Probing for quantum speedup in spin glass problems with planted solutions

Itay Hen,[1] Joshua Job,[2,3] Tameem Albash,[2,3] Troels F. Rønnow,[4] Matthias Troyer,[4] and Daniel Lidar*[2,3,5,6]

[1] *Information Sciences Institute, University of Southern California, Marina del Rey, CA 90292*
[2] *Department of Physics and Astronomy, University of Southern California, Los Angeles, California 90089, USA*
[3] *Center for Quantum Information Science & Technology, University of Southern California, Los Angeles, California 90089, USA*
[4] *Theoretische Physik, ETH Zurich, 8093 Zurich, Switzerland*
[5] *Department of Electrical Engineering, University of Southern California, Los Angeles, California 90089, USA*
[6] *Department of Chemistry, University of Southern California, Los Angeles, California 90089, USA**

(Dated: September 30, 2015)

The availability of quantum annealing devices with hundreds of qubits has made the experimental demonstration of a quantum speedup for optimization problems a coveted, albeit elusive goal. Going beyond earlier studies of random Ising problems, here we introduce a method to construct a set of frustrated Ising-model optimization problems with tunable hardness. We study the performance of a D-Wave Two device (DW2) with up to 503 qubits on these problems and compare it to a suite of classical algorithms, including a highly optimized algorithm designed to compete directly with the DW2. The problems are generated around predetermined ground-state configurations, called planted solutions, which makes them particularly suitable for benchmarking purposes. The problem set exhibits properties familiar from constraint satisfaction (SAT) problems, such as a peak in the typical hardness of the problems, determined by a tunable clause density parameter. We bound the hardness regime where the DW2 device either does not or might exhibit a quantum speedup for our problem set. While we do not find evidence for a speedup for the hardest and most frustrated problems in our problem set, we cannot rule out that a speedup might exist for some of the easier, less frustrated problems. Our empirical findings pertain to the specific D-Wave processor and problem set we studied and leave open the possibility that future processors might exhibit a quantum speedup on the same problem set.

I. INTRODUCTION

Interest in quantum computing is motivated by the potential for quantum speedup – the ability of quantum computers, aided by uniquely quantum features such as entanglement and tunneling, to solve certain computational problems in a manner that scales better with problem size than is possible classically. Despite tremendous progress in building small-scale quantum information processors [1], there is as of yet no conclusive experimental evidence of a quantum speedup. For this reason, there has been much recent interest in building more specialized quantum information processing devices that can achieve relatively large scales, such as quantum simulators [2] and quantum annealers [3, 4]. The latter are designed to solve classically hard optimization problems by exploiting the phenomenon of quantum tunneling [5–14]. Here we report on experimental results that probe the possibility that a putative quantum annealer may be capable of speeding up the solution of certain carefully designed optimization problems. We refer to this either as a *limited* or as a *potential* quantum speedup, since we study the possibility of an advantage relative to a portfolio of classical algorithms that either "correspond" to the quantum annealer (in the sense that they implement a similar algorithmic approach running on classical hardware), or implement a specific, specialized classical algorithm [15]. In addition, for technical reasons detailed below we must operate the putative quantum annealer in a suboptimal regime. With these caveats in mind, we push the experimental boundary in searching for a quantum advantage over a class of important classical algorithms, which includes simulated classical and quantum annealing [16, 17], using quantum hardware. We achieve this by designing Ising model problems that exhibit frustration, a well-known feature of classically-hard optimization problems [18, 19]. In doing so we go beyond the random spin-glass problems of earlier studies [15, 20], by ensuring that the problems we study exhibit a degree of hardness that we can tune, and have at least one "planted" ground state that we know in advance.

