### Recent quantum runtime (dis)advantages

J. Tuziemski, J. Pawłowski, <sup>2,1</sup> P. Tarasiuk, <sup>1</sup> L. Pawela, <sup>3,1</sup> and B. Gardas<sup>3</sup>

<sup>1</sup> Quantumz.io Sp. zo.o., Puławska 12/3, 02-566 Warsaw

<sup>2</sup> Institute of Theoretical Physics, Faculty of Fundamental Problems of Technology, Wrocław University of Science and Technology, 50-370 Wrocław, Poland

<sup>3</sup> Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Bałtycka 5, 44-100 Gliwice, Poland

We (re)evaluate recent claims of quantum advantage in annealing- and gate-based algorithms, testing whether reported speedups survive rigorous end-to-end runtime definitions and comparison against strong classical baselines. Conventional analyses often omit substantial overhead (readout, transpilation, thermalization, etc.) yielding biased assessments. While excluding seemingly not important parts of the simulation may seem reasonable, on most current quantum hardware a clean separation between "pure compute" and "overhead" cannot be experimentally justified. This may distort "supremacy" results. In contrast, for most classical hardware total time  $\approx$  compute + a weakly varying constant leading to robust claims. We scrutinize two important milestones: (1) quantum annealing for approximate QUBO [PRL 134, 160601 (2025)], which uses a sensible timeto- $\varepsilon$  metric but proxies runtime by the annealing time (non-measurable); (2) a restricted Simon's problem [PRX 15, 021082 (2025)], whose advantageous scaling in oracle calls is undisputed; yet, as we demonstrate, estimated runtime of the quantum experiment is  $\sim 100 \times$  slower than a tuned classical baseline. Finally, we show that recently claimed "runtime advantage" of the BF-DCQO hybrid algorithm (arXiv:2505.08663) does not withstand rigorous benchmarking. Therefore, we conclude that runtime-based supremacy remains elusive on NISQ hardware, and credible claims require a careful time accounting with a proper reference selections, and an adequate metric.

#### I. INTRODUCTION

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

Quantum advantage can be defined in various ways, such as in computation or metrology [7]. One approach relies on complexity-theoretic arguments proving better scaling than classical algorithms, as in Simon's algorithm [8], which shows exponential query separation, or Shor's algorithm [9], based on the presumed hardness of factoring [10]. While generic speedups for NP-hard problems are unlikely, restricted instances may allow improvements [11], motivating studies of heuristic quantum algorithms [12]. These may provide complexity statements for specific cases or empirical advantages, given appropriate hardware. Other notions, such as energetic advantage, have also been discussed [13].

As quantum devices improve, attention turns to experimental verification [14]. Early claims based on Random Circuit Sampling [15] were challenged by classical simulation [16, 17], though recent results seem intractable [18]. Further claims include favorable time-to-solution scaling for QUBO optimization on D-Wave annealers [19], query complexity advantage in restricted Simon's problem [20], and runtime benefits for a hybrid classical-quantum al-

gorithm [21]. These claims are significant as they reflect the growing technological maturity of quantum computing. However, a key question remains: do these reported advantages translate into actual runtime improvements — a critical factor for practical adoption? In this work, we reassess various strategies employed in these experiments and show that the answer is negative: no runtime advantage is observed when runtime is properly measured, and an appropriate classical baseline is selected.

In particular, in Sec. II, we examine the findings of Ref. [19], which reported that the runtime of a quantum annealing-based approximate QUBO scaled more favorably than the best classical reference algorithm (therein PT-ICM [22]). We discuss challenges in defining total runtime for such experiments and show that, under a robust definition, no advantage remains. Next, we examine the results of Ref. [21], where the digitized counterdiabatic quantum optimization algorithm, a gate-based approach, was reported to achieve a runtime advantage over classical Simulated Annealing (SA) [23] and CPLEX [24]. We identify several issues with the definitions of classical and quantum runtime used therein, which ultimately eliminates this advantage altogether.

In Sec. III, we address the reported query complexity advantage for a restricted Simon's problem [20] on noisy gate-based IBM quantum computers. We investigate whether fewer oracle calls lead to shorter runtimes and find that, although the classical algorithm indeed exhibits exponentially worse scaling in oracle calls, in the regime considered in [20] its runtime is actually two orders of magnitude shorter than the quantum algorithm.

Finally, in Sec. IV, we discuss the importance of selecting an appropriate classical reference algorithm and,

focusing on runtime as the key metric, outline essential criteria for this choice. We revisit the reference selection made in [21] and argue that a proper choice already eliminates any quantum-classical runtime advantage. This scenario closely parallels a recent discussion in [25], where it was shown that an appropriately chosen classical baseline can demonstrate robust scaling and effectively remove claims of quantum supremacy in discrete approximate optimization (cf. Ref. [19]) under operationally meaningful conditions. For completeness, we also verified the performance of a D-Wave quantum annealer on the problems studied in [21], and found that adiabatic quantum computing offers no runtime advantage either. On the contrary, these instances appear to be particularly challenging for the D-Wave machine, as they require a complex embedding that ultimately degrades solution quality.

## II. QUANTUM ADVANTAGE IN ALGORITHM RUNTIME

In this section, we address the problem of defining runtime in a manner suitable for quantum computation and investigate how sensitive claims of runtime advantage are to changes in this definition. Our focus is on approximate quadratic unconstrained binary optimization (QUBO), where we assume that the problem has been mapped to an instance of the Ising model, whose ground state encodes the solution [26]. 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  $J_{ij}$  are coupling strengths between spins, and  $h_i$  are local magnetic fields [27].

In the case of approximate optimization the standard figure of merit is time-to-epsilon,  $TT\varepsilon$  (sometimes used interchangeably with time-to-approximation-ratio  $TT_{\mathcal{R}}$  [28]), which quantifies the time needed to obtain a solution within  $\varepsilon$  fraction of the ground state energy,

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

where  $t_f$  is the time taken to generate the solution and  $p_{E \leq E_0 + \varepsilon | E_0|}$  is the probability of finding a solution with energy E, within  $\varepsilon$  optimality gap of the true ground state energy  $E_0$  in cases when the solution is known, or a reference energy otherwise (e.g. the best energy found by a reference solver). In most cases the probability distribution  $p_{E \leq E_0 + \varepsilon | E_0|}$  remains unknown, and it needs to be replaced by an estimator, whose computation involves firstly a non-trivial optimization of the solver settings, and subsequently a number of independent runs of the solver with optimal parameters, from which the estimate

is established. The employed function of success probability estimates the number of times an algorithm needs to be run to find a solution within the desired optimality.

Investigation of  $TT\varepsilon$  for specific classes of optimization problems, or its scaling with the problem size serves as a standard comparison tool between different heuristic algorithms, including the quantum ones [19]. However, it is clear from the above definition that a proper definition of the runtime  $t_f$  is absolutely crucial for fair and unbiased comparison.

In most cases defining and measuring runtime for classical algorithms poses no problem. The set of tools for straightforward runtime measurements is also available [29–31]. On the other hand, there are two fundamental challenges concerning quantum runtime. The first one concerns the proper definition of quantum runtime, as one must determine which of various quantum computation stages should be included in the total runtime. Secondly, one needs to establish a robust method of measuring times of those stages. In the subsections these issues would be addressed separately for quantum annealing and gate based devices.

