Demonstration of a Scaling Advantage for a Quantum Annealer over Simulated Annealing

Tameem Albash[1,2,3] and Daniel A. Lidar[1,2,4,5]

[1]*Department of Physics and Astronomy, University of Southern California, Los Angeles, California 90089, USA*

[2]*Center for Quantum Information Science & Technology, University of Southern California, Los Angeles, California 90089, USA*

[3]*Information Sciences Institute, University of Southern California, Marina del Rey, California 90292, USA*

[4]*Department of Electrical Engineering, University of Southern California, Los Angeles, California 90089, USA*

[5]*Department of Chemistry, University of Southern California, Los Angeles, California 90089, USA*

(Received 23 October 2017; revised manuscript received 12 June 2018; published 19 July 2018)

The observation of an unequivocal quantum speedup remains an elusive objective for quantum computing. A more modest goal is to demonstrate a scaling advantage over a class of classical algorithms for a computational problem running on quantum hardware. The D-Wave quantum annealing processors have been at the forefront of experimental attempts to address this goal, given their relatively large numbers of qubits and programmability. A complete determination of the optimal time-to-solution using these processors has not been possible to date, preventing definitive conclusions about the presence of a scaling advantage. The main technical obstacle has been the inability to verify an optimal annealing time within the available range. Here, we overcome this obstacle using a class of problem instances constructed by systematically combining many-spin frustrated loops with few-qubit gadgets exhibiting a tunneling event—a combination that we find to promote the presence of tunneling energy barriers in the relevant semiclassical energy landscape of the full problem—and we observe an optimal annealing time using a D-Wave 2000Q processor over a range spanning up to more than 2000 qubits. We identify the gadgets as being responsible for the optimal annealing time, whose existence allows us to perform an optimal time-to-solution benchmarking analysis. We perform a comparison to several classical algorithms, including simulated annealing, spin-vector Monte Carlo, and discrete-time simulated quantum annealing (SQA), and establish the first example of a scaling advantage for an experimental quantum annealer over classical simulated annealing. Namely, we find that the D-Wave device exhibits certifiably better scaling than simulated annealing, with 95% confidence, over the range of problem sizes that we can test. However, we do not find evidence for a quantum speedup: SQA exhibits the best scaling for annealing algorithms by a significant margin. This is a finding of independent interest, since we associate SQA's advantage with its ability to transverse energy barriers in the semiclassical energy landscape by mimicking tunneling. Our construction of instance classes with verifiably optimal annealing times opens up the possibility of generating many new such classes based on a similar principle of promoting the presence of energy barriers that can be overcome more efficiently using quantum rather than thermal fluctuations, paving the way for further definitive assessments of scaling advantages using current and future quantum annealing devices.

DOI: 10.1103/PhysRevX.8.031016 Subject Areas: Quantum Physics, Quantum Information

I. INTRODUCTION

The elusive and tantalizing goal of experimentally demonstrating a quantum speedup is being actively pursued using a variety of quantum computing platforms. The holy grail is an exponential speedup, such as expected with Shor's algorithm for factoring integers [1], or with the simulation of quantum systems [2–4]. This goal is still substantially out of reach given the relatively small scale of current universal quantum computers and quantum simulators (∼20–70 qubits [5–10]), which prevents the implementation of fault tolerant quantum error correction on a scale that would enable quantum circuits to be executed reliably despite decoherence and noise. However, there is reason for optimism [11] that current "noisy intermediate scale quantum" (NISQ) era [12] quantum computers will be capable of demonstrating the important milestone of "quantum supremacy" [13], a less ambitious quantum speedup

goal than that associated with application-level computational tasks such as factoring or quantum simulation.

The largest quantum information processing devices currently available are quantum annealers, featuring several thousands of noisy qubits and programmable qubit-qubit interactions. Unlike universal quantum computers that operate using quantum gates and the principles of the circuit model [14], these devices specialize primarily in solving combinatorial optimization problems, and are designed to represent physical implementations of quantum annealing (QA) [15] and the quantum adiabatic algorithm [16]. While the algorithmic focus in the domain of universal quantum computers has been on demonstrating quantum simulation and quantum supremacy, in QA the primary focus has been on benchmarking the algorithmic performance of quantum annealers against classical algorithms [17–27], an effort that has not yet been undertaken with gate model quantum computers. This difference is explained primarily by the relatively large number of qubits available in QA, which enables scaling tests over several orders of magnitude of problem sizes. Despite the large body of work on benchmarking quantum annealers, conclusive evidence about how their performance scales with problem size has until now been unattainable. The primary reason, as we discuss in detail, is that it has not been possible to identify an optimal annealing time for any class of problem instances, an obstacle that was first pointed out in Ref. [18].

Here, we overcome this obstacle by introducing a new class of problem instances that exhibit an optimal annealing time, and present for the first time a complete algorithmic scaling analysis of a hardware quantum annealer (the D-Wave 2000Q device [34]), up to the largest available problem size of more than 2000 spins or qubits [35].

This advance allows us furthermore to demonstrate the first certifiable observation of an algorithmic scaling advantage obtained using quantum annealing hardware over an important general purpose classical algorithm, namely, over simulated annealing with single-spin updates (SA) [36]. Without the identification of an optimal annealing time, one can certify a scaling disadvantage for the hardware quantum annealer, but not an advantage [20]; this holds for all earlier scaling analyses presented for quantum annealing hardware [17–25].

The advantage of QA over SA we demonstrate holds for a class of problem instances (called "logical planted" and defined below) that we constructed by systematically combining a distribution of frustrated cycles of coupled spins over the entire hardware graph and small "gadgets" made of a relatively small number of qubits (here we used eight) that have a small quantum gap (on the order of the temperature) and that exhibit a tunneling event that can be established via numerical solution of the Schrödinger equation. Since both features are flexible, our construction provides a recipe for generalizing our results to a broader class of problem instances. We show that the optimal annealing time arises due to the gadgets, in the sense that instances based only on frustrated loops do not exhibit an optimal annealing time.

To establish the presence of many-qubit tunneling, we resort to large-scale simulations using the discrete-time simulated quantum annealing (SQA) algorithm [37], which we contrast with the spin-vector Monte Carlo (SVMC) algorithm [38]. Both SQA and SVMC are transverse-field annealing algorithms, and thus more closely model QA than a temperature-annealing algorithm such as SA does. But while the SVMC algorithm is purely classical, in the sense that it provides a time-dependent description in terms of unentangled planar rotors, SQA is a classical algorithm based on the path-integral Monte Carlo formalism, which in its continuous-time limit and for sufficiently many spin updates generates samples from the quantum Gibbs state. In particular, at sufficiently low temperatures SQA can mimic tunneling [39,40] and describe entangled ground states such as those followed by QA. It is the opposite trends exhibited by SQA and SVMC as a function of the simulation temperature that allows us to argue for the occurrence of many-qubit tunneling. We emphasize that the appropriate energy landscape for tunneling is not the classical energy landscape associated with simulated annealing [26] but rather the semiclassical landscape associated with transverse-field annealing [41,42].

Our benchmarking analysis reveals that SQA has the best scaling of all the annealing algorithms we tested for the logical-planted instances, in particular outperforming the quantum annealing hardware. It also outperforms a number of algorithms (described below) designed to specifically exploit features of the "Chimera" hardware graph of the D-Wave devices [43,44]. The fact that SQA performs so well for the logical-planted instance class is in itself a significant and novel finding about the class of logical-planted problem instances, since one might reasonably expect that as hardware quantum annealers continue to improve, SQA will become a lower bound on the performance of such hardware [45]. The reason is that SQA serves as a reasonable classical simulation of a thermally dominated quantum annealer but of course does not actually physically manifest any of the quantum features (unitary dynamics, coherent tunneling, entanglement) that are expected to come into play in a physical realization of sufficiently coherent QA. We show for the logical-planted problem class that, by mimicking tunneling, SQA traverses energy barriers more efficiently as the temperature of the simulation is lowered. We use this to argue that a key reason for the quantum annealer's slowdown relative to SQA is its suboptimally high temperature [46], which causes it to behave more like the SVMC algorithm. Thus, the strong performance of SQA on the logical-planted instance class suggests that this class is a good target or basis for the exploration of an eventual quantum speedup using QA hardware.

We first review and discuss, in Sec. II, the time-to-solution metric, and how to establish optimality. Section III presents our results. First, Sec. III A establishes the empirical evidence for optimal annealing times for our class of problem instances. Then, Sec. III B presents the empirical evidence we found for a QA scaling advantage over SA, but a disadvantage relative to SQA and the SVMC algorithm. In Sec. III C, we introduce and describe the properties of the class of problem instances for which we observe the optimal annealing time and the scaling advantage over SA. We discuss the implications of our results and provide an outlook in Sec. IV. Additional technical details and methods are provided in the Appendixes. A–K.

II. OPTIMAL TIME TO SOLUTION

We consider the standard setting where the goal of the optimizer is to find the optimal solution (i.e., the global minimum of the cost function) and one is interested in minimizing the time taken to find the solution at least once. There is a trade-off between finding the solution with a high probability in a single long run of the algorithm and running the algorithm multiple times with a shorter run time and (usually) a smaller single-run success probability. This trade-off is reflected in the time-to-solution (TTS) metric, which measures the time required to find the ground state at least once with some desired probability p_d (often taken to be 0.99):

$$\mathrm{TTS}(t_f) = t_f R(t_f) \frac{N}{N_{\max}}, \qquad R(t_f) = \frac{\ln(1-p_d)}{\ln[1-p_S(t_f)]}. \quad (1)$$

Here, $p_S(t_f)$ is the success probability of a single-instance run of the algorithm with a run time t_f, and $R(t_f)$ is the required number of runs; success means that the optimal solution was found. The instance size is N, and $N_{\max}$ is the size of the largest instance that the device accommodates (typically set by the total number of qubits); the factor $N/N_{\max}$ accounts for maximal parallel utilization of the device. While R and $N_{\max}/N$ should correspond to whole numbers, we do not round them here since this can result in sharp TTS changes that complicate the extraction of scaling with N; see Appendix A for more details.

However, when considering the performance of an algorithm evaluated over an ensemble of randomly chosen instances from the same class, we are typically interested not in the TTS of a single instance but in a given quantile q of the TTS distribution over such instances at a given problem size, $N \in [N_{\min}, N_{\max}]$. We denote the qth quantile of the TTS evaluated at $t_f(N)$ by $\langle \mathrm{TTS}(t_f) \rangle_q$, and suppress the N dependence for simplicity. Since the goal of optimization is to find the solution as rapidly as possible, there is an optimal t_f value for a given quantile q, t_q^*, where $\langle \mathrm{TTS}(t_f) \rangle_q$ is minimized, and we denote $\langle \mathrm{TTS} \rangle_q^* \equiv \langle \mathrm{TTS}(t_q^*) \rangle_q$. While the success probability of individual instances may exhibit many minima (as in the case of coherent evolution when oscillations in the success probability are observed as the annealing time is varied), the quantile of the TTS distribution exhibits only one minimum because the many minima of the individual instances are unlikely to coincide, and there is no ambiguity in the determination of an optimal t_f value. Finally, it is important to note that since quantiles are solver dependent, a comparison between different solvers at the same quantile involves different sets of instances.

We can now state the precise nature of the critical obstacle alluded to above: if $t_q^* < t_{\min}$, where $t_{\min}$ is the smallest possible annealing time on the given quantum annealer, then it becomes impossible to determine $\langle \mathrm{TTS} \rangle_q^*$. As was shown in Refs. [18,20], when operating with a suboptimal t_f, one can easily be led to false conclusions about the scaling with N of $\langle \mathrm{TTS}(t_f) \rangle_q$ compared to the all-important scaling as captured by $\langle \mathrm{TTS} \rangle_q^*$, and even be led to conclude that there is a scaling advantage where there is none.

None of the experimental quantum annealing benchmarking studies to date [17–26,47] have provided a complete scaling assessment, precisely because it has not been possible to verify that $t_q^* > t_{\min}$ (for any quantile). The culprit was the absence of a suitable class of problem instances for which an optimal annealing time could be verified. Here we report on a class of instances that exhibits an optimal annealing time greater than $t_{\min} = 5\ \mu$s on the D-Wave 2000Q (DW2KQ, fourth generation, for which the largest energy scale is ~50 GHz in $\hbar = 1$ units; see Appendix B) and the D-Wave 2X (DW2X, third generation, for which the largest energy scale is ~40 GHz). In the main text, we focus on the DW2KQ, and we provide results from the DW2X in Appendixes G and K. This allows us to obtain the first complete optimal-TTS scaling results for an experimental quantum annealer, defined as the TTS scaling obtained from certifiably optimal annealing times.

The D-Wave processors we use in our study are designed to implement quantum annealing using a transverse-field Ising Hamiltonian:

$$H(s) = A(s)H_X + B(s)H_P, \quad (2)$$

where $s = t/t_f \in [0,1]$, $H_X = -\sum_{i\in\mathcal{V}} \sigma_i^x$, and $H_P = \sum_{i\in\mathcal{V}} h_i \sigma_i^z + \sum_{(i,j)\in\mathcal{E}} J_{ij}\sigma_i^z\sigma_j^z$ is the Ising, or "problem" Hamiltonian whose ground state we are after. The σ_i^x and σ_i^z are the Pauli matrices acting on superconducting flux qubits that occupy the vertices $\mathcal{V}$ of a Chimera hardware graph $\mathcal{G}$ with edge set $\mathcal{E}$ [43,44] and the local fields h_i and couplings J_{ij} are programmable analog parameters. The system is initialized in or near the ground state of the initial Hamiltonian $H(0)$, and the annealing schedules $A(s)$ and $B(s)$, which set the energy scale, are described in Appendix B, along with further technical and operational details, including a schematic of the Chimera graph. The DW2KQ processor comprises 16×16 unit cells, so we can consider $L \times L$ subgraphs up to $L_{\max} = 16$

for our analysis, where each subgraph comprises L^2 unit cells, and each complete unit cell comprises 8 qubits (a small number of unit cells are incomplete, as a total of 21 out 2048 qubits are inoperative).

III. RESULTS

We start by describing our key results: the evidence for optimal annealing times, and the evidence for a scaling advantage of a physical quantum annealer over SA, along with its scaling disadvantage against the SQA and SVMC algorithms (we review these algorithms and discuss how we implemented and timed them in Appendix C). We then describe in detail the construction of the class of problem instances exhibiting these properties and the role of tunneling in explaining them.

A. Evidence for optimal annealing times

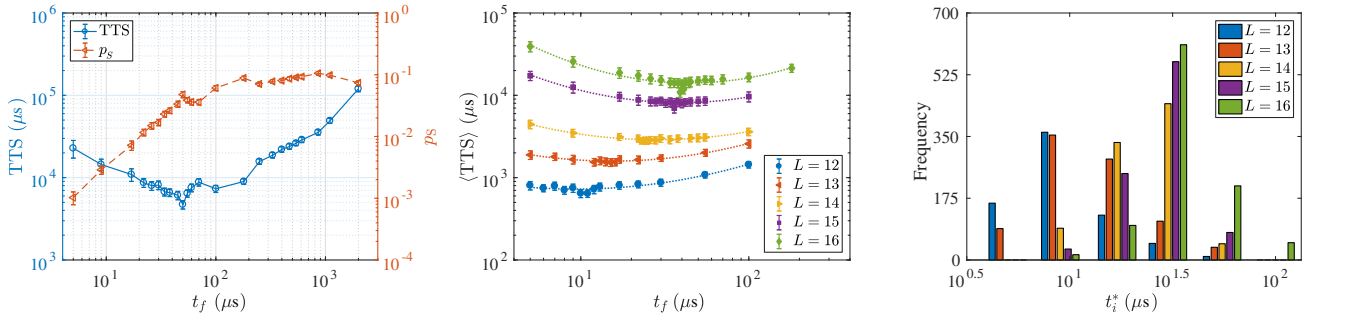
We first present the evidence for optimal annealing times in Fig. 1. Figure 1(a) shows the TTS for a single representative $L = 16$ instance from a class we call logical-planted problems. The unambiguous minimum at $t_f = 50\ \mu$s is the optimal annealing time for this instance. The presence of a minimum is a robust feature: Fig. 1(b) shows that the optimal annealing time feature persists for the median TTS ($\langle \mathrm{TTS} \rangle^*_{0.5}$), at all sizes $L \in [12, 16]$. In all previous benchmarking work, only the rise in $\langle \mathrm{TTS} \rangle$ as a function of t_f was observed, i.e., t^* was always below $t_{\min}$, thus precluding the identification of an optimal annealing time. The increase in the optimal annealing time from $L = 12$ to $L = 16$ seen in Fig. 1(b) can be attributed to the general increase with problem size of the per-instance optimal annealing time as shown in Fig. 1(c), which shows the distribution of optimal annealing times over all the logical-planted problem instances we tested.

B. Evidence for a scaling advantage for QA hardware over simulated annealing, and a disadvantage against SQA and SVMC

Having established accessible optimal annealing times ($\geq 5\ \mu$s) for the logical-planted instances, we are now ready to present a complete optimal-TTS scaling analysis. Our results for the dependence of $\langle \mathrm{TTS} \rangle^*$ on problem size are shown in Fig. 2, where we compare the DW2KQ results to three classical algorithms: SA with single-spin updates [36], SQA based on the discrete-time path-integral quantum Monte Carlo algorithm [37], and the SVMC algorithm [38]. Figure 3 summarizes the performance of each algorithm in terms of the coefficients of exponential and polynomial fits, respectively, for several quantiles and two simulation temperatures for SQA and SVMC algorithms (a hybrid polynomial-exponential fit does not work as well; see Appendix D).

