\right)^{\top} s. \tag{D1}$$

This formulation underlies the δ_{rv} update cost formulas presented in Algorithm 8 and δ_r for Algorithm 9.

APPENDIX E: REPRODUCIBILITY

To ensure the reproducibility of our results, we provide a GitHub repository [61]. This contains structured HUBO instances for three-body Ising models (along with generation code), a Python client for simulated bifurcation machine demonstrations, and a CUDA-optimized Julia implementation of simulated annealing. The repository also includes a Python program for executing quantum circuits on IBM Brisbane, a Julia solver for Simon's problem with various oracle configurations, and all data and scripts used to generate the paper's figures.

APPENDIX F: DETAILS OF CLOUD QUANTUM COMPUTING DEMONSTRATIONS

The quantum annealing data presented in Figs. 1 and 5 were obtained between May 15 and July 14, 2025 on D-Wave Advantage 4.1 na-west-1, which consisted of 5627 qubits (note that in December 2025 qubit #840 was removed from that device). The configuration data are presented in Table I [75].

The quantum annealing data presented in Fig. 4 were obtained on September 27, 2025 on D-Wave Advantage2 1.6. The configuration data are presented in Table II [76].

APPENDIX G: HARDWARE

All classical benchmarks in this work were performed on a workstation equipped with dual Intel(R) Xeon(R) Platinum 8462Y+ CPUs, 4 NVIDIA H100 SXM GPUs, and 1TB of RAM. All results were obtained using a single GPU, unless stated otherwise.

[1] R. P. Feynman, Simulating physics with computers, Int. J. Theor. Phys. **21**, 467 (1982).

[2] D. Deutsch, Quantum theory, the Church-Turing principle and the universal quantum computer, Proc. R. Soc. Lond. A **400**, 97 (1985).

[3] IonQ, IonQ roadmap (2025), https://ionq.com/blog/ionqs-accelerated-roadmap-turning-quantum-ambition-into-reality.

[4] IBM, IBM quantum roadmap (2025), https://www.ibm.com/roadmaps/quantum/.

[5] Quantinuum, Quantinuum roadmap (2025), https://www.quantinuum.com/press-releases/quantinuum-unveils-accelerated-roadmap-to-achieve-universal-fault-tolerant-quantum-computing-by-2030.

[6] D-Wave, The D-Wave *Clarity* roadmap (2025), https://www.dwavesys.com/media/xvjpraig/clarity-roadmap_digital_v2.pdf accessed: 2025-01-30.

[7] D. Castelvecchi, Quantum computers will finally be useful: What's behind the revolution, Nature **650**, 24 (2026).

[8] H.-Y. Huang, S. Choi, J. R. McClean, and J. Preskill, The vast world of quantum advantage, arXiv:2508.05720.

[9] D. R. Simon, On the power of quantum computation, SIAM J. Comput. **26**, 1474 (1997).

[10] P. Shor, in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, NW Washington, DC United States, 1994), p. 124.

[11] L. J. Stockmeyer, The polynomial-time hierarchy, Theor. Comput. Sci. **3**, 1 (1976).

[12] O. Lanes, M. Beji, A. D. Corcoles, C. Dalyac, J. M. Gambetta, L. Henriet, A. Javadi-Abhari, A. Kandala, A. Mezzacapo, C. Porter, *et al.*, A framework for quantum advantage, arXiv:2506.20658.

[13] A. Abbas *et al.*, Challenges and opportunities in quantum optimization, Nat. Rev. Phys. **6**, 718 (2024).

[14] F. Meier and H. Yamasaki, Energy-consumption advantage of quantum computation, PRX Energy **4**, 023008 (2025).

[15] K. Bertels, E. Turki, T. Sarac, A. Sarkar, and I. Ashraf, Quantum computing—A new scientific revolution in the making, arXiv:2106.11840.

[16] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell *et al.*, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).

[17] Y. A. Liu, X. L. Liu, F. N. Li, H. Fu, Y. Yang, J. Song, P. Zhao, Z. Wang, D. Peng, H. Chen *et al.*, in *Proc.—Int. Conf. High Perform. Comput. Netw. Storage Anal., SC '21* (Association for Computing Machinery, New York, 2021).

[18] F. Pan, K. Chen, and P. Zhang, Solving the sampling problem of the Sycamore quantum circuits, Phys. Rev. Lett. **129**, 090502 (2022).

[19] A. Morvan, B. Villalonga, X. Mi, S. Mandrà, A. Bengtsson, P. V. Klimov, Z. Chen, S. Hong, C. Erickson, I. K. Drozdov *et al.*, Phase transitions in random circuit sampling, Nature **634**, 328 (2024).

[20] H. Munoz-Bauza and D. Lidar, Scaling advantage in approximate optimization with quantum annealing, Phys. Rev. Lett. **134**, 160601 (2025).

[21] P. Singkanipa, V. Kasatkin, Z. Zhou, G. Quiroz, and D. A. Lidar, Demonstration of algorithmic quantum speedup for an abelian hidden subgroup problem, Phys. Rev. X **15**, 021082 (2025).

