

VeloxQ: A Fast and Efficient QUBO Solver

J. Pawłowski,^{1,2} J. Tuziemski,² P. Tarasiuk,² A. Przybysz,² R. Adamski,² K. Hendzel,² L. Pawela,^{3,2} and B. Gardas^{3,2}

¹*Institute of Theoretical Physics, Faculty of Fundamental Problems of Technology,
Wrocław University of Science and Technology, 50-370 Wrocław, Poland*

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

³*Institute of Theoretical and Applied Informatics,
Polish Academy of Sciences, Bałtycka 5, 44-100 Gliwice, Poland*

We introduce VeloxQ, a fast and efficient solver for Quadratic Unconstrained Binary Optimization (QUBO) problems, which are central to numerous real-world optimization tasks. Unlike other physics-inspired approaches to optimization problems, such as quantum annealing and quantum computing, VeloxQ does not require substantial progress of technology to unlock its full potential. We benchmark VeloxQ against the state-of-the-art QUBO solvers based on emerging technologies. Our comparison includes quantum annealers, specifically D-Wave’s Advantage and Advantage2 prototype platforms, the digital-quantum algorithm designed to solve Higher-Order Unconstrained Binary Optimization (HUBO) developed by Kipu Quantum, physics-inspired algorithms: Simulated Bifurcation and Parallel Annealing and an algorithm based on tropical tensor networks. We also take into account modern developments of conventional algorithms: Branch and Bound algorithm, an optimal implementation of the brute-force algorithm and BEIT QUBO solver. Our results show that VeloxQ not only matches but often surpasses the mentioned solvers in solution quality and runtime. Additionally, VeloxQ demonstrates excellent scalability being the only solver capable of solving large-scale optimization problems, including up to 2×10^8 sparsely connected variables, that are currently intractable for its competitors. These findings position VeloxQ as a powerful and practical tool for tackling large-scale QUBO and HUBO problems, offering a compelling alternative to existing quantum and classical optimization methods.

I. INTRODUCTION

Increasing complexity of the real-world optimization problems presents a considerable challenge for the state-of-the-art approaches. Growing size and complexity of industry-relevant problems may soon become a crucial factor hindering performance of the well-established exact and heuristics methods. Recent progress in new computational paradigms, including quantum computing and physics-inspired algorithms, provides new opportunities for development of novel optimization algorithms that can prove superior to existing methods. For example, simulation of dynamics of certain classical systems has recently proved to be a promising heuristic method of solving QUBO problems [1–3] that, in contrast to other physics-inspired optimization algorithms such as Simulated Annealing [4], can take advantage of the conventional parallel hardware. Here we present VeloxQ [5]: The QUBO solver that, similarly to other emerging technologies such as quantum computing, goes beyond conventional approaches to solving QUBO problems by adapting a novel physics-inspired methodology. However, unlike quantum computing, it does not require any technology developments as it can fully exploit the enormous potential of conventional computing technology. In this way VeloxQ bridges established and emerging approaches, and facilitates faster uptake of novel physics-inspired optimization technologies and, when the technology matures, can be readily integrated into hybrid quantum-classical workflows to maximize strengths of both technologies. In fact, a hybrid solver that allows

to combine various approaches to optimization problems, including VeloxQ and quantum computing will be presented elsewhere.

The aim of this paper is to provide an extensive set of benchmarks that comprehensively characterize VeloxQ features such as solution quality and runtime. The space of QUBO solvers offers several promising solutions, which we compare with VeloxQ solver. We present benchmarks against quantum-analog solvers offered by D-Wave, the quantum-digital optimization algorithm developed by Kipu Quantum [6], the optimal implementation of the brute force QUBO solver [7–9], the QUBO solver with ground-state certification offered by BEIT [10, 11], physics-inspired algorithm based on tropical tensor networks [12], parallel annealing [3] and simulated bifurcation algorithm [1], as well as modern developments of classical Branch and Bound algorithm [13, 14]. The benchmarks include instances that are favorable for the solvers compared with VeloxQ, what allows to fully recognize those solver strengths, and assures that the excellent performance of VeloxQ observed in benchmarks cannot be attributed to the biased choice of instances. The results of benchmarks consistently show that VeloxQ either matches or surpasses performance of the mentioned solvers. VeloxQ distinguishes itself from other benchmarked solutions by its scalability: It is the only solver able to achieve remarkable solution quality and runtime for large-scale optimization problems that are intractable for its competitors (however, as for some solvers in the comparison VeloxQ results are not certified, i.e. the found solution is not guaranteed to be the optimal one).

For example, VeloxQ is able to solve optimization problem based on Zephyr graph Z_{1750} that involves more than 9.8×10^7 variables. We estimate, based on D-Wave devices release dates, that a quantum annealer able to handle such a problem would be available no sooner than in 30 years, see Fig. 1. A similar conclusion concerning required hardware can be drawn for the case of digital quantum algorithm developed by Kipu Quantum, where VeloxQ can handle an instance of 2×10^8 variables.

There are notable emerging QUBO solution technologies that we do not include in this comparison. Memcomputing is one of the promising approaches to the QUBO optimization problem [15], however we were not able to get access to any solver utilizing this approach. A lot of efforts are devoted to investigations of Coherent Ising Machines [16], in the context of hardware [17], and theoretical understanding of the relation between binary solution and its continuous relaxation [18]. We decided not to include Coherent Ising Machines into our study as they behaviour is similar to the Simulated Bifurcation algorithm. We also implemented annealer based on the Hamiltonian Monte Carlo [19] but the initial results of this approach were not satisfying and we did not proceed with an extensive benchmark. Approaches based on classical thermodynamics, like quadratic programming enhanced by thermodynamic linear algebra subroutines [20, 21], were not taken into account as they require specialized hardware that is currently unavailable, and they do not allow to solve binary problems natively.

In cases, when the instance size allowed it, the reference solution was generated by CPLEX [22]. Those cases provide an indirect benchmark of VeloxQ against the well-established classical QUBO solution methods, and they clearly show that the performance of VeloxQ in terms of solution quality and runtime is favorable as compared to standard tools adopted in the industry. Here again the most prominent feature of VeloxQ distinguishing it from well-established conventional solvers is its performance on large-scale instances: Runtime of VeloxQ was less than two minutes, and we were not able to establish the required runtime for CPLEX (our estimate is that it is on the order of weeks rather than days).

To make the paper self-contained in Sec. II the definition of QUBO problems is provided, and its relation to the Ising model, as well as the Higher Order Unconstrained Binary Optimization problems. In Sec. III state-of-the art approaches to solving QUBO problems are presented. Subsequently, benchmarks and their results are described: In Sec. IV benchmarks against D-Wave quantum annealers, in Sec. V against digital quantum algorithm developed by Kipu Quantum, in Sec. VI against solvers with certified solutions including BEIT solver [10, 11] and a tropical tensor networks approach. In Sec. VII results of benchmarks against physics-inspired algorithms: parallel annealing and simulated bifurcation are presented, and in Sec. VIII against refined Branch and Bound algorithm proposed by Quantagonia [14]. We summarize our findings in Sec. IX.

Examples of benchmark instances, as well as the scripts that were used to generate the instances are available in the online repository [23].

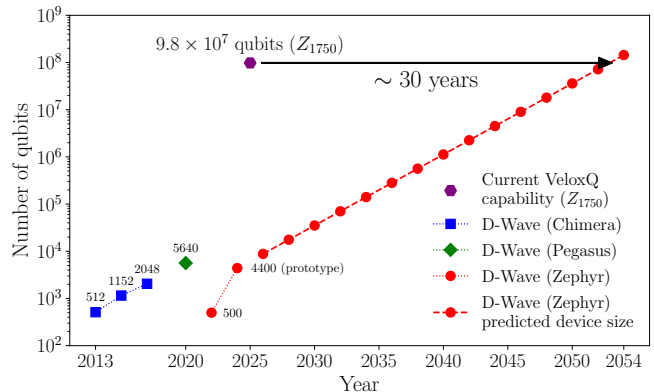


FIG. 1. Comparison of past, current and predicted qubit count of D-Wave’s quantum annealers, to current capabilities of VeloxQ — Z_{1750} with $\sim 9.8 \times 10^7$ variables (purple hexagon). Assuming a doubling period of two years (estimated from historical data), we can roughly estimate that a quantum device capable of natively handling such instances will be available around 2054.

II. QUADRATIC UNCONSTRAINED BINARY OPTIMIZATION

QUBO problems constitute an important class of optimization problems with applications in many areas ranging from operational research to portfolio optimization [24]. They encode the task of finding a binary vector $\mathbf{x} \in \{0, 1\}^N$ that minimizes a quadratic function $Q(\mathbf{x})$,

$$Q(\mathbf{x}) = \sum_{i < j} Q_{ij} x_i x_j. \quad (1)$$

The optimization problem encoded in QUBO can be equivalently formulated in terms of the Ising model. The latter originates from statistical physics, and describes energy associated with a particular configuration of discrete variables (spins) $\mathbf{s} \in \{-1, 1\}^N$:

$$H(\mathbf{s}) = \sum_{i < j} J_{ij} s_i s_j + \sum_i h_i s_i. \quad (2)$$

The transformations between QUBO and Ising formulations is presented in Appendix A. The Ising-QUBO relation opens possibility for tackling optimization problems with analog quantum devices such as quantum annealers, as well as gate-based quantum computers [25]. Despite the equivalence there are case (e.g. random instances), in which application of a QUBO solver to a QUBO formulation of the Ising problem may perform poorly [26].

In general QUBO belongs to the NP-hard problems, and many NP-problems can be embedded into

QUBO [27], such as graph coloring [28], max-cut [29], traveling salesman [30] or boolean satisfiability [31], as well as Higher Ordered Unconstrained Binary Optimization problems [32], whose cost function includes multi-variable terms of order \mathcal{P} :

$$P(\mathbf{x}) = \sum_{\substack{i_1, \dots, i_N \\ i_1 + \dots + i_N \leq \mathcal{P}}} a_{i_1 \dots i_N} x_1^{i_1} \dots x_N^{i_N}. \quad (3)$$

Third-order HUBO problems will be used to benchmark our solver against proposed quantum algorithms capable of solving such HUBO problems natively.

III. STATE OF THE ART QUBO SOLVERS

For the purpose of this paper we will divide solver capable of handling QUBO problems into the following categories: classical, analog-quantum, and digital-quantum solutions, as well as physics inspired algorithms.

To the first group belong solvers that run on conventional hardware. CPLEX [22] and Gurobi [33] solvers have been widely used in the industry for optimization problems, including QUBO problems. They implement both exact and heuristic solving methods [34]. Linear integer problem solvers also can be used to handle QUBO problems, due to the existence of reformulation of QUBO as a linear constrained binary problem. In general, the performance of general purpose classical solvers such as those mentioned above depends strongly on the problem structure and its size. Some classical solvers are designed to handle problems of specific structure e.g. BEIT QUBO solver [10, 11], which accepts instances of Chimera graph topology and restricted size. There are also attempts to utilize deep reinforcement learning to solve QUBO problems [35, 36].

Due to the connection between QUBO problems and Ising models, quantum annealers have been employed to solve QUBO problems [37]. The main limitation of this approach is the number of qubits available on a quantum annealer, and its topology, which restricts the class of problems that can be natively implemented on a device. Problem instances not matching topology of a device require embedding that introduces computational overhead. The possibility of achieving some sort of computation advantage over classical methods is still debated [38], and there is an ongoing investigation of particular use-cases such as protein-folding [39], or grid cost allocation in electricity markets [40]. Due to the importance of quantum Ising model [41] quantum annealers are also important scientific tool [42].

In the context of gate-based quantum computers there exists a family of optimization algorithms exploiting sub-routines such as Quantum Phase Estimation or Grover algorithm [25], and there are cases of quantum algorithms that exhibit super-polynomial speedup over classical algorithms for a certain class of optimization problems [43]. However, those algorithms remain theoretical proposals

since the present generation of quantum hardware, despite the recent progress [44], are still not useful for commercially relevant combinatorial optimization problems. On the other hand, the so-called Variational Quantum Algorithms (e.g. Quantum Approximate Optimization Algorithm [45]) are proposed as quantum heuristic that can be run on available devices. However, in most cases it remains unclear whether such algorithms lead to some advantage over best-known classical algorithms. For a recent overview of the field see [25].

There is also a group of physics-inspired algorithms that utilizes conventional computing technology. Arguably the best known algorithm belonging to this group is Simulated Annealing, which was introduced in 1980s [4], and is still being studied e.g. to develop optimized versions for particular use-cases [46]. To recently developed physics-inspired algorithms belong algorithms such as Simulated Bifurcation [1], and tensor networks [47], or classical algorithms inspired by quantum annealing [3, 48]. It is also interesting to note that the many algorithms in convex optimization, such as Alternating Direction Method of Multipliers [49], can be formulated in terms of dynamical systems [50–52].