#### A. Runtime for annealing quantum devices

Solution of an optimization problem on a quantum annealer consists of the following stages. Firstly, an optimization problem must be loaded into the annealing device. This involves two steps: embedding, which maps a given instance of the Ising model onto the device topology, and programming, which sets the physical properties of the device to realize the embedded problem. Subsequently, the quantum annealing protocol takes place. Here it is important to note that the duration of the annealing is an input parameter to the protocol, and, at least using the standard cloud access, it is not possible to measure it independently [32]. Due to the probabilistic nature of the annealing protocol, multiple runs are required to increase the probability of obtaining a highquality solution. These annealing runs are executed sequentially, and are interrupted by the two processes. The first is readout, which allows to obtain information about the final state of each protocol, whereas the second is thermalization, which restores the device to its initial state. Both processes have their intrinsic runtimes.

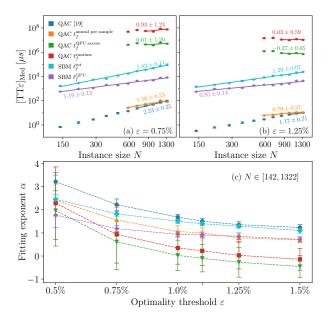
It seems reasonable, therefore, that for quantum annealers, the total runtime should be defined as the sum of all annealing runs, including readout, thermalization, and programming times. This is not a common practice; for instance, in [19], runtime was defined (and not measured) solely as the annealing time, excluding other contributions. Our aim is now to examine how this decision influenced the reported scaling of  $TT\varepsilon$ , and how the scaling changes when the additional stages of the annealing protocol are taken into account. To this end we repeated the experiments using the same instances as in Refs. [19, 33]. When  $t_f$  is set to the annealing time per

sample, we reproduce the results of [19], confirming that the methodology was implemented correctly. We then investigated how the  $\mathrm{TT}\varepsilon$  scaling changes when  $t_f$  is taken as the total QPU time, which includes programming, annealing, readout, and thermalization times, as reported by the cloud interface [34]. To further validate these results, we measured the usage time of the cloud interface itself, which also accounts for access overheads.

In both cases, the results indicate that the runtime is almost constant. The analysis of the scaling exponent support this findings as the uncertainties of the fits are too large to conclude that the exponents are non-zero. Analysis of the different runtime components shows that the dominant contribution to the total runtime comes from the readout, which is on the order of  $200\mu s$  per annealing run. This implies that measuring the final quantum state of the D-Wave quantum annealer is up to two orders of magnitude longer than executing the quantum evolution required to prepare this state, as the annealing time in this experiment varies from  $0.5\mu s$  to  $27\mu s$ . This discrepancy, and the difficulties it poses for investigating  $TT\varepsilon$  scaling, motivated the exclusion of readout time in [19]. However, this argument is not entirely convincing. Measurements are an inescapable part of quantum computing, whether analogue or digital. Information encoded in a quantum state must be read from a quantum device, as only the measurement results provide the solution to a computational problem.

For reference we compared scaling of the quantum annealing protocol to a classical algorithm, Simulated Bifurcation Machine (SBM [25, 35, 36]), cf. App. A1. Since it is a classical algorithm, the runtime can be measured directly and split into components, in particular the pure computation time, in our case performed on GPU –  $t_f^{\text{GPU}}$ , and the overhead time  $t_f^{\rm overhead}$ , which is dominated by the data transfer between CPU and GPU and hyperparameter tuning as exaplained in Appendix A1. We denote the total time from the end user perspective as  $t_f^{\text{tot}} = t_f^{\text{GPU}} + t_f^{\text{overhead}}$ . In Fig. 1 we show that the impact of overhead on the scaling of  $[TT\varepsilon]_{Med}$  is certainly non-negligible, albeit much less pronounced than in the quantum case. Our experiments show that runtime  $\approx$ "compute" + "a weakly varying constant" does not holdfor quantum annealers. In contrast, this approximation remains valid for classical hardware such as GPUs, which explains why scaling behavior is preserved across different methodologies for measuring time.

To properly address this issue, all stages of computation would need to be included. On such time scales, however, the runtime becomes nearly constant for the problem sizes considered. Conclusive scaling results for current annealing devices would therefore require experiments with annealing times exceeding  $200\mu s$  and with substantially more than 5000 qubits.



**FIG. 1.** Time-to-epsilon  $[TT\varepsilon]_{Med}$  scaling with the size N of the Sidon-28 instances, for values of  $\varepsilon = 0.75\%$  in panel (a), and  $\varepsilon = 1.25\%$  in panel (b). Results for QAC (blue) solver are reproduced from Ref. [19], courtesy of the authors. Remaining QAC data concerns experiments performed by us on the instances form Ref. [19]. With the same definition of runtime the results (orange) are consistent with [19], what confirms correctness of methodology implementation. In addition  $[\mathrm{TT}\varepsilon]_{\mathrm{Med}}$  defined using the complete QPU access time  $t_{\scriptscriptstyle f}^{
m QPU\,access}$  as reported by D-Wave's cloud interface (green), as well as runtime measured with cloud access  $t_{\it f}^{\rm runtime}$  (red) are presented. Solid lines are power-law fits  $[{\rm TT}\varepsilon]_{\rm Med} \propto N^{\alpha}$ , with the corresponding exponents  $\alpha$  shown on the plot. The instance size range spanned by the lines denote which data points were used for the respective fits. Both plots clearly demonstrate that with the proper runtime definition  $[TT\varepsilon]_{Med}$ is constant for considered problem sizes. For comparison data for Simulated Bifurcation Machine (SBM) solver[25, 35–37] with  $[TT\varepsilon]_{Med}$  computed using 1 GPU and total runtime  $t_f^{tot}$ (cyan), as well as pure GPU runtime  $t_f^{\rm GPU}$  (magenta) are presented. In this case the scaling of  $[TT\varepsilon]_{Med}$  is more robust with respect to different runtime definitions, which is generally the case for classical algorithms. The bottom panel (c) shows detailed data of the fitting exponent and its uncertainty for values of  $\varepsilon \in \{0.5, 0.75, 1.00, 1.10, 1.25, 1.5\}\%$ 

### B. Runtime for digital quantum devices

For digital quantum computers, one can distinguish computation stages analogous to those discussed for quantum annealers. The first stage is preprocessing, which involves translating a quantum circuit into the topology and native gate set of a chosen device, a process known as transpilation. The transpiled circuit is then executed multiple times (shots), with consecutive runs separated by readout and thermalization. A proper definition of runtime for digital quantum computers should therefore account for all these processes. This perspective was also advocated in [38], which describes a procedure

for measuring their durations on IBM devices.

With this definition of quantum runtime, we revisit the results of Ref. [21]. Their digitized counter-diabatic quantum optimization algorithm was reported to exhibit a runtime advantage over Simulated Annealing (SA [23]) and IBM's proprietary CPLEX solver [24] for a class of Higher-Order Unconstrained Binary Optimization (HUBO) problems. These are probles whose cost function includes multi-variable terms of order N=3, thereby generalizing the Ising model of Eq. (1),

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

The choice of these two particular classical references is discussed in Sec. IV. Here, we focus solely on the employed definition of runtime, as well as its measurement. The algorithm considered in [21] is a hybrid quantum-classical algorithm consisting of two runs of SA and a single run of the quantum routine. This methods begins with a single run of SA providing an initial guess for quantum routine. The state produced by the quantum routine is then fed into a second run of SA, which aims to mitigate possible readout errors.

