Hybrid Sequential Quantum Computing

Pranav Chandarana [ORCID],[1,2] Sebastián V. Romero [ORCID],[1,2] Alejandro Gomez Cadavid [ORCID],[1,2] Anton Simen [ORCID],[1,2] Enrique Solano [ORCID],[1,*] and Narendra N. Hegade [ORCID][1,†]

[1]*Kipu Quantum GmbH, Greifswalderstrasse 212, 10405 Berlin, Germany*
[2]*Department of Physical Chemistry, University of the Basque Country EHU, Apartado 644, 48080 Bilbao, Spain*
(Dated: October 8, 2025)

We introduce hybrid sequential quantum computing (HSQC), a paradigm for combinatorial optimization that systematically integrates classical and quantum methods within a structured, stage-wise workflow. HSQC may involve an arbitrary sequence of classical and quantum processes, as long as the global result outperforms the standalone components. Our testbed begins with classical optimizers to explore the solution landscape, followed by quantum optimization to refine candidate solutions, and concludes with classical solvers to recover nearby or exact-optimal states. We demonstrate two instantiations: (i) a pipeline combining simulated annealing (SA), bias-field digitized counterdiabatic quantum optimization (BF-DCQO), and memetic tabu search (MTS); and (ii) a variant combining SA, BF-DCQO, and a second round of SA. This workflow design is motivated by the complementary strengths of each component. Classical heuristics efficiently find low-energy configurations, but often get trapped in local minima. BF-DCQO exploits quantum resources to tunnel through these barriers and improve solution quality. Due to decoherence and approximations, BF-DCQO may not always yield optimal results. Thus, the best quantum-enhanced state is used to continue with a final classical refinement stage. Applied to challenging higher-order unconstrained binary optimization (HUBO) problems on a 156-qubit heavy-hexagonal superconducting quantum processor, we show that HSQC consistently recovers ground-state solutions in just a few seconds. Compared to standalone classical solvers, HSQC achieves a speedup of up to $700\times$ over SA and up to $9\times$ over MTS in estimated runtimes. These results demonstrate that HSQC provides a flexible and scalable framework capable of delivering up to two orders of magnitude improvement at runtime quantum-advantage level on advanced commercial quantum processors.

I. INTRODUCTION

Machine learning, classical optimization, and classical simulation methods build a coupled toolkit for modern science, each with distinct strengths and limitations that unveil what is computationally feasible to date. Machine learning converts input data into predictive or generative models, relevant to biology [1], material science [2], weather forecasting [3], and conversational agents [4], among others. However, it often requires solving vast high-dimensional problems or it faces trainability issues due to barren plateaus, overfitting, or complex landscapes [5–7]. Given a problem encoded as an objective function subject to constraints, classical optimization aims to find the set of optimal variables that minimize or maximize the objective function. With applications covering a broad set of use-cases like logistics [8–11] and manufacturing [12–14], these problems often belong to the NP-hard class, demanding an exponentially growing computational time with the problem size, heuristics, and decomposition strategies. Classical simulations of physical systems range from Monte Carlo methods [15–17], which efficiently sample equilibrium ensembles but with slow convergence, as well as density functional theory [18–21], which investigates the electronic structure of many-body systems by first-principles but struggles with

strongly correlated systems [22, 23].

On the other hand, recent developments in quantum technologies are turning quantum computers into reliable devices capable of addressing computationally complex tasks, which may overcome the issues aforementioned and ultimately solve problems beyond the reach of classical computation. Apart from purely quantum algorithms, several implementations have been put forth combining a classical precomputation followed by a quantum routine, aiming for a beneficial interplay between both computing paradigms. Some examples are given by a classical warm-start of quantum optimization routines [24], classical preparation of initial states for ground state preparation of chemical compounds [25, 26], and hybrid decomposition pipelines [27–30], among others. There are also recent proposals reversing the order, where a short quantum precomputation is followed by a more intense classical routine, sometimes referred to as *quantum accelerators* [31–34]. However, to our knowledge, none of these classical-to-quantum or quantum-to-classical approaches has been able to show any source of runtime quantum advantage to date.

In this work, we introduce hybrid sequential quantum computing (HSQC), a paradigm that leverages a selective combination of quantum and classical computing routines to solve combinatorial optimization problems where standalone methods struggle (see Fig. 1). Classical algorithms are well established, yet many require resources that grow rapidly, often exponentially, to obtain accurate or practically valuable solutions, with some
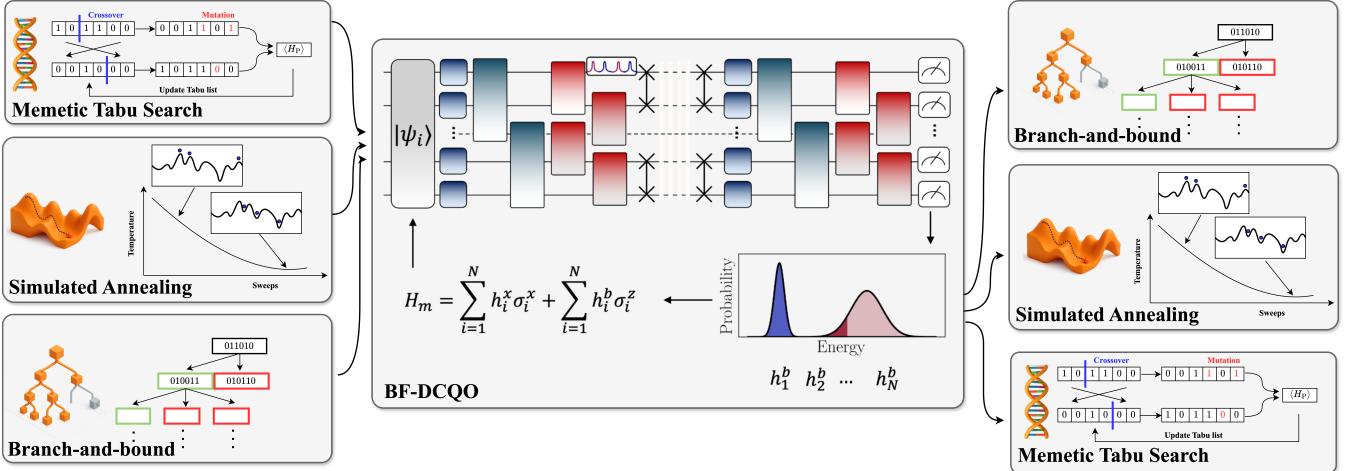
FIG. 1. **Schematic of the hybrid sequential quantum computing (HSQC) framework.** The optimization begins by formulating the target problem as a higher-order unconstrained binary optimization (HUBO) problem. A classical optimization routine first explores the energy landscape to identify promising low-energy configurations, continuing until further improvements become computationally expensive. The best configuration is then used to initialize the BF-DCQO protocol. The QPU then executes a controlled counterdiabatic evolution that guides the system through the energy landscape by exploiting quantum superposition and tunneling to escape local minima and converge toward lower-energy states. The resulting output is passed to a final classical optimization stage, which may employ heuristic or exact methods, to reach the ground state or obtain high-quality approximations. The runtime allocation across the stages is chosen adaptively based on the problem characteristics.

problems remaining out of reach. Quantum computing offers a potential remedy, but current devices are limited by gate and readout fidelities, coherence times, and hardware connectivity. HSQC targets this gap by selectively invoking each routine where it is strongest, combining their advantages and mitigating limitations, in line with recent proposals that combine different platforms [35]. Motivated by recent quantum speedup claims in optimization [32, 36–45] and without loss of generality, we experimentally validate HSQC by solving higher-order unconstrained binary optimization (HUBO) problems by sandwiching the bias-field digitized counterdiabatic quantum optimization (BF-DCQO) algorithm [45–48] on IBM quantum hardware [49] between different classical solvers, namely simulated annealing (SA) [50] and memetic tabu search (MTS) [51]. For completeness, we also solve them using D-Wave quantum annealers [52] for different annealing times as well as their hybrid solvers [31], both requiring extra qubits to quadratize higher-order terms. Our results show a significant speedup in both scenarios, against the involved solvers running independently, and the D-Wave solvers. From the results drawn, HSQC is capable of boosting well-established classical routines by providing faster and better solutions using current quantum hardware, with potential applications in quantum machine learning, material simulation, and quantum chemistry, apart from quantum optimization.

II. RESULTS

A. Higher-order binary optimization

HUBO extends quadratic unconstrained binary optimization (QUBO) by allowing interactions among up to p binary variables. Many academic and industrial optimization problems can be formulated as HUBO problems [53]. The cost function is

$$F(z) = \sum_{r=1}^{p} \sum_{(a_1,\ldots,a_r) \in \mathbb{P}_k} W_{a_1\ldots a_r} \, z_{a_1} \cdots z_{a_r}, \quad (1)$$

where $z_a \in \{0, 1\}$, $\mathbb{P}_k$ is a hypergraph containing the indices of the k-body terms, p is the maximum interaction order, and the couplings $W_{a_1\ldots a_r}$ are drawn from a specified distribution.

Instance difficulty is governed by four factors: the variable count n; the statistics of the couplings $W_{a_1\ldots a_r}$; the highest interaction order p; and the interaction density of the underlying hypergraph. We consider up to $n = 156$ variables on IBM superconducting processors [49], fixing $p = 3$ for hardware compatibility. Mapping each bit to a spin via $z_a = \frac{1-\sigma_a^z}{2}$ converts Eq. (1) into the problem Hamiltonian

$$\begin{aligned} H_{\mathrm{P}} &= \sum_{a=1}^{N} h_a \, \sigma_a^z + \sum_{(u,v) \in \mathbb{P}_{2b}} J_{uv} \, \sigma_u^z \sigma_v^z \\ &+ \sum_{(u,v,w) \in \mathbb{P}_{3b}} K_{uvw} \, \sigma_u^z \sigma_v^z \sigma_w^z, \end{aligned} \quad (2)$$

TABLE I. **Classical hardware and software specifications.**

Processor	AMD (KVM processor) (48×2.3 GHz)
RAM	123 GB
OS	Debian GNU/Linux 12 (bookworm) ($\times 64$)
CPLEX [56]	v22.1.2.0 C++ 11.4.0

whose ground state minimizes $F(z)$, with $\mathbb{P}_{\{2b,3b\}}$ containing the indices of the two- and three-body terms, and N the number of qubits. Since our mapping from variables to qubits is one-to-one, we have that $n = N$.