The putative quantum annealer used in our work is the D-Wave Two (DW2) device [21]. This device is designed to solve optimization problems by evolving a known initial configuration — the ground state of a transverse field $H_X = \sum_{i\in V} \sigma_i^x$, where σ_i^x is the Pauli spin-1/2 matrix acting on spin i — towards the ground state of a classical Ising-model Hamiltonian which serves as a cost function that encodes the problem that is to be solved:

$$H_{\rm Ising} = \sum_{(i,j)\in E} J_{ij}\sigma_i^z\sigma_j^z + \sum_{i\in V} h_i\sigma_i^z \ . \qquad (1)$$

The variables $\{\sigma_i^z\}$ denote either classical Ising-spin variables that take values ± 1 or Pauli spin-1/2 matrices, the $\{J_{ij}\}$ are programmable coupling parameters, and the $\{h_i\}$ are programmable local longitudinal fields. The N spin variables are realized as superconducting flux qubits and occupy the vertices V of a graph G with edge set

* lidar@usc.edu

E. Here G is the D-Wave "Chimera" hardware graph [21, 22]. Further details, including a visualization of the Chimera graph and the annealing schedule that interpolates between H_X and $H_{\rm Ising}$, are provided in Appendix A.

The dual questions of the computational power and underlying physics of the D-Wave devices — the 512-qubit DW2 and its predecessor, the 128-qubit DW1 — have generated a fair amount of interest and debate. A major concern is to what extent quantum effects determine the performance of these devices, given that they are inherently noisy and operate at temperatures (∼20 mK) where thermal effects are expected to play a significant role, causing decoherence, excitation and relaxation. Several studies have addressed these concerns [3, 20, 23–32]. Entanglement has been experimentally detected during the annealing process [33], and multiqubit tunneling involving up to 8 qubits has been demonstrated to play a functional role in determining the output of a DW2 device programmed to solve a simple non-convex optimization problem [34]. However, the role of quantum effects in determining the computational performance of the D-Wave devices on hard optimization problems involving many variables remains an open problem. A direct approach to try to settle the question is to demonstrate a quantum speedup. Such a demonstration has so far been an elusive goal, possibly because the random Ising problems chosen in previous benchmarking tests [15, 20] were too easy to solve for the classical algorithms against which the D-Wave devices were compared; namely such problems exhibit a spin-glass phase only at zero temperature [35]. The Sherrington-Kirkpatrick model with random ± 1 couplings, exhibiting a positive spin-glass critical temperature, was tested on a DW2 device, but the problem sizes considered were too small (due to the need to embed a complete graph into the Chimera graph) to test for a speedup [31]. The approach we outline next allows us to directly probe for a quantum speedup using frustrated Ising spin glass problems with a tunable degree of hardness, though we do not know whether these problems exhibit a positive-temperature spin-glass phase.

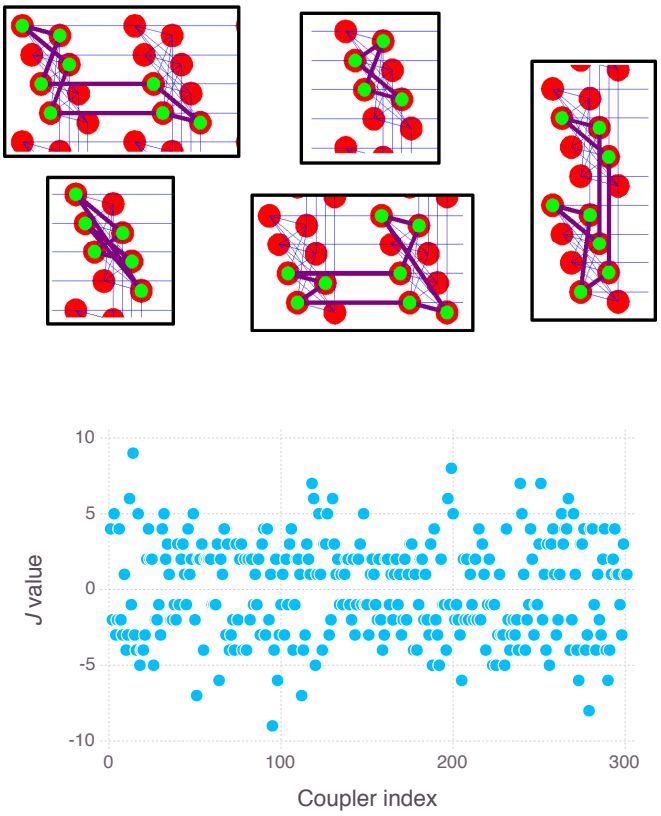


FIG. 1. **Examples of randomly generated loops and couplings on the DW2 Chimera graph.** Top: qubits and couplings participating in the loops are highlighted in green and purple, respectively. Only even-length loops are embeddable on the Chimera graph. Bottom: distribution of J values for a sample problem instance with $N = 126$ spins and edges, and 101 loops. It is virtually impossible to recover the loop-Hamiltonians H_j from a given $H_{\rm Ising}$. The couplings are all eventually rescaled to lie in $[-1, 1]$. We always set the local fields h_i to zero as non-zero fields tended to make the problems easier.