[22] P. Chandarana, A. G. Cadavid, S. V. Romero, A. Simen, E. Solano, and N. N. Hegade, Runtime quantum advantage with digital quantum optimization, arXiv:2505.08663.

[23] Z. Zhu, A. J. Ochoa, and H. G. Katzgraber, Efficient cluster algorithm for spin glasses in any space dimension, Phys. Rev. Lett. **115**, 077201 (2015).

[24] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by simulated annealing, Science **220**, 671 (1983).

[25] CPLEX, IBM ILOG, users manual for CPLEX (2024), https://www.ibm.com/docs/en/icos/22.1.1?topic=optimizers-users-manual-cplex.

[26] J. Pawłowski, P. Tarasiuk, J. Tuziemski, Ł. Pawela, and B. Gardas, Closing the quantum-classical scaling gap in approximate optimization, arXiv:2505.22514.

[27] S. Boettcher, Analysis of the relation between quadratic unconstrained binary optimization and the spin-glass ground-state problem, Phys. Rev. Res. **1**, 033142 (2019).

[28] G. F. Newell and E. W. Montroll, On the theory of the Ising model of ferromagnetism, Rev. Mod. Phys. **25**, 353 (1953).

[29] M. Mohseni, M. M. Rams, S. V. Isakov, D. Eppens, S. Pielawa, J. Strumpfer, S. Boixo, and H. Neven, Sampling diverse near-optimal solutions via algorithmic quantum annealing, Phys. Rev. E **108**, 065303 (2023).

[30] NVIDIA, Nsight systems user guide (2025), https://docs.nvidia.com/nsight-systems/UserGuide/index.html.

[31] Intel, Intel vtune profiler (2025), https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.html.

[32] AMD, AMD profiler (2025), https://rocm.docs.amd.com/en/latest/.

[33] D-Wave, D-wave QPU solver parameters (2025), https://docs.dwavequantum.com/en/latest/quantum_research/solver_parameters.html.

[34] H. Munoz Bauza and D. Lidar, Scaling advantage in approximate optimization with quantum annealing—Spin-glass instances, Phys. Rev. Lett. **134**, 160601 (2025).

[35] D-Wave, D-Wave operation and timing (2025), https://docs.dwavequantum.com/en/latest/quantum_research/operation_timing.html.

[36] H. Goto, Bifurcation-based adiabatic quantum computation with a nonlinear oscillator network, Sci. Rep. **6**, 21686 (2016).

[37] H. Goto, K. Endo, M. Suzuki, Y. Sakai, T. Kanao, Y. Hamakawa, R. Hidaka, M. Yamasaki, and K. Tatsumura, High-performance combinatorial optimization based on classical mechanics, Sci. Adv. **7**, eabe7953 (2021).

[38] T. Koch *et al.*, Quantum optimization benchmark library—The intractable decathlon, arXiv:2504.03832.

[39] H. Goto, K. Tatsumura, and A. R. Dixon, Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems, Sci. Adv. **5**, eaav2372 (2019).

[40] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, Defining and detecting quantum speedup, Science **345**, 420 (2014).

[41] E. Alessandroni, S. Ramos-Calderer, I. Roth, E. Traversi, and L. Aolita, Alleviating the quantum big-*M* problem, npj Quantum Inf. **11**, 125 (2025).

[42] M. Larocca, S. Thanasilp, S. Wang, K. Sharma, J. Biamonte, P. J. Coles, L. Cincio, J. R. McClean, Z. Holmes, and M. Cerezo, Barren plateaus in variational quantum computing, Nat. Rev. Phys. **7**, 174 (2025).

[43] S. Schulz, D. Willsch, and K. Michielsen, Learning-driven annealing with adaptive Hamiltonian modification for solving large-scale problems on quantum devices, Quantum **9**, 1898 (2025).

[44] R. Kothari, T. Lee, M. Szegedy, and I. Newman, *Query Complexity*, G—Reference, Information and Interdisciplinary Subjects Series (World Scientific, 2025).

[45] A. Ambainis, in *International Congress of Mathematicians 2018 (ICM 2018)* (World Scientific, 2018).

[46] D. Deutsch and R. Jozsa, Rapid solution of problems by quantum computation, Proc. R. Soc. A **439**, 553 (1992).

[47] L. K. Grover, in *Proc. Annu. ACM Symp. Theory Comput.* (Association for Computing Machinery, New York NY, United States, 1996), p. 212.

[48] E. M. Stoudenmire and X. Waintal, Opening the black box inside Grover's algorithm, Phys. Rev. X **14**, 041029 (2024).