To create challenging yet hardware-compatible instances, we follow the graph-coloring strategy of Refs. [35, 54]: nearest-neighbor two- and three-body couplings are first assigned on the native topology, then a SWAP layer is applied; this process is repeated for n_{swap} layers (see Methods). The resulting dense interaction maps for H_{P} yield HUBO problems that are difficult for classical solvers while embedding efficiently on today's quantum devices.

To evaluate the effectiveness of the HSQC strategy, we consider a suite of challenging HUBO instances defined on $N = 156$ qubits. Each instance incorporates $n_{\text{swap}} = 2$ layers (see Methods), resulting in a problem Hamiltonian H_{P} that includes 156 one-body terms, 125 two-body terms, and 616 three-body terms. The coupling coefficients h, J, and K are sampled from a Cauchy distribution, which has been shown to produce classically hard instances, particularly for algorithms like SA [45].

As a concluding remark, D-Wave quantum annealers are well-suited for solving QUBO problems formulated as two-body spin glasses (Eq. (1) setting $p = 2$), since their analog hardware naturally evolves according to a transverse-field Ising model. Therefore, in order to solve HUBO instances using their devices, a HUBO-to-QUBO mapping is required, demanding extra qubits and introducing constraints (weighted by a tunable Lagrange multiplier) that are not naturally present in the original HUBO formulation. Additionally, once the Hamiltonian is transformed, a suitable embedding across the coupling map of the device has to be found, which might increase the final number of qubits required. Note that in this case, $n \neq N$. In our experiments, the first conversion is done using DIMOD library and the subsequent embedding on hardware is done using the D-WAVE OCEAN SDK [55] which, given a minor and a target graph, heuristically attempts to map the minor into the target, where additional qubits may be required to finally embed the resulting QUBO problem. A more detailed analysis can be seen in Methods.

B.   SA + BF-DCQO + MTS

In this section, we present a HSQC strategy that combines three optimization solvers: SA, BF-DCQO, and MTS. SA [50] is a stochastic optimization method
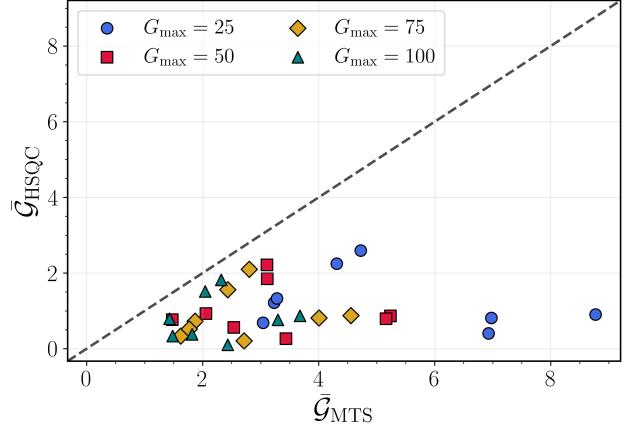

FIG. 2. **Performance comparison between MTS and HSQC.** For different number of generations G_{max}, optimality gaps $\mathcal{G}$ obtained across the eight instances tested in Table VIII. Data points lying in the lower part of the plot mean that HSQC approach performed better than MTS.

inspired by the process of physical annealing. It explores the energy landscape by accepting both downhill (energy-lowering) and, with decreasing probability, uphill (energy-increasing) moves. This feature enables the algorithm to escape local minima and converge toward near-optimal solutions. Within the HSQC framework, SA is executed first over n_{runs} independent trials, each consisting of n_{sweep} sweeps (see Methods).

Continued from the best SA solution, the second stage of HSQC applies BF-DCQO, a digitized quantum algorithm based on counterdiabatic (CD) protocols [57]. The algorithm implements a fast, discrete-time quantum evolution governed by a CD Hamiltonian H_{cd}, which interpolates between a mixer Hamiltonian H_m and a problem Hamiltonian H_{P}. In the bias-field variant [46–48], the mixer Hamiltonian is dynamically updated based on measurement-driven feedback $H_m = \sum_{i=1}^{N} \left(h_i^x \sigma_i^x + h_i^b \sigma_i^z \right)$, where the transverse fields h_i^x are fixed, while the longitudinal bias fields h_i^b are iteratively adjusted using the rule $h_i^b = \pm \langle \sigma_i^z \rangle$. This iterative feedback loop is executed for n_{iter} steps, progressively improving the initialization of the quantum state prior to evolution (see Methods).

The final component of the HSQC protocol is MTS, a memetic metaheuristic that merges evolutionary algorithms [58] with local search guided by a tabu list [59]. Starting from an initial population of P candidate bitstrings, constructed by cloning the best solution $\mathbf{s}_{\text{ws}}$ obtained from BF-DCQO, the algorithm proceeds through G_{max} generations. In each generation, pairs of parents are selected, recombined via crossover, and mutated with a generation-dependent mutation rate

$$\mu_g = \mu_{\text{end}} + (\mu_{\text{start}} - \mu_{\text{end}}) \frac{\ln(G_{\text{max}} + 1 - g)}{\ln(G_{\text{max}} + 1)}, \quad (3)$$

where g denotes the current generation. Each offspring

is further refined via a local tabu search before being reintroduced into the population.

The complete HSQC workflow proceeds as follows. First, SA is run for n_{sweep} sweeps across n_{runs} trials, resulting in a classical runtime of $T_{\text{SA}} = n_{\text{sweep}} \times n_{\text{runs}} \times 0.6 \times 10^{-5}$ seconds. The lowest-energy configuration from SA is used to initialize the bias fields h_i^b for the subsequent BF-DCQO stage. BF-DCQO is executed on quantum hardware for n_{iter} iterations and n_{shots} shots, with total runtime given by $T_{\text{BFDCQO}} = T_{\text{CPU}} + T_{\text{QPU}}$. The quantum sampling time is set to $T_{\text{QPU}} = n_{\text{shots}} \times 10^{-4}$ seconds [60], and each iteration includes a classical local search phase lasting $T_{\text{CPU}} = n_{\text{sweep}}^{\text{loc}} \times 0.6 \times 10^{-5}$ seconds to mitigate bit-flip noise introduced by the hardware [45, 61, 62]. The constant factors are computed based on the CPU performance on which the heuristics are executed (see Methods).

Finally, the best solution $\mathbf{s}_{\text{ws}}$ from BF-DCQO is duplicated to seed the MTS population. MTS is then executed for G_{max} generations using a tabu list of length I_{tabu} and the adaptive mutation schedule described above. The corresponding time is $T_{\text{MTS}} = n_{\text{bitflip}} \times 5.740 \times 10^{-8}$ seconds. Therefore, the total time of the HSQC reads

$$
T = \underbrace{n_{\text{sweep}} \times n_{\text{runs}} \times 0.6 \times 10^{-5}}_{\text{SA}}
$$
$$
+ \underbrace{n_{\text{shots}} \times 10^{-4} + n_{\text{sweep}}^{\text{loc}} \times 0.6 \times 10^{-5}}_{\text{BF-DCQO}} \quad (4)
$$
$$
+ \underbrace{n_{\text{bitflip}} \times 5.740 \times 10^{-8}}_{\text{MTS}} \text{ s.}
$$

This hybrid, sequential strategy allows each solver to build upon the progress of the previous stage, enabling deeper exploration of the solution space and faster convergence to high-quality solutions. For each instance, we execute the first two stages of the HSQC workflow, SA followed by BF-DCQO, within a fixed time budget, yielding an initial population for MTS. We then run this MTS multiple times and compare its performance against standalone MTS (with a randomly sampled initialization), standard SA, and the exact solver CPLEX. This setup allows us to isolate and assess the contribution of quantum acceleration within the HSQC framework. Regarding MTS, for each run, we fix $I_{\text{tabu}} = 10$, $\mu_{\text{start}} = 0.1$ and $\mu_{\text{end}} = 0.001$ and generation $G_{\text{max}} \in \{25, 50, 75, 100\}$ as hyperparameters.

Figure 2 presents a direct comparison between the performance of the MTS algorithm and the HSQC workflow, evaluated using the minimum optimality gap $\bar{\mathcal{G}} = 100\,(\bar{E}_{\text{best}} - E_{\text{GS}})/|E_{\text{GS}}|$, where E_{GS} denotes the ground-state energy obtained from CPLEX [56]. Details of the experimental setup, including statistics for E_{min} and the average best energy $\bar{E}_{\text{best}}$, are provided in Table VIII and discussed further in the Methods section.

We observe that all data points fall below the equivalence line $\bar{\mathcal{G}}_{\text{MTS}} = \bar{\mathcal{G}}_{\text{HSQC}}$, clearly demonstrating the consistent advantage of incorporating HSQC. Particularly for smaller generation limits (e.g., $G_{\text{max}} = 25$), data

cluster toward the lower right corner, reflecting substantial gains in solution quality due to the HSQC. As G_{max} increases, the performance gap between MTS and HSQC narrows, with data points approaching the equivalence line. This is expected: more generations allow MTS to further refine solutions, while the relative improvement offered by HSQC diminishes due to saturation and the growing density of local minima near the optimum.

Nonetheless, HSQC maintains superior performance across all tested generation limits, yielding lower $\bar{E}_{\text{best}}$ and, in many instances, smaller E_{min}. Notably, HSQC successfully recovers the exact ground state for two problem instances with just $G_{\text{max}} = 75$, a feat unattained by standalone MTS in any configuration. These findings underscore HSQC's robustness and efficiency, showing it not only accelerates convergence but also enhances the likelihood of identifying exact or near-exact solutions.

To make a fair runtime-based comparison, we select the best-case HUBO instance and repeat the HSQC (fixed SA + BF-DCQO), MTS, and SA procedures over 10 independent seeds. The corresponding results are summarized in Table II. Since HSQC successfully finds the GS for this instance, we use it as a benchmark to assess how long it takes for standalone SA and MTS to achieve the same result.

To this end, we modify the MTS algorithm to allow for a maximum of $G_{\text{max}} = 20000$ generations, and introduce a stopping criterion based on the target energy $E_{\text{target}} = -223.1855$, corresponding to the ground-state energy. The MTS column in Table II reports the minimum energy reached for each seed, along with the total runtime T_{MTS}, computed using Eq. (4). If E_{target} is reached before exhausting all generations, the algorithm stops early.