IV. BENCHMARKS AGAINST D-WAVE QUANTUM ANNEALERS

The aim of this benchmark is to compare our flagship product, VeloxQ solver, against the state-of-the-art quantum annealers, represented by the D-Wave Advantage and D-Wave Advantage2 prototype platforms. Due to the underlying topologies of those quantum processing units (QPUs), Pegasus P_{16} and Zephyr Z_6 respectively (cf. Fig. 2), most real-world problems have to be mapped to the hardware graph in a process known as *minor embedding* [53], which introduces, sometimes significant, overheads in terms of total time of solution and qubits required. On the other hand, in the case of VeloxQ solver the minor embedding of a problem is not used as the solver handles natively all problem topologies. Due to this difference between solvers, in order to obtain direct comparison of their performance, in the first benchmark we decided to focus on instances that are native to the D-Wave hardware, and thus do not require minor embedding. That is, we will be solving QUBO problems with cost function given by Eq. (1), where the problem structure match that of undirected (sub)graph of the Pegasus or Zephyr structure (the binary variables are placed on vertices of a (sub)graph, the edges of the (sub)graph correspond to couplings the variables). Elements of the coupling matrix Q are chosen randomly from a uniform distribution on the interval $[-1, 1]$. For a reference solution, we solved the original QUBO problems using an industry-standard optimization suite – CPLEX. It was allowed to run for 20 minutes, which is a much longer timescale than on which both VeloxQ and D-Wave annealers operate.

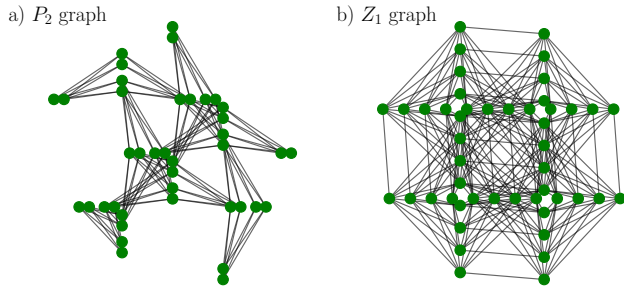


FIG. 2. Illustration of the basic building blocks of the D-Wave’s Pegasus [54] and Zephyr [55] topologies. **a)** Pegasus graph P_2 on 40 qubits, underlying current-gen D-Wave Advantage QPUs. This topology supports K_4 and $K_{6,6}$ graphs natively. **b)** Zephyr graph Z_1 on 48 qubits, forming the basis of the next-gen topology used in the Advantage2 prototype and future devices. Enables direct embedding of K_4 and $K_{8,8}$ graphs.

Real-world optimization problems, such as portfolio optimization [56], are often expressed by QUBO models with the number of variables considerably exceeding number of qubits of the current quantum annealers. To demonstrate scalability of VeloxQ solver we also consider instances of the structure described above in the size-range intractable for the D-Wave annealers (due to available number of qubits). However, we make use of the *dwave-hybrid* Python package, offering the so-called Kerberos hybrid solver that uses quantum annealer to handle subproblems of the initial problem [57]. The benchmark set consist of problems of Pegasus structure up to P_{150} with approx. 5×10^5 variables, and Zephyr structure up to Z_{150} with approx. 7.2×10^5 qubits, going more than 10^3 in problem size beyond the capabilities of current and near-future quantum annealers. Reference solutions for these large scale instances were obtained using an efficient, GPU-based implementation of the SA algorithm [4].

For each benchmark we present results from VeloxQ solver obtained with two different sets of solvers parameters. Results denoted as ‘AutoTune VeloxQ’ were obtained with the default set of parameters, without any prior knowledge about the structure of the problems, which sometimes results in a longer runtime. The results denoted as ‘Custom VeloxQ’ use the in-built flexibility of VeloxQ to adjust the solver parameters that control the trade-off between the solution quality and runtime, based on some knowledge about the problem and/or prior solver runs.

For examples of benchmark instances, as well as scripts that were used to generate them see the online repository [23].

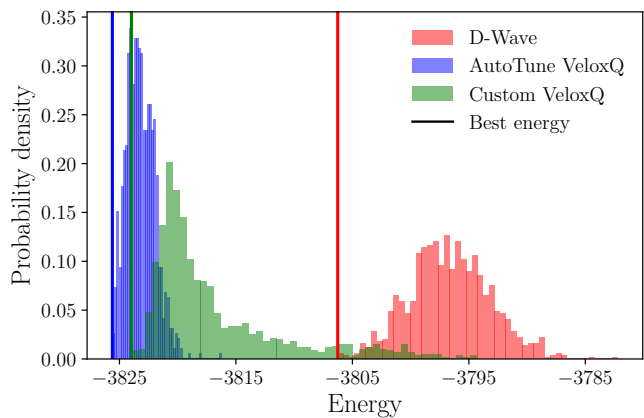


FIG. 3. Low-energy spectra for a random QUBO problem on P_{16} graph, consisting of 2^{10} states each, obtained with D-Wave Advantage platform (red), AutoTune VeloxQ (blue) and Custom VeloxQ (green). The distributions produced by VeloxQ are qualitatively different from the D-Wave’s, shifted towards lower energies and, in case of AutoTune VeloxQ, more concentrated around the mean. Custom VeloxQ spectrum, tuned for shorter runtime, is more spread out, with longer high-energy tail, yet the lowest-energy state is reasonably close to the best configuration from the AutoTune VeloxQ. This highlights the versatility of our solver, allowing for easily controlled trade-off between the solution quality and the runtime.

A. Pegasus topology

To test the statistical performance on comprehensive set of problem sizes, initially we generated 20 random instances for each Pegasus graph being a subgraph of P_{16} , that is P_2, P_3, \dots, P_{16} . All possible couplings and biases were chosen from the uniform distribution on the interval $[-1, 1]$. We measure the optimality gap defined with respect to the CPLEX solution \mathbf{s}_{ref}

$$g(\mathbf{s}, \mathbf{s}_{\text{ref}}) = \frac{H(\mathbf{s}) - H(\mathbf{s}_{\text{ref}})}{|H(\mathbf{s}_{\text{ref}})|}, \quad (4)$$

and the total runtime, including overheads related to the communication with the solver through an appropriate API. Note, that due to invariability of the true optimal solutions, *i.e.* ground states, the gap, henceforth referred to as reference gap, can be negative. Such cases indicate that the solution obtained by the solver is better than the reference one.

Due to the inherently probabilistic nature of quantum measurements, the D-Wave quantum annealers produce an ensemble of samples, from which the best is usually chosen as the final solution. However, when treating the QPU as just a part of a hybrid solver, dealing with selected subproblems, it might be beneficial to use more than just one results of local optimization, to avoid the risk of getting stuck in a local minimum. VeloxQ, even though it is a classical solver, emulates this behavior and also produces a low-energy spectrum of solutions. An

example of such spectra, both for VeloxQ and D-Wave annealer, is presented in Fig. 3. In subsequent benchmarks AutoTune VeloxQ and D-Wave annealers are set to produce 2^{10} samples, and the best solution is chosen to calculate the reference gap. Both quantities are averaged over realizations of a given graph size (see figures description for details). Results of the experiments are presented in Fig. 4. By default, VeloxQ solver operates with a carefully chosen default set of parameters with an instance-depended tuning. The panel a) in Fig. 4 demonstrates that even with default settings (denoted by AutoTune VeloxQ), our solver obtains solutions very close to the reference ones, and even outperforms them for the largest instances. Despite the fact that the instances are tailored to match the structure of D-Wave Advantage platform, the annealer does not present any advantage over VeloxQ in terms of solutions quality. The use of the quantum annealer results in an order of magnitude shorter time-to-solution for the considered instances (Fig. 4) than those of VeloxQ solver with default settings. However, by an appropriately tuning VeloxQ parameters we are able to achieve a similar or better runtime, while maintaining the high quality of the solutions.

To investigate the regime beyond native QPU capabilities, we prepared random instances of Pegasus topology with size parameter $m \in \{20, 30, \dots, 150\}$, five for each m , and computed reference solutions with SA algorithm [4]. The results are presented in Fig. 4 c) (reference gap) and d) (runtime). VeloxQ solver is able to consistently obtain high-quality solutions, matching the hybrid Kerberos solver solution quality, and outperforms Kerberos in terms of runtime as the time-to-solution is ten times faster. By adjusting VeloxQ solver parameters, the runtime can be further reduced by an order of magnitude at a relatively small cost in the solution quality. Crucially, the Kerberos solver with default parameters was not able to produce solutions for instances larger than P_{60} , that is $\sim 85 \times 10^3$ variables, whereas VeloxQ performed well across the whole range of problem sizes, up to P_{150} with $\sim 500 \times 10^3$ variables.

For completeness, we have tested the other end on the difficulty spectrum for quantum annealers, that is problems with all-to-all connectivity. We prepared random Ising models on complete graphs with up to 160 vertices, translating to 2536 qubits after embedding. The experiments were conducted, and data gathered, in the same fashion as for instances with Pegasus topology. They are presented in Fig. 5. Even though there are specialized routines for clique embedding, the overheads related to newly introduced auxiliary qubits, significantly degrades the performance of the D-Wave Advantage sampler, with optimality gaps reaching double digits for the largest instances. VeloxQ solver, on the other hand, is able to solve the problems with high accuracy and in extremely short time, even in comparison to the annealer with the embedding time excluded.

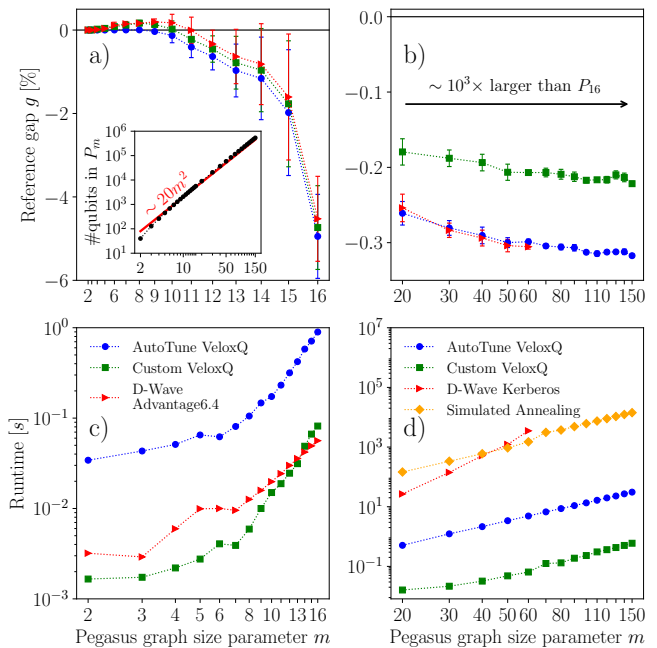


FIG. 4. Performance of VeloxQ solver on problems with Pegasus topology, with D-Wave comparison where possible. Panels a), c): respectively, reference gap w.r.t. to CPLEX reference solution and total runtime (averaged over 20 random problems), for instances being subgraphs of P_{16} , i.e. the Pegasus graph underlying the QPU of the D-Wave Advantage6.4 platform. Error bars mark the standard deviation. Inset shows how number of qubits scales with the size parameter of the graph. Panels b), d): the same as in a), c) but for larger instances of Pegasus topology, up to P_{150} (averaged over 5 random problems). Reference solutions in this regime were obtained using Simulated Annealing (SA) algorithm. VeloxQ solver is able to match both the quality and the time to solution of the D-Wave’s Pegasus-based QPU in the regime $P_m \leq 16$. For larger problems, our solver, with default parameters, obtains samples of quality very close to the Kerberos hybrid solver, but in significantly shorter time, up to almost three orders of magnitude. If the time to solution is a priority, custom settings allow for a runtime reduction of another order of magnitude, with only a slight decrease in the solution quality. The Kerberos solver was limited to solve instances beyond P_{60} , as for larger ones to running out of memory on a machine with 1 TB of RAM.