II. FRUSTRATED ISING PROBLEMS WITH PLANTED SOLUTIONS

In this section we introduce a method for generating families of benchmark problems that have a certain degree of "tunable hardness", achieved by adjusting the amount of frustration, a well-known concept from spin-glass theory [36]. In frustrated optimization problems no configuration of the variables simultaneously minimizes all terms in the cost function, often causing classical algorithms to get stuck in local minima, since the global minimum of the problem satisfies only a fraction of the Ising couplings and/or local fields. We construct our problems around "planted solutions" – an idea borrowed from constraint satisfaction (SAT) problems [37, 38]. The planted solution represents a ground state configuration of Eq. (1) that minimizes the energy and is known in advance. This knowledge circumvents the need to verify the ground state energy using exact (provable) solvers, which rapidly become too expensive computationally as the number of variables grows, and which were employed in earlier benchmarking studies [15, 20].

To create Ising-type optimization problems with planted solutions, we make use of frustrated "local Ising Hamiltonians" H_j, i.e., Ising problem instances defined on sub-graphs of the Chimera graph in lieu of the clauses appearing in the SAT formulas. The total Hamiltonian for each problem is then of the form $H_{\rm Ising} = \sum_{j=1}^{M} H_j$, where the sum is over the (possibly partially overlapping) local Hamiltonians. Similarly to SAT problems, the size

of these local Hamiltonians, or clauses, does not scale with the size of the problem. Moreover, to ensure that the planted solution is a ground state of the total Hamiltonian, we construct the clauses so that each is minimized by that portion of the planted solution that has support over the corresponding subgraph.[1] The planted solution is therefore determined prior to constructing the local Hamiltonians, by assigning random values to the bits on the nodes of the graph. The above process generates a Hamiltonian with the property that the planted solution is a simultaneous ground state of all the frustrated local Hamiltonians.[2]

The various clauses H_j can be generated in many different ways. This freedom allows for the generation of many different types of instances, and here we present one method. An N-qubit M-clause instance is generated as follows.

1. A random configuration of N bits corresponding to the participating spins of the Chimera graph is generated. This configuration constitutes the planted solution of the instance.

2. M random loops are constructed along the edges of the Chimera graph. The loops are constructed by placing random walkers on random nodes of the Chimera graph, where each edge is determined at random from the set of all edges connected to the last edge. The random walk is terminated once the random walker crosses its path, i.e., when a node that has already been visited is encountered again. If this node is not the origin of the loop the "tail" of the path is discarded. Examples of such loops are given in Fig. 1. We distinguish between "short loops" of length $\ell = 4, 6$, and "long loops" of length $\ell \geq 8$, as these give rise to peaks in hardness at different loop densities. Here we focus on long loops; results for short loops, which tend to generate significantly harder problem instances, will be presented elsewhere.

3. On each loop, a clause H_j is defined by assigning $J_{ij} = \pm 1$ couplings to the edges of the loop in such a way that the planted solution minimizes H_j. As a first step, the J_{ij}'s are set to the ferromagnetic $J_{ij} = -s_i s_j$, where the s_i are the planted solution values. One of the couplings in the loop is then chosen at random and its sign is flipped. This ensures that no spin configuration can satisfy every edge in that loop, and the planted solution remains a global minimum of the loop, but is now a frustrated ground state.[3]

4. The total Hamiltonian is then formed by adding up the M-loop clauses H_j. Note that loops can partially overlap, thereby also potentially "canceling out" each other's frustration, a useful feature that will give rise to an easy-hard-easy pattern we discuss below. Since the planted solution is a ground state of each of the H_j's, it is also a ground state of the total Hamiltonian $H_{\rm Ising}$.