For the HSQC protocol, we set the time $T_{\text{SA}} = 0.6$ seconds, based on $n_{\text{sweep}}^{\text{SA}} = 1000$ and $n_{\text{runs}}^{\text{SA}} = 100$. The lowest-energy bitstring from SA is then used to initialize the bias fields in BF-DCQO, which is executed using $n_{\text{shots}} = 5000$ shots. For the local-search, we set $n_{\text{sweep}}^{\text{loc}} = 900$ sweeps to mitigate bitflip errors. The best result from BF-DCQO is subsequently passed to MTS for initializing the population.

We observe that standalone MTS successfully reaches the ground state in only 4 out of 10 trials, with runtimes reaching up to $T_{\text{MTS}} \approx 18$ seconds. In contrast, HSQC consistently finds the ground state in all 10 trials, with a significantly lower and consistent total runtime of approximately $T \approx 1.5$ seconds. This clearly demonstrates a runtime advantage of HSQC over standalone MTS.

To complete the comparison, we also evaluate standalone SA. Specifically, we perform 10 independent trials using $n_{\text{sweep}} = 50000$ and $n_{\text{runs}} = 1000$, resulting in a total runtime of approximately $T_{\text{SA}} = 300$ seconds per trial. In 5 out of the 10 trials the ground state is recovered. While this represents a higher success rate than standalone MTS, it comes at a significantly greater computational cost. In contrast, HSQC achieves ground-state solutions in all 10 trials, with a much lower and

TABLE II. **Per-seed minimum energies and runtimes for MTS, SA, and HSQC.** Results correspond to the best-case instance, with each method repeated ten times using different random seeds. For MTS, all trials for a given seed were initialized identically and executed with $G_{\max} = 20000$, employing early stopping upon reaching the ground state. For SA, each seed was run using $n_{\text{sweep}}^{\text{SA}} = 50000$ sweeps and $n_{\text{runs}} = 1000$ independent runs. For HSQC, the bitstring obtained from BF-DCQO was fixed, and the MTS with initial population from BF-DCQO, was executed with $G_{\max} = 20000$ and early stopping at the ground state energy. The total runtime of HSQC is decomposed into contributions from SA (T_{SA}), BF-DCQO (T_{BFDCQO}), and MTS (T_{MTS}). For reference, the exact CPLEX runtime is $T_{\text{CPLEX}} = 22.896$ s and is independent of seed.

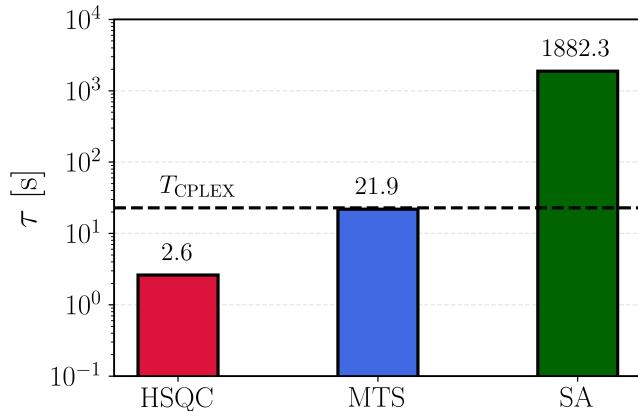| Seed	T_{CPLEX} [s]	MTS		SA		HSQC				
		Min energy	T_{MTS} [s]	Min energy	T_{SA} [s]	Min energy	T_{SA} [s]	T_{BFDCQO} [s]	T_{MTS} [s]	T [s]
0	22.896	-223.186	5.168	-223.186	300.0	-223.186	0.600	0.505	0.475	1.580
1	22.896	-223.186	2.274	-222.900	300.0	-223.186	0.600	0.505	0.177	1.282
2	22.896	-219.769	18.071	-222.841	300.0	-223.186	0.600	0.505	0.276	1.382
3	22.896	-219.769	18.071	-223.186	300.0	-223.186	0.600	0.505	0.884	1.989
4	22.896	-221.434	18.071	-222.841	300.0	-223.186	0.600	0.505	0.470	1.575
5	22.896	-222.900	18.071	-223.186	300.0	-223.186	0.600	0.505	0.390	1.495
6	22.896	-222.900	18.071	-223.186	300.0	-223.186	0.600	0.505	0.615	1.720
7	22.896	-223.186	1.187	-222.674	300.0	-223.186	0.600	0.505	0.715	1.821
8	22.896	-223.186	0.274	-222.874	300.0	-223.186	0.600	0.505	0.119	1.224
9	22.896	-221.294	18.071	-223.186	300.0	-223.186	0.600	0.505	0.264	1.370

FIG. 3. **Minimum τ values for various standalone optimization routines for Instance 8.** Comparing minimum τ values for MTS, HSQC and SA obtained from Table III and Table IV with $T_{\text{CPLEX}} = 22.896$ s value as a reference.

consistent runtime of approximately 1.5 seconds, demonstrating its runtime advantage over both SA and MTS.

Additionally, we note that the runtime for CPLEX on this instance is approximately 23 seconds. While this includes the time required to prove optimality, it is still longer than the total time required by HSQC to find the ground state. It is important to highlight that HSQC, as a hybrid workflow composed of multiple heuristic components, is inherently probabilistic in nature. Therefore, any comparison with deterministic exact solvers like CPLEX must account for this. However, since HSQC successfully recovers the ground state in all 10 trials for this instance, the runtime comparison remains meaningful and highlights HSQC's practical efficiency even against exact methods.

As a final benchmark, we evaluate the estimated time for convergence τ for Instance 8 to compare the performance of HSQC, SA, and MTS heuristics, accounting for

their inherently probabilistic nature. We define the convergence time τ as

$$\tau = t_f \frac{\log(1 - 0.99)}{\log(1 - p_{\text{gs}})}, \tag{5}$$

where t_f denotes the time per trial, which depends on the specific solver, and p_{gs} is the empirical probability of reaching the ground state across all trials. The factor 0.99 corresponds to the desired confidence level.

For Instance 8, we observe the first occurrences of ground-state solutions starting from $G_{\max} = 75$ (see Table VIII). Therefore, we perform 100 independent trials at $G_{\max} \in \{75, 100, 200, 500\}$ and estimate p_{gs} for both MTS and HSQC. Using these probabilities, we compute the corresponding τ values using Eq. (5). For HSQC, we include the cumulative overhead from the SA and BF-DCQO stages, i.e., $T_{\text{SA}} + T_{\text{BFDCQO}} = 1.05$ s, in the trial time t_f to ensure a fair runtime comparison.

In Table III, we report the p_{gs}, t_f and τ values for HSQC and MTS. We can observe that the p_{gs} values are considerably higher for HSQC than MTS. Therefore, even after adding the constant time, the τ values of HSQC are much lower than MTS. Specifically for MTS the lowest value we obtained is around $\tau = 21.86$ s, whereas for HSQC the minimum value goes down to $\tau = 2.6$ s. We run a similar study for the SA case in Table IV where, again, we run 100 independent trials using $n_{\text{runs}} = 1000$ and $n_{\text{sweep}} \in \{30000, 40000, 50000, 60000, 70000, 100000\}$ and compute the convergence time τ. In this case, we use Eq. (4) to estimate the time per SA trial t_f. From the results drawn, the best result obtained comes from the combination of $n_{\text{runs}} = 1000$ and $n_{\text{sweep}} = 50000$, returning $\tau = 1882.30$ s, value that is around 700 times greater than HSQC method. As a side note, we observe that for Instance 8 a saturation point appears when going beyond $n_{\text{sweep}} = 50000$, where increasing the number of sweeps

TABLE III. **Ground-state probability and runtimes for MTS and HSQC.** Each row corresponds to a maximum generation budget $G_{\max}$. Columns report the ground-state success probability p_{gs}, the mean runtime t_f, and the estimated time τ.

$G_{\max}$	MTS			HSQC		
	p_{gs}	t_f [s]	τ [s]	p_{gs}	t_f [s]	τ [s]
75	0.01	0.068	32.491	0.07	0.068	5.452
100	0.02	0.091	**21.864**	0.12	0.091	4.386
200	0.02	0.181	42.447	0.41	0.181	2.688
500	0.08	0.452	26.086	0.75	0.452	**2.608**

TABLE IV. **SA ground-state probability and runtimes.** Each row corresponds to a SA sweep budget. Columns report the ground-state success probability p_{gs}, the runtime t_f, and the estimated time τ.

n_{runs}	n_{sweep}	SA		
		p_{gs}	t_f [s]	τ [s]
1000	30000	0.26	180.0	2752.96
1000	40000	0.32	240.0	2865.82
1000	50000	0.52	300.0	**1882.30**
1000	60000	0.35	360.0	3848.48
1000	70000	0.37	420.0	4186.20
1000	100000	0.36	600.0	6191.31

does not help to increase the probability of reaching the ground state energy. The complexity of these instances comes from a distribution that has shown to be challenging for SA [45]. Fig 3, shows a comparison of the minimum τ values for HSQC, MTS and SA for Instance 8 with T_{CPLEX} for reference as well.

Overall, these results provide compelling evidence that the HSQC paradigm offers a superior balance between solution quality and runtime efficiency. In particular, for dense HUBO problems, HSQC achieves optimal solutions more reliably and faster than standalone classical optimizers such as MTS and SA.

C. SA + BF-DCQO + SA

Inspired by the previous hybrid combination of methods, we now perform the same study, but now substituting only the MTS part with SA again. Therefore, in this section, we compare the running times and best energies obtained for CPLEX, BF-DCQO, SA, and HSQC, with HSQC results obtained by initializing an SA routine with the 100 lowest energy-valued outcomes of BF-DCQO instead of randomly generated bitstrings. Consequently,

TABLE V. **Per-seed minimum energies and runtimes for SA and HSQC for best-performing instance: instance 1.** Hyperparameters are set to $n_{sweep}^{SA} = 10000$ sweeps for SA, $n_{sweep}^{HSQC} = 8167$ for HSQC.