B. Zephyr topology

Finally, we proceed to the newest D-Wave platform, the Advantage2 prototype. As of the beginning of 2025, the device’s topology is limited to the Zephyr graph Z_6 , consisting of approximately 1200 qubits, which imposes a limit of problem sizes for which we can conduct a 1-to-1 comparison with the QPU. Nevertheless, we prepared randomized optimization problems native to the Zephyr graphs of size up to Z_{15} with 7000+ qubits, which is the promised structure of the final Advantage2 device, coming in the near future [58]. Motivated by the excellent

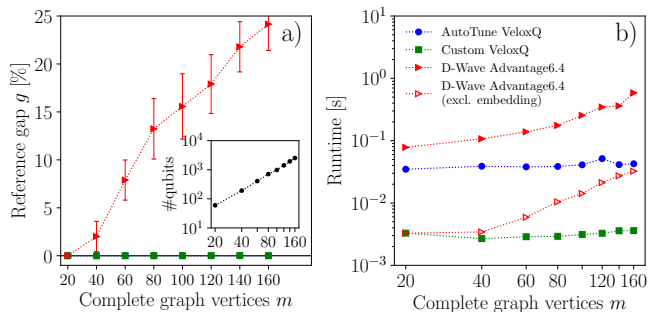


FIG. 5. Performance of VeloxQ solver in comparison to quantum annealer, on problems with all-to-all connectivity. Panel **a)**: optimality gap (averaged over 20 problem realizations) w.r.t. to CPLEX reference solution. Inset shows how many qubits are needed to embed a complete graph into P_{16} Pegasus graph. Panel **b)**: total runtime. Empty symbols denote results for D-Wave machine with excluded time of embedding. Due to the overheads related to the embedding process, the D-Wave Advantage sampler produces solutions of poor accuracy. VeloxQ, being in principle insensitive to the problem topology, is able to produce solutions very likely to be close to the ground states (reference gap close to zero), in a time much shorter than the quantum annealer.

performance of VeloxQ on the Pegasus topology, we decided to extend the range of the benchmark even further, far beyond the capabilities of yet-to-be-released quantum hardware, to a Z_{1750} graph with more than 9.8×10^7 qubits. We estimate that a device capable of handling such instances would be available in approximately 30 years (cf. Fig. 1). This prediction is based on the assumption that the current prototype device with Zephyr topology (Advantage2 QPU with 4400 qubits) would double its size every two years, what is in accordance with previous D-Wave devices releases. We also assume and that scaling challenges such as hardware reliability (e.g. number of idle qubits), control electronics, and communication bandwidth, would not hinder development of annealing devices. Technical details of the computational experiments remain the same as in the previous case, with the new results presented on Fig. 6. Once again, VeloxQ is able to closely match both the quantum annealer and the quantum-enabled hybrid solver, in terms of solution quality and total runtime. Moreover, we confirm the scalability of our solver, and its promising performance even in regimes far beyond the capabilities of next-generation, Zephyr-based quantum hardware.

V. BENCHMARKS AGAINST KIPU QUANTUM SOLVER

In this section we compare VeloxQ solver to the Kipu Quantum solver BF-DCQO [6], which is a digital-quantum computer algorithm for solving higher-order unconstrained binary optimization (HUBO) problems. QUBO solvers, such as VeloxQ or quantum annealers,

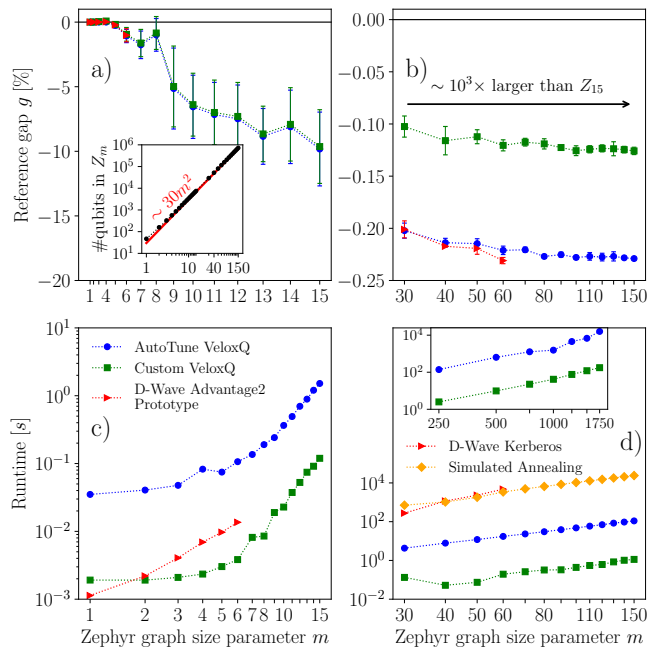


FIG. 6. Performance of VeloxQ solver on problems with Zephyr topology, with D-Wave comparison where possible (dashed red lines indicate regime of problems possible to solve directly on the D-Wave Advantage2 prototype). Panels **a)**, **c)**: respectively, optimality gap w.r.t. to CPLEX reference solution and total runtime (averaged over 20 problem realizations), for instances of the Zephyr type. Inset shows how number of qubits scales with the size parameter of the graph. Panels **b)**, **d)**: the same as in **a)**, **c)** but for larger instances of Zephyr topology, up to Z_{150} . Reference solutions in this regime were obtained using Simulated Annealing (SA) algorithm. Similarly to the case of Pegasus benchmarks, VeloxQ solver is able to match rather closely the D-Wave’s Advantage2 prototype performance both in the small instance regime ($Z_m \leq 6$), where direct QPU sampling is possible, and in the large instance regime, where the annealer is used as a part of the hybrid workflow. Kerberos solver once again was memory-constrained to instances no larger than Z_{60} . Inset in panel **d)** shows the runtime of VeloxQ solver for much larger Zephyr instances, up to Z_{1750} with more than 90×10^6 qubits. Here, we do not make any claims about the quality of the solutions, as it is difficult to obtain reference solutions for such large instances. Nevertheless, VeloxQ is able to produce samples in a reasonable time, and the runtime keeps its consistent scaling behavior.

can be used to solve HUBO problems (such as 2-SAT) at the expense of dealing with problem of increased variable-size and complexity (the necessary reduction of a HUBO problem leads to a QUBO problem with increased number of variables and increased complexity [32]). The advertised advantage of the Kipu Quantum algorithm over QUBO solvers is that it solves HUBO problems natively and the costly reduction HUBO-QUBO reduction is avoided. In the benchmarks we compare performance of VeloxQ to results obtained with Kipu Quantum solver BF-DCQO [6].

A. Higher-order unconstrained binary optimization

The benchmark consists of two types of problems: NP-complete 3-Satisfiability (3-SAT) and one-dimensional random spin glass with three-body interactions. These problems were selected to facilitate a direct comparison with the results of the Kipu Quantum solver BF-DCQO, which in part were obtained using digital quantum hardware - IBM FEZ quantum platform (the chain structure of these problems is particularly well suited to the topology of existing IBM quantum processors) [6].

For examples of benchmark instances, as well as scripts that were used to generate them see the online repository [23].

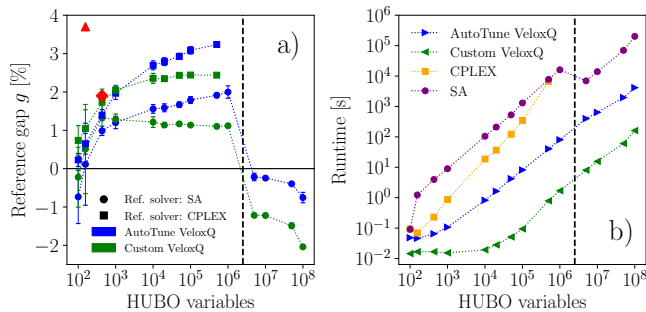


FIG. 7. Results of the benchmarks for random 3 body Ising model, Eq. (5). **a)** Quality of the solutions obtained by VeloxQ solver, measured by the relative gap to the reference solution from CPLEX (squares) and Simulated Annealing (SA) (circles), obtained with default (blue) and custom (green) settings. Red triangle and diamond denote results for, respectively, BF-DCQO on real and simulated quantum hardware, as reported in [6]. **b)** Runtimes of VeloxQ solver for default (blue right triangles), and custom (green left triangles) settings. We also present the time needed to obtain the reference solutions by CPLEX (orange squares) and SA (purple circles). SA for instances with more than 10^6 variables (indicated with vertical dashed line) was run with modified parameters, to ensure the results were obtained in reasonable time. Despite the factor of 2 overhead introduced by HUBO→QUBO reduction, VeloxQ demonstrates excellent performance in problems spanning 7 orders of magnitude in size, from 100 to 10^8 variables. The quality of the solutions is within approx. 3% of the reference solution in the small to intermediate range, and even outperforms the SA results for the largest instances (CPLEX failed to produce solutions in this regime). Furthermore, this is achieved in a time of up to 3 orders of magnitude shorter than the reference solvers.

Due to the fact that test instances were not provided in [6], we generated our own benchmark set. In the case of instance of HUBO problems corresponding to a one-dimensional spin glass the cost function is equivalent to the Hamiltonian

$$H^{\text{NN}}(\mathbf{s}) = \sum_{i=1}^N h_i s_i + \sum_{i=1}^{N-1} J_i s_i s_{i+1} + \sum_{i=1}^{N-2} K_i s_i s_{i+1} s_{i+2}, \quad (5)$$

where the couplings h_i, J_i, K_i are drawn randomly from a Gaussian distribution with zero mean and unit variance.

The cost function corresponding to the weighted MAX 3-SAT problem is given by the Hamiltonian

$$H^{\text{MW3S}}(\mathbf{s}) = \sum_{i=1}^{N-2} \frac{\omega_i}{8} [1 + (-1)^{c_i} s_i] [1 + (-1)^{c_{i+1}} s_{i+1}] \times [1 + (-1)^{c_{i+2}} s_{i+2}], \quad (6)$$

and it aims to maximize the weighted sum of satisfied clauses, where each clause consists of three propositional variables and is weighted by a random value ω_i , drawn from a uniform distribution on the interval $[0, 1]$. The c_i are either 0 or 1, chosen randomly, and determine whether the corresponding propositional variable is negated or not.

As VeloxQ solver can currently only handle second-order cost functions natively, we perform a reduction of the form

$$\pm s_i s_j s_k \rightarrow 3 \pm (s_i + s_j + s_k + 2s^{\text{aux}}) + 2s^{\text{aux}} (s_i + s_j + s_k) + s_i s_j + s_j s_k + s_i s_k, \quad (7)$$

where s^{aux} is an auxiliary variable [32]. This has a negative impact on the achievable problem sizes — we need to introduce one additional variable per each third-order term, which can quickly become prohibitive for large problems. However, as demonstrated by the results in the following sections, our solver is characterized by excellent scalability and can still handle HUBO problems of considerable size.

B. Benchmarks results

The left panel of the Figs. 7, 8 demonstrates the quality of the solutions obtained by VeloxQ solver, measured by the relative distance to two kinds of reference solution: obtained by the industry-standard CPLEX solver, and the SA algorithm. The right panel displays the corresponding time needed to obtain the solution.

Across a broad range of problem sizes, from a modest 100 variables, up to considerable 10^8 (2×10^8 variables after HUBO → QUBO reduction) VeloxQ demonstrates excellent performance, with reference gaps barely exceeding 3% in the structureless random case, and staying below 0.01% for the structured MAX 3-SAT problems. For the largest instances, with more than 10^6 variables it even outperforms best known solutions from other solvers. Comparison of runtimes with CPLEX demonstrates another advantage of VeloxQ solver, namely a predictable and consistent scaling with problem size, depending in principle only on the density of the problem and not any particular structure. Moreover, it significantly outperforms a native HUBO solver proposed by Kipu Quantum [6] (in terms of solution quality and tractable problem sizes), both in the case of computer simulation (red

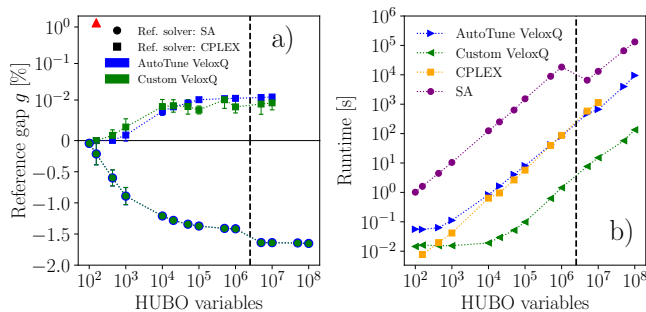


FIG. 8. Results for weighted MAX 3-SAT problem, given by Eq. (6). Contents of panels a) and b) are analogous to Fig. 7. Interestingly, CPLEX is able to exploit the structure of this class of problems to arrive at solutions of very high quality in a short time, in particular much better and faster than SA, the other reference solver. Moreover, its runtime scales with system size similarly to other solvers, in contrary to the case of structureless random instances. Nevertheless, VeloxQ is able to find solutions within 0.01% of CPLEX, in a time more than an order of magnitude shorter for instances larger than 10^4 variables and similar time for smaller problems.

diamonds) and execution on a real quantum computing platform from IBM (red triangles), providing a compelling argument for the usefulness of scalable, native QUBO solvers for higher-order problems.