Ising-type optimization problems with planted solutions, such as those we have generated, have several attractive properties that we utilize later on: i) Having a ground-state configuration allows us to readily precompute a measure of frustration, e.g., the fraction of frustrated couplings of the planted solution. We shall show that this type of measure correlates well with the hardness of the problem, as defined in terms of the success probability of finding the ground state or the scaling of the time-to-solution. ii) By changing the number of clauses M we can create different classes of problems, each with a "clause density" $\alpha = M/N$, analogous to problem generation in SAT.[4] The clause density can be used to tune through a SAT-type phase transition [40], i.e., it may be used to control the hardness of the generated problems. Here too, we shall see that the clause density plays an important role in setting the hardness of the problems. Note that when the energy is unchanged under a spin-flip the solution is degenerate, so that our planted solution need not be unique.

III. ALGORITHMS AND SCALING

A judicious choice of classical algorithms is required to ensure that our test of a limited or potential quantum speedup is meaningful. We considered (i) simulated annealing (SA), a well-known, powerful and generic heuristic solver [16]; (ii) simulated quantum annealing (SQA) [8–11], a quantum Monte Carlo algorithm that has been shown to be consistent with the D-Wave devices [20, 32];

[1] To see that the planted solution minimizes the total Hamiltonian, assume that a configuration x_* is a minimum of $f_j(x)$ for all j, where $\{f_j(x)\}$ is a set of arbitrary real-valued functions. Then, by definition, for each j, $f_j(x) \geq f_j(x_*)$ for all possible configurations x. Let us now define $f(x) \equiv \sum_j f_j(x)$. It follows then that $f(x) \geq \sum_j f_j(x_*)$. Since also $f(x_*) = \sum_j f_j(x_*)$, x_* is a minimizing configuration of $f(x)$.

[2] Somewhat confusingly from our perspective of utilizing frustration, such Hamiltonians are sometimes called "frustration-free" [39].

[3] To see that there can be no spin configuration with an energy lower than that of the planted solution, consider a given loop and a given spin in that loop; note that every spin participates in two couplings. Either both couplings are satisfied after the sign flip, or one is satisfied and the other is not. Correspondingly, flipping that spin will thus either raise the energy or leave it unchanged. This is true for all spins in the loop.

[4] Note that for small values of M, the number of spins actually participating in an instance will be smaller than N, the number of spins on the graph from which the clauses are chosen.

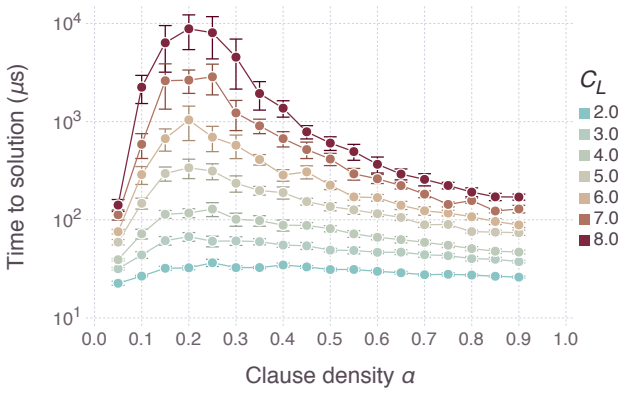
(iii) the Shin-Smolin-Smith-Vazirani (SSSV) thermal rotor model, that was specifically designed to mimic the D-Wave devices in their classical limit [26]; (iv) the Hamze-Freitas-Selby (HFS) algorithm [41, 42], an algorithm that is fine-tuned for the Chimera graph and appears to be the most competitive at this time. Of these, the HFS algorithm is the only one that is designed to exploit the scaling of the treewidth of the Chimera graph (see Appendix A) , which renders it particularly efficient in a comparison against the D-Wave devices [43].