Seed	SA	HSQC	$T_{SA} = T_{HSQC}$ [s]
0	-214.6752	**-216.1362**	6.0
1	-215.8490	**-215.8609**	6.0
2	-215.3549	**-215.5230**	6.0
3	-214.3795	**-216.4535**	6.0
4	-215.6728	**-215.9188**	6.0
5	-214.3795	**-215.5230**	6.0
6	-215.3549	**-217.9757**	6.0
7	-215.2783	**-215.5230**	6.0
8	-214.3083	**-215.8609**	6.0
9	-214.3515	**-215.5230**	6.0

the total time of HSQC is now

$$
T_{\text{HSQC}} = \underbrace{n_{\text{sweep}}^{\text{SA1}} \times n_{\text{runs}}^{\text{SA1}} \times 0.6 \times 10^{-5}}_{\text{SA}}
$$
$$
+ \underbrace{n_{\text{shots}} \times 10^{-4} + n_{\text{sweep}}^{\text{loc}} \times 0.6 \times 10^{-5}}_{\text{BF-DCQO}} \quad (6)
$$
$$
+ \underbrace{n_{\text{sweep}}^{\text{SA2}} \times n_{\text{runs}}^{\text{SA2}} \times 0.6 \times 10^{-5}}_{\text{SA}} \text{ s.}
$$

We use again as number of sweeps-to-time conversion Eq. (4) and $n_{\text{runs}} = 100$. We use the same parameters as in Sec. II B for the first simulated annealing

For this variant, we solve the same instances tackled in Table VIII but now substituting the MTS part with SA. In Table IX, we attach the minimum and mean best energies for SA and HSQC for a different number of sweeps $n_{\text{sweep}}^{\text{SA2}} \in \{1000, 2000, 5000, 10000\}$, where the HSQC approach outperforms SA in most of the cases. We also run an intense SA routine with $n_{\text{sweep}} = 10^6$ (leading to $600\,\text{s}$ of execution), where the ground state energy was found only for three out of the eight instances tested, indicating their complexity for SA. In Fig. 4 we compare the results obtained in Table IX for all instances in terms of the mean of their best approximation ratios, with most of these values lying in the region where HSQC performed better than SA.

Therefore, rather than exact ground states, for this HSQC variant, we can look instead for better and faster approximate solutions. For that purpose, setting again $n_{\text{runs}} = 100$, we can fix the number of sweeps considered for SA, $n_{\text{sweep}}^{\text{SA}}$, and extract the number of sweeps of HSQC, $n_{\text{sweep}}^{\text{HSQC}}$, such that $T_{SA} = T_{\text{HSQC}}$. Using the same values of $T_{\{SA, BF-DCQO\}}$ as in Table II, we obtain the relationship $n_{\text{sweep}}^{\text{HSQC}} \approx n_{\text{sweep}}^{\text{SA}} - 1833$. In Table V, we compare for instance 10 the minimum energies obtained when $n_{\text{sweep}}^{\text{SA}} = 10000$ ($n_{\text{sweep}}^{\text{HSQC}} = 8167$) is set, which translates into $6\,\text{s}$ of execution. We can observe that in the ten trials, HSQC returned better solutions in the same amount of time.
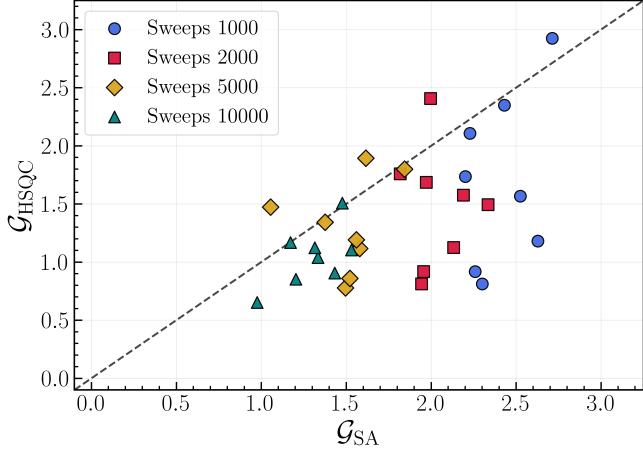
FIG. 4. **Performance comparison between SA and HSQC.** For different number of sweeps, optimality gaps obtained across the eight instances tested in Table IX. Data points lying in the lower part of the plot mean that the HSQC approach performed better than SA.

III. DISCUSSION

In this work, we introduce hybrid sequential quantum computing, an optimization framework that integrates simulated annealing, bias-field digitized counterdiabatic quantum optimization, and memetic tabu search, to solve dense higher-order unconstrained binary optimization problems. This approach demonstrates how classical and quantum components can be orchestrated sequentially to exploit their complementary strengths, achieving both rapid convergence and high-quality solutions under the constraints of near-term hardware. Our results reveal that HSQC consistently outperforms standalone SA and MTS in terms of both solution quality and time-to-solution. These findings provide direct evidence of runtime quantum advantage, where incorporating quantum subroutines such as BF-DCQO enables faster and more reliable convergence toward optimal solutions.

A key enabler of this performance is the quantum-generated bitstrings from BF-DCQO to initialize the MTS stage. This steers the classical search toward promising regions of the solution space and away from poor local optima. Across various generation limits, HSQC achieved lower optimality gaps and better minimum energies than standalone MTS, consistently improving both average and worst-case performance. Importantly, HSQC is also well-suited to the limitations and opportunities of upcoming quantum devices. HSQC is designed to find high-quality solutions within fixed time budgets, indicated by the fact that the results can match CPLEX in time-to-solution on certain instances. Its modular design allows each stage to be improved independently, offering flexibility for future enhancements and scalability to larger systems.

Looking ahead, extending HSQC to two-dimensional quantum architectures with a higher degree of con-

nectivity presents a natural and promising next step. Such architectures offer greater connectivity and support for more complex problem instances, increasing the likelihood of runtime separation between classical and quantum solvers. As quantum hardware continues to evolve, we believe HSQC provides a compelling blueprint for industry-relevant hybrid optimization protocols and practical demonstrations of runtime quantum advantage beyond the given benchmarks. To adapt HSQC to industrially relevant problems and specific hardware and classical solvers remains as future work. As quantum hardware scales and more complex hybrid sequences become viable, the disruptive emergence of generative AI and intelligent agents is expected to automate, adapt, and accelerate the HSQC workflows, unlocking its full potential for real-world optimization at unprecedented speed and scale.

IV. METHODS

A. Instance generator

The instance generator employed in this study builds upon and extends the methodology introduced in Ref. [45]. Our objective is to generate classically challenging problem instances by co-designing the interaction structure and assigning coupling strengths in a hardware-aware manner.

We begin with empty interaction layers, denoted $\Lambda_2 = \varnothing$ and $\Lambda_3 = \varnothing$, alongside an initial hardware connectivity map $\mathcal{C}^{(0)}$. Using graph-coloring techniques inspired by Ref. [54], we identify sets of mutually non-overlapping two- and three-qubit interactions that are compatible with the hardware constraints and can be executed in parallel. These interactions are organized into the families $\mathbb{P}_{2b} = \{\mathbb{P}_{2b}^{(1)}, \ldots, \mathbb{P}_{2b}^{(R_2)}\}$ and $\mathbb{P}_{3b} = \{\mathbb{P}_{3b}^{(1)}, \ldots, \mathbb{P}_{3b}^{(R_3)}\}$, where R_2 and R_3 denote the number of parallelizable layers for the two- and three-body interactions, respectively.

To construct a specific instance, we select ρ_{2b} and ρ_{3b} layers from the respective families and update the interaction layers as $\Lambda_2 \leftarrow \Lambda_2 \cup \{\mathbb{P}_{2b}^{(i)}\}_{i=1}^{\rho_{2b}}$ and $\Lambda_3 \leftarrow \Lambda_3 \cup \{\mathbb{P}_{3b}^{(j)}\}_{j=1}^{\rho_{3b}}$. Next, we interpret the first two-body layer $\mathbb{P}_{2b}^{(1)}$ as a SWAP operation, which permutes qubit labels and updates the connectivity map to a new configuration $\mathcal{C}^{(1)}$. This process is repeated iteratively for n_{swap} rounds, using the leading layer from $\mathbb{P}_{2b}$ in each round to recursively update the connectivity map, as outlined in Algorithm 1. All simulations in this study use $\mathcal{C}^{(0)}$ corresponding to IBM's Heron processor, which features a 156-qubit heavy-hexagonal lattice architecture [63].

B. BF-DCQO and circuit decomposition

The counterdiabatic Hamiltonian is constructed using a first-order nested commutator approximation of the

---

Algorithm 1 Instance layout generation with swap-based connectivity updates

---

Require: Number of swap rounds n_{swap}; initial connectivity map $\mathcal{C}^{(0)}$; integers ρ_{2b}, ρ_{3b}; function GraphColoring$(\mathcal{C})$ that returns the families $\mathbb{P}_{2b} = \{\mathbb{P}_{2b}^{(1)}, \ldots, \mathbb{P}_{2b}^{(R_2)}\}$ and $\mathbb{P}_{3b} = \{\mathbb{P}_{3b}^{(1)}, \ldots, \mathbb{P}_{3b}^{(R_3)}\}$ of mutually non-overlapping two- and three-body interaction layers compatible with $\mathcal{C}$; function SwapRegister$(\mathcal{C}, I)$ that permutes qubit labels in $\mathcal{C}$ according to a set of disjoint pairs I and returns the updated connectivity map.