[49] T. Hoefler, T. Häner, and M. Troyer, Disentangling hype from practicality: On realistically achieving quantum advantage, Commun. ACM **66**, 82 (2023).
[50] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, Quantum computing with Qiskit, arXiv:2405.08810.
[51] IBM, IBM quantum compute resources (2025), https://quantum.cloud.ibm.com/computers.
[52] S. Chowdhury, N. A. Aadit, A. Grimaldi, E. Raimondo, A. Raut, P. A. Lott, J. H. Mentink, M. M. Rams, F. Ricci-Tersenghi, M. Chiappini, L. S. Theogarajan, T. Srimani, G. Finocchio, M. Mohseni, and K. Y. Camsari, Pushing the boundary of quantum advantage in hard combinatorial optimization with probabilistic computers, Nature Commun. **16**, 9193 (2025).
[53] J. Alman, R. Duan, V. V. Williams, Y. Xu, Z. Xu, and R. Zhou, More asymmetry yields faster matrix multiplication, arXiv:2404.16349.
[54] X. Yi, A study of performance programming of CPU, GPU accelerated computers and SIMD architecture, arXiv:2409.10661.
[55] R. Kastner, J. Matai, and S. Neuendorffer, Parallel programming for FPGAs, arXiv:1805.03648.
[56] J. Hou, A. Barzegar, and H. G. Katzgraber, Direct comparison of stochastic driven nonlinear dynamical systems for combinatorial optimization, Phys. Rev. E **112**, 035301 (2025).
[57] T. Aonishi, T. Nagasawa, T. Koizumi, M. D. S. H. Gunathilaka, K. Mimura, M. Okada, S. Kako, and Y. Yamamoto, Highly versatile FPGA-implemented cyber coherent Ising machine, IEEE Access **12**, 175843 (2024).
[58] J. Pawłowski, J. Tuziemski, P. Tarasiuk, A. Przybysz, R. Adamski, K. Hendzel, Ł. Pawela, and B. Gardas, VeloxQ: A fast and efficient QUBO solver, arXiv:2501.19221.
[59] D. Delahaye, S. Chaimatanan, and M. Mongeau, in *Handbook of Metaheuristics*, edited by M. Gendreau and J.-Y. Potvin (Springer International Publishing, Cham, 2019), p. 1.
[60] The authors did not respond to our request to share their HUBO instances.
[61] J. Tuziemski, J. Pawłowski, P. Tarasiuk, L. Pawela, and B. Gardas, Recent quantum run-time (dis)advantages—Code repository, https://github.com/quantumz-io/quanutm-runtime-disadvantage (2025), GitHub repository.
[62] P. Farré, E. Ordog, K. Chern, and C. C. McGeoch, Comparing quantum annealing and BF-DCQO, arXiv:2509.14358.
[63] IBM, IBM quantum platform—Processor types (2025), https://quantum.cloud.ibm.com/docs/en/guides/processor-types.
[64] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, Validating quantum computers using randomized model circuits, Phys. Rev. A **100**, 032328 (2019).
[65] A. Wack, H. Paik, A. Javadi-Abhari, P. Jurcevic, I. Faro, J. M. Gambetta, and J. M. Chow, Quality, speed, and scale: Three key attributes to measure the performance of near-term quantum computers, arXiv:2110.14108.
[66] T. Zhang and J. Han, in *2023 IEEE 23rd International Conference on Nanotechnology (NANO)* (IEEE, Jeju-do Island, Korea, 2023), p. 715.
[67] J. Fulman, Random matrix theory over finite fields: A survey, arXiv:0003195.
[68] C. Chamberland, G. Zhu, T. J. Yoder, J. B. Hertzberg, and A. W. Cross, Topological and subsystem codes on low-degree graphs with flag qubits, Phys. Rev. X **10**, 011022 (2020).
[69] N. Dattani, Quadratization in discrete optimization and quantum mechanics, arXiv:1901.04405.
[70] T. Kanao and H. Goto, Simulated bifurcation for higher-order cost functions, Appl. Phys. Express **16**, 014501 (2023).
[71] D. J. Earl and M. W. Deem, Parallel tempering: Theory, applications, and new perspectives, Phys. Chem. Chem. Phys. **7**, 3910 (2005).
[72] A. Russkov, R. Chulkevich, and L. N. Shchur, Algorithm for replica redistribution in an implementation of the population annealing method on a hybrid supercomputer architecture, Comput. Phys. Commun. **261**, 107786 (2021).
[73] J. Machta, Population annealing with weighted averages: A Monte Carlo method for rough free-energy landscapes, Phys. Rev. E **82**, 026704 (2010).
[74] S. V. Romero, A.-M. Visuri, A. G. Cadavid, E. Solano, and N. N. Hegade, Bias-field digitized counterdiabatic quantum algorithm for higher-order binary optimization, Commun. Phys. **8**, 348 (2025).
[75] D.-W. S. Inc., Advantage_system4.1 solver properties (2025), https://docs.dwavequantum.com/en/latest/quantum_research/solver_properties_specific.html#advantage-system4-1, accessed: 2025-11-07.
[76] D-Wave, QPU hardware and properties (2026), https://docs.dwavequantum.com/en/latest/quantum_research/index_about.html.