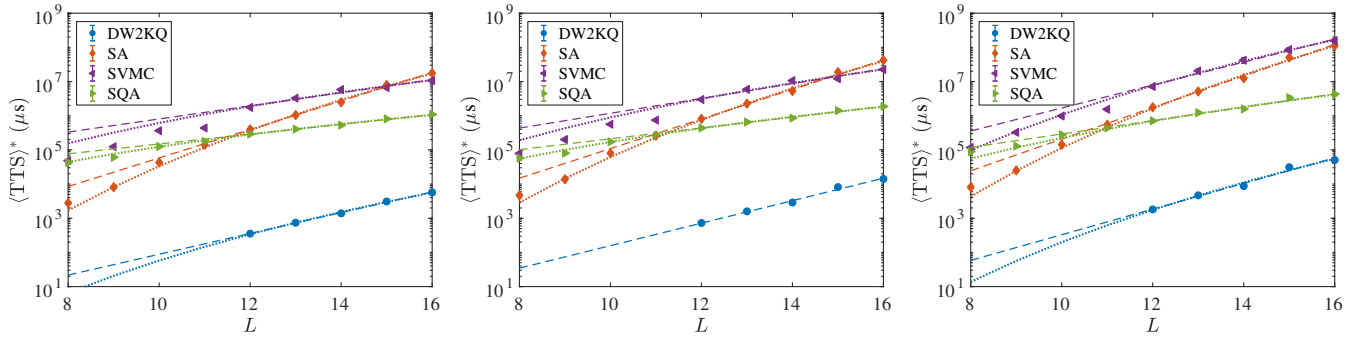
The results presented in Fig. 3 demonstrate a (95% confidence) scaling advantage for the DW2KQ over SA in the case of the logical-planted instances, for the entire range of quantiles [0.25, 0.9]. This represents the first observation of a scaling advantage over SA on an experimental quantum annealer.

However, the SQA algorithm outperforms the DW2KQ in all quantiles and at both the colder inverse temperature of $\beta = 2.5$ and the warmer $\beta = 0.51$ (which corresponds to the operating temperature of the DW2KQ). The SVMC algorithm outperforms the DW2KQ in all quantiles at the warmer inverse temperature of $\beta = 0.51$ and in all quantiles at $\beta = 2.5$ except $q = 0.9$, where the error bars are too large



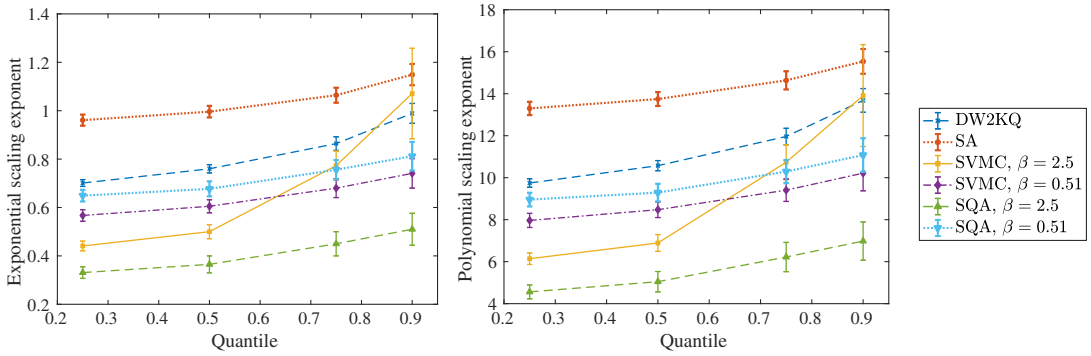
(a) TTS for representative instance (b) TTS for ensemble (c) distribution of optimal annealing times

FIG. 1. Optimal annealing time and optimal TTS. Result shown are for the "logical-planted" instance class. (a) TTS (blue solid line) and p_S (red dashed line) for a representative problem instance at size $L = 16$. A clear minimum in the TTS is visible at $t^* = 50\ \mu$s, thus demonstrating the existence of an optimal annealing time for this particular instance (but not by itself for the problem class). Note that the decreasing or increasing TTS is associated with p_S growing sufficiently fast or too slowly, respectively, with increasing t_f. (b) Median TTS as a function of annealing time for $L \geq 12$, from 1000 instances. Dotted curves represent best-fit quadratic curves to the data (see Appendix G for the scaling of t^* with L, and Appendix K for details on the fitting procedure). The position of the minimum of these curves gives t^*. The position of $\langle \mathrm{TTS} \rangle^*$ shifts to larger t_f as the system size increases. An optimum could not be established for $L < 12$ for this instance class; i.e., it appears that $t^* < 5\ \mu$s when $L < 12$. (c) The distribution of per-instance optimal annealing times t_i^* for different system sizes, as inferred directly from the positions of the minima as shown in (a). It is evident that the number of instances with higher optimal annealing times increases along with the system size, in agreement with (b).



(a) $q = 0.25$ (b) $q = 0.50$ (c) $q = 0.75$

FIG. 2. Scaling of the optimal TTS with problem size. Result shown are for the logical-planted instance class. The data points represent the DW2KQ (blue circles) and three classical solvers: SA (red diamonds), SVMC (purple left triangles), and SQA (green right triangles) algorithms. The dashed and dotted curves correspond, respectively, to exponential and polynomial best fits with parameters shown in Fig. 3 (also given in table format in Table III in the Appendix). Panels (a)–(c) correspond to the 25th quantile, median, and 75th quantile, respectively. SVMC and SQA were run with $\beta = 2.5$ here. Additional simulation parameters for SA, SVMC, and SQA algorithms are given in Appendix C. The data symbols obscure the error bars, representing the 95% confidence interval for each optimal TTS data point [computed from the fit of $\ln\langle \mathrm{TTS} \rangle$ to a quadratic function, as explained in Appendix K].



(a) fit to $a\exp(bL)$ (b) fit to aL^b

FIG. 3. Scaling coefficients for the logical-planted instances. The data shown are for the coefficient b in fits to (a) $a\exp(bL)$ and (b) aL^b for the logical-planted instances using $L \in [12, 16]$ for different quantiles and different solvers. Results are shown for the DW2KQ, SA (with a final inverse temperature of $\beta = 5$), SVMC, and SQA for two different inverse temperatures. The value $\beta = 0.51$ corresponds to the operating temperature of the DW2KQ of 15 mK.

to make a statistically significant determination. Thus, the scaling advantage over SA we observe is definitively not an unqualified quantum speedup.

We note that our results are robust to modifying the SA annealing schedule from a quadratic to a linear function in β (see Appendix E), and under a change of the metric to the so-called "quantile-of-ratios speedup" [18] (see Appendix F).

We also note that of all the solvers featured in Fig. 3, the scaling of the SVMC algorithm at $\beta = 2.5$ increases fastest from the easiest to the hardest quantile. As we discuss in more detail below, and is clear from Fig. 3, the SVMC and SQA performance depends strongly on the temperature at which the simulations are run. Specifically, we find that the SVMC algorithm performs better at higher quantiles at *higher* temperatures, whereas SQA performs better at all quantiles at *lower* temperatures. We attribute this to harder instances involving an energy barrier that the SVMC algorithm must thermally hop over, while SQA can mimic tunneling through. This also suggests that the DW2KQ performance is severely hindered by its suboptimally high temperature. To explain this, we next motivate and discuss how we constructed our problem instances.

C. Construction of problem instances with an optimal annealing time

Having presented the evidence for optimality and the scaling analysis, we next describe the instance class with these properties. The two key properties we wish our instances to possess are (1) a guarantee of knowing the ground state energy (a useful feature for benchmarking optimizers at ever-growing problem sizes) and (2) an optimal annealing time on the D-Wave processors.

1. Planted solutions

In order to guarantee a known ground state energy, we construct "planted-solution" instances. The method builds the problem Hamiltonian as a sum of frustrated loop Hamiltonians H_ℓ, such that

$$H_P = \sum_\ell H_\ell \tag{3}$$

itself is "frustration free"; i.e., the planted solution is the simultaneous ground state of all H_ℓ terms and hence is the ground state of H_P [20]. Without loss of generality, we can always pick the planted solution to be the $|0\cdots0\rangle$ (all-zero state) configuration, where henceforth the states $|0\rangle$ and $|1\rangle$ denote the eigenstates of the σ^z operator with $+1$ and -1 eigenvalues, respectively. We consider planted solutions defined on the logical graph formed by the complete unit cells of the D-Wave hardware graph (i.e., without faulty qubits or couplers; in the case of an ideal Chimera graph, this would form a square lattice) [22]. Frustrated loops are then built on this logical graph, where logical couplings between adjacent unit cells are imposed only when all four physical interunit cell couplings are available. The intraunit cell couplers are then all set to be ferromagnetic, guaranteeing that the planted solution on the hardware graph is the planted solution on the logical graph with all physical spins in the unit cell set to their corresponding logical spin value. We refer to these as logical-planted instances. In Appendix G, we introduce "hardware-planted" instances and demonstrate that they also exhibit an optimal annealing time.

2. Gadgets

In order to identify problem instances that exhibit an optimal annealing time, we first recall that previous studies of planted-solution instances on the D-Wave processors [20–22] found a TTS that rises monotonically as a function of the annealing time. Keeping Fig. 1(a) in mind, a decreasing or increasing TTS results from the success probability rising sufficiently fast or too slowly, respectively, with increasing annealing time (we formalize this in Sec. III C 4 below). In the case where the system is very weakly coupled to its thermal environment, we can expect a competition between adiabaticity (unitary dynamics) and open system effects such as thermal excitations [48,49], resulting in a peak in the success probability and a minimum in the TTS. Though we note that it is unlikely that the DW2KQ operates entirely in the weak-coupling regime (the minimum gap associated with the gadget is already below the temperature energy scale, as shown in Fig. 4, and we expect the minimum gap of the large instances to be smaller), from this perspective, shifting the minimum in the TTS to larger t_f values corresponds to prolonging the timescale over which adiabaticity dominates over open system effects. One way to try to accomplish this

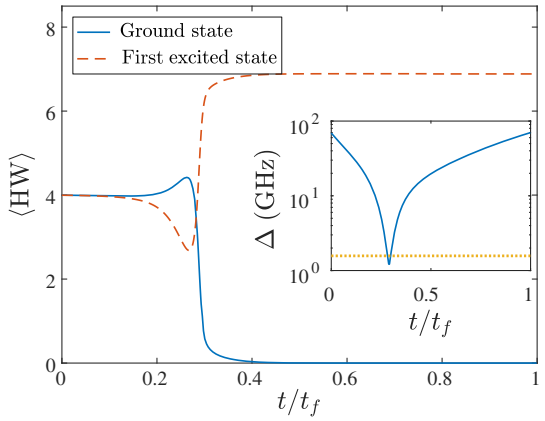


FIG. 4. Expectation values of the Hamming weight operator. Shown are the ground state and first excited state expectation values of $\mathrm{HW} = \frac{1}{2}\sum_{i=1}^{n}(1-\sigma_i^z)$ for the 8-qubit gadget using the DW2KQ annealing schedule. Inset: The ground state energy gap to the first excited state, as calculated using the DW2KQ annealing schedule. The dotted line corresponds to the operating temperature of the device.

is by enhancing the role of finite-range tunneling in the dynamics. Motivated by this insight, and by recent work on the possibility of a computational role of finite multiqubit tunneling in quantum annealers [25,41], we introduce a key modification and supplement the planted-solution instance Hamiltonian [Eq. (3)] with terms corresponding to the addition of identical 8-qubit gadgets that exhibit tunneling during their anneal:

$$H'_P = H_P + \sum_{i\in\mathcal{S}} H_{G_i}. \tag{4}$$

Here, H_{G_i} denotes the gadget Hamiltonian in unit cell i, and the gadgets are placed into randomly chosen unit cells: $\mathcal{S}$ denotes a randomly chosen subset comprising a fraction p of complete unit cells (we use $p = 0.1$). The specific 8-qubit gadget we used fits into the unit cell of the D-Wave processors, and its connectivity and parameters are depicted in Fig. 5. The ground state of the gadget is the all-zero state

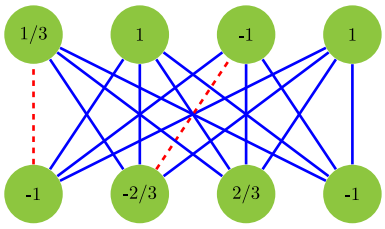


FIG. 5. The 8-qubit gadget used in the instance construction. The qubits (green circles) are arranged in a complete bipartite graph. Blue (red) lines correspond to ferromagnetic (antiferromagnetic) Ising couplers with magnitude 1. The value of the local fields on the qubits are given inside the circles, with a negative value indicating a spin-up preference.

of the eight qubits, so that the ground state of the full Hamiltonian remains the all-zero state. The first excited state of the gadget is doubly degenerate with average Hamming weight seven.

Generically, one would not expect the annealing properties of H_G to be shared by H'_P, but below we show to what extent they are for our instances.

3. Tunneling

We next establish in what sense our gadget exhibits tunneling. We show in Fig. 4 the expectation value of the Hamming weight operator $\mathrm{HW} = \frac{1}{2}\sum_{i=1}^{n}(1-\sigma_i^z)$ in the ground state and first excited state, computed by numerically solving the time-dependent Schrödinger equation for the evolution of the gadget. This expectation value exhibits a sharp change at the same point in the evolution where the minimum gap occurs (shown in the inset). The ground state reorients itself to the $|0\cdots0\rangle$ state, while the first excited state reorients to align closely with the $|1\cdots1\rangle$ state. This already suggests a tunneling transition, but in order to confirm this we wish to establish the presence of an energy barrier in the semiclassical potential that the quantum system must tunnel through during the anneal. Such tunneling transitions have been well studied in the context of systems with qubit-permutation invariance [41,42,50–52], but less so in the context of systems such as ours without this symmetry.

The semiclassical potential as derived from the spin-coherent path-integral formalism [53] is given by the expectation value of $H(t)$ in the spin-coherent state $|\Omega(\vec{\theta},\vec{\varphi})\rangle = \otimes_{i=1}^{n}[\cos(\theta_i/2)|0\rangle_i + e^{i\varphi_i}\sin(\theta_i/2)|1\rangle_i]$. In the context of the transverse-field Ising Hamiltonian [Eq. (2)], the semiclassical potential becomes

$$V(\vec{\theta},\vec{\varphi},t) = -A(t)\sum_i \sin(\theta_i)\cos(\varphi_i) + B(t)\left(\sum_{i\in\mathcal{V}} h_i\cos\theta_i + \sum_{(i,j)\in\mathcal{E}} J_{ij}\cos(\theta_i)\cos(\theta_j)\right). \tag{5}$$

Equation (5) provides a multidimensional energy landscape for the quantum annealing protocol. Unfortunately, due to the absence of any symmetries, it is infeasible to exhaustively explore this landscape and identify the actual location of barriers, even under the simplification where $\varphi_i = 0,\ \forall\ i$. Instead, as proxies for a direct calculation of tunneling transition matrix elements or an instanton analysis [54], we consider the behavior of the SVMC and SQA algorithms. The SVMC algorithm performs Metropolis updates on the potential energy landscape, Eq. (5) [38]. Since this algorithm can only thermally "hop" over energy barriers, we expect its performance to deteriorate with decreasing temperature in the presence of a relevant energy barrier. On the other hand, a path-integral Monte Carlo based approach like SQA should be able to not only thermally hop over these barriers but also mimic tunneling through them [39,40,55], which should benefit from a decreasing temperature. Therefore, we expect to be able to identify tunneling energy barrier bottlenecks in the quantum anneal by contrasting the temperature dependence of the performance of SVMC and SQA.

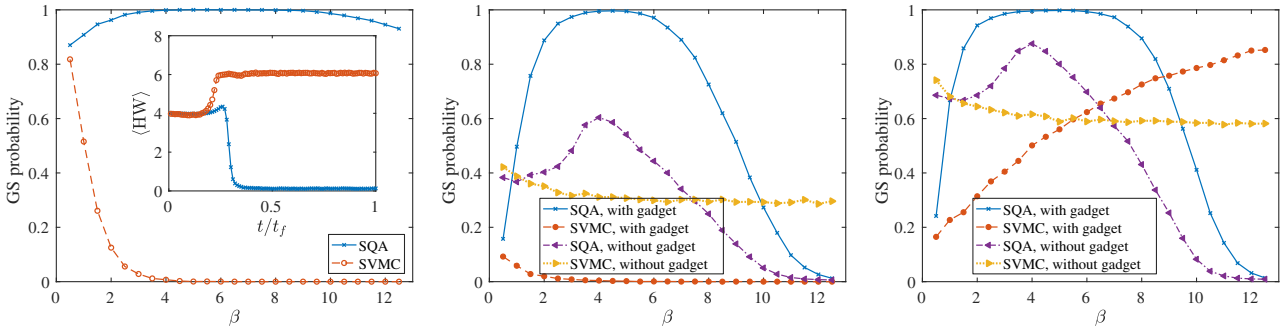
Figure 6(a) shows that for our 8-qubit gadget, SQA and SVMC algorithms behave as expected in the presence of a tunneling energy barrier: the success probability of the SVMC algorithm decreases with decreasing temperature, whereas the success probability of SQA increases with decreasing temperature. As shown in the inset and comparing to Fig. 4, we see that the SVMC algorithm is effectively trapped in the higher excited states, while SQA is able to follow the ground state.