1: Initialize $\Lambda_2 \leftarrow \varnothing$ and $\Lambda_3 \leftarrow \varnothing$
2: Set $\mathcal{C} \leftarrow \mathcal{C}^{(0)}$
3: **for** $r = 1$ to n_{swap} **do**
4:    $(\mathbb{P}_{2b}, \mathbb{P}_{3b}) \leftarrow$ GraphColoring$(\mathcal{C})$ ▷ $\mathbb{P}_{2b} = \{\mathbb{P}_{2b}^{(i)}\}_{i=1}^{R_2}$, $\mathbb{P}_{3b} = \{\mathbb{P}_{3b}^{(j)}\}_{j=1}^{R_3}$
5:    $\Lambda_2 \leftarrow \Lambda_2 \cup \{\mathbb{P}_{2b}^{(i)}\}_{i=1}^{\rho_{2b}}$ ▷ Select ρ_{2b} two-body layers
6:    $\Lambda_3 \leftarrow \Lambda_3 \cup \{\mathbb{P}_{3b}^{(j)}\}_{j=1}^{\rho_{3b}}$ ▷ Select ρ_{3b} three-body layers
7:    **if** $r < n_{\text{swap}}$ **then**
8:       $\mathcal{C} \leftarrow$ SwapRegister$\left(\mathcal{C}, \mathbb{P}_{2b}^{(1)}\right)$ ▷ Interpret leading two-body layer as SWAP
9:    **end if**
10: **end for**
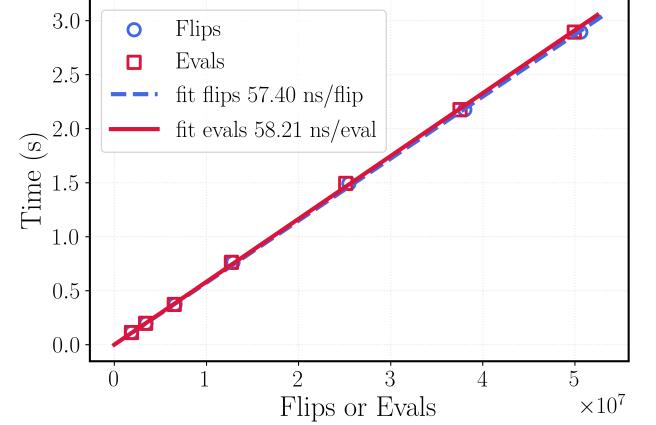11: **return** (Λ_2, Λ_3)

FIG. 5. **Estimating per-flip runtime for MTS.** Total runtime of the Memetic tabu Search (MTS) algorithm as a function of the number of bit-flips (function evaluations). A linear fit is applied to extract the average time per flip, which is subsequently used to estimate MTS runtimes throughout the manuscript.

adiabatic gauge potential, resulting in:

$$
\begin{aligned}
H_{cd} = -2\beta_1(t)\Bigg[&\sum_{i=1}^{N} h_i^x h_i^z \sigma_i^y \\
&+ \sum_{(m,n)\in\mathbb{P}_{2b}} J_{mn}\left(h_m^x \sigma_m^y \sigma_n^z + h_n^x \sigma_m^z \sigma_n^y\right) \\
&+ \sum_{(p,q,r)\in\mathbb{P}_{3b}} K_{pqr}\left(h_p^x \sigma_p^y \sigma_q^z \sigma_r^z + h_q^x \sigma_p^z \sigma_q^y \sigma_r^z + h_r^x \sigma_p^z \sigma_q^z \sigma_r^y\right)\Bigg],
\end{aligned}
\tag{7}
$$

where the time-dependent scaling function $\beta_1(t)$ is analytically derived in Ref. [47]. The algorithm begins by preparing the ground state of H_m using a product state $|\psi_i\rangle = \bigotimes_{i=1}^{N} R_y(\theta_i)|0\rangle$, with the angle θ_i determined by

$$
\theta_i = \tan^{-1}\left(\frac{h_i^x}{h_i^b + \sqrt{(h_i^b)^2 + (h_i^x)^2}}\right).
$$

This state is evolved under H_{cd} in the impulse regime, where time evolution is digitized and includes contributions from one-body, two-body, and three-body interactions.

To construct the quantum circuit, all single-qubit gates are applied in parallel, followed by parallel layers of three-body and then two-body interaction terms. A SWAP layer is applied at the end of each sequence, and this entire sequence is repeated for n SWAP layers. At each layer, only $\mathbb{P}_{3b}$ three-body and $\mathbb{P}_{2b}$ two-body terms are applied, following the ordering strategy outlined in Algorithm 1. The deliberate ordering, placing three-body interactions before two-body terms, ensures more efficient gate compilation and helps cancel redundant operations introduced by subsequent swaps.

All the experiments were executed on the IBM Quantum backends `ibm_kingston`, `ibm_marrakesh`, or `ibm_aachen`, depending on queue time and hardware availability [49]. We transpiled the BF-DCQO circuits using Qiskit's transpiler at optimization level 3 [64], targeting IBM's hardware-native gate set $\{\text{CZ}, R_z(\theta), \sqrt{X}, X\}$, where $\text{CZ} = \text{diag}(1,1,1,-1)$ and $R_z(\theta) = \exp(-i\theta\sigma^z/2)$. To further reduce circuit depth and improve hardware efficiency, we incorporated fractional gates [65], natively supported on IBM's Heron QPUs. Specifically, we used $R_{zz}(\theta) = \exp(-i\theta\sigma_0^z\sigma_1^z/2)$ for $0 < \theta \leq \pi/2$, and $R_x(\theta) = \exp(-i\theta\sigma^x/2)$ for arbitrary θ, which allowed us to implement entangling operations more compactly.

After each evolution step, n_{shots} measurements are performed in the computational basis. From the outcome distribution, the n_{CVaR} lowest-energy bitstrings are selected according to the conditional-value-at-risk (CVaR) approach [47, 66, 67]. These configurations inform the update of the bias fields h_i^b for the next iteration of the algorithm. The overall process is repeated for n_{iter} iterations to progressively refine the solution quality.

C. Memetic tabu search

The goal is to discover a binary string $\mathbf{s} \in \{0,1\}^n$ that minimizes an energy functional $E(\mathbf{s})$ defined by an Ising-style problem instance. The *memetic tabu search* (MTS) couples two complementary search strategies: (i) a tabu-guided bit-flip local search that intensively exploits the energy landscape, and (ii) a population-based evolutionary layer that enables global exploration and information

Algorithm 2 Local tabu Search

Require: incumbent $\mathbf{s}$, energy $E(\mathbf{s})$, tabu length L, max iterations I_{tabu}, (optional) target E_{target}
Ensure: locally–optimized string $\mathbf{s}^\star$ and energy $E^\star$
1: initialize FIFO tabu list $\mathcal{T} \leftarrow \emptyset$
2: $\mathbf{s}^\star \leftarrow \mathbf{s}, E^\star \leftarrow E(\mathbf{s})$
3: **if** $E^\star \le E_{\text{target}}$ **then**
4:     **return** $(\mathbf{s}^\star, E^\star)$
5: **end if**
6: **for** $k = 1$ to I_{tabu} **do**
7:     $\mathcal{N} \leftarrow$ Hamming-1 neighbors of $\mathbf{s}$ not in $\mathcal{T}$
8:     **if** $\mathcal{N} = \emptyset$ **then**
9:         **break**
10:     **end if**
11:     choose $\mathbf{v} \in \mathcal{N}$ with minimum $E(\mathbf{v})$
12:     append $\mathbf{s}$ to $\mathcal{T}$; discard oldest if $|\mathcal{T}| > L$
13:     $\mathbf{s} \leftarrow \mathbf{v}, E(\mathbf{s}) \leftarrow E(\mathbf{v})$
14:     **if** $E(\mathbf{s}) < E^\star$ **then**
15:         $\mathbf{s}^\star \leftarrow \mathbf{s}, E^\star \leftarrow E(\mathbf{s})$
16:         **if** $E^\star \le E_{\text{target}}$ **then**
17:             **return** $(\mathbf{s}^\star, E^\star)$
18:         **end if**
19:     **end if**
20: **end for**
21: **return** $(\mathbf{s}^\star, E^\star)$

Algorithm 3 Memetic tabu Search

Require: bit length n, population size P, generations $G_{\max}$, tabu iterations I_{tabu}, mutation range $(\mu_{\text{start}}, \mu_{\text{end}})$, (optional) bitstring $\mathbf{s}_{\text{ws}}$, (optional) target E_{target}
Ensure: best string $\mathbf{s}_{\text{best}}$ and energy E_{best}
1: build initial population $\mathcal{P}$:
2: **if** bitstring not supplied **then**
3:     sample one string uniformly at random
4: **end if**
5: **for** each candidate until $|\mathcal{P}| = P$ **do**
6:     refine with **Local tabu Search**
7:     **if** target reached **then**
8:         **return** best candidate
9:     **end if**
10: **end for**
11: $(\mathbf{s}_{\text{best}}, E_{\text{best}}) \leftarrow$ best member of $\mathcal{P}$
12: **for** $g = 0$ to $G_{\max} - 1$ **do**
13:     $\mu_g \leftarrow \mu_{\text{end}} + (\mu_{\text{start}} - \mu_{\text{end}}) \dfrac{\ln(G_{\max} + 1 - g)}{\ln(G_{\max} + 1)}$ ▷
    decaying mutation
14:     $\mathcal{O} \leftarrow \emptyset$ ▷ offspring set
15:     **for** $i = 1$ to P **do**
16:         pick two distinct parents uniformly from $\mathcal{P}$
17:         single-point crossover $\rightarrow$ child $\mathbf{c}$
18:         mutate each bit of $\mathbf{c}$ with probability μ_g
19:         refine $\mathbf{c}$ with **Local tabu Search**
20:         **if** target reached **then**
21:             **return** $(\mathbf{c}, E(\mathbf{c}))$
22:         **end if**
23:         add $\mathbf{c}$ to $\mathcal{O}$
24:     **end for**
25:     $\mathcal{P} \leftarrow P$ best strings from $\mathcal{P} \cup \mathcal{O}$ ▷ elitist replacement
26:     **if** best energy in $\mathcal{P}$ improves E_{best} **then**
27:         update $(\mathbf{s}_{\text{best}}, E_{\text{best}})$
28:     **end if**
29: **end for**
30: **return** $(\mathbf{s}_{\text{best}}, E_{\text{best}})$

---

exchange among candidate solutions.

The local search component begins from an incumbent string and repeatedly inspects its Hamming-1 neighborhood. At each iteration, every one-bit variant is evaluated, but any configuration found in a short, fixed-length *tabu list* is skipped. The admissible neighbor with the lowest energy becomes the new incumbent, while the previous incumbent is appended to the tabu list. This process continues for a bounded number of iterations or until a user-defined energy target is reached, whichever comes first.

To initialize the population, a set of P candidate strings is generated. Optionally, a bitstring may be supplied, otherwise, all strings are sampled uniformly at random. Each initial string is immediately refined via the local tabu search, ensuring that the algorithm begins from locally optimized states rather than raw random samples.

The evolutionary layer proceeds for a prescribed number of generations. In each generation, two parents are selected at random without replacement, combined via single-point crossover, and then subjected to bit-wise mutation. The mutation probability follows a logarithmic decay schedule as shown in Eq. (3). Here, g denotes the current generation and $G_{\max}$ is the total number of generations. This schedule starts aggressively to promote exploration and gradually cools as the search progresses. Each offspring is immediately refined by the local tabu search, yielding a memetically improved individual. Once all offspring have been generated and refined, the union of parents and offspring is ranked by energy, and only the top P solutions are retained. This elitist replacement scheme ensures that the best solution found so far is never lost between generations, while allowing better newcomers to enter the population. The algorithm terminates when any one of the following criteria is met: (i) an individual reaches the desired target energy; (ii) the maximum number of generations is reached; or (iii) no improvement occurs during local refinement for any individual in a generation, indicating stagnation.