VI. BENCHMARKS AGAINST SOLVERS WITH GROUND STATE CERTIFICATION

A generic QUBO problem is NP-hard, what implies that one cannot expect to find the ground state in polynomial time [59]. Nevertheless, provably optimal solutions, even for relatively small problems, are very valuable e.g. for assessing the performance of heuristic solvers, classical or quantum. Beyond benchmarking, exact solvers can be useful as subroutines in various hybrid algorithms, which decompose large scale problems into smaller, manageable chunks and then merge the individual solutions to find a low-energy state of the original problem [60].

We can distinguish essentially two types of exact QUBO/Ising solvers: brute-force (BF), which systematically explores the entire solution space, and specialized solvers that exploit the structure of a problem to guarantee reaching a ground truth. The former type is completely general, i.e. able to handle any QUBO instance regardless of its structure, but due to exponential growth of the solution space, it is only feasible for small problems, up to ~ 60 variables with sophisticated implementation [9] on modern hardware. In the latter case, the generality is sacrificed for the sake of efficiency, as the solvers are tailored to specific problem classes, and in turn can handle larger instances, sometimes even up to ~ 1000 variables for sufficiently simple topologies [12].

In this section we compare VeloxQ to solvers with ground state certification. First, VeloxQ is benchmarked

against the efficient implementation of the Brute-force solver [9]. Then, as examples of solvers exploiting the problem structure, we take into consideration BEIT’s solver [10, 11], and a method based on tropical tensor networks [3].

For examples of benchmark instances, as well as scripts that were used to generate them see the online repository [23].

A. Brute force solver

To illustrate the ability of VeloxQ to produce exact solutions of small problems, we compare it against a state-of-the-art, GPU-accelerated exact solver based on distributed brute-force approach [9]. Due to the exponential growth of the solution space, we consider problems with up to 60 variables with all-to-all connectivity and random couplings from range $[-1, 1]$, which can be solved exactly in approximately three days, utilizing 2×4 NVIDIA H100 GPUs. Remarkably, VeloxQ solver is able to match the quality of the solutions, i.e. obtain the same ground states, in a time of order of less than a millisecond, and essentially constant across all considered instance size. Detailed runtime results are presented in Fig. 9.

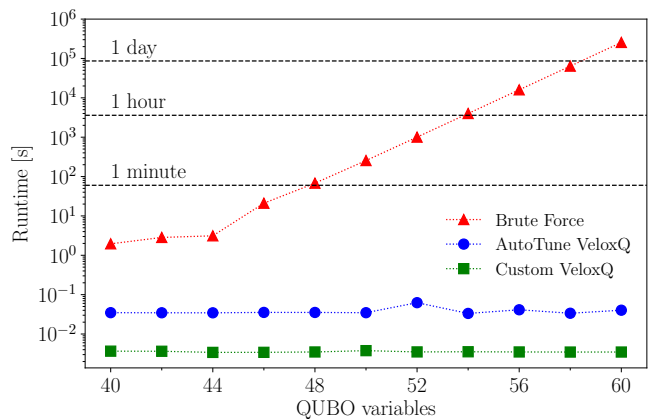


FIG. 9. Benchmark results of VeloxQ against state-of-the-art brute force solver (BF). Red triangles denote the BF results, whereas green squares and blue circles represent VeloxQ in Custom and AutoTune modes, respectively. Horizontal guidelines correspond to runtimes of one second, minute and hour. The runtime of the BF solver grows exponentially with the problem size, and for instances larger than 58 variables already exceeds the one-day mark. VeloxQ runs in constant time regardless of the variable count, and is able to match the quality of the BF solutions in a fraction of a millisecond.

As for the larger problem sizes in Fig. 5, we have already demonstrated that VeloxQ solver is capable of solving fully connected random (structureless) problems with up to 160 variables, on timescales of order of milliseconds, obtaining configurations of vanishing gap with respect to the ones obtained with the industry standard CPLEX solver, taking up to 20 minutes for the largest instances.

Note, that the maximal size of the problem was limited by the maximal clique embedded into the Pegasus graph P_{16} , and not by VeloxQ itself. Even though our solver does not offer a full certification of the ground state, most of the computed states in this case are likely to correspond to the ground truth, or at least be very close to it. At the same time, the considered problem sizes are far beyond capabilities of even the best brute-force algorithms.

B. Structure-exploiting solvers

The aim of the second part is to compare VeloxQ against the other class of exact solvers, in particular two concrete examples: an exact QUBO solver for the Chimera topology, offered as one of the commercial solutions by BEIT Inc. [10, 11], and a specialized solver based on the concept of Tropical Tensor Networks (TTN), introduced by [12]. Out of the three considered options, the proprietary BEIT’s solver, utilizing dynamic programming on bounded treewidth graphs, is the most limited in terms of the problem structure, as it can only handle problems with up to 1024 qubits, arranged in a 8×16 Chimera lattice. Furthermore, the QUBO couplings are restricted to integers in the range $[-31, 31]$. The TTN solver on the other hand, is more general, as it has no inherent limitations on the problem structure. It exploits the tensor network representation of the Ising Hamiltonian, and an observation that the zero-temperature limit of the partition function (which encodes the ground state energy) finds a natural formulation in terms of the so-called tropical algebra, which is a semiring over $\mathbb{R} \cup \{-\infty\}$ with addition replaced by the maximum operation, $x \oplus y = \max(x, y)$, and multiplication by the sum, $x \otimes y = x + y$. To obtain the corresponding state, automatic differentiation of the contraction outcome with respect to the tensor elements is performed. However, the complexity of tensor network contraction depends on the structure of the graph underlying the QUBO problem, and so the TTN solver performs best on relatively sparse problems.

To assert the fairness of the benchmark, we investigate the performance of these three solvers on a most general problem that can be handled by all of them, namely random Chimera instances, with sizes ranging from 128 ($C_{2,8}$) to 1024 ($C_{8,16}$) variables, and random integer couplings in the range $[-31, 31]$. Importantly, all solvers obtained the lowest energy solution for instances they were able to process, and so the comparison is based on the time needed to reach the solution. The results are presented in Fig. 10. Unfortunately, BEIT’s chimera solver is available only through the AWS cloud service, which makes the true runtime obscured by the request overhead [10, 11]. This is visible in the almost constant time to solution for instances of different sizes, so it is safe to assume that the actual computation time is much shorter. Regardless, effective runtime of approximately 2 minutes

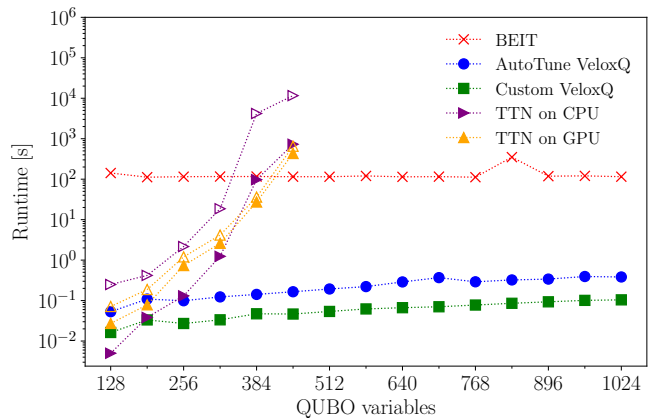


FIG. 10. Results of the benchmark of VeloxQ against solvers with ground state certification. Because all solvers found optimal solutions, we present only the runtimes. Red crosses denote the BEIT’s solver (obscured by AWS-related overheads), whereas blue circles and green squares represent VeloxQ in AutoTune and Custom modes, respectively. Filled triangles correspond to the TTN solver operating in energy-only mode and empty triangles include differentiating through the network to obtain the state. Contraction of a Tropical Tensor network seems to perform better on CPU than on GPU, at least for the energy computation mode. Even though VeloxQ solver is not guaranteed to return the ground state, for relatively small instances it is able to match the solution quality of specialized solvers, without sacrificing its generality and scalability.

for a certified ground truth on 1024 variables is still an impressive result. Both the BEIT’s solver and VeloxQ always return the state alongside its energy. However, the nature of TTN solver allows the user to choose between computing just the lowest energy, or both the energy and the corresponding state. In case of CPU computations, the latter option is significantly more time-consuming, with up to 3 orders of magnitude increase in the runtime. Interestingly, the GPU version does not suffer from such a dramatic slowdown, and the difference is almost negligible. Nevertheless, the complexity of tensor contraction increases rapidly with the problem size, and the TTN solver cannot handle the problems larger than 448 variables, corresponding to the $C_{8,7}$ Chimera graph.

VeloxQ solver consistently delivered optimal configurations in time much shorter than other solvers, except for the two smallest instances. These results further corroborate the claim about versatility of our solver and prove that if the ground state certification is not a strict requirement, it can be a very competitive alternative to both open-source and proprietary, specialized solutions, even for tasks of modest size.

VII. BENCHMARKS AGAINST PHYSICS-INSPIRED ALGORITHMS

Ever since the introduction of the SA algorithm in the 1980s [4], physical processes have been a rich source of inspiration for optimization algorithms. The idea is to map the optimization problem onto a physical system in such a way that the optimal solution corresponds to the lowest energy state, the so-called ground state, of the system. The system is then evolved according to the laws of physics, which, under appropriate conditions, favor the evolution towards low-energy states, akin to a ball rolling down a hill. In this section, we compare VeloxQ against two such state-of-the-art algorithms: Parallel Annealing (PA) and Simulated Bifurcation (SB), both having their roots in quantum adiabatic optimization.

PA is a variant of an annealing algorithm, designed to leverage the massively parallel nature of modern hardware accelerators [3]. As this is a classical algorithm, quantum spins are first replaced by classical spins, i.e. binary variables taking values ± 1 . These spins are subsequently relaxed to analog variables in the range $[-1, 1]$, emulating the idea of quantum superposition. Real classical spins are recovered via the sign function. The initial Hamiltonian is taken to be a convex function with easy to find global minimum (ground state), such as x^2 , and gradually shifted towards the target Hamiltonian, encoding the QUBO problem in Ising form. Since this is a simulation of a physical process, the analog spins do not update automatically — a suitable update rule must be chosen, and in the case of PA, it is a modified version of gradient descent, called the straight through estimator. For the purposes of this benchmark, we have created a custom, GPU-accelerated implementation of the PA algorithm. Technical details about PA are outlined in Appendix B.

Simulated Bifurcation Machine (SBM), an Ising solver implementing the idea of simulated bifurcation, was first proposed in 2016 [1] by Toshiba’s researchers, in form of a quantum computer consisting of a nonlinear oscillator network, adiabatically driven through a bifurcation point, and generating a superposition of quantum states that encodes the solution to a given combinatorial optimization problem. Since an appropriate quantum hardware is not yet readily available, and the simulation of a system working as general purpose quantum computer is computationally infeasible, in subsequent work a sequence of approximations was proposed, yielding a classical nonlinear dynamical system, governed by non-autonomous Hamiltonian equations of motion for harmonic oscillators with quartic nonlinearity, coupled via interaction that encodes the Ising problem [2]. The equation of an individual oscillator is also known under the name Duffing equation. This quantum-inspired solver leverages *classical* chaos and bifurcation to explore the solution space. Initially dominated by the parabolic minimum, post-bifurcation the energy landscape governing the evolution approximately encodes the local minima of

the Ising term, and thus the state of the system flows towards the low-energy solutions of the optimization problem, which then can be extracted by taking the sign of the analog variables. Furthermore, the chaotic behavior of this system of equations allows for a natural parallelization, as many initial states can be simultaneously evolved and the best solution can be selected. A commercial optimization suite based on a modified version of SBM algorithm [61], called SQBM+, is available from Toshiba Digital Solutions Corporation. Due to restricted solver access, however, we could not compare SQBM+ directly with VeloxQ. Instead, to the best of our abilities, we have replicated the original SBM algorithm, with GPU acceleration, and used it as a reference in the benchmarks. Technical details of SBM are summarized in Appendix B.

Since our physics-inspired solvers of choice are topology-agnostic, and so is VeloxQ, we have decided to benchmark them on three types of random instances with planted solutions: 3-regular 3-XORSAT equation planting [62], tile planting [63], and Wishart ensemble [64]. This way we can create an ensemble of problems with exactly known ground state energies, to compare the solvers without the need for a reference solver. Tile planting and Wishart ensemble instances are especially interesting, because apart from known ground state, their ‘hardness’ is continuously tunable, whereby ‘hardness’ we understand empirically observed typical difficulty, and not more precise computer-science notions of computational complexity. To streamline the preparation of problems, we employ an open-source Python-based tool called *Chook* [65]. For examples of benchmark instances, as well as scripts that were used to generate them see the online repository [23].