Next, we use the same technique to probe our planted-solution instances with and without the gadget. We show the behavior of two very different $L = 16$ instances in Fig. 6. For one of the instances [Fig. 6(b)], the introduction of the gadget adversely affects the performance of the SVMC algorithm, pushing the success probability to zero for increasing inverse temperature β. For SQA, the improvement in performance as β increases is significantly sharper with the gadget. For the second instance [Fig. 6(c)], we see that while SQA's behavior is almost identical, the SVMC algorithm exhibits an improving performance with increasing β in the presence of the gadget, suggesting an absence of an energy barrier. This analysis demonstrates that the tunneling properties induced by our 8-qubit gadget can be inherited by the problem instances even at the largest problem size, and that the success probability exhibits a strong temperature dependence. For further details on the behavior of an ensemble of instances see Appendix G.

Unfortunately, we cannot directly probe tunneling or study the temperature dependence on the D-Wave processors. To the extent that SQA models the behavior of the physical quantum annealer, one may choose to interpret the evidence we have presented above as evidence for the role of tunneling energy barriers induced by the gadgets.

4. The gadget is responsible for the observed optimal annealing time

To more directly understand the effect of our gadget on the hardware quantum annealer, it is instructive to contrast the scaling behavior with and without the gadget. Toward that end, we fit the empirical success probability p_S to a power law of the form $b(t_f)^a$ (see Appendix H for the fit quality). We show in Fig. 7 the distribution of the scaling coefficient a for 100 instances for the logical-planted instances with and without the gadget. The two distributions differ substantially: the instances with the gadget exhibit larger coefficients, almost all with a value greater than 1, resulting in a significantly larger initial rate of



(a) gadget only (b) instance exhibiting tunneling (c) instance not exhibiting tunneling

FIG. 6. Probability of reaching the ground state (GS) at the end of the anneal using SVMC and SQA for different simulation inverse temperatures β. (a) Simulation results for the 8-qubit gadget only. SQA's success probability increases over a wide range of decreasing temperatures, whereas the SVMC algorithm rapidly deteriorates as the temperature decreases. Inset: Expectation values of the Hamming weight operator $\mathrm{HW} = \frac{1}{2}\sum_{i=1}^{n}(1-\sigma_i^z)$ for the 8-qubit gadget for SVMC and SQA using $\beta = 2.5$. Compare to the behavior of the ground state and first excited state shown in Fig. 4. For the SVMC algorithm, to compute the expectation value of the Hamming weight operator at intermediate s values, we can either project the state to the computational basis (shown) or use the spin-coherent state; the results are almost indistinguishable. For SQA, we average over the Hamming weight of the configurations in the imaginary-time direction. (b),(c) Probability of reaching the ground state at the end of the anneal using SVMC and SQA for different simulation inverse temperatures β using two different instances, with and without the 8-qubit gadget, at the largest available size $L = 16$. (b) An instance that exhibits a clear signature of a tunneling energy barrier when the gadget is introduced. (c) An instance that does not exhibit a signature for a tunneling energy barrier even with the gadget. In all panels, SVMC and SQA simulations used 8 M and 3 M sweeps, respectively, and both algorithms use the DW2KQ annealing schedule. The drop in success probability at large β for SQA is because spin updates become less efficient at high β and more spin updates are required to maintain the high success probability.

increase in $p_S(t_f)$ than for the instances without the gadget. This, in turn, leads to the observed initial decrease in the TTS with increasing annealing time: upon expanding the logarithm in Eq. (1) for small p_S, we find that $\mathrm{TTS}(t_f) \propto t_f/p_S(t_f) = (t_f)^{1-a}$, so that $\mathrm{TTS}(t_f)$ decreases with t_f provided $a > 1$; this is consistent with Fig. 1(a), where the slope of $p_S(t_f)$ is indeed seen to be initially > 1, then dropping to < 1 for larger t_f. Since the TTS must eventually increase with the annealing time [ideally, for sufficiently large t_f, $R(t_f) \to 1$, at which point $\mathrm{TTS}(t_f) \propto t_f$], this helps to explain why the instances with the gadget exhibit an optimal annealing time. It also suggests a useful heuristic for future studies attempting to identify problem instance classes with an optimal annealing time: the instances should have the property that if $p_S(t_f) = g(t_f) \ll 1$ for some function g, then $\mathrm{TTS}(t_f) \propto t_f/g(t_f)$ must be decreasing for some range of $t_f > t_{\min}$ values. This is compatible with any faster-than-linear form for g. We already alluded to a competition between adiabaticity and thermal excitations as being potentially responsible for an optimal annealing time. Another mechanism, that appears to be more consistent with the fact that the DW2KQ is not operating in the weak-coupling regime, is that thermal relaxation is fast for small t_f and then slows down for sufficiently large t_f, presumably since the system has already entered the quasistatic regime [56].

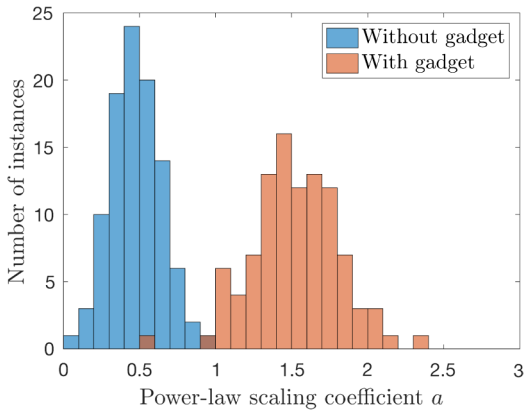


FIG. 7. Empirical scaling behavior with and without the gadget. Shown is the distribution of the power-law scaling coefficient a obtained after fitting $\ln p_S$ to $a \ln t_f + b$ for 100 instances at $L = 16$, run on the DW2KQ processor. We choose $t_f \in [5, 50]$ μs since this is the range over which the TTS decreases, as seen in Fig. 1(a). The instances with the gadget typically exhibit a larger scaling coefficient, which leads to the observation of an optimal annealing time. In (a) and (b) error bars represent 95% confidence intervals (2σ) calculated using 1000 bootstraps of 100 gauge transformations [17].

IV. DISCUSSION AND OUTLOOK

The key result of this work is the demonstration of an algorithmic scaling advantage for QA hardware over the SA algorithm for a family of problem instances constructed with frustrated loops and a small gadget that exhibits tunneling. It is worth emphasizing why the combinatorial optimization and quantum annealing communities have

often focused on suboptimal heuristics such as SA (see, e.g., Ref. [41]). SA is not only a very general metaheuristic, but it is also often viewed as the inspiration for QA, with thermal fluctuations replaced by quantum fluctuations [15]. Because of this correspondence between SA and QA, a demonstration of superior performance by QA can presumably be attributed to an advantage of the quantum approach over the thermal approach. The goal is then to leverage this advantage to a broader range of problems. However, as we have argued, temperature annealing as in SA is actually quite different from transverse-field annealing, so that the analogy between SA and QA needs to be treated with care. Another concern we face is that while the accuracy threshold theorem provides a theoretical guarantee that for sufficiently low noise levels and through the use of quantum error correction a finite-size device can be scaled up fault tolerantly [57], in the absence of such an asymptotic guarantee for quantum annealing a finite-size device provides evidence of what can be expected at larger, future sizes, only provided the device temperature, coupling to the environment, and calibration and accuracy errors, can be appropriately scaled down.

In light of this, what is the significance of our demonstration of a QA scaling advantage over SA? We believe that an important clue lies in the fact that the SVMC algorithm also exhibits an advantage over SA for these problems. The SA and SVMC algorithms can both be viewed not only as classical analogues of QA, but also as implementing two of its possible classical limits [15,38,58]. While SA performs updates on the classical energy landscape associated with the Ising Hamiltonian, the SVMC algorithm performs updates on the semiclassical potential associated with the quantum anneal. A scaling difference between the two, with an advantage for the SVMC algorithm, suggests that thermal updates on the semiclassical energy landscape is more efficient. While it is unclear whether the quantum effects in the D-Wave devices that have already been demonstrated on a smaller scale ($N \lesssim 16$) [17,25,41,59–63] remain operative at the much larger scales we have employed in our study, the fact that the DW2KQ also exhibits an advantage over SA suggests that it must be evolving in a landscape that also allows for better scaling. To be specific, this is the landscape associated with transverse-field annealing as opposed to temperature annealing. It is in this sense that quantum effects that are necessarily absent from SA and *might* be present in the quantum annealer can provide an advantage. This is especially significant since there is a large overlap between the instances solved at the median quantile by the DW2KQ and all three of the classical algorithms, including SA, as shown in Fig. 8. This means that if any quantum effects are responsible for the scaling advantage of the DW2KQ over the SA algorithm, then they are operative in largely the same set of problem instances, so that these instances may define a target class for quantum enhanced optimization.

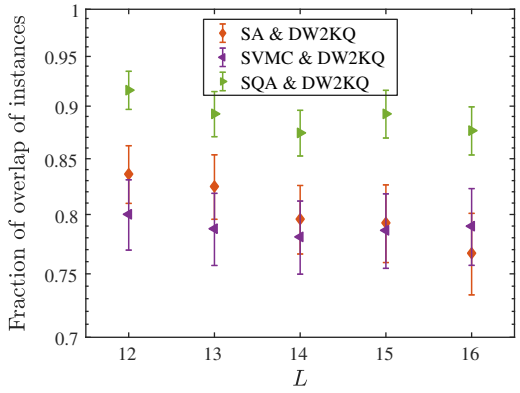


FIG. 8. Overlap of the instances that fall below the median TTS for the classical solvers and the DW2KQ. We calculate the TTS for 1000 instances with each solver's respective optimal annealing time for the median at a given size L, and check which instances fall below the median TTS. Shown is the (normalized) fraction of the overlap of the instances between the solvers. Further details are given in Appendix I. Error bars represent 95% confidence intervals (2σ) calculated using 1000 bootstraps of 1000 instances.

Likewise, SQA also evolves on the same semiclassical energy landscape as the SVMC algorithm, but in addition to thermal updates is also capable of mimicking tunneling. The fact that SQA's scaling is far superior to that of SVMC's, and that this improves as the simulation temperature is lowered, shows that tunneling is effective at enhancing SQA's performance for the logical-planted instances. What does this tell us about the possibility that the relative performance of the algorithms is indicative of quantum effects in the DW2KQ device? It is known that the scaling of the quantum Monte Carlo algorithm can be as efficient [39,40,64] or less efficient [55] than the incoherent tunneling rate scaling of a true quantum annealer. Therefore, the fact that SQA overwhelmingly outperforms the DW2KQ but that the DW2KQ still outperforms SA suggests that the device is dominated by classical dynamics with a very small quantum component. While only speculative at this point, this type of situation might be the best we can hope for in the current generation of highly noisy quantum annealers, without some form of quantum error correction or suppression [65–69].

Figure 3 shows how the scaling of both SVMC and SQA is strongly affected when we increase their temperatures to the DW2KQ dilution fridge temperature. In both cases, for the median and lower percentile, the performance of the SQA and SVMC algorithms is hurt at this higher temperature relative to the colder temperature. This strongly suggests that the DW2KQ's performance for this class of instances is severely impacted by its temperature, consistent with general expectations [46]. However, we also find that the SVMC algorithm scales better than the DW2KQ.

One would expect that if the SVMC algorithm is the classical limit of the device, then the DW2KQ should perform at least as well as the SVMC algorithm. One possible explanation for the violation of this expectation is that additional noise sources, such as implementation errors, further degrade the performance of the DW2KQ relative to solvers run on digital classical computers [21].

We note that the performance of the SVMC algorithm improves at higher percentiles at higher temperatures relative to colder temperatures, which is consistent with the algorithm being able to thermally hop over energy barriers. This indicates that temperature is another algorithmic parameter that should be optimized separately for each quantile, a point we leave for future studies.

We emphasize that the *discrete-time* SQA algorithm studied here should not be interpreted as a true model of a thermal quantum system. It has been demonstrated that time discretization may result in improved residual energy minimization performance over the continuum case [70], although this does not necessarily translate into a scaling performance advantage. Nevertheless, the superior performance of the SQA algorithm we have observed is an interesting finding in its own right: we are unaware of another example of an Ising model cost function where SQA with closed boundary conditions bests SA as a ground state solver (Ref. [25] reports an example but uses SQA with open boundary conditions). We have attributed this advantage to a more favorable energy landscape with the presence of tunneling barriers than can be traversed efficiently, but to test whether the observed scaling advantage would hold for a thermalizing quantum annealer requires quantum Monte Carlo simulations without Trotter errors [71–73]; unfortunately, at the >2000 qubits scale we have worked with here, this is computationally prohibitive. The same is true for master equation simulations [41,74], even when implemented using the quantum trajectories method [75].

We emphasize that the instances presented here are not necessarily computationally hard, as suggested by the fact that, considering the entire range of sizes we tested, the quality of the polynomial fits is better than that of the exponential fits (see Fig. 2). In the absence of the gadget, the logical-planted instances are defined on a square lattice and can be solved in polynomial time using the exact minimum-weight perfect-matching (MWPM) algorithm [76]. However, we have confirmed that this algorithm performs poorly once the gadget is included, as expected when local fields are present (see Appendix J). Therefore, it is natural that algorithms optimized with respect to the problem structure demonstrate superior performance. For example, simulated annealing with both single-spin and multispin updates (SAC), with the latter being simultaneous updates of all the spins comprising a unit cell (superspin approximation [23]), scales significantly better than SA with single-spin updates but still does not perform as well as SQA (see Appendix J). Furthermore, there are many other classical algorithms that do not implement the same algorithmic approach as quantum annealing, such as the Hamze-Freitas-Selby (HFS) [77,78] algorithm. The latter exploits the low tree width of the Chimera connectivity graph and has in all studies to date been the top performer for Chimera-type instances. In contrast, here we find that the scaling performance of the HFS algorithm lies between the DW2KQ and SAC (see Appendix J). Another competitive algorithm is parallel tempering with isoenergetic cluster moves [79,80]. We can expect that a more highly connected hardware graph will prevent algorithms such as HFS or SAC from being efficient; which architectures may lend themselves to an unqualified quantum speedup remains an open research question.

In this work we focused on the task of finding any ground state, and did not address the question of how well quantum annealing can uniformly sample the ground states, commonly referred to as ground state "fair sampling" and a problem that belongs to the complexity class #P. It is well established that quantum annealing with the standard transverse-field driver Hamiltonian samples the ground states in a biased manner [59,81–83], and our work does not establish to what extent this bias exists for the class of instances we study, nor whether the different algorithms studied exhibit a different bias. Addressing this question provides another approach for searching for a quantum advantage beyond the standard optimization approach [60,82].

Meanwhile, our hybrid frustration-tunneling-based instance construction approach defines a clear path forward by concretely establishing the possibility of generating instance classes with accessible optimal annealing times, amenable to a complete scaling analysis. By "mining" those instances which exhibit the largest separation between SQA and the top performing alternative classical algorithms, while corroborating that the performance of hardware-based QA is also competitive on the same instances (at a minimum it should certainly continue to beat SA), we expect to be able to identify the features that give rise to a quantum advantage and learn how to amplify the difference. This procedure can be iterated, all the while ensuring that optimal annealing times can be ascertained, in order to amplify the separation. Current QA hardware may simply be too hot and incoherent to exhibit an amplification leading to an unqualified scaling advantage over all classical algorithms for the resulting instance class. This is especially true for instances amenable to SQA simulations that can reproduce the incoherent tunneling rates of a noisy quantum annealer [39,40,64], in which case more coherent devices will be necessary. Nevertheless, the principles we establish here will at the very least provide a means of testing this exciting possibility.

ACKNOWLEDGMENTS

We thank D-Wave Systems Inc. for providing access to the DW2KQ device in Burnaby, Canada. T. A. thanks James and Andrew King for useful discussions on the GPU implementations of the classical algorithms. This work was supported under ARO Grant No. W911NF-12-1-0523, ARO MURI Grants No. W911NF-11-1-0268 and No. W911NF-15-1-0582, and NSF Grant No. INSPIRE-1551064. Computation for the work described in this paper was supported by the University of Southern California's Center for High-Performance Computing and by the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract No. DE-AC05-00OR22725. The research is based upon work partially supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the U.S. Army Research Office Contract No. W911NF-17-C-0050. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

APPENDIX A: TIME TO SOLUTION AND OPTIMALITY

We provide a derivation of the TTS expression given in Eq. (1) (see also, e.g., the Supplemental Material of Ref. [18]). Let us assume that the probability of observing the ground state energy in any given repetition is $p_S(t_f)$, and we ask how many repetitions R must be performed to observe the ground state energy at least once. The probability of not observing the ground state energy once in R trials is $[1-p_S(t_f)]^R$. Therefore, to observe the ground state energy at least once with probability p_d is

$$1-p_d=[1-p_S(t_f)]^R. \tag{A1}$$

Solving for R gives the expression in Eq. (1) in the main text. Technically, the number of repetitions is defined as $R(t_f)=\lceil[\ln(1-p_d)]/\{\ln[1-p_S(t_f)]\}\rceil$, but we do not include the ceiling operation in the calculation of $R(t_f)$ in this work, since this can result in sharp TTS changes that complicate the extraction of a scaling. Similarly, the ratio $N/N_{\max}$ should be $(\lfloor N_{\max}/N\rfloor)^{-1}$. The TTS should in principle also include all time costs accrued by running the algorithm multiple times, such as state initialization and state readout times, as well as multiple programming times if different gauges are used (see Appendix B). We do not include these here either, because at least on the D-Wave processors, the readout and programming times are several factors larger than the annealing time and hence can effectively mask the scaling behavior. Instead, we restrict t_f to be the run time between state preparation and readout for all our algorithms.