The total runtime thus comprises both classical and quantum components. The classical runtime was estimated based on the time required to perform a single pass of SA. Therefore, the reported times are not the result of a physical measurement, but rather estimates derived from the number of passes performed. Curiously, the authors did not account for overheads associated with SA, which, if included, would affect their conclusions, as the overheads provided by the authors are on the order of 1.6s. We emphasize that a robust protocol for measuring classical computational time would allow for avoiding such ambiguities. The quantum runtime was estimated under the assumption that the quantum devices used in the experiments can execute 10<sup>4</sup> circuit repetitions per second. This estimate, however, excludes overheads such as programming and transpilation time, as discussed above. For such experiments a better methodology, specifically designed for IBM quantum devices, could be applied, cf. Ref. [38].

To conclude, there are a number of severe problems with the definition and measurement of runtime for both classical and quantum algorithms, which constitute the digitized counter-diabatic quantum optimization algorithm. Due to these concerns, and the lack of an appropriate metric, we conclude that at the present stage the runtime advantage claimed in Ref. [21] is unfounded. Furthermore, even if such a non-standard approach to measuring runtime were to be justified, the claimed "supremacy" would not withstand comparison against a strong classical reference. We explicitly address this point in Sec. IV, where we highlight further methodological problems in Ref. [21]. For instance, applying their methodology to the same class of problem instances shows that other classical methods (and even optimized

SA) can achieve comparable or superior results, thereby undermining the BF-DCQO supremacy claim. A detailed analysis is provided in App. A3.

#### C. Runtime for classical-quantum algorithms

Heuristic algorithms generally rely on a set of hyperparameters that must be tuned to achieve optimal performance [39]. Such tuning is often computationally expensive [40, 41]. If the reported runtime of a solver excludes the cost of hyperparameter optimization, it cannot be regarded as evidence of a runtime advantage. Unless the chosen hyperparameters are demonstrably problem-independent, or the tuning effort is uniform across all instances. Furthermore, in hybrid quantum–classical approaches, the reported runtime should also include the overhead associated with transferring data between the classical host and the quantum device. This practice is standard in classical–classical settings, such as combined CPU–GPU computations, and there is no justification for treating quantum–classical methods differently.

In this context, let us discuss the results of Ref. [42], where a new quantum-classical solver for large-scale spin glasses has been proposed, and it was claimed that it outperforms other quantum and classical algorithms both in terms of solution quality and the runtime. The solver is based on a sequential process, in which at every step the algorithm needs a set of hyperparameters. However, the runtimes presented by the authors do not explicitly include execution times needed to obtain those hyperparameters. Moreover, a tuning subroutine for hyperparameters is not described. Therefore, while the quality of solutions is impressive, the runtime advantage claim is not justified. Again, a proper metric in this context is  $TT\varepsilon$ , with all the necessary time components included, and optimization performed [39].

# 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 [43, 44]. 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 [8, 45, 46]. However, once the restriction on limited problem knowledge is lifted, for instance, 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, e.g., the recent discussion in [47] 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 the claim that an algorithm requiring fewer oracle queries is more efficient can 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 runtime 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 [48]. Because of these considerations, it has been argued that a practical quantum advantage can be achieved only by algorithms offering super-quadratic speedups [48]. Runtime 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 [20]. In the original problem, the task is to find a hidden period (a bitstring) encoded in an unknown 2to-1 function. In contrast, the authors of [20] study Nbit 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 as  $N_w \equiv \sum_j {N \choose j}$ , where N is the total number of bits, and w the maximal allowed Hamming weight. Theoretical considerations are presented that suggest possibility of a quantum advantage for such defined problem even for noise, uncorrected quantum computers. The exponential advantage manifest itself in a polylogarithmic scaling of the score function as a function of the total number of possible periods (the classical model scales polynomial with this parameter). The authors verify then these claims experimentally. 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 runtime of the algorithm?

To compare the runtimes of quantum and classical algorithms for Simon's problem, we implemented a classical brute-force algorithm running on a single GPU. In Ref. [19], 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 NISO device. A detailed discussion of this compiler can be found in Ref. [19]. For a fair comparison, we implemented a classical oracle corresponding to the quantum oracle described in Ref. [19], and we included only the oracle operation in the runtime. Other computations performed by the compiler were excluded. We assume that any additional classical computations required for the oracle would take the same runtime 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  $f_b: \{0,1\}^n \mapsto \{0,1\}^n$ ,

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

where  $b=0^{n-i}1^i$  is the function period of Hamming weight i [19]. The additional steps transforming this function into a generic 2-to-1 function are performed by the classical compiler [20]. In our implementation we relied on the same function form.

The quantum algorithm for Simon's problem was implemented following the approach of Ref. [20], with its runtime estimated using Qiskit functionalities [49]. 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 [50]. The number of shots was chosen according to the oracle call bound given in Eq. (16) of Ref. [20]. The results are summarized in Fig. 2. The runtime analysis clearly demonstrates that the exponential query complexity speedup reported in [20] 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 0.035s to find the solution, while the estimated runtime for the quantum algorithm is on the order of 2s, assuming  $s = 10^5$  shots as in the experimental demonstration of Ref. [20]. The projected problem size at which the quantum algorithm would achieve a runtime advantage is N = 60. However, as already discussed in Ref. [20], 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 a specifically tailored hardware, such as FPGA, would result in even larger separation between classical and quantum runtimes [51].

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 runtime 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-matrix multiplication, which scales as  $O(n^{2.371339})$  [52], 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 [11]). This was, for example, the case with the first random circuit sampling-based supremacy results [15–17]. 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 runtime 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  $\mathrm{TT}\varepsilon$ , which measures the expected time required to obtain an approximate solution within a given precision, cf. Eq. 2.

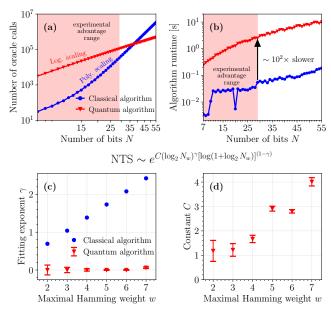


FIG. 2. For the restricted Simon's problem the quantum advantage manifests itself in a favorable polylogarithmic scaling of the protocol score function with the total number of periods  $N_w = \sum_j \binom{N}{j}$  as compared to the exponential scaling for the classical algorithm. The score function depends on number of oracle queries and a probability of protocol success. (a) Total number of oracle queries for the protocol and (b) runtime as a function of total number of bits N, as obtained via execution of 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 experimentally in [20]. For the considered problem sizes and oracle periods the classical algorithm runtime is shorter than the quantum one, we predict that the runtime crossover occurs for problem sizes N=60. For example, in the case of N=29 bits, what corresponds to the implementation presented in Ref. [20], the runtime of the classical algorithm is two order of magnitude shorter than the quantum algorithm. The quantum implementation is based on oracle constructed in Ref. [20], and the number of shots required for the algorithm was established using Eq. (16) of Ref. [20]. In this case the quantum circuit was transpiled to take into account connectivity and native gate set of IBM Brisbane, and runtime was estimated using Qiskit functionalities. The classical algorithm was implemented on a GPU. The fitting parameters of the score function for quantum and classical algorithm are presented in panels (c), (d). Parameter  $\gamma = 0$  indicates that the score scales polylogarithmicly, 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 [20].