A. 3-regular 3-XORSAT equation planting

The starting point for the generation of 3-regular 3-XORSAT (3R3X) instances, is a system of linear equations over \mathbb{Z}_2 ,

$$\sum_{j=1}^n a_{ij}x_j = b_i \pmod{2}, \quad \text{for } i = 1, \dots, m, \quad (8)$$

where $x_j \in \{0, 1\}$ are binary variables. Specifically, we consider a case wherein each equation involves exactly three randomly chosen variables, and each variable appears in exactly three equations — hence the name 3-regular 3-XORSAT. Crucially, this system of equation can be solved in polynomial time, using e.g. Gaussian elimination. Its solution will be the planted ground state of the corresponding optimization problem. Translated into Ising form, these equations can be written as

$$\prod_{j:a_{ij}=1} s_j = (-1)^{b_i}, \quad \text{for } i = 1, \dots, m, \quad (9)$$

where $s_j \in \{-1, 1\}$ are the Ising spins. By squaring and adding the equations, we can obtain the relevant Ising Hamiltonian

$$\begin{aligned}
 H_{3R3X} &= \sum_{i=1}^m \left((-1)^{b_i} - \prod_{j:a_{ij}=1} s_j \right)^2 \\
 &\triangleq - \sum_{i=1}^m (-1)^{b_i} \prod_{j:a_{ij}=1} s_j \\
 &= - \sum_{i=1}^m J_i s_{i_1} s_{i_2} s_{i_3},
 \end{aligned} \tag{10}$$

where \triangleq denotes equality up to an irrelevant constant term. For each equation i there are exactly three indices $j \in \{i_1, i_2, i_3\}$ such that $a_{ij} = 1$, and the couplings $J_i = (-1)^{b_i}$ encode the right-hand sides of the equations. Thus, the cost function for this problem is composed of m , 3-body Ising terms, which can be reduced to the usual 2-body variant with the reduction scheme given by Eq. (7), yielding finally a $2m$ variable QUBO/Ising problem. Obtaining an optimal solution of such a problem is a challenging task for QUBO solvers, because the low-lying excited states are far away (in the sense of Hamming distance) from the ground state, and so it is easy to get stuck in a local minimum [62].

Using the procedure outlined above, we have prepared a set of random 3R3X instances with sizes ranging from 10 to 100×10^3 variables, 10 per each size. In Fig. 11 we present the averaged results of optimality gap and runtimes. The error bars are omitted for the sake of clarity. The outcome of this computational experiment demonstrates that VeloxQ stands its ground among modern, physics-inspired solvers for combinatorial optimization problems. In particular, the two sets of results denoted as ‘‘Custom VeloxQ 1’’ and ‘‘Custom VeloxQ 2’’ are a testament to the flexibility of our product, which, using knowledge about a given type of problem, can be tuned to either favor the quality of solution or the runtime, depending on the end user’s needs.

B. Tile planting

Tile planting (TP) instances are certain type of square lattice Ising models with periodic boundary conditions. They are constructed by partitioning the lattice graph $G = (V, E)$ into unit cell tiles $C \in \mathcal{C}$, creating a checkerboard pattern. Each tile C consists of four vertices, and each vertex is shared by two tiles. For each subgraph, the Hamiltonian is defined as

$$H_{\text{tiled}}^C = - \sum_{\langle i,j \rangle \in E[C]} J_{ij} s_i s_j, \tag{11}$$

where the sum runs over all edges in the tile. The full Hamiltonian is then given by the sum over all tiles

$$H_{\text{tiled}} = \sum_{C \in \mathcal{C}} H_{\text{tiled}}^C. \tag{12}$$

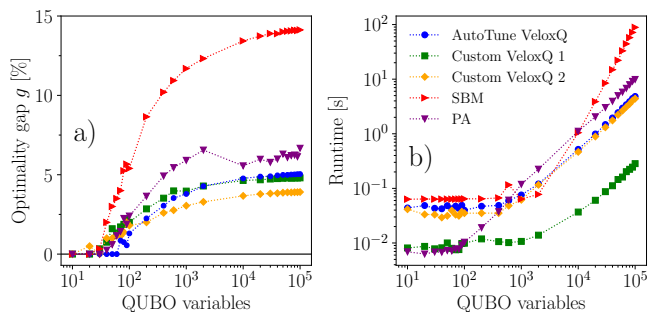


FIG. 11. Results of the benchmarks of VeloxQ against the PA and SBM solvers on 3R3X instances, **a)** optimality gap and **b)** runtime in seconds. We show data points for VeloxQ solver in AutoTune mode (blue circles) and two manually optimized versions, for runtime (green squares) and for quality of the solution (orange diamonds). For problems with up to 50 variables, the optimization procedure consistently returns ground state solution. In case of larger instances optimality gap increases, but does not exceed 5%. High-quality solutions are also delivered by parallel annealing (purple inverted triangles), with a runtime that differs only by a constant factor from VeloxQ in the default version. The SBM (red right-facing triangles) underperforms significantly in this benchmark, with both optimality gaps growing rapidly with systems size, finally reaching almost 15% and runtimes demonstrating worse scaling in the 1000+ variable regime. Moreover, this result was obtained after extensive tuning of the SBM parameters, which should be compared with the satisfying performance of VeloxQ out-of-the-box (AutoTune mode).

Importantly, if all the tile Hamiltonians have a common ground state, then the full Hamiltonian will also share the same ground state configuration. This fact allows for a construction of instances with known minimal energy value [63]. The remaining freedom of choosing the coupling constants J_{ij} allows one to tune the level of frustration in the system, and thus the hardness of the problem. Without loss of generality (due to gauge freedom), we can assume that our target ground state is a fully polarized, ferromagnetic state $\mathbf{t} = (+1, +1, \dots, +1)$. Then, four types of nonequivalent tiles are introduced, C_i for $i = 1, \dots, 4$, such that tile of type i has i ground states, always including the fully polarized ferromagnetic states. A complete TP instance is obtained by randomly selecting the tile type for each tile, according to a given probability distribution (p_1, p_2, p_3, p_4) . It was shown that this three-dimensional space of control parameters allows for a fine-tuning of the problem hardness [63]. For the purpose of our benchmark, we choose a particular 1-dimensional subspace in this parameter space, given by $S = \{(0, p_2, 0, 1 - p_2) \mid p_2 \in [0, 1]\}$, as it is the most versatile one-dimensional subspace out of those exhibiting an easy-hard transition. In general, there are three such subspaces, since it is the fraction of C_2 tiles that control the problem hardness.

Results for two kinds of TP instances from $C_2 - C_4$ subspace, easy ($p_2 = 0.2$) and hard ($p_2 = 0.8$), on square

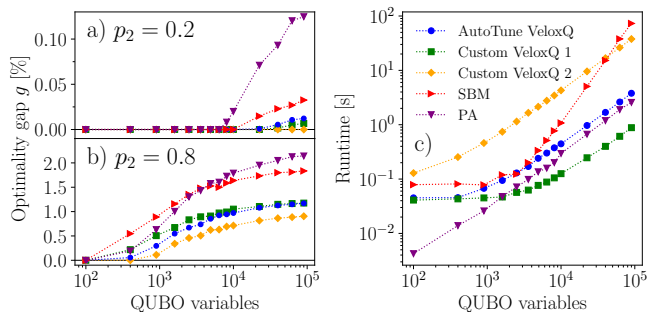


FIG. 12. Results of the benchmarks of VeloxQ against the PA and SBM solvers on TP instances, optimality gap in the **a)** ‘easy’ regime ($p_2 = 0.2$) and **b)** ‘hard’ ($p_2 = 0.8$) regime. Panel **c)** shows the runtimes only in the ($p_2 = 0.8$) case, since value of p_2 does not affect the runtime. We show data points for VeloxQ solver in AutoTune mode (blue circles) and two manually optimized versions, for runtime (green squares) and for quality of the solution (orange diamonds). Quality-optimized VeloxQ is the only solver able to obtain ground states for all sizes in the easy regime, and consistently outperforms other solvers in the hard regime. If the runtime is the main concern, it is possible to decrease it by a factor of 3, while still maintaining the optimality gap below 1% in the $p_2 = 0.8$ case. Parallel annealing (purple inverted triangles) behaves similarly to VeloxQ in most of the easy regime ($N \leq 10^4$), but it produces noticeably worse (although still of high quality) solutions beyond it. Nevertheless, it has an advantage for problem sizes below 10^3 variables, since it is able to deliver solutions remarkably fast. After an appropriate hyperparameter tuning, the SBM (red right-facing triangles) in its original formulation demonstrates competitive performance, emerging on top of PA in the easy regime, and on par with it in the hard regime. Still, VeloxQ solver is able to outperform it in both cases.

lattices with side of length $L \in [10, 300]$, are presented in Fig. 12. Since these instances are also randomized, for each size we have prepared 10 copies and averaged the results, again omitting the error bars for clarity. With a little optimization, VeloxQ can consistently find the ground state for even the largest instances with 90k variables, in the easy regime. Nevertheless, the default settings are already sufficient to outperform PA and SBM in both difficulty regimes.

C. Wishart ensemble

Both 3R3X and TP problems are formulated as Ising models with rather sparse coupling matrix. On the other hand, Wishart ensemble planting (WP) provides a way to construct instances with known ground state energy and all possible pairwise couplings. We begin by choosing a ground state configuration, which again can be chosen as the fully polarized ferromagnetic state \mathbf{t} , and subsequently concealed by gauge randomization. Let now W

be an $N \times M$ real matrix, such that

$$W^T \mathbf{t} = 0, \quad (13)$$

i.e. describing a set of M linear equations in N variables, with the ground state \mathbf{t} as the solution. The ratio $\alpha = M/N$ of equations to variables will determine the hardness of the problem. This property guarantees, that an Ising model defined as

$$H_{\text{WP}} = -\frac{1}{2} \sum_{i,j} J_{ij} s_i s_j, \quad (14)$$

with

$$\tilde{J} = -\frac{1}{N} W W^T \quad (15)$$

$$J = \tilde{J} - \text{diag}(\tilde{J}), \quad (16)$$

has a ground state $\mathbf{s} = \mathbf{t}$ with energy

$$E_0 = -\frac{1}{2} \mathbf{t}^T J \mathbf{t} = \frac{1}{2} \text{Tr}(\tilde{J}). \quad (17)$$

The actual construction of the matrix W is carried out by sampling each column vector \mathbf{w} from a multivariate Gaussian distribution with zero mean, and covariance matrix Σ , given by

$$\Sigma = \frac{N}{N-1} \left[\mathbb{1}_N - \frac{1}{N} \mathbf{t} \mathbf{t}^T \right], \quad (18)$$

where \mathbf{t} denotes our planted ground state. It can be shown, that $\mathbf{w}^T \mathbf{t} = 0$ for all columns of W , which concludes the construction. The resulting random matrix $W W^T$ follows a matrix version of χ^2 distribution, known as the Wishart distribution [64].

When the equation-to-variable ratio $\alpha \leq 1$, there exists a stable set of paramagnetic configurations (uncorrelated with planted ground state), which essentially behaves as a local minimum trap for classical heuristic solvers. The generated problem is likely to be solved with high probability only if the initialization happens by chance within the ground-state basin of attraction (assuming no escape mechanism is present). Interestingly, the parameter α influences the size of this basin, with higher values increasing the chances of a successful solution due to a fortunate initialization. Therefore, in Fig. 13 we present the benchmark results for two regimes, $\alpha = 0.2$ (hard) and $\alpha = 1.0$ (easy). Since the WP instances are fully connected, the number of variables is limited to $N \leq 5000$. For each of the presented system sizes, ten instances were generated and the results averaged. An interesting observation regarding the nature of ‘hardness’ in the Wishart ensemble, can be made on the basis of the results presented in Fig. 13. In the easy regime, each solver produces a ground state solution up to a certain, solver-dependent system size, after which the solution quality rapidly degrades, finally settling at the optimality gap of around 10–15%. It is worth noting, that VeloxQ can be

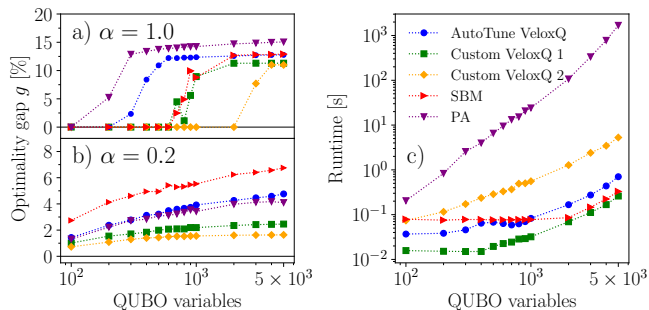


FIG. 13. Results of the benchmarks of VeloxQ against the PA and SBM solvers on WP instances, optimality gap in the **a)** ‘easy’ regime ($\alpha = 0.2$) and **b)** ‘hard’ ($\alpha = 0.8$) regime. Similarly to TP case, panel **c)** shows the runtimes only in the $\alpha = 0.2$, since value of α does not affect the runtime of considered solvers. We see that the default VeloxQ settings, denote by AutoTune mode (blue circles), are not optimal for this class of problems. However, thanks to its flexibility, it is straightforward to tune the performance to deliver simultaneously better solutions and faster runtimes (green squares) or to focus just on the quality of the sampled states (orange diamonds). The PA (purple inverted triangles) underperforms in the easy regime, but is able to match the default VeloxQ performance in the hard regime (although with a noticeable increase in runtime). On the other hand, Simulated Bifurcation yields excellent results remarkably fast in the easy regime, but its performance deteriorates in the hard regime.

tuned to produce optimal configurations for the broadest range of system sizes (up to $N = 1000$), and the best results beyond this range. The hard regime is characterized by a qualitatively different behavior, with all considered solvers failing to deliver ground state solutions even for the smallest instances with $N = 100$. Even though the optimality gap increases with system size, the change is not as drastic as in the easy regime, and VeloxQ is able to maintain the gap below 5% for default setting and at the level of 1% for the quality-optimized version.