To see how an analysis of the TTS that does not account for optimal annealing times can lead one astray, consider the following extreme example: suppose t_f is too large at all problem sizes $N\in[N_{\min},N_{\max}]$, such that $R(t_f)=1$ always suffices to find the global minimum; in this case, $\langle \mathrm{TTS}(t_f)\rangle \propto Nt_f$ for all N (i.e., is constant except for the parallelization factor N), which, except for trivial problems, must obviously be false.

APPENDIX B: D-WAVE QUANTUM ANNEALERS

We used the D-Wave 2000Q (DW2KQ) processor housed at Burnaby that features 2023 functional qubits and 5874 programmable couplers. We also used the DW2X processor housed at USC/ISI that features 1098 functional qubits and 3049 programmable couplers. The minimum annealing times for all D-Wave processors involved in benchmarking studies to date are 5 μs for the D-Wave One, D-Wave Two X, and D-Wave 2000Q, and 20 μs for the D-Wave Two. Additional details about the processors are provided below. For each instance, we ran 100 random gauges (also known as spin-reversal transforms). A gauge is the application of a particular bit-flip transformation to the σ^z operators in the problem Hamiltonian; i.e., $H'_P \mapsto \prod_{i=1}^N(\sigma_i^x)^{s_i}H'_P\prod_{i=1}^N(\sigma_i^x)^{s_i}$, where each $s_i\in\{0,1\}$. This transformation does not change the eigenvalues of the transverse-field Hamiltonian, and it is meant to minimize the effect of local biases and precision errors on the device [59]. For each gauge we took $n_{\mathrm{reads}}=1000$ readouts, unless constrained by $t_f n_{\mathrm{reads}}<10^6\mu$s. For example, for $t_f=2$ ms, we only took 400 readouts per gauge.

The annealing schedules of the D-Wave 2000Q (DW2KQ) processor housed at Burnaby and the DW2X processor housed at USC/ISI devices are shown in Fig. 9, in units of GHz. These schedules are not measured but computed and reported by D-Wave Systems Inc. based

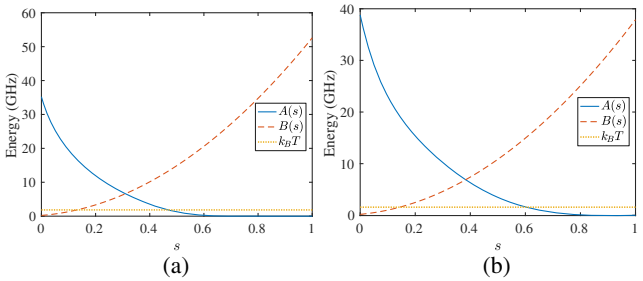


FIG. 9. Annealing schedules for (a) the DW2KQ and (b) the DW2X. The units are such that $\hbar=1$. As a reference, we include the operating temperatures of the devices, corresponding to 14.1 mK for the DW2KQ and 12.5 mK for the DW2X. Note the different vertical axis scales.

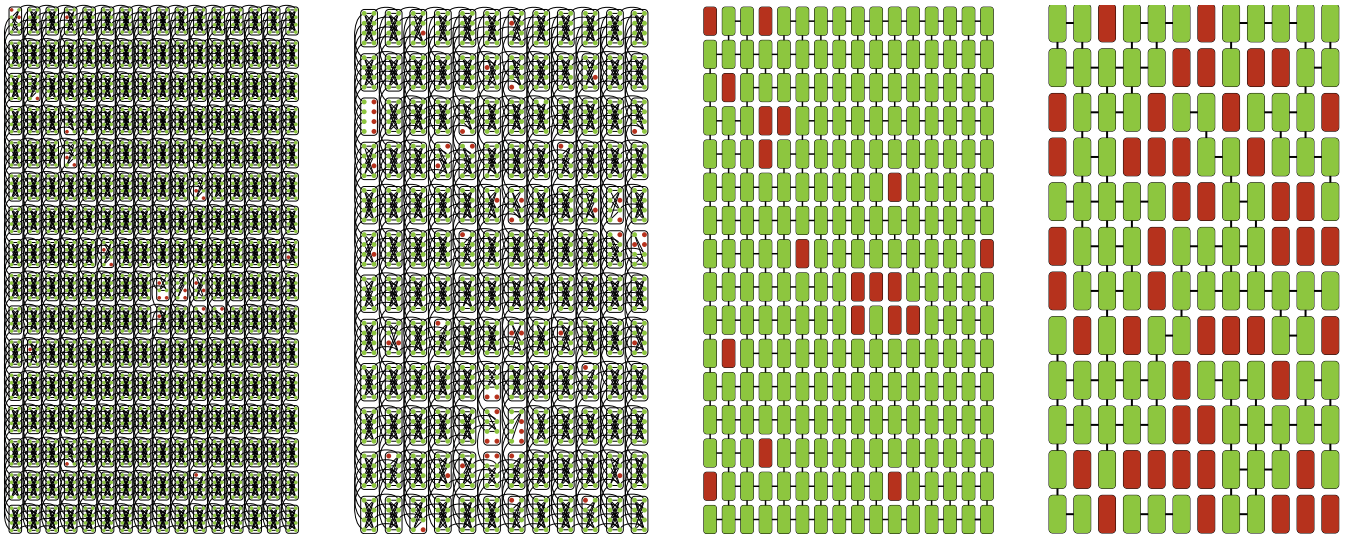


FIG. 10. Hardware and logical graphs for instance generation. (a) DW2KQ hardware graph, (b) DW2X hardware graph, (c) DW2KQ logical graph, (d) DW2X logical graph. For DW2KQ (DW2X) subgraphs of size $L \leq 16$ ($L \leq 12$) were chosen starting from the lower right-hand corner. (a),(b) Available qubits are shown in green, and unavailable qubits are shown in red. Programmable couplers are shown as black lines connecting qubits. (c),(d) Complete unit cells are shown in green, and incomplete ones are shown in red. Logical couplers are shown as black lines between the unit cells. The unit cells are numbered from 0, starting from the top left-hand corner and moving across rows to the right.

on their flux qubit models. The Chimera hardware graphs of the DW2KQ and DW2X processors we used in this work are shown in Fig. 10.

In the main text, we focused on the annealing time. There are several other relevant timescales that we present here for completeness. We used the default initial state preparation time (t_{initial}). The readout time for the DW2KQ is $t_{\text{readout}} = 124.98\ \mu$s. A complete characterization of the required run time (the "wall-clock time") would include the thermalization and readout times in each independent run of the quantum annealer. Furthermore, since we program the same instance multiple times using different gauges, the programming time of $t_{\text{program}} = 6987.80\ \mu$s needs to be accounted for. In total, the wall-clock TTS would be given by

$$\text{TTS}_{\text{wall clock}} = G t_{\text{program}} + (t_f + t_{\text{initial}} + t_{\text{readout}}) \frac{R}{\lfloor \frac{N_{\max}}{N} \rfloor}, \tag{B1}$$

where G is the number of gauges, and R is the total number of runs, divided equally among the G gauges. However, since these timescales can be much larger than the optimal annealing time, they can mask the scaling of the TTS, and hence we focus just on the annealing time, as in previous work [17,18]. In principle, the initial state preparation time can be reduced and optimized along with the annealing time if included as part of the TTS, but we have not explored in this work how this impacts performance.

APPENDIX C: SIMULATION PARAMETERS AND TIMING

Our implementation of the SA, SQA, and SVMC algorithms is based on the graphics processing unit (GPU) implementation used in Ref. [22] and described in more detail in Ref. [84]. We briefly describe our CUDA implementation of these algorithms here for completeness. In what follows, a sweep is a single Monte Carlo update of all the spins. For all implementations, we use the default cuRAND random number generator (XORWOW). We compile the CUDA code using the "-use_fast_math" flag, which, we note, may not be suitable for Monte Carlo simulations that require accurate calculations of thermal expectation values.

We first discuss our implementation of SA [36]. Each GPU thread updates the eight spins in a single unit cell. Because the Chimera graph is bipartite, each thread updates the four spins in one partition followed by the four spins in the second partition. A key feature of the implementation is that the eight local fields, 16 intercell couplers, and 16 intracell couplers are stored in the memory registers of the GPU. Only the spin configuration is stored on local memory. This minimizes the cost of retrieving data from global memory. We use the GPU intrinsic math function for the calculation of the Metropolis acceptance probability in order to maximize execution speed. As many copies n_{copies} as allowed by register memory are run in parallel in separate GPU blocks. Therefore, we have for the timing of SA

$$\text{TTS} = \tau_{\text{sweep}} n_{\text{sweep}} \frac{R}{n_{\text{copies}}}, \tag{C1}$$

where n_{sweep} is the number of sweeps and τ_{sweep} is the time required to perform a single sweep. Because n_{copies} depends on the total number of threads (L^2) and hence on the problem size, this can be equivalently written as

$$\text{TTS} = 8L^2 n_{\text{sweep}} R / f_{\text{SA}}, \tag{C2}$$

where f_{SA} is the number of total spin updates per unit time performed by the GPU. For consistency we use the timings reported in Ref. [84] for runs performed on an NVIDIA GTX 980, which have $f_{\text{SA}} = 50\ \text{ns}^{-1}$. For SA, we use a temperature annealing schedule that is the DW2X annealing schedule for $B(s)$ (shown in Appendix B) times $\beta = 0.132$ (in units where the maximum Ising coupling strength $|J_{ij}|$ is 1), such that $\beta B(1) \approx 5$.

The implementation of the SVMC algorithm follows the same structure as SA, except that the spin configuration is replaced by angles $\{\theta_i\} \in (0, 2\pi]$ [38]. The energy potential along the anneal is given by

$$V(s) = -A(s)\sum_i \sin\theta_i + B(s)\sum_{i<j} J_{ij}\cos\theta_i\cos\theta_j, \tag{C3}$$

a special case of Eq. (5) in the main text. An update involves drawing a random angle $\in (0, 2\pi]$, and it is accepted according to the Metropolis-Hastings rule [85,86] with $\beta = 2.5$ for the logical-planted instances and $\beta = 0.51$ for the hardware-planted instances (in units where the maximum Ising coupling strength $|J_{ij}|$ is 1). We use the GPU intrinsic math function for the calculation of the cosine, sine, and Metropolis acceptance probability in order to maximize the speed of the algorithm. The timing of the SVMC algorithm is the same as in Eq. (C2) but with $f_{\text{SVMC}} = 29\ \text{ns}^{-1}$ replacing f_{SA}. We use the DW2X annealing schedule for $A(s)$ and $B(s)$ shown in Appendix B; this schedule keeps $A(s) > 0$ longer than that of the DW2KQ, which favors the SVMC algorithm, since once $A(s) = 0$ the system becomes the classical Ising model and the most efficient updates use $\theta = 0, \pi$, but SVMC chooses angles randomly.

The implementation of SQA follows the same structure as SA, and also uses the DW2X schedule for similar reasons as just mentioned for the SVMC algorithm. We restrict the Trotter slicing to 64 in order to fit the spins along the imaginary-time direction into a 64-bit word. A sweep involves performing a single Wolff cluster update [87] along the imaginary-time direction for each spin. Once a cluster of spins is picked, it is flipped according to the Metropolis-Hastings rule using the Ising energy of the cluster with $\beta = 2.5$ for the logical-planted instances and $\beta = 4.25$ for the hardware-planted instances. We use the GPU intrinsic math function for the calculation of the Metropolis acceptance probability in order to maximize execution speed. At the end of the anneal, one of the 64 slices is picked randomly as the final classical state. The timing of SQA is the same as in Eq. (C2) but with $f_{\text{SQA}} = 5\ \text{ns}^{-1}$ replacing f_{SA}.

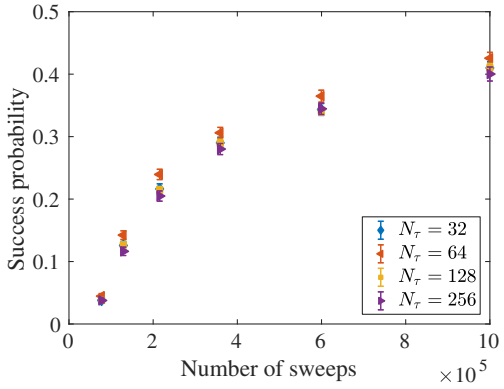


FIG. 11. Performance of SQA as we vary the number of Trotter slices. Shown are CPU simulation results for the success probability for a single logical-planted instance, using 32, 64, 128, and 256 Trotter slices. The success probability is maximized for 64 slices. Error bars give the 95% confidence interval generated by performing 1000 bootstraps.

We note that increasing the number of Trotter slices, while decreasing the Trotter error, appears to reduce the final success probability for one of the instances we have checked (see Fig. 11, where the peak success probability occurs for 64 slices), an effect noted in Ref. [70]. Studying this effect over the entire set of instances is computationally prohibitive at our > 2000 qubits scale.

APPENDIX D: ALTERNATIVE FITS

In Fig. 2 of the main text (see also Figs. 17 and 20 below), we present exponential and polynomial fits to the optimal TTS as a function of L. Here we show that a hybrid three-parameter fit, i.e., $\ln \text{TTS} = a + b\ln L + cL$, does not give reasonable fits with good confidence bounds for all solvers. We restrict our attention to the hardware-planted instances, since in that case we have 9 sizes for the fit. Table I gives the results of the fits for the median; we see that the estimate for the exponential scaling coefficient c of the classical solvers is especially poor, likely due to an insufficient number of data points for a three-parameter fit.

APPENDIX E: SA WITH A LINEAR SCHEDULE

We used the DW2X annealing schedule in Fig. 9(b) for the SQA, SVMC, and SA simulations. Further optimization

TABLE I. The coefficient (a, b, c) in fits of $\ln\langle\text{TTS}\rangle^*$ to $a + b\ln L + cL$ for the hardware-planted instances using $L \in [8, 16]$. Errors are 95% confidence intervals.

Solver	a	b	c
DW2KQ	-6.953 ± 1.442	5.017 ± 1.020	0.380 ± 0.090
SA	0.493 ± 1.102	9.521 ± 0.764	-0.193 ± 0.065
SQA	15.893 ± 5.396	0.248 ± 3.933	0.508 ± 0.360
SVMC	3.050 ± 1.814	9.316 ± 1.241	-0.075 ± 0.104

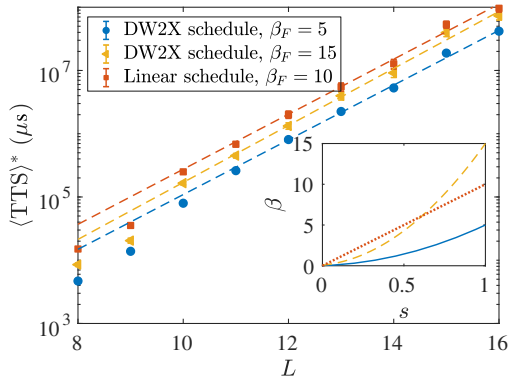


FIG. 12. Median scaling for SA on the logical-planted instances with three different annealing schedules. We compare the median TTS for SA using three different annealing schedules, $0.132B(s)$, where $B(s)$ is from the DW2X annealing schedule in Fig. 9(b), $0.396B(s)$, and a linear schedule. The dashed lines correspond to the exponential fits $\exp(a + bL)$ with $a = 12.457 \pm 0.332$, $b = 0.996 \pm 0.24$ and $a = 12.489 \pm 0.888$, $b = 1.037 \pm 0.064$ for the DW2X schedules and $a = 13.321 \pm 0.934$, $b = 1.002 \pm 0.066$ for the linear schedule. Inset: The annealing schedules in the inverse temperature β as a function of the dimensionless parameter s.

of this schedule is likely to improve the overall performance of the algorithms, although it is not evident whether it will substantially change their scaling with problem size. As an example, we provide results for the median TTS for SA using the DW2X schedule with a different overall temperature $\beta = 0.396$ and a linear schedule in Fig. 12, where we observe that the different schedules only shift the TTS curve but do not change the scaling within the statistical error bars. This indicates that our SA scaling results are robust to minor modifications of the schedule.

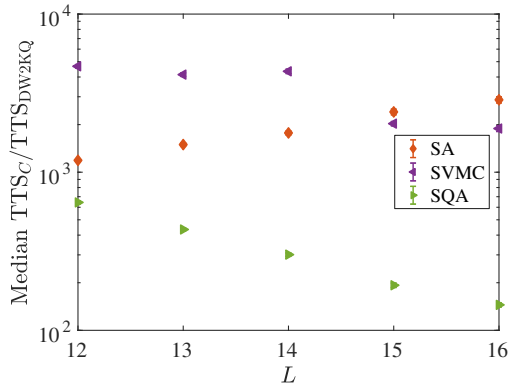


FIG. 13. Median quantile of ratios for the logical-planted instances. We show the ratio of the different classical solvers $C = \{\mathrm{SA}, \mathrm{SQA}, \mathrm{SVMC}\}$ to the DW2KQ. A positive slope, as for SA and SVMC algorithms, indicates a scaling advantage for the DW2KQ, while a negative slope, as for SQA, indicates a slowdown. The data symbols obscure the error bars representing the 95% confidence intervals (2σ) calculated using 1000 bootstraps of 1000 instances.