For this figure of merit, an optimal reference classical algorithm should ideally combine solution quality with short runtime. 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 [53] or FPGAs [54]. In this regard, a particularly interesting class of classical heuristics is based on the dynamics of Hamiltonian nonlinear

systems [25, 35-37, 55-57], as these methods combine good performance with short execution times. Recently, it was shown that one algorithm from this class-the Simulated Bifurcation Machine (SBM)-achieves scaling of  $TT\varepsilon$ comparable to, or better than, that of quantum annealing [25]. This result effectively closes the previously reported quantum-classical gap in approximate optimization announced in [19], where the chosen classical reference was Parallel Tempering with Isoenergetic Cluster Moves (PT-ICM). The PT-ICM algorithm belongs to the class of physics-inspired methods based on temperature annealing [58]. 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 [25].

Secondly, a classical reference algorithm should be designed to solve the same class of problems. Otherwise, the comparison is biased as it favours the quantum algorithm. Potential worst performance of a chosen classical algorithm is not an evidence of a quantum algorithm advantage but it merely reflects the fact that the problem cannot be solved natively by the classical reference. For example, in [21], the investigated counterdiabatic quantum optimization algorithm 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 runtime of CPLEX compared to their quantum algorithm. Setting aside the fact that the overheads of counter-diabatic quantum optimization were not accounted for (see the discussion in Sec. II), the relevant question is how the performance of the counter-diabatic 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. [21], 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 considered instances is constructed iteratively, starting from the coupling graph  $C_0$  of 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 gets 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 BF-DCQO algorithm, since by changing the number of SWAP iterations, one can directly control the depth of the quantum circuit

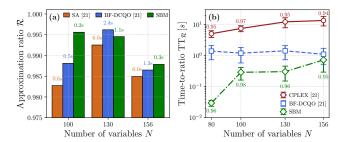
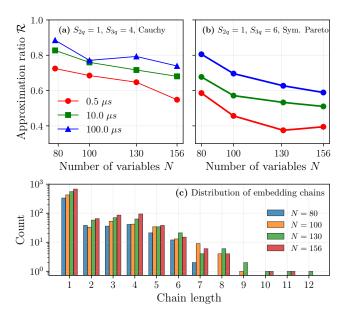


FIG. 3. Partial reproduction of Fig. 5 from Ref. [21], with additional results from SBM solver. Panel (a) shows approximation ratio  $\mathcal{R}$  achieved in a time annotated on top of the bars, for instance of type  $S_{2q} = 1$ ,  $S_{3q} = 4$  and couplings from Cauchy distribution. The SA and BF-DCQO results from Ref. [21] correspond to a single, best performing instance. Since the exact instances used by the authors of Ref. [21] had not been disclosed, we generated our own instances and show SBM results for the best performing instance. This highlights the danger of cherry-picking results, and the simplicity of "manufacturing" supremacy claims. Finally, panel (b) shows the value of time-to-ratio  $TT_{\mathcal{R}}$  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. [21], where they were obtained as a result of averaging over 5 random instances. Similarly, we constructed our own instances and show SBM results averaged over 5 of them. In both cases SBM outperforms other solvers, casting doubt on the supremacy claim of Ref. [21]. Moreover, in App. A3 we present a more thorough analysis of these instances, and show how to optimize HUBO SA to outperform BF-DCQO.

necessary for the quantum part of the algorithm. The authors of Ref. [21] 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. [21] were not made publicly available [59]. Consequently, we generated our own instances, closely following the description in Ref. [21] (cf. App. A3), and we make both these instances and the generation routine available in the accompanying GitHub [60].

To make a connection with the supremacy claims of Ref. [21], we employ the same figure of merit, which is the time-to-ratio  $TT_{\mathcal{R}}$  [28], defined as the time required to find a solution with an energy lower than  $\mathcal{R}E_{GS}$ , where  $E_{\rm GS}$  is the ground state energy of the problem instance. While this metric is conceptually similar to  $[TT\varepsilon]_{Med}$ , its execution in Ref. [21] is questionable, since it does not reflect the stochastic nature of investigated solvers. In Fig. 3, we compare SBM with the results of BF-DCQO, as well as SA and CPLEX taken from Ref. [21]. The authors choose to present their results only for either the best performing instance or as an average over just a few instances. In App. A3, we argue in detail why this approach is not satisfactory, 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

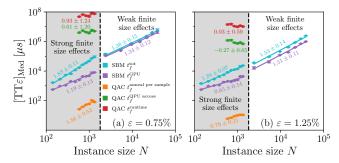


**FIG. 4.** (a)-(b) Approximation ratio  $\mathcal{R}$  achieved by D-Wave's Advantage 1.6 quantum annealer on the same HUBO instances as in Fig. 3, computed for each instance type and size, by selecting the best result out of 5 shots with  $2^{\bar{10}}$  samples each. Forward annealing scheme was used, with annealing times of  $0.5\mu s$  (red),  $10\mu s$  (green) and  $100\mu s$  (blue). The instances were first reduced to QUBO form in the same way as in the case of SBM results (see App. A3 for details), and then embedded into Advantage2 1.6 working graph. The results are, unsurprisingly, much worse than all other considered solvers. They can be explained by investigating the distribution of chain lengths in the necessary embedding, shown in panel (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.

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 easy it is to "manufacture" such claims by cherry-picking results.

Nonetheless, an important point is to be made by looking at a 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, the difference between SBM and BF-DCQO is miniscule ( $\Delta \mathcal{R} \simeq 0.002$ ), 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 runtime to achieve it.

This analysis illustrates how important it is to select a suitable classical reference algorithm when assessing



**FIG. 5.** Illustration of impact of solver-related overhead on scaling behavior of  $[TT\varepsilon]_{Med}$ , as defined in Eq. (2), using a certain type of Ising instances, relevant for near-term quantum devices (see Ref. [25] 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.

quantum advantage claims. Let us also mention that the results of [21] were recently also disputed in [61], 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 Advantage 21.6 based on the Zephyr topology. The results are presented in Fig. 4. Since the considered instances are not native to the topology of the QPU, an embedding procedure must be carried out. Fig. 4(c) shows the distribution of chain lengths in the embedding, with a tail extending to very long chains. This, unsurprisingly, results in a very poor performance, even after cherry-picking the best result out of 35 random instances, and 5 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 the performance, and thus one should be very careful when making such claims on the basis of experiments with hardware-native problems only.

#### 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 experiments 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—on 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 comparable to quantum devices, such as GPU clusters or FP-GAs, 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 [25] for the SBM, a heuristic inspired by nonlinear Hamiltonian dynamics [35–37]. 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 one of the next disruptive technologies. However, realizing this potential requires not only technological progress but also the identification of (potentially industry-relevant) use cases where quantum computing can deliver a significant advantage over existing classical methods. In this work, we argue that such an advantage should be reflected in a shorter time-to-solution achieved by a quantum algorithm. From this perspective, we (re)examined recent claims of quantum advantage and found that, under a proper definition of runtime and with an appropriate choice of the classical reference, none of them demonstrate a genuine runtime supremacy. In particular, we analyzed two recent milestone results:

Quantum Annealing (approximate QUBO). Prior work [19] proxied runtime by the annealing time. However, a careful runtime measurements show the median  $TT\varepsilon$  is essentially flat over the tested sizes. This is due to the readout per shot being  $\approx 200\mu s$  while the annealing durations are only  $0.5-27\mu s$ . Measurement errors are too large to draw meaningful conclusions; thus no scaling can be inferred, cf. Fig. 1. Furthermore, this "supremacy" would not survive against stronger classical baseline (such as SBM) even if the annealing time would faithfully reflect the device's runtime, as shown in [25].