VIII. BENCHMARKS AGAINST MODERN B&B ALGORITHMS

Branch and Bound (B&B) [66] is a foundational framework for solving combinatorial optimization problems, including QUBO and Ising model formulations. Over the years, numerous enhancements have been developed to improve its efficiency, focusing on solution-space pruning techniques and accelerating convergence. In this section, we compare VeloxQ solver against selected B&B algorithms, including modern refinement of the algorithm developed by Quantagonia [14].

This comparison aims to underscore the differences in performance and scalability among the tested approaches, with a focus on demonstrating the potential advantages of VeloxQ in multiple domains. Key evaluation metrics include solution quality, computational cost,

and scalability to larger problem instances. The selected B&B methods construct solutions iteratively by solving incremental subproblems, progressively appending spins to build the solution. This approach aligns naturally with the binary decision tree representation over fixed $\{-1, 1\}$ spin-like values. Memory efficiency is enhanced by maintaining a pool of stored partial solutions, while subtree selection is guided by heuristic bound functions. These functions are critical components that determine the effectiveness of each method. The choice of bound functions allows for a diverse range of methods [67], including modern implementations adopted in the industry [13, 14, 68]. For examples of benchmark instances, as well as scripts that were used to generate them see the online repository [23].

As the reference methods used for comparison to VeloxQ, we implemented search algorithms based on two bound functions. A standard approach to B&B algorithms is represented by B_{base} [24], in which the prefix subproblem energy is directly used as the bounding premise. The modern refinement of B&B is denoted as B_{SPD} , in which the bound relates to the subproblems transformed to semi-positive definite matrix formulations. This allows estimating energy minima for subproblems. Such an approach, reflecting selected state-of-the-art techniques and modern advancements in optimization, aligns with the algorithmic outline described in [14]. SPD-based methods are highly adaptable, including successful applications to solving QUBO and Ising model problems in various computational environments, including quantum computing [13, 69]. Our implementation, optimized for the CUDA runtime environment to facilitate a practical comparison with VeloxQ, is detailed in Appendix C.

The computation time and solution optimality achieved by B&B methods using B_{base} and B_{SPD} , in comparison to VeloxQ, are presented in Fig. 14. To ensure a diverse range of matrix sizes for benchmarking, pseudo-random dense matrices were generated. While B&B methods can operate on sparse matrices, the computational overhead is primarily determined by matrix size, which corresponds to the height of the search tree. As such, dense matrices were chosen as the most suitable for this study. Matrix sizes were distributed within the range of 1000 to 10 000. Problem instances with smaller matrices were excluded, as they do not reflect the scope of the method and are often more efficiently solved by brute-force approaches. Sizes up to 10^4 were chosen as a practical upper limit for our computational setup, with individual runs requiring less than three hours.

The results of the benchmark show that VeloxQ obtains comparable or better solution quality than both B&B algorithms. The most profound difference concerns the runtime: even for the smallest instances VeloxQ is two orders of magnitude faster than the B&B algorithms, and this difference increases with the growing instance size.

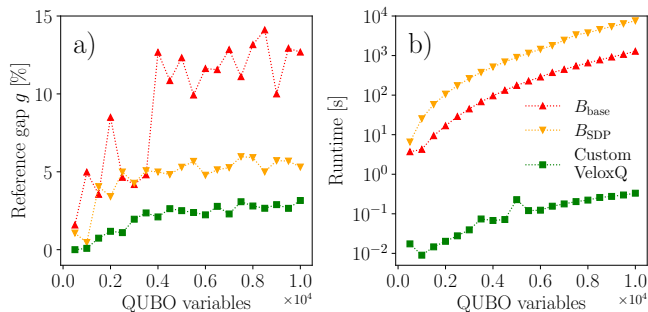


FIG. 14. B&B methods: **a)** reference gap and **b)** runtimes for dense, pseudo-random instances with up to 10 000 variables. For instances exceeding 4 000 variables, the B_{base} variant became unreliable, pruning significant subtrees, and resulting in an optimality gap exceeding 10%. In contrast, the B_{SDP} variant maintained consistency, achieving results within twice the energy gap of VeloxQ. This reliability came at the cost of computational efficiency, with B_{SDP} being approximately $10\times$ slower than B_{base} . VeloxQ, however, consistently achieved comparable or better accuracy while being $100\text{--}300\times$ faster, positioning it as the superior choice for instances in this size range. VeloxQ values in the plot were computed using Custom VeloxQ with 150 internal states. Reference energies were approximated by invoking VeloxQ iteratively, starting with 1000 states and increasing by 50% until the best energy value was repeated three consecutive times, ensuring robust convergence.

IX. SUMMARY

The aim of this paper was to provide a versatile set of benchmarks that allow to compare VeloxQ solver to the most promising QUBO solvers based on emerging technologies. All approaches to QUBO optimization were taken into consideration. Solvers based on emerging quantum technologies were investigated: D-Wave quantum annealers, and the algorithm developed for digital quantum computers by Kipu Quantum. We also included physics-inspired algorithms based on tropical tensor networks. Among conventional classical algorithms we chose BEIT’s solver, the efficient implementation of the brute-force algorithm [9], and the modern development of Branch and Bound algorithm. The broad choice of test instances accommodates strengths of particular solvers: For this reason, in addition to problems with all-to-all connectivity, we extensively studied problems that fit natively on quantum annealer’s topology, or in case of digital quantum algorithms transpile directly on digital quantum hardware. In this way the performance comparison is not biased by an unfavorable instance choice for the solvers VeloxQ is benchmarked against. This methodology allows highlighting versatility of VeloxQ: Using default set of parameters it was usually able to obtain solution of the problem with quality and time either matching those of competitive solvers or surpassing them. In all remaining cases, a straightforward tuning of VeloxQ parameters allowed to achieve the same or better

results.

Comparison with the annealers shows that, for instances natively fitting topology of an annealer, quality of VeloxQ solutions is comparable to that provided by the annealer. We tested also instances of increasing variable size, comparing VeloxQ to D-Wave hybrid solver, in which annealers are used to speed up some subroutines. Our test show that solution quality of VeloxQ is comparable that of D-Wave hybrid solver, but achieved in much shorter time. VeloxQ is also able to handle large-scale instances that are intractable to the D-Wave hybrid solver. D-Wave annealers excel in time-to-solution for small instances that naturally align with the topology of D-Wave devices, offering an advantage in such cases. However, in this case VeloxQ can outperform D-Wave solvers upon appropriate hyperparameter readjustment. The considerable advantage of VeloxQ over D-Wave solvers is clearly visible for problems with all-to-all connectivity: Quality of D-Wave solver solution significantly deteriorates with the growing problem size, whereas VeloxQ produces highly accurate solution in shorter time.

VeloxQ performs also favorably for weighted MAX-3-SAT instances that are natively formulated in terms of HUBO problems. Based on comparison with the data presented in [6], our results indicate that VeloxQ outperforms the quantum-digital algorithm developed by Kipu Quantum. Although the Kipu’s algorithm is severely restricted by the number of qubits of the present gate-based quantum computers, we decided to test VeloxQ in regime of ultra-large instances, intractable to digital quantum solutions in the foreseeable future. VeloxQ solved QUBO instances consisting of up to $\sim 200 \times 10^6$ variables outperforming significantly the reference solvers CPLEX and SA in terms of both solution quality, and time-to-solution.

The fact that we chose CPLEX as a solver providing reference solutions serves as an indirect comparison of VeloxQ to the industry-standard QUBO solution methods. It is worth to point out that on the chosen set of instances the solution quality of both solvers is comparable, whereas VeloxQ runtime is supreme over that of CPLEX, as for the largest instances in our benchmarks CPLEX was not able to produce a meaningful solution in a manageable time.

Comparison of results of VeloxQ and solvers providing solution certification reveals that for modest size problems VeloxQ produces optimal solutions in time shorter than the considered solvers. This conclusion holds for the efficient implementation of the brute force solver [9], as well as solvers exploiting underlying structure of the optimization problem based on tropical tensor networks as well as BEIT’s solver (note that in this case the comparison is not straightforward as, due to the cloud access, we could not get obtain the bare solver runtime).

VeloxQ was benchmarked against physical-inspired algorithms parallel annealing [3], and simulated bifurcation [1]. In these benchmarks instances with planted solutions were used generated accordingly to 3-regular 3-XORSAT

equation planting, tile planting, and Wishart ensemble planting. Simulated bifurcation algorithm is advantageous over VeloxQ only for certain type of Wishart instances. Parallel Annealing produces solutions of comparable quality to the ones of VeloxQ, but for small instance sizes the time-to-solution of Parallel Annealing can be shorter than that of VeloxQ with default parameters, except for Wishart instance, for which Parallel Annealing requires longer runtime for all problem sizes. As in the case of D-Wave, the change of default VeloxQ parameters can lead to the superior runtime over that of Parallel Annealing and Simulated Bifurcation algorithms, while the quality of solutions of these algorithms remain similar.

Performance of VeloxQ was also benchmarked against the modern development of the conventional Branch and Bound algorithm proposed by Quantagonia. VeloxQ obtains solutions of comparable or better quality, and outperforms this family of algorithms both in terms of the runtime: It is at least two orders of magnitude than the considered algorithms.

The results presented in this paper allow us to conclude that VeloxQ is an efficient large-scale QUBO solver offering competitive performance in time-to-solution and solutions quality, and superiority in the regime of ultra-large instances, to the leading quantum and classical optimization methods. Moreover, it offers unparalleled versatility across diverse instance problem structures and sizes. Thus, it presents a compelling alternative to the existing QUBO solution tools.

ACKNOWLEDGMENTS

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, titled *VeloxQ: Utilizing Dynamic Systems in Decision-Making Processes Based on Knowledge Acquired Through Machine Learning, Across Various Levels of Complexity, in Industrial Process Optimization*, that aims to develop digital solutions for solving combinatorial optimization problems. The focus is on leveraging quantum-inspired algorithms and artificial intelligence to enhance decision-making processes in industrial optimization contexts.

Appendix A: Transformations between QUBO and Ising model formulations

This appendix follows conventions of [26]. To represent a QUBO problem as an instance of Ising model one transforms the binary variables $x_i = \frac{1}{2}(1 + \sigma_i)$. In this case the parameters of the related Ising model are given

by

$$h_i = \sum_{j=1}^N Q_{ij}, \quad (\text{A1})$$

$$J_{ij} = \begin{cases} Q_{ij}, & i \neq j, \\ 0, & i = j, \end{cases} \quad (\text{A2})$$

and values of the relevant quadratic functions are related as

$$Q(\mathbf{x}) = \frac{1}{2}H(\mathbf{s}) - \frac{1}{2}C, \quad (\text{A3})$$

with $C = \sum_{i=1}^N \sum_{j=i+1}^N Q_{ij} + \sum_{i=1}^N Q_{ii}$.

On the other hand, an Ising model can be represented as a QUBO problem via discrete variable transformation $s_i = 2x_i - 1$. The parameters of the related QUBO model are given by

$$Q_{ij} = \begin{cases} J_{ij}, & i \neq j, \\ h_i - \sum_{l=1}^N J_{il}, & i = j, \end{cases} \quad (\text{A4})$$

and the relation between the functions is

$$H(\mathbf{s}) = 2Q(\mathbf{x}) + C, \quad (\text{A5})$$

with $C = -\sum_{i=1}^N \sum_{j=i+1}^N J_{ij} + \sum_{i=1}^N h_i$.