APPENDIX F: QUANTILE OF RATIOS

The benchmarking analysis we performed in the main text is akin to the "ratio-of-quantiles" comparison performed in Ref. [18], where an alternative metric for speedups was also defined, called the "quantile of ratios." For this case, we find the annealing time that minimizes the TTS for each instance *individually*, and the per-instance optimal TTS, denoted TTS_i^*, is the minimal TTS for each instance individually. For each instance, the ratio of TTS_i^* for two different solvers is calculated, and different quantiles over the set of ratios are taken. We show in Fig. 13 the results for the median ratio using the logical-planted instances. The advantage of the DW2KQ relative to SA continues to hold, and SQA continues to exhibit the best scaling.

APPENDIX G: GADGET AND INSTANCE CONSTRUCTION

The key new ingredient in our instance construction is an 8-qubit gadget that fits into the unit cell of the D-Wave processors. The gadget has a bipartite $K_{4,4}$ graph connectivity with the following Ising parameters, as also depicted in Fig. 5 in the main text:

$$\vec{h}^T = (-1, -2/3, 2/3, -1, 1/3, 1, -1, 1),$$
$$J_{1,5} = +1, \quad J_{1,6} = -1, \quad J_{1,7} = -1, \quad J_{1,8} = -1,$$
$$J_{2,5} = -1, \quad J_{2,6} = -1, \quad J_{2,7} = +1, \quad J_{2,8} = -1,$$
$$J_{3,5} = -1, \quad J_{3,6} = -1, \quad J_{3,7} = -1, \quad J_{3,8} = -1,$$
$$J_{4,5} = -1, \quad J_{4,6} = -1, \quad J_{4,7} = -1, \quad J_{4,8} = -1. \tag{G1}$$

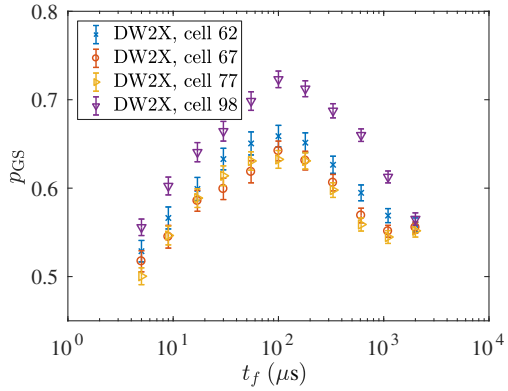
The logical-planted class of instances involves constructing planted-solution instances on the logical graph of the DW2KQ. The construction of the planted instance is similar to that of Ref. [22]. We define the logical graph of the DW2KQ as being composed of vertices corresponding to only the complete unit cells (with no faulty qubits or couplers). We also included one unit cell that was missing a single intracell coupler (unit cell 251), since having this missing coupler does not change the analysis. This is a minor difference relative to Ref. [22], where only complete unit cells were used. The edges of the logical graph correspond to having all four intercell couplers. We did remove the logical edge between unit cells 251 and 252. On an ideal Chimera graph, this would form a square grid. We constructed an Ising Hamiltonian as a sum of $\lfloor \alpha L^2 \rfloor$ frustrated loops, where we picked $\alpha = 0.65$. We again chose to plant the all-zero state. We constructed loops as follows. Choose a random vertex on the graph as the starting vertex, and randomly pick an available edge. If that edge does not already have $|J| = 3$, add the vertex connecting it to the chain until a loop is formed. Continue until the chain forms a loop by hitting a member of the chain. Only the loop and not the tail is kept.

Accept the loop if it includes more than 4 vertices; this means that the minimum loop has 6 vertices. This then generates a planted-solution instance on the logical graph. In order to embed it on the hardware graph, turn on all the available couplings in the unit cell to be ferromagnetic with $J = -3$. In the notation of Ref. [22], this amounts to constructing instances with $R = \rho = 3$.

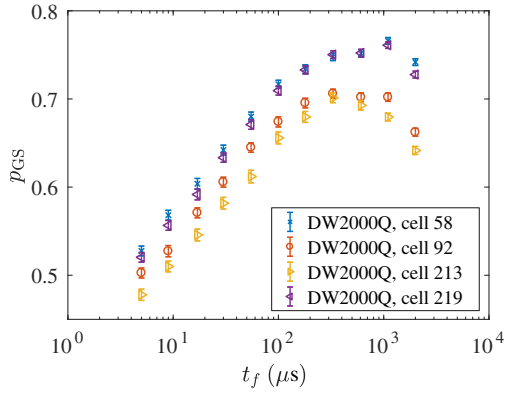
Finally, we randomly placed our gadget into a fraction $p = 0.1$ of all the connected unit cells in the planted-solution instance, and added these terms to the Ising Hamiltonian. (The gadget on unit cell 251 has the same ground state even with the one missing coupling.) The final Hamiltonian now has a maximum range of 6; i.e., $|J_{ij}| \leq 6$ for all couplers. Again, the ground state of the final Hamiltonian remains the all-zero state.

1. 8-qubit gadget

The key ingredient in our study is an 8-qubit gadget that fits into the unit cell of the D-Wave processors. Figure 14 compares the results for the 8-qubit gadget on the two



(a) DW2X



(b) DW2KQ

FIG. 14. Results for the 8-qubit gadget on the DW2KQ and DW2X. (a) The gadget on four different unit cells of the DW2X. (b) The gadget on four different unit cells of the DW2KQ. On both devices we used 1000 gauges with 1000 anneals per gauge. Error bars on the data points are 2σ calculated using 1000 bootstraps of each gauge.

D-Wave processor generations, for different representative unit cells. The success probability exhibits a single maximum, with the peaks occurring at different annealing times on the two devices. While there is some variation in the magnitude of the success probability depending on which unit cell is used, the position of the peak remains robust. We note, however, that the position of the peak differs on the two devices (around 100 μs on the DW2X and around 300 μs on the DW2KQ), indicating that the physical characteristics of the two devices are different beyond simply having different connectivity graphs.

2. Hardware-planted instances

Here, we describe a class of instances we call hardware planted (not discussed in the main text) that also exhibits an optimal annealing time within the accessible range of the DW2KQ, as we demonstrate below. The class is defined by constructing planted-solution instances on the hardware graph of the DW2KQ, shown in Fig. 10(a). This method builds an Ising Hamiltonian as a sum of $\lfloor \alpha 8L^2 \rfloor$ frustrated loops, where the all-zero state is a ground state of all loops (somewhat confusingly, the Hamiltonian is thus frustration free in the terminology of Ref. [88]). We picked $\alpha = 0.35$ (this value is approximately where the peak in hardness occurs for the HFS algorithm, described in Appendix J). We constructed loops as follows. Choose a random vertex on the graph as the starting vertex. From the (at most six) available edges connected to this vertex, randomly pick one. If this new vertex has not been visited already, it is added to the chain. Continue until the chain forms a loop by hitting a member of the chain. Only the loop and not the tail is kept. The loop is discarded if any of the couplings along the loop already have $|J| = 3$, and if the loop does not visit at least two unit cells [21]. The second condition means that the shortest possible loop includes six vertices (within each unit cell the degree of each vertex is four, but including other unit cells the degree is six, except for unit cells along the boundary of the Chimera graph, where the degree can be five). Along the loop, choose the couplings to satisfy the planted solution; i.e., set them all to be ferromagnetic. Then randomly pick a single coupling and flip it. The couplings along the loop are added to the already-present coupling values on the graph. This process is repeated until $\lfloor \alpha 8L^2 \rfloor$ loops are generated for the chosen value of α.

Finally, we randomly placed our gadget into pL^2 complete unit cells (without faulty qubits or couplers), where in this work we set $p = 0.1$, and added these terms to the Ising Hamiltonian. The final Hamiltonian now has a maximum range of 6; i.e., $|J_{ij}| \leq 6$ for all couplers. The ground state of the final Hamiltonian remains the all-zero state because this state is the ground state of all loop and gadget terms in the Hamiltonian.

We provide in Fig. 15 analogous results to those in Fig. 1 of the main text for the hardware-planted instances. In

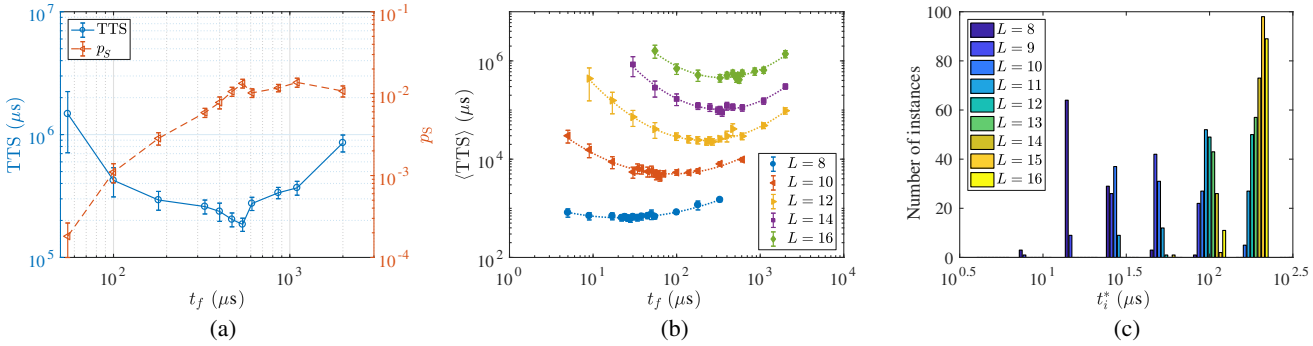


FIG. 15. Optimal annealing time and optimal TTS for the hardware-planted instance class on the DW2KQ. (a) TTS (blue solid line) and p_S (red dashed line) for a representative problem instance at size $L = 16$. A clear minimum in the TTS is visible at $t^* \approx 550$ μs, thus demonstrating the existence of an optimal annealing time for this instance. (b) Median TTS as a function of annealing time for $L \geq 8$. Dotted curves represent best-fit quadratic curves to the data (see Appendix K for details). The position of $\langle \mathrm{TTS} \rangle^*$ shifts to larger t_f as the system size increases. An optimum could not be established for $L < 8$ for this instance class; i.e., it appears that $t^* < 5$ μs when $L < 8$. (c) The distribution of instance optimal annealing times for different system sizes, as inferred directly from the positions of the minima as shown in (a). It is evident that the number of instances with higher optimal annealing times increases along with the system size, in agreement with (b). In (a) and (b) error bars represent 95% confidence intervals (2σ) calculated using 1000 bootstraps of 100 gauge transformations [17].

Fig. 15(a), we show a representative instance at $L = 16$ that exhibits an optimal annealing time above 500 μs. In Fig. 15(b), we show that the median TTS exhibits a clear minimum for sizes $L \in [8, 16]$ (no minimum was observed for $L < 8$), which moves to higher annealing time values with increasing problem size. This is reflected in the distribution of instance optimal annealing times, as shown in Fig. 15(c). The steady increase in the hardness of the instances with problem size is reflected in the upward shift of the minimum TTS in Fig. 15(b).

Apart from the obvious difference of the existence of optimal annealing times at smaller sizes ($L \geq 8$ compared to $L \geq 12$), the optimal annealing time is significantly higher for the hardware-planted instances than for the logical-planted instance class, as summarized in Fig. 16. The optimal annealing time is seen to increase with problem size in all cases, rising faster for the DW2KQ than for the DW2X, but eventually flattening for both types of problem instances. The increase is consistent with both the possibility of benefit from a longer adiabatic evolution time or from a longer thermal relaxation time at larger problem sizes.

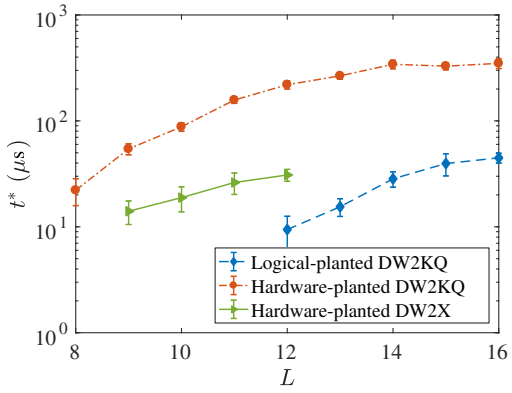
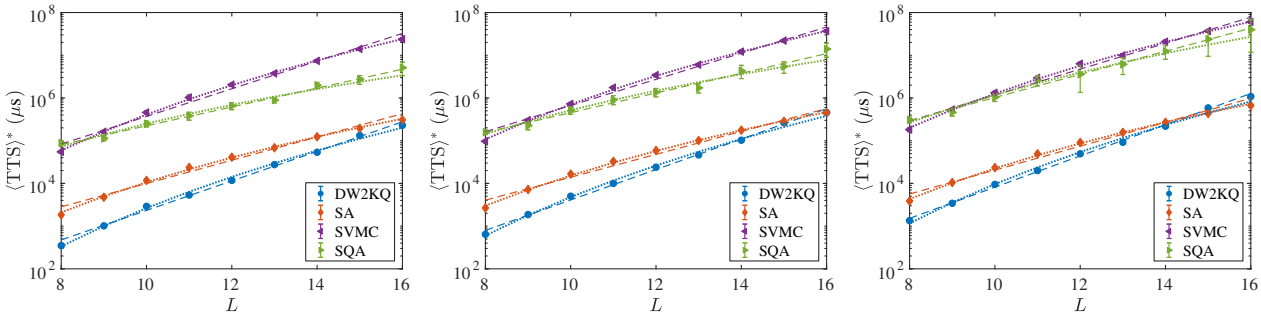


FIG. 16. Scaling of t^* with problem size for the two problem classes. Error bars represent the 95% confidence interval for the location of t^* by fitting to a quadratic function as described in Appendix K.

In Fig. 17, we present the scaling results for the hardware-planted instances at three different quantiles, in analogy to Fig. 2 in the main text. The simulation parameters for the solvers are identical except that we use colder temperatures for SA and SQA. For SA we use $\beta = 0.396$ [this corresponds to $\beta B(1) \approx 15$], while for SQA we use $\beta = 4.25$. The scaling coefficients are summarized in Table II. We again find that SQA has the smallest scaling coefficient. The scaling coefficient of the DW2KQ is larger than that of all the classical solvers we tested, so for this class of instances we can definitively rule out the possibility of scaling advantage against the solvers we tested.

3. Correlating SQA and SVMC algorithms for logical-planted instances

While a detailed analysis for each instance such as shown in Fig. 6 in the main text is prohibitive, we correlate in Fig. 18 the performance of SQA and SVMC algorithms at $\beta = 2.5$ and a relatively large number of sweeps. We observe that for almost all the instances, SQA finds the ground state with a significantly higher success probability and substantially fewer spin updates. Furthermore, a significant (but not overwhelming) number of instances hug the vertical axis of the scatter plot, corresponding to instances where the SVMC algorithm completely fails to find the ground state but SQA succeeds with a nonvanishing probability.



(a) $q = 0.25$ (b) $q = 0.50$ (c) $q = 0.75$

FIG. 17. Scaling of the optimal TTS with problem size for hardware-planted instances. The data points represent the DW2KQ (blue circles) and three classical solvers, SA (red diamonds), SVMC (purple left triangle), and SQA (green right triangle). The dashed and dotted curves correspond, respectively, to exponential and polynomial best fits with parameters given in Table II. Panels (a)–(c) correspond to the 25th quantile, median, and 75th quantile, respectively. Simulation parameters for SA, SVMC, and SQA are given in Appendix C. The data symbols obscure the error bars, representing the 95% confidence interval for each optimal TTS data point (computed from the fit of $\ln\langle \mathrm{TTS} \rangle$ to a quadratic function as explained in Appendix K).

TABLE II. The coefficient b in fits to (a) $\exp(a + bL)$ and (b) $\exp(a)L^b$ and the coefficient a in fits (c) $\exp(a + bL)$ and (d) $\exp(a)L^b$ for the hardware-planted instances using $L \in [8, 16]$. q denotes the quantile. Errors are 95% confidence intervals.