To estimate the runtime of the algorithm, we executed MTS with increasing generation limits $G_{\max} \in \{5, 10, 20, 40, 80, 120, 160\}$ on a HUBO instance of size $N = 156$. For each run we measured the elapsed time as a function of both the number of bit flips and the number of function evaluations performed. The results are shown in Fig. 5. By fitting a linear model to these data and extrapolating, we obtained estimates for the time per bit flip and the time per function evaluation. Since both quantities grow at nearly identical rates, we define the runtime of MTS as $T_{\text{MTS}} = n_{\text{bitflip}} \times 5.7 \times 10^{-8}$ s.

TABLE VI. **D-Wave results under different annealing times.** Making use of D-Wave ADVANTAGE2_SYSTEM1.6 quantum annealer, we solve all eight HUBO instances, setting as annealing times $t_a \in \{0.5, 20, 100, 1000, 2000\}$ µs, using $n_{\text{shots}} \in \{39920, 37030, 28570, 8000, 4440\}$, respetively, to guarantee total sampling times ranging from $5.5 - 9.5$ s on average. For each of them, we attach the minimum energies obtained and optimality gaps $\mathcal{G}$, including the actual ground states (GS) as a reference. Best results per instance are in bold.

Instance	GS	$t_a = 0.5$ µs		$t_a = 20$ µs		$t_a = 100$ µs		$t_a = 1000$ µs		$t_a = 2000$ µs	
		$E_{\min}$	$\mathcal{G}$ (%)	$E_{\min}$	$\mathcal{G}$ (%)	$E_{\min}$	$\mathcal{G}$ (%)	$E_{\min}$	$\mathcal{G}$ (%)	$E_{\min}$	$\mathcal{G}$ (%)
1	-218.098	-161.185	26.095	-166.253	23.771	-170.542	21.805	**-174.652**	19.920	-172.054	21.112
2	-194.771	-131.297	32.589	-141.377	27.414	-146.813	24.623	**-147.922**	24.053	-147.918	24.055
3	-191.360	-133.978	29.986	-140.384	26.639	-149.367	21.945	**-154.865**	19.071	-147.603	22.866
4	-242.283	-172.027	28.997	-182.116	24.833	-188.185	22.328	-185.723	23.345	**-195.150**	19.454
5	-221.722	-141.397	36.228	**-163.465**	26.275	-160.719	27.513	-158.650	28.446	-157.407	29.007
6	-229.319	-134.108	41.519	-148.808	35.109	-154.888	32.457	**-163.039**	28.903	-152.001	33.716
7	-219.571	-131.745	39.999	-145.718	33.635	-153.470	30.105	**-155.455**	29.201	-147.872	32.654
8	-223.186	-142.857	35.992	-151.062	32.316	-156.517	29.871	**-158.339**	29.055	-154.380	30.829

D. Simulated annealing

We apply SA directly to the HUBO formulation of Eq. (2). All spins are initialized at random, and an upper bound on the largest single-spin energy change, $\Delta E^{\max} = \max_i \Delta E_i^{\max}$, is computed. This bound defines the initial temperature, $T_{\text{init}} = \Delta E^{\max}$, while the final temperature is set to $T_{\text{final}} = 0.01\,T_{\text{init}}$. A geometric cooling schedule spans these limits, assigning one temperature per sweep.

Each run comprises n_{sweep} sweeps. At the start of every sweep, the spin indices are randomly permuted. Visiting spins in this order, we compute the exact energy change ΔE for a proposed flip and accept the move according to the Metropolis–Hastings rule [16, 68]. After all sweeps have concluded, the lowest energy encountered in the run is recorded. The entire procedure is repeated for n_{runs} independent runs, and the best overall configuration is retained. To make full use of available hardware, runs are parallelized across all CPU cores (see Table I) and time per sweep is set to the value found in Ref. [45].

E. Extended Results

In this section, we start by presenting and discussing extended experimental results using D-Wave after solving the eight instances with their quantum annealing devices directly and their hybrid solvers through the LEAPHYBRIDBQMSAMPLER class after HUBO-to-QUBO conversion with the DIMOD library [55]. In Table VI we present the obtained results on D-Wave's ADVANTAGE2_SYSTEM1.6 for different annealing times $t_a \in [0.5, 2000]$ µs, indicating the minimum energies obtained and their corresponding optimality gaps. We set as annealing times $t_a \in \{0.5, 20, 100, 1000, 2000\}$ µs, using as number of samples $n_{\text{shots}} \in \{39920, 37030, 28570, 8000, 4440\}$, respetively, to guarantee total QPU sampling times ranging approximately from $5.5 - 9.5$ s on average.

For the HUBO-to-QUBO conversion, we set as La-

grange multiplier for the penalty terms $w = 1 + W_{\max}$, with $W_{\max}$ the largest strength in absolute value of the HUBO problem, which are on average roughly $W_{\max} \approx 6$ for our instances. After conversion, our 156-qubit HUBO instances require 580 qubits in its quadratic form, and posterior embedding on hardware requires 1000 to 1400 qubits, which is between $6\times$ to $9\times$ greater than the number of qubits present in the original HUBO form. We can observe that, as expected, for larger annealing times, better solutions are obtained in general. However, the optimality gaps obtained range from $20\% - 35\%$, showcasing that the results obtained on D-Wave hardware are not as good as our HSQC approach. The qubit overhead and the inclusion of constraints not naturally present in the original problem might be some of the reasons behind.

Given that D-Wave also offers hybrid classical-quantum solutions for a wide range of optimization problems [31], we again solve the eight instances to compare our HSQC results with their hybrid workflow. Essentially, given a time limit T (in seconds) to solve the input problem, their service starts by running in parallel different heuristic methods on CPUs and GPUs, whose set of solutions might be used to further refine them by means of quantum resources in a later stage [69]. Despite the classical algorithms involved and the recipe behind their decomposition pipeline being proprietary, previous documentation and technical reports indicate that QB-SOLV [27] and KERBEROS [70] have been taken into account, whose heuristics involve tabu search and simulated annealing.

In Table VII we present the obtained results across all instances using as time limits $T = 3$ and 4 s employing again the same HUBO-to-QUBO conversion procedure as for the Table VI results. For this case, we observe in general that the larger the time limit set the better solutions obtained. However, despite the hybrid solutions are lower in energy when compared to the standalone quantum annealing ones (mostly due to the additional classical heuristics), the obtained results are still worse than the ones reported for our HSQC approach.

Apart from this, we present extended experimen-

TABLE VII. **D-Wave hybrid solver results.** Across all eight HUBO instances, D-Wave hybrid workflow best solutions and their corresponding optimality gaps (computed as $\mathcal{G} = 100(E_{\min} - \mathrm{GS})/|\mathrm{GS}|$) using the Greedy energy as $E_{\min}$ under time limits $T = 3$ and $4\,\mathrm{s}$. Additionally, the actual ground states (GS) and QPU access times T_{QPU} are attached. Best results per instance are in bold.

Instance	GS	Time limit: $T = 3\,\mathrm{s}$					Time limit: $T = 4\,\mathrm{s}$				
		Raw	Greedy	$\mathcal{G}$ (%)	T_{QPU} [ms]	T [ms]	Raw	Greedy	$\mathcal{G}$ (%)	T_{QPU} [ms]	T [ms]
1	-218.098	-186.937	-195.166	10.515	51.812	2997.692	-191.620	**-208.302**	4.492	155.422	4000.550
2	-194.771	-165.975	**-178.039**	8.591	103.616	2991.615	-164.750	-173.883	10.724	155.431	3998.272
3	-191.360	-160.174	**-179.938**	5.969	103.612	2989.480	-163.832	-178.545	6.697	155.415	3990.059
4	-242.283	-219.711	-226.620	6.465	103.610	3002.142	-218.911	**-226.954**	6.327	155.419	3987.376
5	-221.722	-187.307	-193.554	12.704	103.621	2996.628	-192.608	**-200.948**	9.369	155.420	3996.789
6	-229.319	-197.674	-208.123	9.243	103.614	2997.377	-203.128	**-217.395**	5.200	155.418	3988.413
7	-219.571	-190.788	**-206.462**	5.970	103.617	2992.475	-189.893	-201.422	8.266	155.429	3986.460
8	-223.186	-191.627	**-203.340**	8.892	103.621	2989.875	-191.040	-203.035	9.029	155.422	4001.009

TABLE VIII. **Performance and runtime comparison between MTS and HSQC.** Columns list instance index, ground state energy (GS), BF-DCQO energy, and for several $G_{\max}$ values, the minimum and mean-best energies and optimality gaps from MTS and HSQC. Lower energy and lower gap are indicate better performance and $\mathcal{G} = 100\,(E_{\min} - E_{\mathrm{GS}})/|E_{\mathrm{GS}}|$. Best results are in bold.