Appendix B: Details about Parallel Annealing and Simulated Bifurcation

Simulated annealing (SA) is a well-known heuristic optimization algorithm that employs thermal fluctuations to escape local minima. During the evolution of the system, the temperature is gradually decreased, leading to decreasing fluctuations and a higher probability of accepting only downhill moves. Ultimately, the system settles in some low-energy state. On the other hand, Parallel Annealing (PA) is inspired by the quantum adiabatic annealing, in which the system's Hamiltonian is time-dependent, and gradually switches from the initial form, with known ground state, to the target Hamiltonian. According to the Adiabatic Theorem [70] from quantum mechanics, if this evolution is performed slowly enough, the system will remain in the ground state of the Hamiltonian at all times, thus reaching the ground state of the target Hamiltonian and solving the optimization problem. PA adapts this idea to the realm of classical heuristic optimization, by considering a time-dependent, classical Hamiltonian of the form

$$H(t) = \lambda(t)H_{\text{initial}} + H_{\text{target}}, \quad (\text{B1})$$

where t plays the role of the time parameter, and $\lambda(t)$ is a function that gradually decreases from a sufficiently

large value to zero. By analogy with the quantum case, H_{initial} is chosen to be a function with easily obtainable global minimum. In the original work [3], the authors employed a quadratic function $H_{\text{initial}} = 1/2 \sum_i x_i^2$. Since there is no actual physical process that would evolve towards low-energy configurations, a manual update scheme has to be introduced. An intermediate, analog spin variables x_i , taking values in the interval $[-1, 1]$, are used as a proxy for the true binary variables. The classical spins can be in turn retrieved via the sign function, $s_i = \text{sign}(x_i)$. The update rule for the analog variables is given by a gradient descent step,

$$x_i(t + \Delta t) = x_i(t) - \eta \nabla H(t), \quad (\text{B2})$$

with the gradient computed as a function of continuous variables, but the target part evaluated with binarized spins:

$$\begin{aligned} \nabla H(t) &= \lambda(t) \nabla H_{\text{initial}} + \nabla H_{\text{target}} \\ &= \lambda(t) \mathbf{x} + \mathbf{J} \mathbf{s} + \mathbf{h}. \end{aligned} \quad (\text{B3})$$

Drawing inspiration from Neural Network training, clipping [71],

$$\mathbf{x} \rightarrow \max(-1, \min(1, \mathbf{x})), \quad (\text{B4})$$

and momentum [72]

$$\mathbf{m}(t + 1) = \alpha \mathbf{m}(t) - \eta \nabla H(t), \quad (\text{B5})$$

$$\mathbf{x}(t + 1) = \mathbf{x}(t) + \mathbf{m}(t + 1), \quad (\text{B6})$$

are also utilized to stabilize the optimization process.

Another optimization algorithm inspired by quantum adiabatic computing is the Simulated Bifurcation Machine (SBM) [2]. It is a classical approximation of a network of nonlinear quantum oscillator, described by a chaotic system of Hamilton equations:

$$H = \frac{a_0}{2} \sum_{i=1}^N p_i^2 + \sum_{i=1}^N \left(\frac{q_i^4}{4} + \frac{a_0 - a(t)}{2} q_i^2 \right) - c_0 \sum_{i=1}^N \left(h_i q_i + \frac{1}{2} \sum_{j=1}^N J_{ij} q_i q_j \right), \quad (\text{B7})$$

$$\dot{q}_i = \frac{\partial H}{\partial p_i} = a_0 p_i, \quad (\text{B8})$$

$$\dot{p}_i = -\frac{\partial H}{\partial q_i} = -[q_i^2 + a_0 - a(t)] q_i + c_0 \left(\sum_{j=1}^N J_{ij} q_j + h_i \right). \quad (\text{B9})$$

This process can be viewed as the dynamics of particles with mass a_0^{-1} , positions $q_i \in \mathbb{R}$, and momenta $p_i \in \mathbb{R}$, in a time-dependent single-particle potential and interacting through Ising-like interactions. The matrix J and vector h represent the Ising minimization problem being solved. Hyperparameters a_0 and c_0 are typically taken to be $a_0 = 1$ and $c_0 = 1/\lambda_{\text{max}}$, where λ_{max} is the largest eigenvalue of J (to bound the extremal values of Ising term), while $a(t)$ is a linearly increasing time-dependent function that drives the system through the bifurcation point, which occurs roughly when $a(t) = a_0$. After the bifurcation, the system's energy landscape approximately encodes the local minima of the Ising term, leading the particles to converge toward low-energy solutions of the binary optimization problem. These solutions can be extracted by taking $s_i = \text{sign}(q_i)$. Additionally, the chaotic dynamics of the SBM equations result in sensitivity to initial conditions, allowing many independent replicas to be run simultaneously from different starting points, further enhancing efficiency and enabling massive parallelization.

Appendix C: Bound functions used in B&B methods

The remarkably straightforward B&B method we adopt utilizes the subproblem energy as the primal bound function. For an Ising model instance defined by (J, h) , given a state s and a subproblem $U \subseteq \{1, \dots, n\}$, the bound is computed as:

$$B_{\text{base}}(J, h, s, U) = \sum_{i \in U} h_i s_i + \sum_{\substack{i, j \in U \\ i < j}} J_{ij} s_i s_j. \quad (\text{C1})$$

While this method is efficient when applied to batches of subproblems incrementally, it does not account for the remaining subsets of spins outside U . This limitation can be addressed by employing alternative bound functions.

SPD formulation. The improved B&B method involves transforming each evaluated subproblem into a form based on a semi-positive definite (SPD) matrix. The relaxed solutions to such subproblems can be used as bound function to B&B algorithm – which follows modern algorithm outline published in [13, 14]. We are going to refer to a generalized, matrix-based formulation

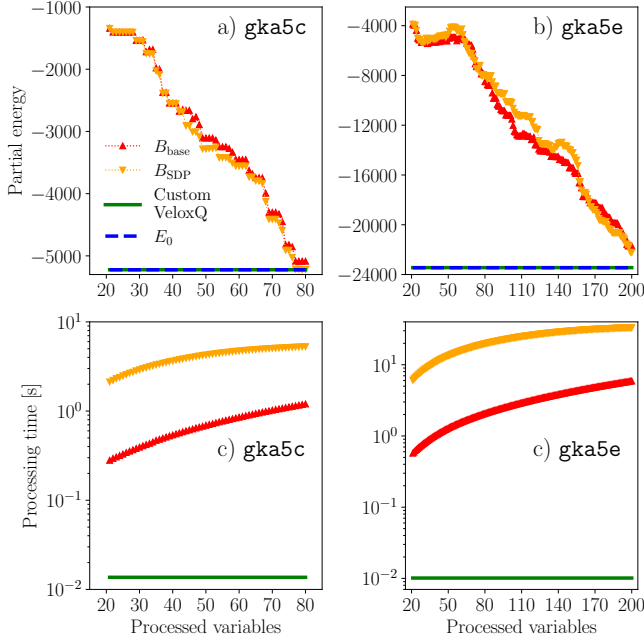


FIG. 15. The B&B progress for selected instances demonstrates that the dynamics of B_{base} and B_{SPD} are roughly similar. B_{base} achieves better intermediate states for **gka5e**, but fails to recognize the more optimal state found by B_{SPD} . While the time characteristics of B_{SPD} is slower initially, as the size of the remaining subproblems decreases, the subsequent iterations of B_{SPD} become progressively faster. However, the overall execution time of either B&B method exceeds the completion time of VeloxQ remarkably quickly, underscoring the significant performance advantages of VeloxQ.

of Ising model for J defined as a symmetric connection matrix:

$$E(J, h, s) = s' J s / 2 + s' h. \quad (\text{C2})$$

This formulation can be extended to matrices that describe numerical problems beyond Ising model, notably - with non-zero diagonal of J . In order to reformulate the problem with a SPD matrix, we shift all the diagonal elements by

$$d(J) = \max(0, -\text{eigmin } J) + \varepsilon, \quad (\text{C3})$$

where $\varepsilon > 0$ constant is included to ensure numerical stability. This ensures that $\tilde{J} = J + d(J)\mathbb{1}$ is a positive definite matrix.

Using \tilde{J} matrix instead of J introduces an offset to the energy that is constant with respect to s

$$E(\tilde{J}, h, s) = E(J, h, s) + d(J) \cdot n, \quad (\text{C4})$$

which means that the binary problems would have the same extrema.

Relaxation of this problem into a continuous domain $r \in \mathbb{R}^n$ leads to the following quadratic SPD form:

$$E(\tilde{J}, h, r) = r^T (\tilde{J} / 2) r + r^T h \rightarrow \min. \quad (\text{C5})$$

Notably, the equality (C4) is ensured only for binary states r . While the SPD formulation does not directly solve the original Ising model, it provides a tractable heuristic, resulting in efficient approximation of solutions.

	N	B_{base}		B_{SPD}		VeloxQ	
		gap	t [s]	gap	t [s]	gap	t [s]
gka1a	50	0.0%	0.41	0.0%	0.98	0.0%	0.01
gka2a	60	0.0%	0.68	0.0%	1.71	0.0%	0.01
gka3a	70	0.7%	1.18	0.7%	2.6	0.0%	0.01
gka4a	80	0.0%	1.1	0.0%	3.44	0.0%	0.01
gka5a	50	0.0%	0.44	0.0%	1.0	0.0%	0.01
gka1c	40	0.0%	0.28	0.0%	0.5	0.0%	0.01
gka2c	50	0.0%	0.5	0.0%	1.1	0.0%	0.01
gka3c	60	1.85%	0.66	0.0%	1.57	0.0%	0.01
gka4c	70	0.0%	0.88	0.92%	2.43	0.0%	0.01
gka5c	80	2.62%	1.04	0.08%	3.31	0.0%	0.01
gka1d	100	4.34%	1.53	0.0%	5.72	0.24%	0.01
gka2d	100	2.27%	1.6	7.73%	6.01	0.0%	0.01
gka3d	100	1.77%	1.74	1.28%	5.9	0.33%	0.01
gka4d	100	5.48%	1.65	2.5%	5.82	0.0%	0.01
gka5d	100	3.16%	1.57	0.64%	5.8	0.0%	0.01
gka1e	200	1.17%	5.41	10.9%	27.6	0.34%	0.05
gka2e	200	11.17%	15.41	6.76%	47.51	0.0%	0.01
gka4e	200	5.72%	5.57	3.8%	27.8	0.0%	0.01
gka5e	200	7.5%	5.49	5.28%	27.53	0.03%	0.01

Table I. Computational results were obtained for selected GKA instances [73]. Files and optimal energies from the Big Mac Library [74] were converted to Ising formulations. The B&B benchmarked methods were configured to maintain 2^{20} states in memory. Using the B_{SPD} bound function resulted in execution times that were 2 to 6 times longer compared to baseline B&B. However, it achieved smaller energy gaps in 75% of cases, especially for larger instances. In comparison, a simple setup of VeloxQ with 150 internal states and AutoTune was used. VeloxQ demonstrated notably better performance, achieving superior results with significantly lower runtime. One instance from the set (**gka1d**) was solved using B&B with B_{SPD} , while tested VeloxQ did not reach the ground state.

Solution. The relaxed SPD problem can be solved using methods such as proximal operators. The optimal relaxed solution is given by:

$$r^*(J, h) = -\tilde{J}^{-1}h. \quad (\text{C6})$$

Since \tilde{J} is positive definite, it is non-singular, ensuring the existence and uniqueness of $r^*(J, h)$.

Bound function for B&B. For the complete Ising problem (J, h) , the heuristic score can be computed as:

$$B_{\text{SPD}}(J, h, s) = H(\tilde{J}, h, -\tilde{J}^{-1}h). \quad (\text{C7})$$

In the context of B&B, the bound function is supposed to estimate the optimal energy of a remaining subproblem. For a subset of spins $U \subseteq \{1, \dots, n\}$, let $\tilde{U} = \{1, \dots, n\} \setminus U$. Then:

$$B_{\text{SPD}}(J, h, s, U) = B_{\text{SPD}}(J_{\tilde{U} \times \tilde{U}}, h_{\tilde{U}}, s), \quad (\text{C8})$$

where $J_{\tilde{U} \times \tilde{U}}$ and $h_{\tilde{U}}$ represent the reduced matrices and vectors for the remaining subtrees in the search process. This approach involves solving a linear system, which can be practically implemented with batches of states through matrix factorization techniques, such as Cholesky decomposition, chosen based on the sparsity of the \tilde{J} matrix. Additionally, determining the shift pa-

rameter δ requires computing the minimum eigenvalues of the matrix. In our simulations, we employ ARPACK-inspired algorithms [75], adapted for the CUBLAS framework, to ensure efficient computation. Alternative methods, tailored to specific runtime environments, include making this approach feasible for quantum computing applications [69].

To validate this approach, we also performed benchmarks using the GKA dataset [73], which is a recognized standard for evaluating B&B methods for quadratic problems [24], included in the Big Mac Library [74]. The results of these benchmarks, detailing the performance of the selected B&B approaches, are presented in Table I and Fig. 15.