Solver	$q = 0.75$	$q = 0.50$	$q = 0.25$
		(a)	
DW2KQ	0.842 ± 0.01	0.820 ± 0.009	0.796 ± 0.009
SA	0.645 ± 0.008	0.617 ± 0.006	0.628 ± 0.007
SQA	0.594 ± 0.029	0.510 ± 0.018	0.487 ± 0.015
SVMC	0.699 ± 0.012	0.705 ± 0.010	0.746 ± 0.012
HFS		0.796 ± 0.008	
SAC		0.420 ± 0.015	
		(b)	
DW2KQ	9.470 ± 0.111	9.310 ± 0.097	9.228 ± 0.105
SA	7.405 ± 0.089	7.275 ± 0.071	7.277 ± 0.085
SQA	6.461 ± 0.315	5.703 ± 0.199	5.383 ± 0.164
SVMC	8.258 ± 0.143	8.431 ± 0.119	8.669 ± 0.133
HFS		9.231 ± 0.029	
SAC		5.001 ± 0.177	
		(c)	
DW2KQ	0.59 ± 0.12	0.12 ± 0.10	-0.21 ± 0.11
SA	14.30 ± 0.09	14.19 ± 0.08	13.75 ± 0.09
SQA	16.48 ± 0.30	16.44 ± 0.20	15.97 ± 0.16
SVMC	17.26 ± 0.15	16.63 ± 0.13	15.64 ± 0.15
HFS		4.29 ± 0.12	
SAC		12.16 ± 0.18	
		(d)	
DW2KQ	-12.63 ± 0.26	-12.96 ± 0.23	-13.37 ± 0.25
SA	3.79 ± 0.22	3.70 ± 0.18	3.34 ± 0.21
SQA	7.65 ± 0.73	8.51 ± 0.47	8.56 ± 0.39
SVMC	5.31 ± 0.36	4.33 ± 0.30	3.24 ± 0.33
HFS		-8.85 ± 0.07	
SAC		4.88 ± 0.43	

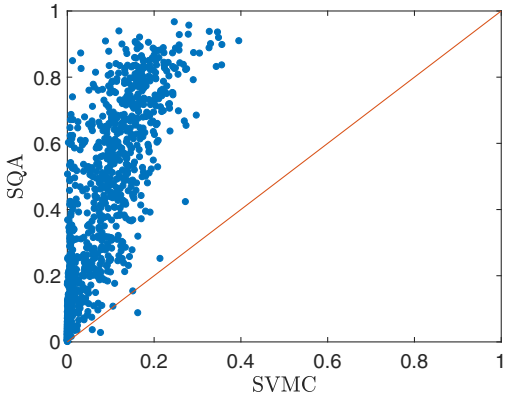


FIG. 18. Correlating the SQA and SVMC algorithm success probabilities. Shown is a scatter plot correlating the highest ground state probability for SQA (up to 2 M sweeps) and SVMC (up to 8 M sweeps) algorithms. For the results shown here both algorithms use the DW2X annealing schedule with an inverse temperature of $\beta = 2.5$.

APPENDIX H: SUCCESS PROBABILITY SCALING

In Fig. 7 of the main text, we showed the power-law scaling coefficient of the success probability extracted for $t_f \leq 50$ μs. Here we provide supplemental data to support the quality of these fits. First, we show the data and fit in Fig. 19(a) for the instance depicted in Fig. 1(a) of the main text as well as its counterpart without the gadget. The data for this instance nicely agree with a polynomial fit. In Fig. 19(b), we show that the uncertainty in the power-law scaling coefficient for the majority of the instances is below 10%, indicating that the polynomial fits to the data points are reasonable.

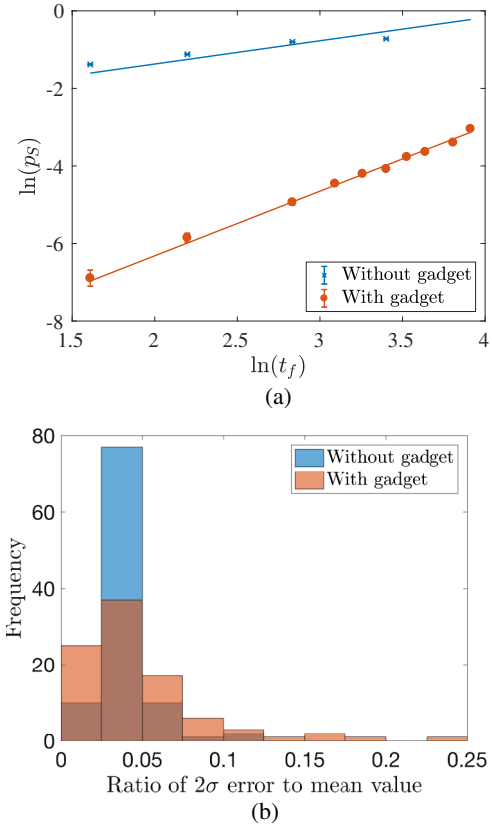


FIG. 19. Fits of the success probability to a power law. (a) The data points represent the DW2KQ $\ln(p_S)$ results for the logical-planted instances with (red circles) or without (blue crosses) the gadget over the range $t_f \in [5, 50]$ μs. The solid lines correspond to the linear best fits $a\ln(t_f) + b$, where $a = 1.546 \pm 0.015$, $b = -8.348 \pm 0.052$ (with the gadget), and $a = 0.367 \pm 0.007$, $b = -1.918 \pm 0.019$ (without the gadget). The 2σ error bars are not visible as they are smaller than the data marker size. (b) The ratio of the 2σ error to the best-fit value of the linear coefficient (shown in Fig. 7 of the main text) for 100 instances of the logical-planted instances without (blue) and with (red) the gadget.

APPENDIX I: CALCULATING THE NORMALIZED OVERLAP OF INSTANCES

In Fig. 8 of the main text we showed the overlap of the logical-planted instances below the median between the classical solvers and the DW2KQ. In order to calculate this quantity, we first fit the ln(TTS) of each instance to the function $a(\ln t_f - b)^2 + c$, and we evaluate the function at the optimal annealing time for the median TTS. This gives us a mean value $\bar{\mathrm{TTS}}_i(t^*)$ and its associated 1σ error $\Delta\mathrm{TTS}_i$ for the ith instance. We then perform 1000 bootstraps over 400 instances, where for each bootstrap we generate two sets of 100 normally distributed random numbers $\eta_{i,(1,2)}$ in order to calculate two TTS realizations for each instance; i.e., $\mathrm{TTS}_{i,(1,2)} = \bar{\mathrm{TTS}}_i(t^*) + \eta_{i,(1,2)}\Delta\mathrm{TTS}_i$. For the two sets of TTS realizations, we calculate the median TTS and find which instances have a TTS below the median. We calculate the overlap fraction of instances between two solvers S and S' for realizations α and β, respectively, which we denote by f_{S_α,S'_β}. The normalized fraction $\bar{f}_{C,\mathrm{DW2KQ}}$ between a solver C and the DW2KQ is then given $f_{C_1,\mathrm{DW2KQ}_2}$ by

$$\bar{f}_{C,\mathrm{DW2KQ}} = f_{C_1,\mathrm{DW2KQ}_2} / \sqrt{f_{C_1,C_2} f_{\mathrm{DW2KQ}_1,\mathrm{DW2KQ}_2}}. \quad \text{(I1)}$$

The normalization ensures that even with the noisy realization of the TTS, the overlap of instances between a solver and itself is one.

APPENDIX J: COMPARISON TO OTHER ALGORITHMS

In this appendix, we present results from testing a number of other algorithms. Of course, for practical reasons we cannot consider all other relevant algorithms (e.g., we do not consider the isoenergetic cluster updates algorithm [80]). Instead, we aimed to find other algorithms in addition to SQA that have a better scaling than the DW2KQ for the logical-planted instances. SQA remains the best-scaling algorithm among those we tested.

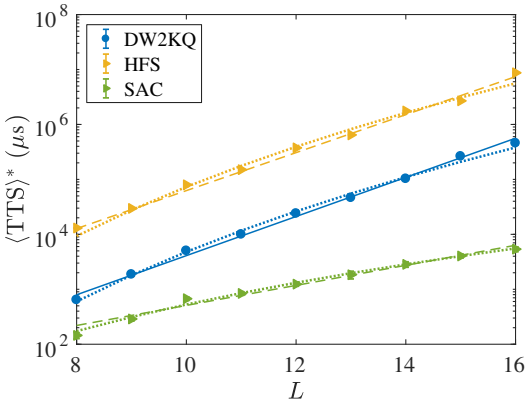
1. HFS

In the main text, we did not make comparisons to the HFS algorithm because it does not implement the same algorithmic approach as the other annealing algorithms. Nevertheless, because it is an algorithm tailored to solve spin-glass problems on the Chimera architecture, it is instructive to compare its performance. For the HFS algorithm, we use the implementation provided by Ref. [89] (which does not utilize a GPU), and we ran it in mode "-S3," meaning that maximal induced trees (tree width 1 in this case) were used. The TTS is given by [20]

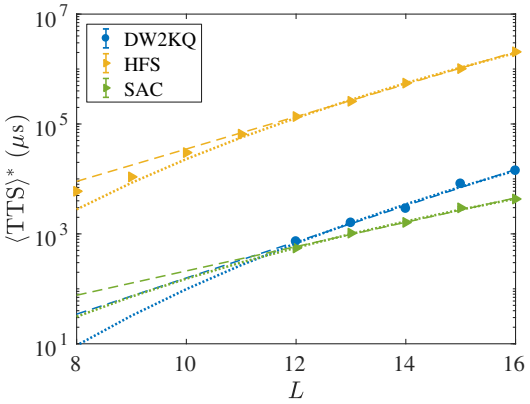
$$\mathrm{TTS}_{\mathrm{HFS}} = \tau_{\mathrm{HFS}} L \left(\frac{5}{4}L + 2\right) n_{\mathrm{trees}} R(n_{\mathrm{trees}}), \quad \text{(J1)}$$

where $\tau_{\mathrm{HFS}} = 0.3$ μs is the time for a single update. $R(n_{\mathrm{trees}})$ is the number of repetitions with n_{trees} tree updates. In principle, the optimal TTS is found by finding the value of n_{trees} that minimizes the TTS, but the implementation of Ref. [89] continues to increase n_{trees} until an exit criterion is reached. Specifically, the algorithm exits when the same lowest energy is found consecutively after $n_{\mathrm{exit}} = 4$ tree updates. We have found that this can be highly nonoptimal, especially for the hardware-planted instances. Therefore, in all our scaling plots, we have optimized the value of n_{trees}. This is an important distinction from all previous work using the HFS algorithm, which to the best of our knowledge did not optimize n_{trees}, and hence the scaling of the HFS algorithm in previous work is likely to be an underestimate of the true scaling, in the very same sense that the D-Wave scaling reported previously underestimates the true scaling.

The behavior of the optimal TTS with problem size is shown in Fig. 20, with the scaling parameter fits given in



(a) hardware planted



(b) logical planted

FIG. 20. Scaling of the median optimal TTS with problem size for DW2KQ versus HFS and SAC. The data points represent the DW2KQ (blue circles), HFS (yellow left triangle), and SAC (green right triangle). The dashed and dotted curves correspond, respectively, to exponential and polynomial best fits with parameters given in Table III. (a) Hardware-planted instances. (b) Logical-planted instances. The data symbols obscure the error bars, representing the 95% confidence interval for each optimal TTS data point [computed from the fit of $\ln\langle \mathrm{TTS} \rangle$ to a quadratic function as explained in Appendix K].

Table III. We find that for the logical-planted instances, the HFS algorithm scales better than the DW2KQ, while for the hardware-planted instances, the scaling of the two is statistically indistinguishable.

For the HFS algorithm, we find that a quadratic fit does not capture the general features of the TTS curve as a function of number of tree updates. Instead, we find that a function of the form $\ln\langle \mathrm{TTS} \rangle = a(\ln n_{\mathrm{tree}})^{-3} + b(\ln n_{\mathrm{tree}}) - (4/3^{3/4})(a^3 b)^{1/4} + c$ captures the data well. The value of c gives the value of $\ln\langle \mathrm{TTS} \rangle^*$. We give the fit values and their confidence intervals in Tables IV and V.

2. SAC

We can also consider simulated annealing with both single and multispin updates (SAC), with the latter being simultaneous updates of all the spins comprising a unit cell

TABLE III. The coefficient b in fits to (a) $\exp(a + bL)$ and (b) $\exp(a)L^b$ and the coefficient a in fits (c) $\exp(a + bL)$ and (d) $\exp(a)L^b$ for the logical-planted instances using $L \in [12, 16]$. q denotes the quantile. Errors are 95% confidence intervals. Given here are the fits for SVMC and SQA at $\beta = 2.5$ only.

Solver	$q = 0.75$	$q = 0.50$	$q = 0.25$
		(a)	
DW2KQ	0.864 ± 0.028	0.760 ± 0.017	0.701 ± 0.014
SA	1.064 ± 0.031	0.996 ± 0.024	0.961 ± 0.023
SVMC	0.773 ± 0.060	0.500 ± 0.029	0.441 ± 0.020
SQA	0.450 ± 0.050	0.365 ± 0.035	0.331 ± 0.024
HFS		0.678 ± 0.013	
SAC		0.510 ± 0.018	
		(b)	
DW2KQ	11.962 ± 0.391	10.573 ± 0.242	9.746 ± 0.201
SA	14.635 ± 0.433	13.746 ± 0.331	13.299 ± 0.316
SVMC	10.735 ± 0.834	6.890 ± 0.399	6.141 ± 0.273
SQA	6.221 ± 0.697	5.047 ± 0.484	4.561 ± 0.331
HFS		9.455 ± 0.183	
SAC		7.134 ± 0.259	
		(c)	
DW2KQ	-2.85 ± 0.40	-2.54 ± 0.25	-2.53 ± 0.21
SA	12.42 ± 0.44	12.46 ± 0.33	12.18 ± 0.32
SVMC	16.88 ± 0.84	19.26 ± 0.39	19.46 ± 0.27
SQA	16.60 ± 0.69	17.13 ± 0.48	17.12 ± 0.33
HFS		4.89 ± 0.19	
SAC		10.39 ± 0.26	
		(d)	
DW2KQ	-22.25 ± 1.03	-19.75 ± 0.64	-18.39 ± 0.53
SA	-11.23 ± 1.14	-9.80 ± 0.87	-9.39 ± 0.83
SVMC	-0.56 ± 2.18	8.11 ± 1.03	9.47 ± 0.72
SQA	6.51 ± 1.82	8.95 ± 1.27	9.74 ± 0.86
HFS		-10.52 ± 0.49	
SAC		-1.23 ± 0.68	

(superspin approximation [23]). This requires the algorithm to know about the underlying hardware graph. The implementation of SAC is identical to that of SA, except each sweep of single-spin updates is followed by a sweep of unit cell updates. The eight spins in the unit cell are flipped, and the move is accepted according to the Metropolis-Hastings rule. Because the unit cell graph is bipartite, unit cells in the first partition are updated first, followed by the unit cells in the second partition. This algorithm can be implemented as efficiently on GPU's as the single-spin SA algorithm since it does not require storing any more data in memory. Because SAC effectively involves updating twice as many spins as SA in a single sweep, the timing of SAC is the same as in Eq. (C2) in Appendix C but with $f_{\mathrm{SAC}} = 25\ \mathrm{ns}^{-1}$. For consistency, we use the same annealing schedule in $B(s)\beta$ for SAC as we did for SA, with $B(s)$ as in Fig. 9(b) (in units where the maximum Ising coupling strength $|J_{ij}|$ is 1). We use $\beta = 0.132$ [this corresponds to $\beta B(1) \approx 5$] for the logical-planted instances and $\beta = 0.396$ [this corresponds to $\beta B(1) \approx 15$] for the hardware-planted instances. We give

TABLE IV. Fit to $ax^{-3} + bx + c - 4(a^3b)^{1/4}/3^{3/4}$ for the median results of the logical-planted instances using HFS with $y = \log \mathrm{TTS}$ and $x = \log n_{\mathrm{trees}}$. The factor c does not include τ_{HFS}.

L	Minimum trees	Maximum trees	a	b	c
8	3	30	0.841 ± 0.082	2.221 ± 0.453	9.897 ± 0.063
9	3	30	0.675 ± 0.078	2.458 ± 0.362	10.502 ± 0.052
10	4	30	0.747 ± 0.116	4.088 ± 0.941	11.518 ± 0.055
11	4	30	0.633 ± 0.117	5.253 ± 0.989	12.292 ± 0.053
12	5	30	0.669 ± 0.087	7.320 ± 0.957	13.029 ± 0.028
13	5	30	0.707 ± 0.104	12.266 ± 1.951	13.668 ± 0.029
14	5	30	0.666 ± 0.103	13.070 ± 2.049	14.430 ± 0.024
15	6	30	0.593 ± 0.095	14.247 ± 1.674	15.038 ± 0.022
16	5	30	0.674 ± 0.086	19.071 ± 1.855	15.742 ± 0.024

TABLE V. Fit to $ax^{-3} + bx + c - 4(a^3b)^{1/4}/3^{3/4}$ for the median results of the hardware-planted instances using HFS with $y = \log \mathrm{TTS}$ and $x = \log n_{\mathrm{trees}}$. The factor c does not include τ_{HFS}.

L	Minimum trees	Maximum trees	a	b	c
8	5	30	0.868 ± 0.068	7.838 ± 0.643	10.679 ± 0.020
9	7	30	0.489 ± 0.102	8.908 ± 1.392	11.500 ± 0.015
10	5	50	0.670 ± 0.041	13.950 ± 0.854	12.491 ± 0.018
11	5	50	0.432 ± 0.040	15.410 ± 0.671	13.125 ± 0.011
12	10	30	0.672 ± 0.263	23.631 ± 5.858	14.038 ± 0.022
13	30	100	1.061 ± 0.154	76.298 ± 12.888	14.582 ± 0.012
14	10	100	0.486 ± 0.086	35.793 ± 3.1606	15.583 ± 0.026
15	35	100	0.943 ± 0.239	96.115 ± 22.116	16.009 ± 0.014
16	35	100	0.878 ± 0.245	69.559 ± 22.095	17.185 ± 0.014

the fit values and their confidence intervals in Tables VI and VII, and the behavior of the optimal TTS with problem size is shown in Fig. 20, with the scaling parameter fits given in Table II. We see that the HFS and DW2KQ algorithms are statistically indistinguishable, and SAC is the top-scaling algorithm for the hardware-planted instances, outperforming even SQA. Results for the logical-planted instances are given in Table III, which for convenience reproduces data

TABLE VI. Fit to $y = a(x-b)^2 + c$ for the median results of the logical-planted instances using SAC with $y = \log \mathrm{TTS}$ and $x = \log t_f$. The factor c does not include f_{SAC}.