SBM achieves robust scaling far beyond the system sizes attainable by current quantum annealers, cf. Fig. 5.

Gate-based oracle query (restricted Simon's problem). Although there is an exponential separation in query-complexity scaling, wall-clock performance favors a tuned classical GPU baseline: for N=29, and the "restriction" w=7 [20], classical runtime is  $\approx 0.05$ s, and the quantum  $\approx 2$ s ( $\sim 100 \times$  slower). The projected crossover of classical and quantum runtimes is at  $N\approx 60$  lies beyond the current capabilities of noise-limited devices, cf. Fig. 2.

BF-DCQO hybrid algorithm. Finally, we scrutinized reported gains for solving HUBO [21] and show they are based on estimates that neglect robust timing protocols and cherry-picking statistics. With such methodology one can produce arbitrary supremacy by design, cf. Fig. 3. It is worth noting that the problem instances considered in [21] were specifically designed to align with IBM's Heron architecture [62], and purportedly to be "challenging" for classical methods. We found that on those instances, the high-quality physics-inspired methods produces diverse solutions and therefore requires much richer statistics to substantiate any claim of supremacy. Cherry-picking the best out of a few runs (5 therein) is not sufficient, cf. App. A3.

In conclusion, across all cases, a practical runtime advantage on current NISQ hardware remains elusive. Credible claims should not omit dominant overheads (readout, thermalization, transpilation, etc.), must employ a proper metric such as  $\mathrm{TT}\varepsilon$  with all the required optimizations, and should benchmark against state-of-the-art, parallel classical solvers impartially selected on a case-by-case basis reflecting the problem class.

In contrast, in [19], all advantages were given to quantum annealing (such as avoiding dense instances and thus eliminating the need for embedding, allowing a relatively large  $\epsilon > 1\%$ , and using annealing time as a proxy for the actual runtime), yet this approach still could not compete for at least "limited supremacy" with a highly optimized classical algorithm executed on a single GPU [25].

#### ACKNOWLEDGMENTS

This project was supported by the National Science Center (NCN), Poland, under Projects: Sonata Bis 10, No. 2020/38/E/ST3/00269 (B.G) 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.

<sup>[1]</sup> R. P. Feynman, Simulating physics with computers, Int J Theor Phys 21, 467–488 (1982).

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

<sup>[3]</sup> IonQ, IonQ Roadmap (2025).

<sup>[4]</sup> IBM, IBM Quantum Roadmap (2025).

<sup>[5]</sup> Quantinuum, Quantinuum Roadmap (2025).

<sup>[6]</sup> D-Wave, The D-Wave Clarity Roadmap (2025), accessed: 2025-01-30.

- [7] H.-Y. Huang, S. Choi, J. R. McClean, and J. Preskill, The vast world of quantum advantage, arXiv 2508.05720 (2025).
- [8] D. R. Simon, On the power of quantum computation, SIAM J. Comput. 26, 1474 (1997).
- [9] P. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in *Proceedings 35th Annual* Symposium on Foundations of Computer Science (1994) pp. 124–134.
- [10] L. J. Stockmeyer, The polynomial-time hierarchy, Theor. Comput. Sci. 3, 1 (1976).
- [11] O. Lanes, M. Beji, A. D. Corcoles, C. Dalyac, J. M. Gambetta, L. Henriet, A. Javadi-Abhari, A. Kandala, A. Mezzacapo, C. Porter, and et al., A Framework for Quantum Advantage (2025), arXiv:2506.20658 [quant-ph].
- [12] A. Abbas, A. Ambainis, B. Augustino, A. Bärtschi, H. Buhrman, C. Coffrin, G. Cortiana, V. Dunjko, D. J. Egger, B. G. Elmegreen, and N. Franco et al., Challenges and opportunities in quantum optimization, Nat. Rev. Phys. 6 (2024).
- [13] F. Meier and H. Yamasaki, Energy-consumption advantage of quantum computation, PRX Energy 4 (2025).
- [14] K. Bertels, E. Turki, T. Sarac, A. Sarkar, and I. Ashraf, Quantum computing – a new scientific revolution in the making, arXiv:2106.11840 (2024).
- [15] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, and D. A. Buell et al., Quantum supremacy using a programmable superconducting processor, Nature 574, 505 (2019).
- [16] Y. A. Liu, X. L. Liu, F. N. Li, H. Fu, Y. Yang, J. Song, P. Zhao, Z. Wang, D. Peng, H. Chen, and et al., Closing the "quantum supremacy" gap: achieving real-time simulation of a random quantum circuit using a new Sunway supercomputer, in *Proc. - Int. Conf. High Perform. Comput. Netw. Storage Anal.*, SC '21 (Association for Computing Machinery, New York, NY, USA, 2021).
- [17] F. Pan, K. Chen, and P. Zhang, Solving the sampling problem of the Sycamore quantum circuits, Phys. Rev. Lett. 129, 090502 (2022).
- [18] A. Morvan, B. Villalonga, X. Mi, S. Mandrà, A. Bengtsson, P. V. Klimov, Z. Chen, S. Hong, C. Erickson, and I. K. Drozdov et al., Phase transitions in random circuit sampling, Nature 634, 328 (2024).
- [19] H. Munoz-Bauza and D. Lidar, Scaling advantage in approximate optimization with quantum annealing, Phys. Rev. Lett. 134, 160601 (2025).
- [20] 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 (2025).
- [21] 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 (2025).
- [22] 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).
- [23] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by Simulated Annealing, Science 220, 671 (1983).
- [24] CPLEX, IBM ILOG, Users manual for CPLEX (2024).
- [25] J. Pawłowski, P. Tarasiuk, J. Tuziemski, L. Pawela, and B. Gardas, Closing the quantum-classical scaling gap in approximate optimization, arXiv:2505.22514 (2025).

- [26] S. Boettcher, Analysis of the relation between quadratic unconstrained binary optimization and the spin-glass ground-state problem, Phys. Rev. Res. 1, 033142 (2019).
- [27] G. F. Newell and E. W. Montroll, On the Theory of the Ising Model of Ferromagnetism, Rev. Mod. Phys. 25, 353 (1953).
- [28] 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 (2023).
- [29] NVIDIA, Nsight systems user guide (2025).
- [30] Intel, Intel vtune profiler (2025).
- [31] AMD, AMD profiler (2025).
- [32] D-Wave, D-wave QPU solver parameters (2025).
- [33] H. Munoz Bauza and D. Lidar, Scaling Advantage in Approximate Optimization with Quantum Annealing -Spin-Glass Instances (2025).
- [34] D-Wave, D-wave operation and timing (2025).
- [35] H. Goto, Bifurcation-based adiabatic quantum computation with a nonlinear oscillator network, Sci. Rep. 6, 21686 (2016).
- [36] H. Goto, K. Endo, M. Suzuki, Y. Sakai, and Taro et al., High-performance combinatorial optimization based on classical mechanics, Sci. Adv. 7, eabe7953 (2021).
- [37] H. Goto, K. Tatsumura, and A. R. Dixon, Combinatorial optimization by simulating adiabatic bifurcations in nonlinear hamiltonian systems, Sci. Adv. 5, eaav2372 (2019).
- [38] T. Koch, D. E. B. Neira, Y. Chen, G. Cortiana, D. J. Egger, R. Heese, N. N. Hegade, A. G. Cadavid, and et al., Quantum optimization benchmark library – the intractable decathlon, arXiv:2504.03832 (2025).
- [39] 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, 420–424 (2014).
- [40] E. Alessandroni, S. Ramos-Calderer, I. Roth, E. Traversi, and L. Aolita, Alleviating the quantum big-m problem, npj Quantum Inf. 11 (2025).
- [41] 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, 174–189 (2025).
- [42] S. Schulz, D. Willsch, and K. Michielsen, Learning-driven annealing with adaptive hamiltonian modification for solving large-scale problems on quantum devices, arXiv:2502.21246 (2025).
- [43] R. Kothari, T. Lee, M. Szegedy, and I. Newman, Query Complexity, G - Reference, Information and Interdisciplinary Subjects Series (World Scientific Publishing Company Pte Limited, 2025).
- [44] A. Ambainis, Understanding quantum algorithms via query complexity, arXiv:1712.06349 (2017).
- [45] D. Deutsch and R. Jozsa, Rapid solution of problems by quantum computation, Proceedings: Mathematical and Physical Sciences 439, 553 (1992).
- [46] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proc. Annu. ACM Symp. Theory Comput.* (1996) pp. 212–219.
- [47] E. M. Stoudenmire and X. Waintal, Opening the black box inside grover's algorithm, Phys. Rev. X 14, 041029 (2024).
- [48] T. Hoefler, T. Häner, and M. Troyer, Disentangling hype from practicality: On realistically achieving quantum advantage, COMMUN ACM 66, 82 (2023).