The D-Wave devices and all the algorithms we considered are probabilistic, and return the correct ground state with some probability of success. We thus perform many runs of the same duration τ for a given problem instance, and estimate the success probability empirically as the number of successes divided by the number of runs. This is repeated for many instances at a given clause density α and number of variables N, and generates a distribution of success probabilities. Let $p(\lambda)$ denote the success probability for a given set of parameters $\lambda = \{N, \alpha, q\}$, where q denotes the qth percentile of this distribution; e.g., half the instances for given N and α have a higher empirical success probability than the median $p(N, \alpha, 0.5)$. The *number of runs* required to find the ground state at least once with a desired probability p_d is [15, 20][5]

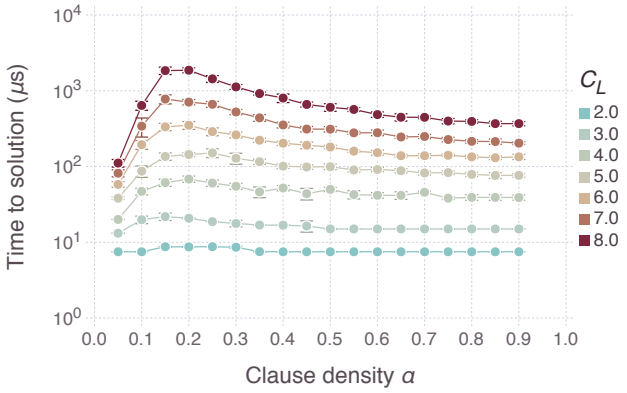
$$r(\lambda) = \frac{\log(1-p_d)}{\log(1-p(\lambda))} \ , \tag{2}$$

and henceforth we set $p_d = 0.99$. Correspondingly, the *time-to-solution* is $\mathrm{TTS}(\lambda) = r(\lambda)\tau$, where for the D-Wave device τ signifies the annealing time t_a (at least 20μs), while for SA, SQA, and SSSV τ is the number of Monte Carlo sweeps s (a complete update of all spins) multiplied by the time per sweep τ_X for algorithm X.[6] For SA, we further distinguish between using it as a *solver* (SAS) or as an *annealer* (SAA): in SAS mode we keep track of the energies found along the annealing schedule (which we take to be linear in the inverse temperature β) and take the lowest, while in SAA mode we always use the final energy found. Thus SAA can never be better than SAS, but is a more faithful model of an analog annealing device. A similar distinction can be made for SQA (i.e., SQAA and SQAS), but we primarily consider the annealer version since it too more closely mimics the operation of DW2 (note that SAA and SQAA were also the modes used in Refs. [15, 20]; SQAS results are shown in Appendix B). For the HFS algorithm, $\mathrm{TTS}(\lambda)$ is calculated directly from the distribution of runtimes obtained from 10^5 identical, independent executions of the algorithm. Further timing details are given in Appendix A.

We next briefly discuss how to properly address resource scaling [15]. The D-Wave devices use N qubits and $O(N)$ couplers to compute the solution of the input problem. Thus, it uses resources which scale linearly in the size of the problem, and we should give our classical solvers the same opportunity. Namely, we should allow for the use of linearly more cores and memory for our classical algorithms as we scale the problem size. For annealers such as SA, SQA, and SSSV, this is trivial as we can exploit the bipartite nature of the Chimera graph to perform spin updates in parallel so that each sweep takes constant time and a linear number of cores and memory. The HFS algorithm is not perfectly parallelizable ~~and~~ but as we explain in more detail in Appendix A, we ~~need to divide the TTS we measure for HFS run on a CPU by $L = \sqrt{N/8}$ for a problem with N qubits~~take



(a) DW2



(b) HFS

FIG. 2. **Time to solution as a function of clause density.** Shown is $\mathrm{TTS}(L, \alpha, 0.5)$ (log scale) for (a) DW2 and (b) HFS, as a function of the clause density. The different colors represent the different Chimera subgraph sizes, which continue to $L = 12$ in the HFS case. In both cases there is a clear peak. From the HFS results we can identify the peak position as being at $\alpha = 0.17 \pm 0.01$, which is consistent with the peak position in the DW2 results. Error bars represent ~~1σ~~ 2σ confidence intervals.

[5] We prefer to define r in this manner, rather than rounding it as in [15, 20], as this simplifies the extraction of scaling coefficients.