L	Minimum t_f	Maximum t_f	a	b	c
12	10	2154	0.101 ± 0.010	4.495 ± 0.099	16.446 ± 0.039
13	10	2154	0.122 ± 0.009	5.104 ± 0.061	17.067 ± 0.037
14	10^2	2154	0.123 ± 0.026	5.504 ± 0.171	17.515 ± 0.031
15	10^2	2154	0.143 ± 0.034	5.876 ± 0.130	18.124 ± 0.046
16	10^2	3593	0.145 ± 0.020	6.247 ± 0.079	18.497 ± 0.034

TABLE VII. Fit to $y = a(x-b)^2 + c$ for the median results of the hardware-planted instances using SAC with $y = \log \mathrm{TTS}$ and $x = \log n_{\mathrm{sweeps}}$. The factor c does not include f_{SAC}.

L	Minimum sweeps	Maximum sweeps	a	b	c
8	46	10^3	0.262 ± 0.102	5.911 ± 0.227	15.098 ± 0.125
9	46	4641	0.217 ± 0.038	6.656 ± 0.110	15.792 ± 0.082
10	166	5994	0.207 ± 0.048	6.907 ± 0.129	16.641 ± 0.078
11	166	5994	0.327 ± 0.059	6.903 ± 0.101	16.859 ± 0.079
12	166	10^4	0.293 ± 0.042	7.480 ± 0.100	17.250 ± 0.089
13	215	10^4	0.277 ± 0.078	7.743 ± 0.146	17.643 ± 0.165
14	278	10^4	0.295 ± 0.075	8.016 ± 0.180	18.075 ± 0.118
15	278	10^4	0.326 ± 0.057	8.160 ± 0.123	18.427 ± 0.083
16	464	10^4	0.325 ± 0.091	8.201 ± 0.166	18.712 ± 0.104

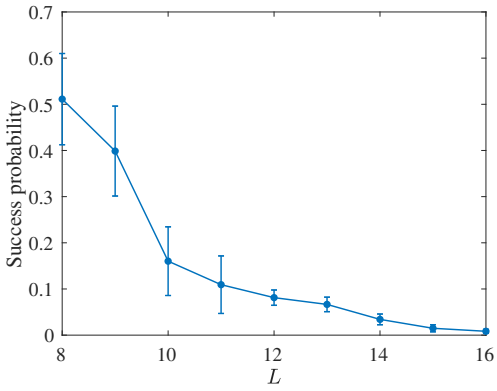


FIG. 21. Success probability for MWPM on the logical-planted instances. The data symbols correspond to the average success probability calculated using 1000 bootstraps of 100 instances for $L < 12$ and of 1000 instances for $L \geq 12$, and the error bars represent the 95% confidence intervals (2σ) calculated from the same bootstrap.

from Fig 3 in the main text. Here we see that SAC outperforms the HFS algorithm, which in turn outperforms DW2KQ, while SQA is the top-scaling algorithm, outperforming SAC.

3. Minimum-weight perfect matching

Before the mapping onto Chimera and the introduction of the gadget, the logical-planted solution instances are defined on a two-dimensional square grid. Here, we check a polynomial-time algorithm for solving the minimum-weight perfect-matching problem [90,91] and show that it cannot be used to efficiently determine the ground state of the logical-planted instances defined on Chimera with the gadget. In order to do so, we map the Ising Hamiltonian on the square grid to a MWPM problem [92], run the Blossom V algorithm [91] to determine the solution to the MWPM problem, and finally map the solution of the MWPM problem to a ground state of the Ising Hamiltonian. While the MWPM algorithm does find the ground state of the planted-solution instance defined on the square grid, it does not necessarily find the ground state of the associated Chimera instance with the gadget. The reason for this is that the gadget reduces the degeneracy of the ground state by selecting only those states for which the unit cell on which the gadget is placed points up. Without knowing this and because of the large ground state degeneracy, the MWPM predominantly selects the wrong ground state. We show in Fig. 21 how the success probability of finding the ground state decreases with increasing problem size. While at $L = 8$, MWPM finds the ground state for approximately half the instances, at $L = 16$, it finds the ground state of only 16 instances out of 1000 instances. This means MWPM is not competitive with SAC, HFS, or SQA.

APPENDIX K: DETERMINING THE OPTIMAL ANNEALING TIME AND OPTIMAL TTS

In order to determine the position of the optimal annealing time t^* and its associated $\langle \mathrm{TTS} \rangle^*$ (we drop the quantile notation q for simplicity) at a given size L for the DW2KQ, DW2X, SA, SQA, and SVMC results, we fit

TABLE VIII. Fit to $y = a(x-b)^2 + c$ for the median results of the logical-planted instances using the DW2KQ with $y = \log \mathrm{TTS}$ and $x = \log t_f$.

L	Minimum t_f	Maximum t_f	a	b	c
12	5	100	0.124 ± 0.040	2.233 ± 0.317	6.588 ± 0.036
13	5	100	0.141 ± 0.044	2.732 ± 0.196	7.371 ± 0.040
14	5	100	0.144 ± 0.040	3.341 ± 0.160	7.968 ± 0.032
15	5	100	0.182 ± 0.050	3.672 ± 0.224	9.007 ± 0.032
16	5	180	0.221 ± 0.031	3.798 ± 0.104	9.557 ± 0.032

TABLE IX. Fit to $y = a(x-b)^2 + c$ for the median results of the hardware-planted instances using the DW2KQ with $y = \log \mathrm{TTS}$ and $x = \log t_f$.

L	Minimum t_f	Maximum t_f	a	b	c
8	5	330	0.115 ± 0.026	3.089 ± 0.266	6.465 ± 0.046
9	5	330	0.148 ± 0.026	3.997 ± 0.122	7.531 ± 0.051
10	5	610	0.203 ± 0.025	4.476 ± 0.092	8.521 ± 0.049
11	5	2000	0.268 ± 0.019	5.059 ± 0.070	9.209 ± 0.055
12	9	2000	0.281 ± 0.029	5.390 ± 0.092	10.084 ± 0.060
13	17	2000	0.305 ± 0.029	5.583 ± 0.070	10.745 ± 0.050
14	30	2000	0.338 ± 0.044	5.834 ± 0.093	11.540 ± 0.064
15	30	2000	0.330 ± 0.038	5.795 ± 0.080	12.479 ± 0.063
16	55	2000	0.342 ± 0.055	5.853 ± 0.107	13.033 ± 0.068

TABLE X. Fit to $y = a(x-b)^2 + c$ for the median results of the hardware-planted instances using the DW2X with $y = \log \mathrm{TTS}$ and $x = \log t_f$.

L	Minimum t_f	Maximum t_f	a	b	c
9	5	2000	0.223 ± 0.026	2.636 ± 0.244	7.688 ± 0.103
10	5	2000	0.240 ± 0.034	2.926 ± 0.270	8.780 ± 0.103
11	5	2000	0.291 ± 0.037	3.265 ± 0.226	9.825 ± 0.112
12	5	2000	0.333 ± 0.030	3.428 ± 0.137	10.591 ± 0.083

TABLE XI. Fit to $y = a(x-b)^2 + c$ for the median results of the logical-planted instances using SA with $y = \log \mathrm{TTS}$ and $x = \log n_{\mathrm{sweeps}}$. The factor c does not include f_{SA}.

L	Minimum sweeps	Maximum sweeps	a	b	c
8	784	14384	0.130 ± 0.029	7.843 ± 0.118	19.272 ± 0.034
9	1000	37926	0.139 ± 0.022	8.58 ± 0.086	20.355 ± 0.040
10	3359	10^5	0.124 ± 0.025	9.408 ± 0.118	22.105 ± 0.037
11	5455	2×10^5	0.125 ± 0.026	10.196 ± 0.118	23.29 ± 0.035
12	5000	10^6	0.122 ± 0.012	10.970 ± 0.075	24.423 ± 0.041
13	10 000	10^6	0.124 ± 0.020	11.485 ± 0.109	25.445 ± 0.054
14	22 000	2×10^6	0.122 ± 0.019	12.057 ± 0.114	26.301 ± 0.044
15	35 000	2×10^6	0.132 ± 0.028	12.601 ± 0.125	27.568 ± 0.059
16	60 000	8×10^6	0.109 ± 0.017	13.103 ± 0.139	28.366 ± 0.052

TABLE XII. Fit to $y = a(x-b)^2 + c$ for the median results of the hardware-planted instances using SAA with $y = \log \mathrm{TTS}$ and $x = \log n_{\mathrm{sweeps}}$. The factor c does not include f_{SA}.

L	Minimum sweeps	Maximum sweeps	a	b	c
8	297	23357	0.127 ± 0.015	7.971 ± 0.072	18.708 ± 0.040
9	483	37926	0.139 ± 0.018	8.772 ± 0.080	19.703 ± 0.049
10	1274	10^5	0.124 ± 0.014	9.373 ± 0.073	20.533 ± 0.032
11	2069	1.2×10^5	0.145 ± 0.020	10.046 ± 0.081	21.221 ± 0.050
12	5455	1.2×10^5	0.154 ± 0.034	10.45 ± 0.111	21.802 ± 0.038
13	5455	3×10^5	0.153 ± 0.024	11.037 ± 0.115	22.344 ± 0.051
14	5455	3×10^5	0.188 ± 0.025	11.429 ± 0.115	22.898 ± 0.051
15	8858	5.6×10^5	0.179 ± 0.017	11.887 ± 0.077	23.378 ± 0.035
16	14 384	10^6	0.172 ± 0.018	12.293 ± 0.073	23.837 ± 0.039

TABLE XIII. Fit to $y = a(x-b)^2 + c$ for the median results of the logical-planted instances using SQA with $y = \log \mathrm{TTS}$ and $x = \log n_{\mathrm{sweeps}}$. The factor c does not include f_{SQA}.

L	Minimum sweeps	Maximum sweeps	a	b	c
8	27 825	359 381	0.339 ± 0.039	11.475 ± 0.046	19.476 ± 0.036
9	27 825	599 484	0.325 ± 0.028	11.792 ± 0.037	19.811 ± 0.037
10	46 415	10^6	0.324 ± 0.033	12.068 ± 0.048	20.577 ± 0.039
11	46 415	599 484	0.451 ± 0.053	12.17 ± 0.050	20.977 ± 0.05
12	77 426	10^6	0.322 ± 0.064	12.296 ± 0.082	21.505 ± 0.061
13	77426	10^6	0.414 ± 0.075	12.391 ± 0.070	21.905 ± 0.073
14	77 426	10^6	0.469 ± 0.078	12.477 ± 0.065	22.171 ± 0.069
15	77 426	10^6	0.575 ± 0.095	12.589 ± 0.071	22.678 ± 0.089
16	77 426	10^6	0.606 ± 0.094	12.653 ± 0.071	22.955 ± 0.073

the $\ln\langle \mathrm{TTS}\rangle$ data for different annealing times to a function of the form $a(\ln t_f - b)^2 + c$. The value of b gives the value of t^*, and the value of c is the associated $\ln\langle \mathrm{TTS}\rangle^*$.

The fit values and their confidence intervals are given in Tables VIII (logical planted) and IX (hardware planted) for the DW2KQ, in Table X for the DW2X (hardware planted only, since logical-planted instances did not exhibit an accessible optimal annealing time on the DW2X), in Tables XI and XII for SA, in Tables XIII and XIV for SQA, and in Tables XV and XVI for the SVMC algorithm.

The outcome of this fitting procedure for the DW2KQ results on the logical-planted instances are shown in Fig. 1(b) of the main text and on the hardware-planted instances in Fig. 15(b). The fits for the DW2X on the hardware-planted instances are shown in Fig. 22. Because the largest problem size we can program on the DW2X is at $L = 12$, we studied only hardware-planted instances on

TABLE XIV. Fit to $y = a(x-b)^2 + c$ for the median results of the hardware-planted instances using SQA with $y = \log \mathrm{TTS}$ and $x = \log n_{\mathrm{sweeps}}$. The factor c does not include f_{SQA}.

L	Minimum sweeps	Maximum sweeps	a	b	c
8	7742	599 484	0.373 ± 0.036	12.005 ± 0.085	20.424 ± 0.082
9	12 915	599 484	0.504 ± 0.050	12.162 ± 0.089	20.985 ± 0.086
10	16 681	599 484	0.567 ± 0.052	12.258 ± 0.075	21.700 ± 0.086
11	27 825	599 484	0.704 ± 0.092	12.382 ± 0.095	22.057 ± 0.117
12	46 415	599 484	0.701 ± 0.164	12.497 ± 0.130	22.632 ± 0.142
13	46 415	10^6	0.613 ± 0.089	12.742 ± 0.089	23.043 ± 0.108
14	46 415	10^6	0.756 ± 0.119	12.791 ± 0.098	23.597 ± 0.142
15	77 426	10^6	0.662 ± 0.173	12.928 ± 0.129	24.069 ± 0.156
16	77 426	10^6	0.710 ± 0.181	12.945 ± 0.139	24.434 ± 0.166

TABLE XV. Fit to $y = a(x-b)^2 + c$ for the median results of the logical-planted instances using the SVMC algorithm with $y = \log \mathrm{TTS}$ and $x = \log n_{\mathrm{sweeps}}$. The factor c does not include f_{SVMC}.

L	Minimum sweeps	Maximum sweeps	a	b	c
8	4641	10^5	0.183 ± 0.032	10.166 ± 0.075	21.545 ± 0.041
9	7742	4×10^5	0.135 ± 0.017	10.919 ± 0.079	22.485 ± 0.037
10	27 825	10^6	0.141 ± 0.023	11.705 ± 0.103	23.515 ± 0.040
11	27 825	4×10^6	0.140 ± 0.011	12.546 ± 0.059	23.797 ± 0.036
12	27 825	4×10^6	0.168 ± 0.012	13.003 ± 0.049	25.184 ± 0.042
13	46 415	4×10^6	0.174 ± 0.023	13.362 ± 0.076	25.864 ± 0.056
14	77 426	3×10^6	0.217 ± 0.041	13.700 ± 0.105	26.457 ± 0.069
15	77 426	8×10^6	0.220 ± 0.023	14.159 ± 0.110	26.582 ± 0.077
16	10^5	8×10^6	0.235 ± 0.024	14.434 ± 0.126	27.208 ± 0.072

TABLE XVI. Fit to $y = a(x-b)^2 + c$ for the median results of the hardware-planted instances using the SVMC algorithm with $y = \log \mathrm{TTS}$ and $x = \log n_{\mathrm{sweeps}}$. The factor c does not include f_{SVMC}.

L	Minimum sweeps	Maximum sweeps	a	b	c
8	4641	10^5	0.087 ± 0.066	10.229 ± 0.406	21.754 ± 0.082
9	2782	4×10^5	0.106 ± 0.026	11.350 ± 0.215	22.890 ± 0.075
10	4641	4×10^5	0.132 ± 0.030	11.961 ± 0.255	23.774 ± 0.061
11	10^5	10^6	0.148 ± 0.042	12.875 ± 0.285	24.651 ± 0.079
12	46 415	4×10^6	0.143 ± 0.023	13.492 ± 0.111	25.321 ± 0.060
13	46 415	4×10^6	0.158 ± 0.023	13.887 ± 0.107	25.884 ± 0.050
14	77 426	4×10^6	0.176 ± 0.048	14.394 ± 0.264	26.592 ± 0.075
15	12 9154	8×10^6	0.179 ± 0.034	14.919 ± 0.167	27.183 ± 0.057
16	21 5443	8×10^6	0.182 ± 0.043	15.215 ± 0.241	27.693 ± 0.054

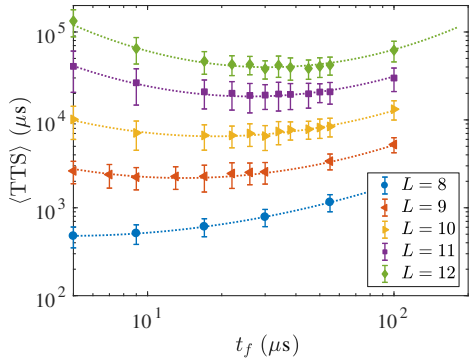


FIG. 22. Median TTS as a function of annealing time for hardware-planted instances on the DW2X. Unlike the DW2KQ, the optimal annealing time for $L = 8$ appears to be smaller than 5 μs; for $L \geq 9$ the optimal annealing time t^* lies within the range of achievable annealing times for the DW2X device. Furthermore, for the same problem size, the DW2X optimal annealing times are consistently smaller than those of the DW2KQ [compare to Fig. 15(b) and recall Fig. 16]. While the difference in the hardware graph may play a role in this, it is likely that the intrinsic differences between the two devices is responsible (see Appendix G 1 for additional comparisons).

this device. Note that because the hardware graph of the DW2X differs from that of the DW2KQ [see Figs. 10(a) and 10(b)], we should not assume that the instances defined on both are necessarily from the same class.