-
- [1] H. Goto, Bifurcation-based adiabatic quantum computation with a nonlinear oscillator network, *Scientific Reports* **6**, 21686 (2016).
- [2] H. Goto, K. Tatsumura, and A. R. Dixon, Combinatorial optimization by simulating adiabatic bifurcations in nonlinear hamiltonian systems, *Science Advances* **5**, eaav2372 (2019).
- [3] M. Jiang, K. Shan, C. He, and C. Li, Efficient combinatorial optimization by quantum-inspired parallel annealing in analogue memristor crossbar, *Nature Communications* **14**, 5927 (2023).
- [4] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by Simulated Annealing, *Science* **220**, 671 (1983).
- [5] Quantumz.io, *VeloxQ QUBO solver* (2025), accessed: 2025-01-31.
- [6] 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).
- [7] K. Jałowiecki, M. M. Rams, and B. Gardas, Brute-forcing spin-glass problems with cuda, *Computer Physics Communications* **260**, 107728 (2021).
- [8] K. Jałowiecki and L. Pawela, Omnisolver: An extensible interface to Ising spin-glass and QUBO solvers, *SoftwareX* **24**, 101559 (2023).
- [9] K. Jałowiecki, L. Pawela, B. Gardas, A. Przybylski, and J. Tuziński, GPU based brute-force solver for QUBO and Ising instances, In preparation (2025).
- [10] BEIT, *BEIT QUBO solver user manual*, accessed: 2024-12-19.
- [11] BEIT, *AWS Marketplace: QUBO Solver by BEIT* (2024), accessed: 2025-01-30.
- [12] J.-G. Liu, L. Wang, and P. Zhang, Tropical tensor network for ground states of spin glasses, *Physical Review Letters* **126** (2021).
- [13] A. Chalkis, T. Kleinert, and B. Sofranac, QUBO Dual Bounds via SDP Plane Projection Method, *arXiv:2312.15026* (2023).
- [14] Quantagonia, How good is good enough?, *Quantagonia Blog* (2023), accessed: 2024-12-11.
- [15] F. Sheldon, F. L. Traversa, and M. Di Ventura, Taming a nonconvex landscape with dynamical long-range order: Memcomputing Ising benchmarks, *Physical Review E* **100** (2019).
- [16] Y. Yamamoto, T. Leleu, S. Ganguli, and H. Mabuchi, Coherent Ising machines — quantum optics and neural network perspectives, *Applied Physics Letters* **117** (2020).
- [17] J. Si, S. Yang, Y. Cen, J. Chen, and Y. Huang et al., Energy-efficient superparamagnetic Ising machine and its application to traveling salesman problems, *Nature Communications* **15**, 3457 (2024).
- [18] J. Wang, D. Ebler, K. Y. M. Wong, D. S. W. Hui, and J. Sun, Bifurcation behaviors shape how continuous physical dynamics solves discrete Ising optimization, *Nature Communications* **14**, 2510 (2023).
- [19] H. Wang, Z. Liu, Z. Xie, L. Li, and Z. Miao et al., Parallel Ising annealer via gradient-based hamiltonian monte carlo, *Quantum Machine Intelligence* **7**, 6 (2025).
- [20] P.-L. Bartosik, K. Donatella, M. Aifer, D. Melanson, and M. Perarnau-Llobet et al., Thermodynamic algorithms for quadratic programming, *arXiv:2411.14224* (2024).
- [21] M. Aifer, K. Donatella, M. H. Gordon, S. Duffield, and T. Ahle et al., Thermodynamic linear algebra, *npj Unconventional Computing* **1**, 13 (2024).
- [22] CPLEX, IBM ILOG, *Users manual for CPLEX* (2024).
- [23] J. Pawłowski, J. Tuziński, P. Tarasiuk, A. Przybylski, and R. A. Adamski et al., *VeloxQ data repository*, accessed: 2025-01-31.
- [24] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, and Z. Lü, The unconstrained binary quadratic programming problem: A survey, *Journal of Combinatorial Optimization* **28**, 58 (2014).
- [25] A. Abbas, A. Ambainis, B. Augustino, A. Bärttschi, and H. Buhman et al., Challenges and opportunities in quantum optimization, *Nature Reviews Physics* **6**, 718 (2024).
- [26] S. Boettcher, Analysis of the relation between quadratic unconstrained binary optimization and the spin-glass ground-state problem, *Physical Review Research* **1**, 033142 (2019).
- [27] F. Glover, G. Kochenberger, R. Hennig, and Y. Du, Quantum bridge analytics I: a tutorial on formulating and using QUBO models, *Annals of Operations Research* **314**, 141 (2022).
- [28] R. Lewis, *Guide to graph colouring* (Springer, 2021).
- [29] C. W. Commander, Maximum Cut Problem, MAX-CUT, in *Encyclopedia of Optimization*, edited by C. A. Floudas and P. M. Pardalos (Springer US, Boston, MA, 2009) pp. 1991–1999.

- [30] K. L. Hoffman, M. Padberg, and G. Rinaldi, Traveling salesman problem, in *Encyclopedia of Operations Research and Management Science*, edited by S. I. Gass and M. C. Fu (Springer US, Boston, MA, 2013) pp. 1573–1578.
- [31] S. Eggersglüß and R. Drechsler, Boolean satisfiability, in *High Quality Test Pattern Generation and Boolean Satisfiability* (Springer US, Boston, MA, 2012) pp. 41–57.
- [32] N. Dattani, Quadraticization in discrete optimization and quantum mechanics, [arxiv:1901.04405](https://arxiv.org/abs/1901.04405) (2019).
- [33] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual* (2024).
- [34] M. Bierlaire, *Optimization: Principles and Algorithms*, 2nd ed. (EPFL Press, Lausanne, 2018).
- [35] C. Fan, M. Shen, Z. Nussinov, Z. Liu, and Y. Sun et al., Searching for spin glass ground states through deep reinforcement learning, *Nature Communications* **14**, 725 (2023).
- [36] S. Boettcher, Deep reinforced learning heuristic tested on spin-glass ground states: The larger picture, *Nature Communications* **14**, 5658 (2023).
- [37] E. J. Crosson and D. A. Lidar, Prospects for quantum enhancement with diabatic quantum annealing, *Nature Reviews Physics* **3**, 466 (2021).
- [38] A. Villanueva, P. Najafi, and H. J. Kappen, Why adiabatic quantum annealing is unlikely to yield speed-up, *Journal of Physics A: Mathematical and Theoretical* **56**, 465304 (2023).
- [39] A. Irbäck, L. Knuthson, S. Mohanty, and C. Peterson, Folding lattice proteins with quantum annealing, *Physical Review Research* **4**, 043013 (2022).
- [40] D. Bucher, D. Porawski, B. Wimmer, J. Nüßlein, and C. O’Meara et al., Grid cost allocation in peer-to-peer electricity markets: Benchmarking classical and quantum optimization approaches, [arXiv: 2501.05253](https://arxiv.org/abs/2501.05253) (2025).
- [41] R. Verresen, Everything is a quantum Ising model, [arXiv:2301.11917](https://arxiv.org/abs/2301.11917) (2023).
- [42] A. D. King, A. Nocera, M. M. Rams, J. Dziarmaga, and R. Wiersema et al., Computational supremacy in quantum simulation, [arxiv:2403.00910](https://arxiv.org/abs/2403.00910) (2024).
- [43] N. Pirnay, V. Ulitzsch, F. Wilde, J. Eisert, and J.-P. Seifert, An in-principle super-polynomial quantum advantage for approximating combinatorial optimization problems via computational learning theory, *Science Advances* **10**, eadj5170 (2024).
- [44] Google Quantum AI and Collaborators, Quantum error correction below the surface code threshold, *Nature* (2024).
- [45] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, [arxiv:1411.4028](https://arxiv.org/abs/1411.4028) (2014).
- [46] S. Isakov, I. Zintchenko, T. Rønnow, and M. Troyer, Optimised simulated annealing for Ising spin glasses, *Computer Physics Communications* **192**, 265–271 (2015).
- [47] A. M. Dziubyna, T. Śmierchalski, B. Gardas, M. M. Rams, and M. Mohseni, Limitations of tensor network approaches for optimization and sampling: A comparison against quantum and classical Ising machines, [arxiv:2411.16431](https://arxiv.org/abs/2411.16431) (2024).
- [48] Q.-G. Zeng, X.-P. Cui, B. Liu, Y. Wang, and P. Mosharev et al., Performance of quantum annealing inspired algorithms for combinatorial optimization problems, *Communications Physics* **7**, 249 (2024).
- [49] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers* (2011).
- [50] G. França, D. P. Robinson, and R. Vidal, *Admm and accelerated admm as continuous dynamical systems* (2018), [arXiv:1805.06579](https://arxiv.org/abs/1805.06579).
- [51] G. França, D. P. Robinson, and R. Vidal, Gradient flows and proximal splitting methods: A unified view on accelerated and stochastic optimization, *Physical Review E* **103** (2021).
- [52] G. França, D. P. Robinson, and R. Vidal, A nonsmooth dynamical systems perspective on accelerated extensions of admm, *IEEE Transactions on Automatic Control* **68**, 2966–2978 (2023).
- [53] D-Wave, *Minor embedding*, accessed: 2025-01-30.
- [54] K. Boothby, P. Bunyk, J. Raymond, and A. Roy, Next-generation topology of D-Wave quantum processors, [arxiv:2003.00133](https://arxiv.org/abs/2003.00133) (2020).
- [55] K. Boothby, A. D. King, and J. Raymond, *Zephyr topology of D-Wave quantum processors* (2021), accessed: 2025-01-31.
- [56] H. M. Markowitz, *Portfolio selection: efficient diversification of investments* (Yale University Press, 1959).
- [57] D-Wave, *D-Wave hybrid framework* (2025), accessed: 2025-01-30.
- [58] D-Wave, *The D-Wave Clarity Roadmap* (2025), accessed: 2025-01-30.
- [59] Z. Lü, F. Glover, and J.-K. Hao, A hybrid metaheuristic approach to solving the ubqp problem, *European Journal of Operational Research* **207**, 1254 (2010).
- [60] I. Zintchenko, M. B. Hastings, and M. Troyer, From local to global ground states in Ising spin glasses, *Physical Review B* **91**, 024201 (2015).
- [61] H. Goto, K. Endo, M. Suzuki, Y. Sakai, and Taro et al., High-performance combinatorial optimization based on classical mechanics, *Science Advances* **7**, eabe7953 (2021).
- [62] I. Hen, Equation Planting: A Tool for Benchmarking Ising Machines, *Physical Review Applied* **12**, 011003 (2019).
- [63] D. Perera, F. Hamze, J. Raymond, M. Weigel, and H. G. Katzgraber, Computational hardness of spin-glass problems with tile-planted solutions, *Physical Review E* **101**, 023316 (2020).
- [64] F. Hamze, J. Raymond, C. A. Pattison, K. Biswas, and H. G. Katzgraber, The Wishart planted ensemble: A Tunably-Rugged pairwise Ising model with a first-order phase transition, *Physical Review E* **101**, 052102 (2020).
- [65] D. Perera, I. Akpabio, F. Hamze, S. Mandra, and N. Rose et al., Chook – A comprehensive suite for generating binary optimization problems with planted solutions, [arXiv:2005.14344](https://arxiv.org/abs/2005.14344) (2021).
- [66] L. A. Wolsey, *Integer Programming* (John Wiley & Sons, 1998).
- [67] T. Häner, K. E. Booth, S. E. Borujeni, and E. Y. Zhu, Solving QUBOs with a quantum-amenable branch and bound method, [arXiv:2407.20185](https://arxiv.org/abs/2407.20185) (2024).
- [68] X. Huo and R. Gu, A vectorized positive semidefinite penalty method for unconstrained binary quadratic programming, [arXiv:2408.04875](https://arxiv.org/abs/2408.04875) (2024).
- [69] F. G.S L. Brandão, R. Kueng, and D. Stilck França, Faster quantum and classical SDP approximations for quadratic binary optimization, *Quantum* **6**, 625 (2022).

- [70] M. Born and V. Fock, Beweis des Adiabatsatzes, *Zeitschrift für Physik* **51**, 165 (1928).
- [71] M. Courbariaux, Y. Bengio, and J.-P. David, BinaryConnect: Training Deep Neural Networks with binary weights during propagations, [arXiv:1511.00363](https://arxiv.org/abs/1511.00363) (2016).
- [72] N. Qian, On the momentum term in gradient descent learning algorithms, *Neural Networks* **12**, 145 (1999).
- [73] F. Glover, G. A. Kochenberger, and B. Alidaee, Adaptive memory tabu search for binary quadratic programs, *Management Science* **44**, 336 (1998).
- [74] A. Wiegele, Biq mac library - binary quadratic and Max-Cut library, <https://biqmac.aau.at/biqmaclib.html>, accessed: 2025-01-16.
- [75] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, Tech. Rep. (Rice University, 1998).