- [49] 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 (2024).
- [50] IBM, IBM Quantum compute resources (2025).
- [51] 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, arXiv:2503.10302 (2025).
- [52] J. Alman, R. Duan, V. V. Williams, Y. Xu, Z. Xu, and R. Zhou, More asymmetry yields faster matrix multiplication, arXiv:2404.16349 (2024).
- [53] X. Yi, A Study of Performance Programming of CPU, GPU accelerated Computers and SIMD Architecture, arXiv:2409.10661 (2024).
- [54] R. Kastner, J. Matai, and S. Neuendorffer, Parallel Programming for FPGAs, arXiv:1805.03648 (2018).
- [55] 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).
- [56] 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, 175843–175865 (2024).
- [57] J. Pawłowski, J. Tuziemski, P. Tarasiuk, A. Przybysz, R. Adamski, K. Hendzel, L. Pawela, and B. Gardas, VeloxQ: A fast and efficient QUBO solver, arXiv:2501.19221 (2025).
- [58] D. Delahaye, S. Chaimatanan, and M. Mongeau, Simulated annealing: From basics to applications, in *Handbook of Metaheuristics*, edited by M. Gendreau and J.-Y. Potvin (Springer International Publishing, Cham, 2019) pp. 1–35.
- [59] The authors did not respond to our request to share their HUBO instances.
- [60] J. Tuziemski, J. Pawłowski, P. Tarasiuk, L. Pawela, and B. Gardas, Recent quantum runtime (dis)advantages – code repository, https://github.com/quantumz-io/ quanutm-runtime-disadvantage (2025), GitHub repository.
- [61] P. Farré, E. Ordog, K. Chern, and C. C. Mc-Geoch, Comparing Quantum Annealing and BF-DCQO, arXiv:2509.14358 (2025).
- [62] IBM, IBM Quantum Platform Processor types (2025).
- [63] T. Zhang and J. Han, Quantized simulated bifurcation for the Ising model, in 2023 IEEE 23rd International Conference on Nanotechnology (NANO) (2023) pp. 715– 720.
- [64] J. Fulman, Random matrix theory over finite fields: a survey, arXiv:0003195 (2001).
- [65] 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 (2020).
- [66] N. Dattani, Quadratization in discrete optimization and quantum mechanics, arxiv:1901.04405 (2019).
- [67] T. Kanao and H. Goto, Simulated bifurcation for higherorder cost functions, Appl. Phys. Express. 16, 014501

- (2022).
- [68] D. J. Earl and M. W. Deem, Parallel tempering: Theory, applications, and new perspectives, Phys. Chem. Chem. Phys. 7, 3910 (2005).
- [69] 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).
- [70] J. Machta, Population annealing with weighted averages: A monte carlo method for rough free-energy landscapes, Phys. Rev. E 82, 026704 (2010).
- [71] 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, arxiv:2409.04477 (2024).

#### Appendix A1: Simulated Bifurcation Machine

To demonstrate that the results for classical algorithms are less sensitive to runtime definition changes we consider a discretized version of Simulated Bifurcation Machine algorithm [25, 35–37], first introduced as dSB in Ref. [36]. The algorithm is based on a non-linear Hamiltonian system, whose motion is governed by the following equations:

$$\dot{q}_i = a_0 p_i,$$

$$\dot{p}_i = -\left[a_0 - a(t)\right] q_i + c_0 \left(\sum_{j=1}^N J_{ij} f(q_j) + h_i\right),$$
(A11)

where f(x) = sign(x) is the signum function. The original formulation of dSB is modified to include a ternary discretization scheme [63], replacing sign with

$$f(x) = \begin{cases} 0 & |x| \le \Delta(t), \\ \operatorname{sign}(x) & |x| > \Delta(t), \end{cases}$$
(A12)

where  $\Delta(t) = 0.7 \frac{t}{T}$  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  $c_0=\frac{0.7a_0}{\sigma\sqrt{N}}$ , where  $\sigma$  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  $a(t)=\frac{t}{T}$ . Bifurcations change the system's energy landscape to one approximately encoding the local minima of the Ising term. This mechanism allows 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  $\Delta t$  - time step, and  $N_s$  – number of steps, which determine jointly the total evolution time  $T=N_s\Delta t$ . Since SBM is a stochastic algorithm it needs to be executed a number of times, which we denote as number of replicas  $N_r$  (this can be done in parallel). For each replica the timestep  $\delta t$  is chosen randomly, for details see [25].

#### Appendix A2: Benchmarking classical algorithm on pseudo-random simulated oracles

The Simon's problem for a function that simulates behavior of an oracle can be solved using a classical algorithm. We have used a solver that not only confirms existence of a non-zero period, but also finds the period. The series of tests and benchmarks we have performed require simulation of oracle-like 2-to-1 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:

$$\forall_{x,y \in \{0,1\}^n} \quad f(x) = f(y) \iff y = x \oplus p. \tag{A21}$$

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

$$f(x) = g(\min_{\prec}(x, x \oplus p)). \tag{A22}$$

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 1-to-1 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,

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 1-to-1 functions such as g. To include all  $f_b$  functions from (4), we have implemented affine functions over GF(2) field:

$$g(x) = A x \oplus b \text{ for } A \in \{0, 1\}^{n \times n}, \text{ rank}_{GF(2)} A = n, b \in \{0, 1\}^n.$$
 (A23)

Notably, Ax is also intended to be computed in GF(2). Requirement that  $rank_{GF(2)}A = n$  is equivalent to A being invertible in GF(2), which is crucial to g being a bijection.

```
Input:
       J, h - coupling matrix and local fields vector specifying an instance of the Ising model,
       \Delta t – 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 = length(h)
q_1 \leftarrow \{\operatorname{rand}(-1,1)\}^n
p_1 \leftarrow \{\operatorname{rand}(-1,1)\}^n
for j = 1, \ldots, N_s do
     q_{i+1} \leftarrow a_0 \Delta t p_i
    p_{i+1} \leftarrow \left\{ \left[ a_0 - a(j\Delta t) \right] q_i + c_0 \left( \sum_{j=1}^N J_{ij} f(q_j) + h_i \right) \right\} \Delta t
     for i = 1, \ldots, n do
          if q_{i+1,j} \geq |1| then
               q_{i+1,j} \leftarrow \operatorname{sign}(q_{i+1,j})
               p_{i+1,j} \leftarrow 0
          end if
     end for
end for
x = \operatorname{sign}(x_{N_c+1})
E = \frac{1}{2}x^T J x + h^T x
return E, x
```

Pseudo-random generation of A and b is straightforward to implement. Constraint on A needs to be considered – probability that a pseudo random  $\{0,1\}^{n\times n}$  matrix will be invertible in GF(2) decreases with n. However, as shown in [64], 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.