[1] P. W. Shor, *Algorithms for Quantum Computation: Discrete Logarithms and Factoring*, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science, 1994*, pp. 124–134, https://doi.org/10.1109/SFCS.1994.365700.

[2] R. P. Feynman, *Simulating Physics with Computers*, Int. J. Theor. Phys. **21**, 467 (1982).

[3] S. Lloyd, *Universal Quantum Simulators*, Science **273**, 1073 (1996).

[4] J. I. Cirac and P. Zoller, *Goals and Opportunities in Quantum Simulation*, Nat. Phys. **8**, 264 (2012).

[5] J. Kelly, *A Preview of Bristlecone, Google's New Quantum Processor*, https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html.

[6] D. Gil, *The Future Is Quantum*, https://www.ibm.com/blogs/research/2017/11/the-future-is-quantum.

[7] 2018 CES: Intel Advances Quantum and Neuromorphic Computing Research, 2018, https://newsroom.intel.com/news/intel-advances-quantum-neuromorphic-computing-research.

[8] J. S. Otterbach *et al.*, *Unsupervised Machine Learning on a Hybrid Quantum Computer*, arXiv:1712.05771.

[9] J. Zhang, G. Pagano, P. W. Hess, A. Kyprianidis, P. Becker, H. Kaplan, A. V. Gorshkov, Z. X. Gong, and C. Monroe, *Observation of a Many-Body Dynamical Phase Transition with a 53-Qubit Quantum Simulator*, Nature (London) **551**, 601 (2017).

[10] H. Bernien, S. Schwartz, A. Keesling, H. Levine, A. Omran, H. Pichler, S. Choi, A. S. Zibrov, M. Endres, M. Greiner, V. Vuletić, and M. D. Lukin, *Probing Many-Body Dynamics on a 51-Atom Quantum Simulator*, Nature (London) **551**, 579 (2017).

[11] C. Neill *et al.*, *A Blueprint for Demonstrating Quantum Supremacy with Superconducting Qubits*, Science **360**, 195 (2018).

[12] J. Preskill, *Quantum Computing in the NISQ Era and Beyond*, arXiv:1801.00862.

[13] A. W. Harrow and A. Montanaro, *Quantum Computational Supremacy*, Nature (London) **549**, 203 (2017).

[14] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, England, 2010).

[15] T. Kadowaki and H. Nishimori, *Quantum Annealing in the Transverse Ising Model*, Phys. Rev. E **58**, 5355 (1998).

[16] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, *A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem*, Science **292**, 472 (2001).

[17] S. Boixo, T. F. Ronnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, *Evidence for Quantum Annealing with More Than One Hundred Qubits*, Nat. Phys. **10**, 218 (2014).

[18] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, *Defining and Detecting Quantum Speedup*, Science **345**, 420 (2014).

[19] J. King, S. Yarkoni, M. M. Nevisi, J. P. Hilton, and C. C. McGeoch, *Benchmarking a Quantum Annealing Processor with the Time-to-Target Metric*, arXiv:1508.05087.

[20] I. Hen, J. Job, T. Albash, T. F. Rønnow, M. Troyer, and D. A. Lidar, *Probing for Quantum Speedup in Spin-Glass Problems with Planted Solutions*, Phys. Rev. A **92**, 042325 (2015).

[21] A. D. King, T. Lanting, and R. Harris, *Performance of a Quantum Annealer on Range-Limited Constraint Satisfaction Problems*, arXiv:1502.02098.

[22] J. King, S. Yarkoni, J. Raymond, I. Ozfidan, A. D. King, M. M. Nevisi, J. P. Hilton, and C. C. McGeoch, *Quantum Annealing Amid Local Ruggedness and Global Frustration*, arXiv:1701.04579.

[23] S. Mandrà, Z. Zhu, W. Wang, A. Perdomo-Ortiz, and H. G. Katzgraber, *Strengths and Weaknesses of Weak-Strong Cluster Problems: A Detailed Overview of State-of-the-Art Classical Heuristics versus Quantum Approaches*, Phys. Rev. A **94**, 022337 (2016).

[24] W. Vinci and D. A. Lidar, *Optimally Stopped Optimization*, Phys. Rev. Applied **6**, 054016 (2016).

[25] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, *What Is the Computational Value of Finite-Range Tunneling?*, Phys. Rev. X **6**, 031015 (2016).

[26] H. G. Katzgraber, F. Hamze, Z. Zhu, A. J. Ochoa, and H. Munoz-Bauza, *Seeking Quantum Speedup through Spin Glasses: The Good, the Bad, and the Ugly*, Phys. Rev. X **5**, 031026 (2015).

[27] Quantum annealers have also been studied outside the context of combinatorial optimization. See, e.g., Refs. [28–33].

[28] S. H. Adachi and M. P. Henderson, *Application of Quantum Annealing to Training of Deep Neural Networks*, arXiv:1510.06356.
[29] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, *Quantum Boltzmann Machine*, Phys. Rev. X **8**, 021050 (2018).
[30] M. Benedetti, J. Realpe-Gómez, R. Biswas, and A. Perdomo-Ortiz, *Quantum-Assisted Learning of Hardware-Embedded Probabilistic Graphical Models*, Phys. Rev. X **7**, 041052 (2017).
[31] A. Mott, J. Job, J.-R. Vlimant, D. Lidar, and M. Spiropulu, *Solving a Higgs Optimization Problem with Quantum Annealing for Machine Learning*, Nature (London) **550**, 375 (2017).
[32] R. Y. Li, R. Di Felice, R. Rohs, and D. A. Lidar, *Quantum Annealing versus Classical Machine Learning Applied to a Simplified Computational Biology Problem*, npj Quantum Inf. **4**, 14 (2018).
[33] A. D. King *et al.*, *Observation of Topological Phenomena in a Programmable Lattice of 1,800 Qubits*, arXiv:1803.02047.
[34] The D-Wave 2000q system, https://www.dwavesys.com/press-releases/d-wave-systems-previews-2000-qubit-quantum-system.
[35] We stress that our observation of an optimal annealing time is not simply due to advances in the quantum annealing hardware; we demonstrate optimal annealing times for these problem instances on the two most recent generations of D-Wave processors.
[36] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by Simulated Annealing*, Science **220**, 671 (1983).
[37] G. E. Santoro, R. Martoňák, E. Tosatti, and R. Car, *Theory of Quantum Annealing of an Ising Spin Glass*, Science **295**, 2427 (2002).
[38] S. W. Shin, G. Smith, J. A. Smolin, and U. Vazirani, *How "Quantum" Is the D-Wave Machine?*, arXiv:1401.7087.
[39] S. V. Isakov, G. Mazzola, V. N. Smelyanskiy, Z. Jiang, S. Boixo, H. Neven, and M. Troyer, *Understanding Quantum Tunneling through Quantum Monte Carlo Simulations*, Phys. Rev. Lett. **117**, 180402 (2016).
[40] Z. Jiang, V. N. Smelyanskiy, S. V. Isakov, S. Boixo, G. Mazzola, M. Troyer, and H. Neven, *Scaling Analysis and Instantons for Thermally Assisted Tunneling and Quantum Monte Carlo Simulations*, Phys. Rev. A **95**, 012322 (2017).
[41] S. Boixo, V. N. Smelyanskiy, A. Shabani, S. V. Isakov, M. Dykman, V. S. Denchev, M. H. Amin, A. Y. Smirnov, M. Mohseni, and H. Neven, *Computational Multiqubit Tunnelling in Programmable Quantum Annealers*, Nat. Commun. **7**, 10327 (2016).
[42] S. Muthukrishnan, T. Albash, and D. A. Lidar, *Tunneling and Speedup in Quantum Optimization for Permutation-Symmetric Problems*, Phys. Rev. X **6**, 031010 (2016).
[43] V. Choi, *Minor-Embedding in Adiabatic Quantum Computation: I. The Parameter Setting Problem*, Quantum Inf. Process. **7**, 193 (2008).
[44] P. I. Bunyk, E. M. Hoskinson, M. W. Johnson, E. Tolkacheva, F. Altomare, A. J. Berkley, R. Harris, J. P. Hilton, T. Lanting, A. J. Przybysz, and J. Whittaker, *Architectural Considerations in the Design of a Superconducting Quantum Annealing Processor*, IEEE Trans. Appl. Supercond. **24**, 1 (2014).
[45] M. B. Hastings and M. H. Freedman, *Obstructions to Classically Simulating the Quantum Adiabatic Algorithm*, Quantum Inf. Comput. **13**, 1038 (2013).
[46] T. Albash, V. Martin-Mayor, and I. Hen, *Temperature Scaling Law for Quantum Annealing Optimizers*, Phys. Rev. Lett. **119**, 110502 (2017).
[47] H. G. Katzgraber, F. Hamze, and R. S. Andrist, *Glassy Chimeras Could be Blind to Quantum Speedup: Designing Better Benchmarks for Quantum Annealing Machines*, Phys. Rev. X **4**, 021008 (2014).
[48] M. Steffen, W. van Dam, T. Hogg, G. Breyta, and I. Chuang, *Experimental Implementation of an Adiabatic Quantum Optimization Algorithm*, Phys. Rev. Lett. **90**, 067903 (2003).
[49] M. S. Sarandy and D. A. Lidar, *Adiabatic Quantum Computation in Open Systems*, Phys. Rev. Lett. **95**, 250503 (2005).
[50] E. Farhi, J. Goldstone, and S. Gutmann, *Quantum Adiabatic Evolution Algorithms versus Simulated Annealing*, arXiv:quant-ph/0201031.
[51] E. Farhi, J. Goldstone, and S. Gutmann, *Quantum Adiabatic Evolution Algorithms with Different Paths*, arXiv:quant-ph/0208135.
[52] G. Schaller and R. Schützhold, *The Role of Symmetries in Adiabatic Quantum Algorithms*, Quantum Inf. Comput. **10**, 0109 (2010).
[53] S. A. Owerre and M. B. Paranjape, *Macroscopic Quantum Tunneling and Quantum-Classical Phase Transitions of the Escape Rate in Large Spin Systems*, Phys. Rep. **546**, 1 (2015).
[54] T. Jörg, F. Krzakala, J. Kurchan, and A. C. Maggs, *Simple Glass Models and Their Quantum Annealing*, Phys. Rev. Lett. **101**, 147204 (2008).
[55] E. Andriyash and M. H. Amin, *Can Quantum Monte Carlo Simulate Quantum Annealing?*, arXiv:1703.09277.
[56] M. H. Amin, *Searching for Quantum Speedup in Quasistatic Quantum Annealers*, Phys. Rev. A **92**, 052323 (2015).
[57] D. Aharonov, A. Kitaev, and J. Preskill, *Fault-Tolerant Quantum Computation with Long-Range Correlated Noise*, Phys. Rev. Lett. **96**, 050504 (2006).
[58] P. J. D. Crowley and A. G. Green, *Anisotropic Landau-Lifshitz-Gilbert Models of Dissipation in Qubits*, Phys. Rev. A **94**, 062106 (2016).
[59] S. Boixo, T. Albash, F. M. Spedalieri, N. Chancellor, and D. A. Lidar, *Experimental Signature of Programmable Quantum Annealing*, Nat. Commun. **4**, 2067 (2013).
[60] T. Albash, T. F. Rønnow, M. Troyer, and D. A. Lidar, *Reexamining Classical and Quantum Models for the D-Wave One Processor*, Eur. Phys. J. Spec. Top. **224**, 111 (2015).
[61] T. Albash, W. Vinci, A. Mishra, P. A. Warburton, and D. A. Lidar, *Consistency Tests of Classical and Quantum Models for a Quantum Annealer*, Phys. Rev. A **91**, 042314 (2015).
[62] T. Lanting *et al.*, *Entanglement in a Quantum Annealing Processor*, Phys. Rev. X **4**, 021041 (2014).
[63] M. W. Johnson *et al.*, *Quantum Annealing with Manufactured Spins*, Nature (London) **473**, 194 (2011).
[64] G. Mazzola, V. N. Smelyanskiy, and M. Troyer, *Quantum Monte Carlo Tunneling from Quantum Chemistry to Quantum Annealing*, Phys. Rev. B **96**, 134305 (2017).

[65] S. P. Jordan, E. Farhi, and P. W. Shor, *Error-Correcting Codes for Adiabatic Quantum Computation*, Phys. Rev. A **74**, 052322 (2006).

[66] A. D. Bookatz, E. Farhi, and L. Zhou, *Error Suppression in Hamiltonian-Based Quantum Computation Using Energy Penalties*, Phys. Rev. A **92**, 022317 (2015).

[67] Z. Jiang and E. G. Rieffel, *Non-Commuting Two-Local Hamiltonians for Quantum Error Suppression*, Quantum Inf. Process. **16**, 89 (2017).

[68] M. Marvian and D. A. Lidar, *Error Suppression for Hamiltonian-Based Quantum Computation Using Subsystem Codes*, Phys. Rev. Lett. **118**, 030504 (2017).

[69] M. Marvian and D. A. Lidar, *Error Suppression for Hamiltonian Quantum Computing in Markovian Environments*, Phys. Rev. A **95**, 032302 (2017).

[70] B. Heim, T. F. Rønnow, S. V. Isakov, and M. Troyer, *Quantum versus Classical Annealing of Ising Spin Glasses*, Science **348**, 215 (2015).

[71] H. Rieger and N. Kawashima, *Application of a Continuous Time Cluster Algorithm to the Two-Dimensional Random Quantum Ising Ferromagnet*, Eur. Phys. J. B **9**, 233 (1999).

[72] A. W. Sandvik, *Stochastic Series Expansion Method for Quantum Ising Models with Arbitrary Interactions*, Phys. Rev. E **68**, 056701 (2003).

[73] T. Albash, G. Wagenbreth, and I. Hen, *Off-Diagonal Expansion Quantum Monte Carlo*, Phys. Rev. E **96**, 063309 (2017).

[74] T. Albash, S. Boixo, D. A. Lidar, and P. Zanardi, *Quantum Adiabatic Markovian Master Equations*, New J. Phys. **14**, 123016 (2012).

[75] K. W. Yip, T. Albash, and D. A. Lidar, *Quantum Trajectories for Time-Dependent Adiabatic Master Equations*, Phys. Rev. A **97**, 022116 (2018).

[76] S. Mandrà, H. G. Katzgraber, and C. Thomas, *The Pitfalls of Planar Spin-Glass Benchmarks: Raising the Bar for Quantum Annealers (Again)*, Quantum Sci. Technol. **2**, 038501 (2017).

[77] F. Hamze and N. de Freitas, in *UAI*, edited by D. M. Chickering and J. Y. Halpern (AUAI Press, Arlington, VA, 2004), pp. 243–250.

[78] A. Selby, *Efficient Subgraph-Based Sampling of Ising-Type Models with Frustration*, arXiv:1409.3934.

[79] J. Houdayer, *A Cluster Monte Carlo Algorithm for 2-Dimensional Spin Glasses*, Eur. Phys. J. B **22**, 479 (2001).

[80] Z. Zhu, A. J. Ochoa, and H. G. Katzgraber, *Efficient Cluster Algorithm for Spin Glasses in Any Space Dimension*, Phys. Rev. Lett. **115**, 077201 (2015).

[81] Y. Matsuda, H. Nishimori, and H. G. Katzgraber, *Quantum Annealing for Problems with Ground-State Degeneracy*, J. Phys. Conf. Ser. **143**, 012003 (2009).

[82] B. H. Zhang, G. Wagenbreth, V. Martin-Mayor, and I. Hen, *Advantages of Unfair Quantum Ground-State Sampling*, Sci. Rep. **7**, 1044 (2017).

[83] S. Mandrà, Z. Zhu, and H. G. Katzgraber, *Exponentially Biased Ground-State Sampling of Quantum Annealing Machines with Transverse-Field Driving Hamiltonians*, Phys. Rev. Lett. **118**, 070502 (2017).

[84] J. King, *Simulating a Quantum Annealer with GPU-Based Monte Carlo Algoritms*, in GTC **2016**, S6380 (2016).

[85] W. K. Hastings, *Monte Carlo Sampling Methods Using Markov Chains and Their Applications*, Biometrika **57**, 97 (1970).

[86] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *Equation of State Calculations by Fast Computing Machines*, J. Chem. Phys. **21**, 1087 (1953).

[87] U. Wolff, *Collective Monte Carlo Updating for Spin Systems*, Phys. Rev. Lett. **62**, 361 (1989).

[88] S. Bravyi and B. Terhal, *Complexity of Stoquastic Frustration-Free Hamiltonians*, SIAM J. Comput. **39**, 1462 (2009).

[89] A. Selby, *Prog-QUBO*, https://github.com/alex1770/QUBO-Chimera.

[90] J. Edmonds, *Paths, Trees, and Flowers*, Can. J. Math. **17**, 449 (1965).

[91] V. Kolmogorov, *Blossom V: A New Implementation of a Minimum Cost Perfect Matching Algorithm*, Math. Program. Comput. **1**, 43 (2009).

[92] L. Bieche, J. P. Uhry, R. Maynard, and R. Rammal, *On the Ground States of the Frustration Model of a Spin Glass by a Matching Method of Graph Theory*, J. Phys. A **13**, 2553 (1980).