Instance	GS	BF-DCQO	Gen	Minimum energy $E_{\min}$		Mean best energy $\overline{E}_{\mathrm{best}}$		Optimality gap $\mathcal{G}$ (%)	
				MTS	HSQC	MTS	HSQC	MTS	HSQC
1	-218.0978	-215.5230	25	-204.0179	**-216.5204**	-198.9679	**-216.1306**	6.456	0.723
			50	-214.8493	**-216.5494**	-206.6768	**-216.2045**	1.489	0.710
			75	-216.2912	**-216.3255**	-208.1639	**-216.1919**	0.828	0.813
			100	-214.7896	**-216.5204**	-210.0768	**-216.1993**	1.517	0.723
2	-194.7714	-188.6820	25	-191.4881	**-193.2080**	-188.4791	**-192.3972**	1.686	0.803
			50	-193.4040	**-194.7714**	-191.8949	**-193.2756**	0.702	0.000
			75	-193.8005	**-194.7714**	-191.6143	**-194.1208**	0.498	0.000
			100	-193.9923	**-194.7714**	-191.8852	**-194.1211**	0.400	0.000
3	-191.3601	-185.3596	25	-185.3193	**-188.3562**	-183.1117	**-187.0558**	3.157	1.570
			50	-187.2410	**-188.7335**	-185.4041	**-187.8235**	2.153	1.373
			75	-187.9333	**-189.3062**	-186.7063	**-188.3787**	1.791	1.073
			100	**-190.3348**	-189.4325	-187.4519	**-188.4597**	0.536	1.007
4	-242.2827	-238.0770	25	-239.2863	**-239.7158**	-234.3365	**-239.0612**	1.237	1.059
			50	**-240.6513**	-240.5283	-237.3003	**-240.0373**	0.673	0.724
			75	-241.4677	**-241.5935**	-237.7555	**-240.5258**	0.336	0.284
			100	**-240.8505**	-240.5830	-238.8252	**-240.3679**	0.591	0.702
5	-221.7218	-219.9218	25	-211.7941	**-219.9219**	-206.2524	**-219.9219**	4.478	0.812
			50	-213.0573	**-220.3947**	-210.2753	**-219.9691**	3.908	0.599
			75	-215.9287	**-219.9219**	-212.8430	**-219.9219**	2.613	0.812
			100	-216.1129	**-221.0083**	-214.4136	**-220.0305**	2.530	0.322
6	-229.3192	-225.7244	25	-221.2532	**-229.1247**	-213.4332	**-228.3942**	3.517	0.085
			50	-227.8142	**-229.1247**	-221.4388	**-228.7137**	0.656	0.085
			75	-228.0460	**-229.1247**	-223.0950	**-228.8468**	0.555	0.085
			100	-228.8585	**-229.1626**	-223.7416	**-229.0780**	0.201	0.068
7	-219.5705	-210.6941	25	-212.7764	**-215.3337**	-209.1954	**-213.8719**	3.094	1.930
			50	**-216.6372**	-215.8933	-212.7452	**-214.7094**	1.336	1.675
			75	**-219.0577**	-216.0782	-213.4183	**-214.9631**	0.234	1.591
			100	**-217.4918**	-216.8694	-214.4788	**-215.5784**	0.947	1.230
8	-223.1855	-221.1362	25	-218.6106	**-222.9956**	-216.3979	**-221.6585**	2.050	0.085
			50	-220.1601	**-222.9956**	-217.5255	**-221.9285**	1.355	0.085
			75	-221.2916	**-223.1856**	-219.2352	**-222.0257**	0.848	0.000
			100	-220.5968	**-223.1856**	-219.1314	**-222.3384**	1.160	0.000

tal results supporting Fig. 2 and Fig. 4. We evaluate eight HUBO instances using both variants of the HSQC pipeline: SA+BF-DCQO+MTS and SA+BF-DCQO+SA. Each instance is executed ten times, and

TABLE IX. **Performance and runtime comparison between SA and HSQC.** Columns list instance index, ground state energy (GS), SA energy (600 s), BF-DCQO energy, and for each number of sweeps the minimum and mean-best energies and optimality gaps from SA and HSQC. Lower energy and lower gap are indicate better performance and $\mathcal{G} = 100\,(E_{\min} - E_{\mathrm{GS}})/|E_{\mathrm{GS}}|$. Best results are in bold.

Instance	GS	BF-DCQO	SA (600 s)	Sweeps	Minimum energy $E_{\min}$		Mean best energy $\overline{E}_{\mathrm{best}}$		Optimality gap $\mathcal{G}$ (%)	
					SA	HSQC	SA	HSQC	SA	HSQC
1	-218.0978	-215.5230	-218.0978	1000	-215.2862	**-215.5230**	-212.3657	**-215.5230**	1.289	1.181
				2000	-215.2182	**-216.7411**	-213.4481	**-215.6448**	1.320	0.622
				5000	-215.9142	**-216.0701**	-214.6536	**-215.6611**	1.001	0.930
				10000	-215.9533	**-217.6037**	-214.9753	**-216.1209**	0.983	0.227
2	-194.7714	-188.6821	-194.1150	1000	**-192.4975**	-191.8835	-190.4311	**-190.6678**	1.168	1.483
				2000	-191.8591	**-193.3848**	-191.2304	**-191.3469**	1.495	0.712
				5000	**-193.7014**	-193.2726	-192.0922	**-192.1573**	0.549	0.770
				10000	-192.8074	**-194.2513**	-192.1736	**-192.7463**	1.008	0.267
3	-191.3602	-185.3597	-190.9482	1000	-188.3781	**-188.8758**	-186.7092	**-186.8656**	1.558	1.298
				2000	-188.9155	**-190.1625**	-187.1693	**-188.3467**	1.278	0.626
				5000	**-189.2850**	-188.4505	**-188.2682**	-187.7365	1.084	1.521
				10000	**-190.2138**	-189.7987	**-188.5346**	-188.4772	0.599	0.816
4	-242.2827	-238.0770	-241.7211	1000	**-239.3209**	-238.0770	-236.9490	**-238.0770**	1.223	1.736
				2000	**-240.1531**	-238.8498	-237.5054	**-238.2007**	0.879	1.417
				5000	**-241.5584**	-240.6580	-238.5045	**-239.3972**	0.299	0.671
				10000	**-240.6361**	-240.2790	-239.0974	**-239.5613**	0.680	0.827
5	-221.7219	-219.9219	-221.7219	1000	-219.7765	**-219.9219**	-216.6210	**-219.9219**	0.877	0.812
				2000	-218.6638	**-219.9219**	-217.4121	**-219.9219**	1.379	0.812
				5000	-219.7485	**-220.3197**	-218.4066	**-219.9977**	0.890	0.632
				10000	**-221.3394**	-220.6576	-219.5597	**-220.2752**	0.173	0.480
6	-229.3193	-225.7245	-229.3127	1000	**-226.5969**	-225.7245	-223.5287	**-225.7245**	1.187	1.568
				2000	-225.8982	**-226.4108**	-223.9656	**-225.8952**	1.492	1.268
				5000	**-228.5780**	-227.5673	**-226.9006**	-225.9407	0.323	0.764
				10000	**-228.0932**	-228.0684	-226.6325	**-226.6413**	0.535	0.546
7	-219.5705	-210.6941	-219.4862	1000	**-215.9558**	-215.2018	**-213.6149**	-213.1492	1.646	1.990
				2000	**-217.3946**	-216.6785	**-215.1870**	-214.2861	0.991	1.317
				5000	**-217.6086**	-216.3228	-215.5243	**-215.6178**	0.894	1.479
				10000	-218.0402	**-218.8007**	-216.2090	**-217.1442**	0.697	0.351
8	-223.1856	-221.1363	-223.1856	1000	-220.7617	**-221.1363**	-218.1452	**-221.1363**	1.086	0.918
				2000	-220.4307	**-221.1363**	-218.8203	**-221.1363**	1.234	0.918
				5000	-221.0350	**-221.7688**	-219.7895	**-221.2686**	0.964	0.635
				10000	-221.2851	**-222.9956**	-220.5006	**-221.2825**	0.852	0.085

we report the minimum energy $E_{\min}$ across all trials, as well as the mean of the best energies obtained in each trial. In addition to the average optimality gap $\bar{\mathcal{G}}$, which is visualized in Fig. 2 and Fig. 4, we also provide the optimality gap $\mathcal{G}$ computed from the minimum energy achieved. Table VIII summarizes results for the MTS variant, showing performance as a function of increasing generation limits $G_{\max} \in \{25, 50, 75, 100\}$. In contrast, Table IX presents results for the SA variant, reporting performance for increasing sweep counts $n_{\mathrm{sweep}} \in \{1000, 2000, 5000, 10000\}$. For context, we also include SA results at $n_{\mathrm{sweep}}^{\mathrm{SA}} = 100{,}000$ and $n_{\mathrm{runs}}^{\mathrm{SA}} = 1000$, corresponding to a reference runtime of $T_{\mathrm{SA}} = 600$ seconds.

[1] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Bal- lard, J. Bambrick, S. W. Bodenstein, D. A. Evans, C.-C. Hung, M. O'Neill, D. Reiman, K. Tunyasuvu-

nakool, Z. Wu, A. Žemgulytė, E. Arvaniti, C. Beattie, O. Bertolli, A. Bridgland, A. Cherepanov, M. Congreve, A. I. Cowen-Rivers, A. Cowie, M. Figurnov, F. B. Fuchs, H. Gladman, R. Jain, Y. A. Khan, C. M. R. Low, K. Perlin, A. Potapenko, P. Savy, S. Singh, A. Stecula, A. Thillaisundaram, C. Tong, S. Yakneen, E. D. Zhong, M. Zielinski, A. Žídek, V. Bapst, P. Kohli, M. Jaderberg, D. Hassabis, and J. M. Jumper, Accurate structure prediction of biomolecular interactions with AlphaFold 3, Nature **630**, 493–500 (2024).

[2] A. Merchant, S. Batzner, S. S. Schoenholz, M. Aykol, G. Cheon, and E. D. Cubuk, Scaling deep learning for materials discovery, Nature **624**, 80–85 (2023).

[3] R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, O. Vinyals, J. Stott, A. Pritzel, S. Mohamed, and P. Battaglia, Learning skillful medium-range global weather forecasting, Science **382**, 1416 (2023).

[4] OpenAI, GPT-4 Technical Report (2024), arXiv:2303.08774 [cs.CL].

[5] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, When is "nearest neighbor" meaningful?, in *Database Theory — ICDT'99*, edited by C. Beeri and P. Buneman (Springer Berlin Heidelberg, Berlin, Heidelberg, 1999) pp. 217–235.

[6] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, Deep double descent: Where bigger models and more data hurt, in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020* (OpenReview.net, 2020).

[7] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, Understanding deep learning (still) requires rethinking generalization, Commun. ACM **64**, 107–115 (2021).

[8] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik **1**, 269–271 (1959).

[9] P. E. Hart, N. J. Nilsson, and B. Raphael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics **4**, 100 (1968).

[10] E. G. Coffman Jr., J. Csirik, G. Galambos, S. Martello, and D. Vigo, Bin packing approximation algorithms: Survey and classification, in *Handbook of Combinatorial Optimization*, edited by P. M. Pardalos, D.-Z. Du, and R. L. Graham (Springer New York, New York, NY, 2013) pp. 455–531.

[11] R. Duan, J. Mao, X. Mao, X. Shu, and L. Yin, Breaking the Sorting Barrier for Directed Single-Source Shortest Paths, in *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, STOC '25 (Association for Computing Machinery, New York, NY, USA, 2025) p. 36–44.

[12] S. Liu, Chain, Generalization of Covering Code, and Deterministic Algorithm for k-SAT, in *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 107, edited by I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2018) pp. 88:1–88:13.