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 non-trivial, and includes all cases described in (4).

The classical Alg. 2 is oblivious to the inner implementation of the oracle. The oracle-like 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 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 runtime measurements from Fig. 2. Constrained vectors can be generated with Alg. 3. The required precomputation of  $S(m, k) = \sum_{v=0}^{k} {m \choose v}$  values involves  $\Theta(n^2)$  computational and memory cost, and is calculated once per n, and reusable across multiple solver calls.

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

The adjusted solver for restricted Simon's problem described in Alg. 4 is a specialized version of Alg. 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

$$v(n, w) = S(\lfloor n/2 \rfloor, w) + S(\lceil n/2 \rceil, w) \leq$$

$$\leq 2S(\lceil n/2 \rceil, w) \leq 2\lceil n/2 \rceil \binom{\lceil n/2 \rceil}{w} =$$

$$= 2\lceil n/2 \rceil \prod_{i=1}^{w} \frac{\lceil n/2 \rceil - i + 1}{i} \leq 2\lceil n/2 \rceil^{w+1}.$$
(A24)

The most dominant component to Alg. 4 complexity is sorting an array of v(n, w) bit vectors of length n. This operation is  $\Theta(v(n, w) \cdot \log(v(n, w)) \cdot n)$ . For small or fixed w, the bound from Eq. A24 applies, so the complexity is in the class  $O(w n^{w+2} \log n)$ . This shows that the proposed classical solver for restricted Simon's problem has a polynomial complexity for fixed w. The polynomial behavior can be observed in Fig. 2.

```
Input:
      n – number of bits,
      w \le n – maximum number of set bits,
     i \in \left\{0,\, \dots,\, \left(\sum_{v=0}^{w} \binom{n}{v}\right) - 1\right\} – index of vector.
     S(m, k) = \sum_{v=0}^{k} {m \choose 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, ..., 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
```

#### ALG. 4. Classical solver to restricted Simon 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} {m \choose 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, ..., S(\lceil n/2 \rceil, \min\{w, \lceil n/2 \rceil\}) do
    t \leftarrow \mathbf{ALG3}(\lceil n/2 \rceil, \min \{w, \lceil n/2 \rceil\}, i, S)
     x_{i+1} \leftarrow (\lceil n/2 \rceil \text{ lowest bits of } t, 0 \ldots)
end for
for i = 1, ..., S(|n/2|, \min\{w, |n/2|\}) do
    t \leftarrow \mathbf{ALG3}(\lceil n/2 \rceil, \min \{w, \lfloor n/2 \rfloor\}, i, S)
    x_{i+2\lceil n/2\rceil} \leftarrow (0\ldots, \lfloor n/2\rfloor \text{ lowest bits of } t)
end for
for i = 1, ..., 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 \mathbf{SortingPermutation}(y_1,\,\ldots,\,y_{2^{\lceil n/2\rceil}+2^{\lfloor n/2\rfloor}-1})
                                                                                                                                   \triangleright done by sorting indices with y as key
for i = 1, ..., 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
```

#### Appendix A3: Detailed analysis of HUBO instances from Ref. [21]

#### 1. HUBO instance construction

We begin with providing a more detailed description of the HUBO instances used in Section IV of the main text. These instances are designed to be challenging, yet well suited for the IBM quantum computers, thus the starting point in their construction is the heavy-hex graph  $C_0$  of IBM Heron architecture [65]. 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,  $P_{2q} = \{P_{2q}^{(1)}, P_{2q}^{(2)}, \dots, P_{2q}^{(M_2)}\}$  and  $P_{3q} = \{P_{3q}^{(1)}, P_{3q}^{(2)}, \dots, P_{3q}^{(M_3)}\}$ , 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  $P_{3q}^{(i)}$  contain 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:  $G_2 \leftarrow G_2 \cup \{P_{2q}^{(i)}\}_{i=1,...,S_{2q}}$  and  $G_3 \leftarrow G_3 \cup \{P_{3q}^{(i)}\}_{i=1,...,S_{3q}}$ . 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 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  $C_0 \rightarrow C_1$  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. [21], 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 [60].

**ALG. 5.** Building conflict graph for 2-body interactions.

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