[6] In our simulations $\tau_{\mathrm{SA}} = 3.54\mu$s, $\tau_{\mathrm{SQA}} = 9.92\mu$s, and $\tau_{\mathrm{SSSV}} = 10.34\mu$s.

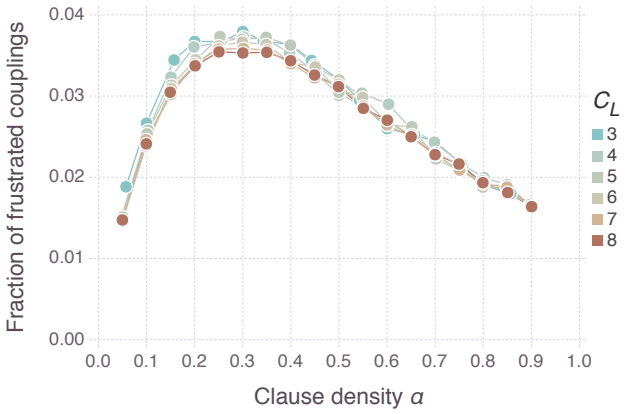


FIG. 3. **Frustration fraction.** Shown is the fraction of frustrated couplings (the number of frustrated couplings divided by the total number of couplings, where a frustrated coupling is defined with respect to the planted solution) as a function of clause density for different Chimera subgraphs C_L, in the case of loops of length ≥ 8, averaged over the 100 instances for each given α and N. There is a broad peak at $\alpha \approx 0.25$. This is the clause density at which there is the largest fraction of frustrated couplings, and is near where we expect the hardest instances to occur, in good agreement with Fig. 2.

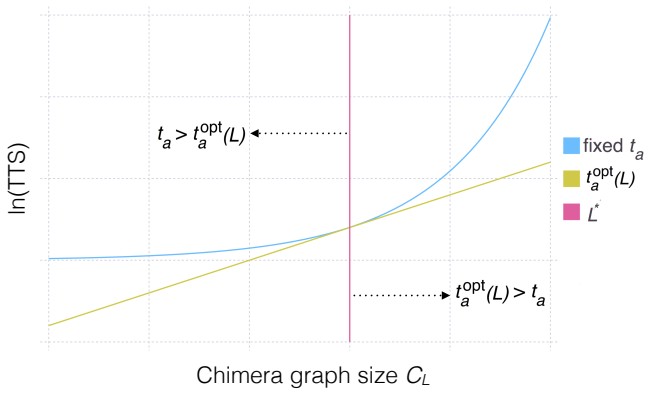


FIG. 4. **Sketch of the relation between the** log(**TTS**) **curves for optimal and suboptimal annealing times.** The annealing time needs to be optimized for each problem size. Blue represents the TTS with a size-independent annealing time t_a. Red represents the optimal TTS corresponding to having an optimal size-dependent annealing time $t_a^{\rm opt}(L)$, i.e. the lower envelope of the full series of fixed annealing time TTS curves. This curve need not be linear as depicted, though we expect it to be linear for NP-hard problems. The blue line upper bounds the red line since by definition $\mathrm{TTS}_{\rm DW2}(\lambda, t_a^{\rm opt}(L)) \leq \mathrm{TTS}_{\rm DW2}(\lambda, t_a)$. The vertical dotted line represents the problem size L^* at which $t_a = t_a^{\rm opt}(L^*)$. To the left of this line $t_a > t_a^{\rm opt}$ and the slope of the fixed-t_a TTS curve lower-bounds the slope of the optimal TTS curve, since for very small problem sizes a large t_a results in insensitivity to problem size, and the success probability is essentially constant. The opposite happens to the right of this line, where $t_a < t_a^{\rm opt}$, and where the success probability rapidly drops with L at fixed t_a.

this into account as well. Finally, note that dynamic programming can always find the true ground state in a time that is exponential in the Chimera graph treewidth, i.e., that scales as $\exp(c\sqrt{N})$ [22]. The natural size scale in our study is the square Chimera subgraph C_L comprising L^2 unit cells of 8 qubits each, i.e., $N = 8L^2$. Therefore, henceforth we replace N by L so that $\lambda = \{L, \alpha, q\}$.