[13] H. Xiong, S. Shi, D. Ren, and J. Hu, A survey of job shop scheduling problem: The types and models, Computers & Operations Research **142**, 105731 (2022).

[14] C. Sundermann, T. Heß, M. Nieke, P. M. Bittner, J. M. Young, T. Thüm, and I. Schaefer, Evaluating state-of-the-art # SAT solvers on industrial configuration spaces, Empirical Software Engineering **28**, 29 (2023).

[15] N. Metropolis and S. Ulam, The Monte Carlo Method, Journal of the American Statistical Association **44**, 335 (1949).

[16] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, Equation of State Calculations by Fast Computing Machines, The Journal of Chemical Physics **21**, 1087 (1953).

[17] A. Barbu and S.-C. Zhu, *Monte Carlo Methods* (Springer Singapore, Singapore, 2020).

[18] P. Hohenberg and W. Kohn, Inhomogeneous Electron Gas, Phys. Rev. **136**, B864 (1964).

[19] W. Kohn and L. J. Sham, Self-Consistent Equations Including Exchange and Correlation Effects, Phys. Rev. **140**, A1133 (1965).

[20] J. P. Perdew, K. Burke, and M. Ernzerhof, Generalized Gradient Approximation Made Simple, Phys. Rev. Lett. **77**, 3865 (1996).

[21] R. O. Jones, Density functional theory: Its origins, rise to prominence, and future, Rev. Mod. Phys. **87**, 897 (2015).

[22] A. Georges, G. Kotliar, W. Krauth, and M. J. Rozenberg, Dynamical mean-field theory of strongly correlated fermion systems and the limit of infinite dimensions, Rev. Mod. Phys. **68**, 13 (1996).

[23] A. J. Cohen, P. Mori-Sánchez, and W. Yang, Insights into Current Limitations of Density Functional Theory, Science **321**, 792 (2008).

[24] D. J. Egger, J. Mareček, and S. Woerner, Warm-starting quantum optimization, Quantum **5**, 479 (2021).

[25] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nature Communications **5**, 4213 (2014).

[26] D. W. Berry, Y. Tong, T. Khattar, A. White, T. I. Kim, G. H. Low, S. Boixo, Z. Ding, L. Lin, S. Lee, G. K.-L. Chan, R. Babbush, and N. C. Rubin, Rapid Initial-State Preparation for the Quantum Simulation of Strongly Correlated Molecules, PRX Quantum **6**, 020327 (2025).

[27] M. Booth, S. P. Reinhardt, and A. Roy, *Partitioning Optimization Problems for Hybrid Classical/Quantum Execution*, Technical Report 14-1006A-A (D-Wave Systems Inc., Burnaby, BC, Canada, 2017).

[28] Z. Zhou, Y. Du, X. Tian, and D. Tao, QAOA-in-QAOA: Solving Large-Scale MaxCut Problems on Small Quantum Machines, Phys. Rev. Appl. **19**, 024027 (2023).

[29] M. Y. Naghmouchi and W. d. S. Coelho, Mixed-integer linear programming solver using benders decomposition assisted by a neutral-atom quantum processor, Phys. Rev. A **110**, 012434 (2024).

[30] A. Acharya, R. Yalovetzky, P. Minssen, S. Chakrabarti, R. Shaydulin, R. Raymond, Y. Sun, D. Herman, R. S. Andrist, G. Salton, M. J. A. Schuetz, H. G. Katzgraber, and M. Pistoia, Decomposition pipeline for large-scale portfolio optimization with applications to near-term quantum computing, Phys. Rev. Res. **7**, 023142 (2025).

[31] D-Wave Systems Inc., *D-Wave Hybrid Solver Service: An Overview*, Whitepaper 14-1039A-B (D-Wave Systems Inc., Burnaby, BC, Canada, 2020).

[32] A. Montanaro, Quantum speedup of branch-and-bound algorithms, Phys. Rev. Res. **2**, 013056 (2020).

[33] Z. Peng, D. de Roux, and D. E. B. Neira, Hybrid Quantum Branch-and-Bound Method for Quadratic Unconstrained Binary Optimization (2025), arXiv:2509.11040 [math.OC].

[34] S. Jeong, J. Park, and J. Ahn, Quantum-Enhanced Simulated Annealing Using Rydberg Atoms, Advanced Quantum Technologies n/a, e2500070 (2025).

[35] S. V. Romero, A. G. Cadavid, E. Solano, and N. N. Hegade, Sequential Quantum Computing (2025), arXiv:2506.20655 [quant-ph].

[36] C. Durr and P. Hoyer, A Quantum Algorithm for Finding the Minimum (1999), arXiv:quant-ph/9607014 [quant-ph].

[37] R. D. Somma, S. Boixo, H. Barnum, and E. Knill, Quantum Simulations of Classical Annealing Processes, Phys. Rev. Lett. 101, 130504 (2008).

[38] P. Wocjan and A. Abeyesinghe, Speedup via quantum sampling, Phys. Rev. A 78, 042336 (2008).

[39] M. B. Hastings, A Short Path Quantum Algorithm for Exact Optimization, Quantum 2, 78 (2018).

[40] A. Montanaro, Quantum-Walk Speedup of Backtracking Algorithms, Theory of Computing 14, 1 (2018).

[41] S. Chakrabarti, P. Minssen, R. Yalovetzky, and M. Pistoia, Universal Quantum Speedup for Branch-and-Bound, Branch-and-Cut, and Tree-Search Algorithms (2022), arXiv:2210.03210 [quant-ph].

[42] 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).

[43] S. Boulebnane and A. Montanaro, Solving Boolean Satisfiability Problems With The Quantum Approximate Optimization Algorithm, PRX Quantum 5, 030348 (2024).

[44] H. Munoz-Bauza and D. Lidar, Scaling Advantage in Approximate Optimization with Quantum Annealing, Phys. Rev. Lett. 134, 160601 (2025).

[45] P. Chandarana, A. G. Cadavid, S. V. Romero, A. Simen, E. Solano, and N. N. Hegade, Runtime Quantum Advantage with Digital Quantum Optimization (2025), arXiv:2505.08663 [quant-ph].

[46] A. G. Cadavid, A. Dalal, A. Simen, E. Solano, and N. N. Hegade, Bias-field digitized counterdiabatic quantum optimization, Phys. Rev. Res. 7, L022010 (2025).

[47] S. V. Romero, A.-M. Visuri, A. G. Cadavid, A. Simen, E. Solano, and N. N. Hegade, Bias-field digitized counterdiabatic quantum algorithm for higher-order binary optimization, Communications Physics 8, 348 (2025).

[48] IBM Quantum, Iskay Quantum Optimizer - A Qiskit Function by Kipu Quantum, https://docs.quantum.ibm.com/guides/kipu-optimization (2025).

[49] IBM Quantum, https://quantum.cloud.ibm.com/ (2025).

[50] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by Simulated Annealing, Science 220, 671 (1983).

[51] A. Silva, L. C. Coelho, and M. Darvish, Quadratic assignment problem variants: A survey and an effective parallel memetic iterated tabu search, European Journal of Operational Research 292, 1066 (2021).

[52] D-Wave Systems, https://www.dwavesys.com/ (2025).

[53] A. Perdomo-Ortiz, A. Feldman, A. Ozaeta, S. V. Isakov, Z. Zhu, B. O'Gorman, H. G. Katzgraber, A. Diedrich, H. Neven, J. de Kleer, B. Lackey, and R. Biswas, Readiness of quantum optimization machines for industrial applications, Phys. Rev. Appl. 12, 014004 (2019).

[54] Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. van den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, and A. Kandala, Evidence for the utility of quantum computing before fault tolerance, Nature 618, 500–505 (2023).

[55] Ocean Developer Tools, https://docs.ocean.dwavesys.com.

[56] I. I. Cplex, V12.10.0: User's Manual for CPLEX, International Business Machines Corporation 46, 157 (2009).

[57] N. N. Hegade, K. Paul, Y. Ding, M. Sanz, F. Albarrán-Arriagada, E. Solano, and X. Chen, Shortcuts to adiabaticity in digitized adiabatic quantum computing, Physical Review Applied 15, 024038 (2021).

[58] T. Bartz-Beielstein, J. Branke, J. Mehnen, and O. Mersmann, Evolutionary algorithms, WIREs Data Mining and Knowledge Discovery 4, 178 (2014).

[59] F. Glover, E. Taillard, and D. de Werra, A User's Guide to Tabu Search, Annals of Operations Research 41, 3 (1993).

[60] A. Kotil, E. Pelofske, S. Riedmüller, D. J. Egger, S. Eidenbenz, T. Koch, and S. Woerner, Quantum Approximate Multi-Objective Optimization (2025), arXiv:2503.22797 [quant-ph].

[61] A. Simen, S. V. Romero, A. G. Cadavid, E. Solano, and N. N. Hegade, Branch-and-bound digitized counterdiabatic quantum optimization (2025), arXiv:2504.15367 [quant-ph].

[62] S. V. Romero, A. G. Cadavid, P. Nikačević, E. Solano, N. N. Hegade, M. A. Lopez-Ruiz, C. Girotto, M. Yamada, P. K. Barkoutsos, A. Kaushik, and M. Roetteler, Protein folding with an all-to-all trapped-ion quantum computer (2025), arXiv:2506.07866 [quant-ph].

[63] C. Chamberland, G. Zhu, T. J. Yoder, J. B. Hertzberg, and A. W. Cross, Topological and Subsystem Codes on Low-Degree Graphs with Flag Qubits, Phys. Rev. X 10, 011022 (2020).

[64] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, Quantum computing with Qiskit (2024), arXiv:2405.08810 [quant-ph].

[65] IBM Quantum, Fractional gates (2025).

[66] P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner, Improving Variational Quantum Optimization using CVaR, Quantum 4, 256 (2020).

[67] S. V. Barron, D. J. Egger, E. Pelofske, A. Bärtschi, S. Eidenbenz, M. Lehmkuehler, and S. Woerner, Provable bounds for noise-free expectation values computed from noisy samples, Nature Computational Science 4, 865–875 (2024).

[68] W. K. Hastings, Monte carlo sampling methods using markov chains and their applications, Biometrika 57, 97 (1970).

[69] D-Wave Systems Inc., Hybrid Solver for Constrained Quadratic Models, Whitepaper 14-1055A-A (D-Wave Systems Inc., Burnaby, BC, Canada, 2021).

[70] Kerberos hybrid sampler, D-Wave Systems Inc. (2025).