```
ALG. 6. Building conflict graph for 3-body interactions.
Input:
     C = (V, E) — connectivity graph
     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\}
                                                                                                                                           \triangleright Neighbors of v
    for (u, w) \in \{(u, w) : u, w \in N_v \land (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, \dots, |\mathcal{I}|\}, \emptyset)
                                                                                                                                \triangleright Initialize conflict graph
for i < j \le |\mathcal{I}| do
    if \mathcal{I}[i] \cap \mathcal{I}[j] \neq \emptyset then
                                                                                                                             \triangleright Interactions share a qubit
        Add edge (i, j) to C
    end if
end for
return C
                                                       ALG. 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_{\text{swap}} – number of SWAP iterations,
```

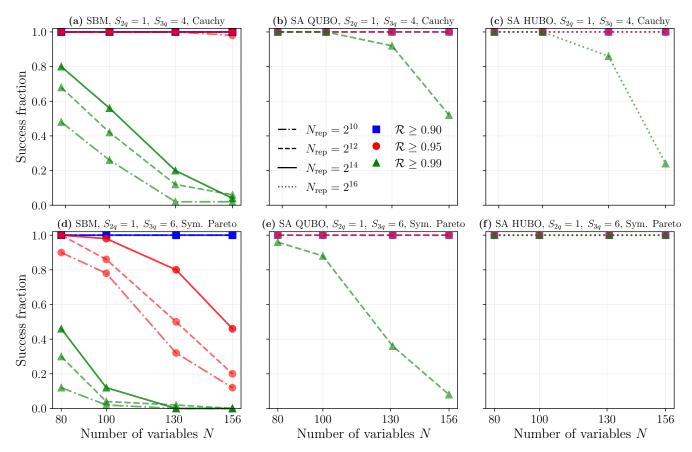
```
dist – coupling distribution
Output:
      J, K – dictionaries with two-body and three-body interactions defining the HUBO instance
G_2 \leftarrow \emptyset, G_3 \leftarrow \emptyset
C \leftarrow C_0
for n = 0, 1, ..., n_{\text{swap}} do
    C^{(2)} \leftarrow \mathbf{BuildConflictGraph}(C, 2\text{-body}), \ C^{(3)} \leftarrow \mathbf{BuildConflictGraph}(C, 3\text{-body})
    P_{2q} \leftarrow \mathbf{GraphColoring}(C^{(2)}), P_{3q} \leftarrow \mathbf{GraphColoring}(C^{(3)})
    for i = 1, ..., 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_{3a}^{(i)}
    end for
    if swap < n_{\text{swap}} then
         C \leftarrow \mathbf{ApplySWAP}(C, P_{2a}^{(1)})
                                                                                                                ▶ Permute qubit pairs from first 2-body set
    end if
end for
for (m,n) \in G_2 do
    J_{mn} \leftarrow \mathbf{SampleCoupling}(\mathbf{dist})
end for
for (p,q,r) \in G_3 do
    K_{pqr} \leftarrow \mathbf{SampleCoupling}(\mathbf{dist})
end for
return J, K
```

#### 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 [66]:

$$\pm s_i s_j s_k \to 3 \pm (s_1 + s_2 + s_3 + 2s_{\text{aux}}) + 2s_{\text{aux}} (s_i + s_j + s_k) + s_i s_j + s_i s_k + s_j s_k, \tag{A31}$$

which introduces one auxiliary spin  $s_{\rm aux}$  per third-order term, and preserves the spectrum of the original problem. In the GitHub repository [60], 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 [67], it is much more involved. Thus, we only consider the HUBO-native version of SA, which we discuss in greater detail in App. A4.



**FIG. A1.** Success fraction as a function of instance size, for 50 considered instances of each type. Panels (a) and (d) show results for SBM, (b) and (e) for SA in QUBO form, and (c) and (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 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.

We start by analyzing the success fraction, defined as the fraction of instances for which a given algorithm found solution better than or equal to a certain threshold. The results are shown in Fig. A1, 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, Simulated Bifurcation seems to struggle much more than Simulated Annealing, at least when it comes to pure solution quality. However, when discussing approximate optimization algorithms, one should always be mindful of the time-quality trade-off. Let us now consider the runtime of the algorithms, to see a different picture emerge. In Fig. A2, we show the approximation ratio  $\mathcal{R}$ , averaged over 50 couplings realizations and 10 (5)

independent runs of SBM (SA), as a function of runtime, for each considered instance type and size. Immediately, we can 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 improvent via extending it to natively handle higher-order interactions [67].

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 dangers of cherry-picking results, which were discussed in the main text, here are clearly visible in the insets of Fig. A2, which show the rather broad distribution of best approximation ratios obtained for each instance type and size. To stress it again here, in Fig. A3 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.

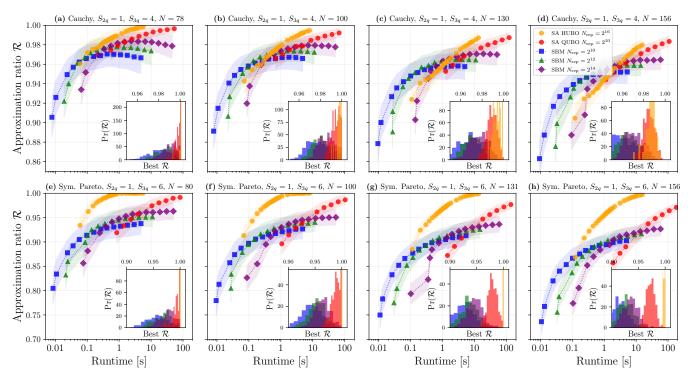


FIG. A2. Approximation ratio  $\mathcal{R}$  as a function of runtime, averaged over 50 instances and 10 (5) independent runs of SBM (SA), for each instance type and size. Shaded area corresponds to one standard deviation, and insets show the distribution of  $\mathcal{R}$  obtained, by aggregating best 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 runtime low. In terms of quality-runtime trade-off, it holds its ground even againts the HUBO SA variant, which does not suffer overheads from rank reduction. In particular, SBM is able to, on average, 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 seems to be "easy" for the HUBO SA. This is promising, given that it is possible to extend SBM so that it can handle higher-order interactions natively [67]. Finally, insets reveal 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 A4: Simulated Annealing for QUBO add HUBO

Simulated Annealing (SA) is one of the most widely used algorithms for finding approximate solutions to combinatorial optimization problems [23]. It relies on thermal fluctuations to probabilistically escape local minima during the search process and has inspired more sophisticated variants such as parallel tempering [68], parallel annealing [69], and population annealing [70], among others [57].

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

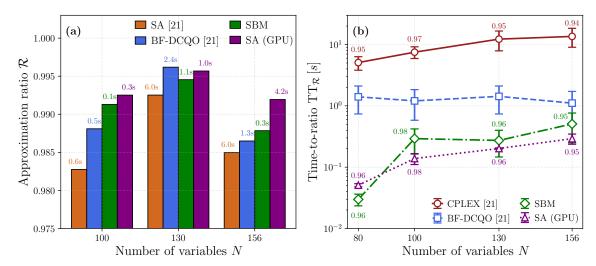


FIG. A3. Reproduction of Fig. 3 from the main text, with additional results from an optimized, GPU-based implementation of Simulated Annealing natively handling third-order interactions (cf. ALG. 9). This simulation demonstrates that an optimized HUBO SA method is already sufficient to outperform the BF-DCQO hybrid approach of Ref. [21].

convergence is limited by the chosen number of steps or runtime. As a result, the gap from the global optimum can remain large, especially for dense graphs [71]. Here we briefly outline the algorithm and provide pseudo-code 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 pseudo-random 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  $\delta$  – 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: parallelization strategy, data structures to store matrices, optimized recomputation of  $\delta$  values. When the batch of trajectories sufficiently large, parallelizing over trajectories is effective, and avoids any potential data races, regardless of the specific approach to  $\delta$  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 Alg. 9.

In the quadratic case, described as Alg. 8,  $\delta$  values can be updated incrementally after each accepted flip. This makes it feasible to keep the full  $\delta_{rv}$  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  $\delta_r$  vector for each variable v. This approach guarantees that we always use  $\delta$  values in sync with states as they are stored in the memory. As shown in Alg. 9, focusing on one variable at time is more efficient than updating the entire matrix at once. This mitigates the overhead of more frequent  $\delta$  recomputation.

For the technical implementation purposes, we assume that 2- and 3-body interactions are presented in symmetric arrays, i.e.  $J_{ij} = J_{ji}$  and  $K_{ijk} = K_{jik} = K_{ikj}$  for  $i, j, k \in \{1, n\}$ , n being the number of variables. We also assume  $J_{ii} = 0$  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:

$$P(s) = \left(\frac{1}{6} \left(\sum_{i=1}^{n} (K_i s) s_i\right) + \frac{1}{2} J s + h\right)^{\top} s.$$
(A41)

This formulation underlies the  $\delta_{rv}$  update cost formulas presented in Alg. 8 and  $\delta_r$  for Alg. 9.

#### Appendix A5: Reproducibility

To ensure the reproducibility of our results, we provide a GitHub repository [60]. It contains structured HUBO instances for three-body Ising models (along with generation code), a Python client for Simulated Bifurcation Machine experiments, 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 A6: 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 otherwisee.

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

#### 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} \text{ matrix})
\beta \leftarrow num\_steps-long geometric sequence from 1/T_0 to 1/T_1
for step = 1, ..., num\_steps do
    for pass \in \{1, \ldots, num\_passes\} do
        for v \in \{1, ..., 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)
                                                                                                \triangleright K_v - v-th matrix of K. J_v, (s^\top)_v - v-th rows
             for r \in \{1, ..., m\} do
                                                                                                ▷ parallelized with no need for atomic operations
                                                                                \triangleright Rand yielding a pseudo-random number from (0, 1) range
                 if \delta_r < 0 or Rand() < \exp(-\delta_r \, \beta_{step}) then
                      s_{rv} \leftarrow -s_{rv}
                 end if
             end for
        end for
    end for
end for
return s
```