IV. PROBING FOR A QUANTUM SPEEDUP

We now come to our main goal, which is to probe for a limited or potential quantum speedup on our frustrated Ising problem set. We reserve the term "limited speedup" for our comparisons with SA, SQA, and SSSV, while the term "potential speedup" refers to the comparison with the HFS algorithm, which unlike the other three algorithms, does not implement a similar algorithmic approach to a quantum annealer.

A. Dependence on clause density

We first analyze the effect of the clause density. This is shown in Fig. 2, where we plot $\mathrm{TTS}(L, \alpha, 0.5)$ for the DW2 and the HFS algorithm.[7] We note that the worst-case TTS of ~ 10ms is smaller than that observed for random Ising problems in Ref. [15] [$\sim$100ms for range 7, C_8, and $q = 0.5$ (median)]. However, as we shall demonstrate, the classical algorithms against which the DW2 was benchmarked in Ref. [15] (SA and SQA) scale significantly less favorably in the present case, i.e., whereas in Ref. [15] no possibility of a limited speedup against SA was observed for the median, here we will find that such a possibility remains. In this sense, the problem instances considered here are relatively harder for the classical solvers than those of Ref. [15].

For our choice of random loop characteristics, the time-to-solution peaks at a clause density $\alpha \approx 0.17$, reflecting the hardness of the problems in that regime. To correlate the hardness of the instances with their degree of frustration, we plot the frustration fraction, defined as the ratio of the number of unsatisfied edges with respect to the planted solution to the total number of edges on the graph, as a function of clause density. The frustration fraction curve, shown in Fig. 3, has a peak at $\alpha \approx 0.25$, confirming that frustration and hardness are

[7] In this study we focus mostly on the median since with 100 instances per setting, higher quantiles tend be noise-dominated.

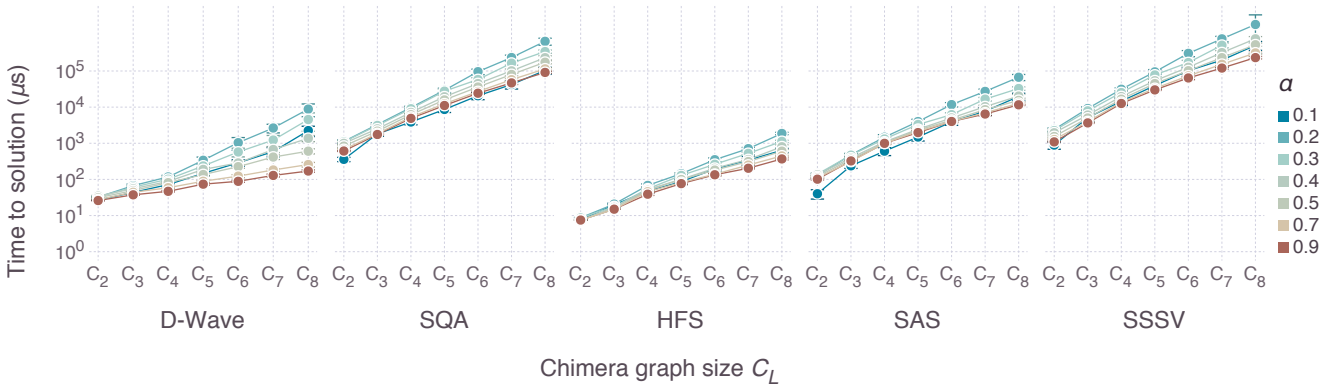


FIG. 5. **Median time-to-solution over instances.** Plotted is TTS$(L, \alpha, 0.5)$ (log scale) as a function of the Chimera subgraph size C_L, for a range of clause densities and for all solvers we tested. Note that only the scaling matters and not the actual TTS, since it is determined by constant factors that vary from processor to processor, compiler options, etc. ~~The SQA, SSSV, and SA algorithm's TTS have parallelism already taken into account;~~ All algorithms timing reflects the ~~HFS algorithm TTS was divided by $\sqrt{N/8}$ to account~~ result after accounting for ~~partial parallelization~~parallelism, as described in Appendix A. Error bars represent ~~1σ~~ 2σ confidence intervals. The DW2 annealing time is $t_a = 20\mu$s.

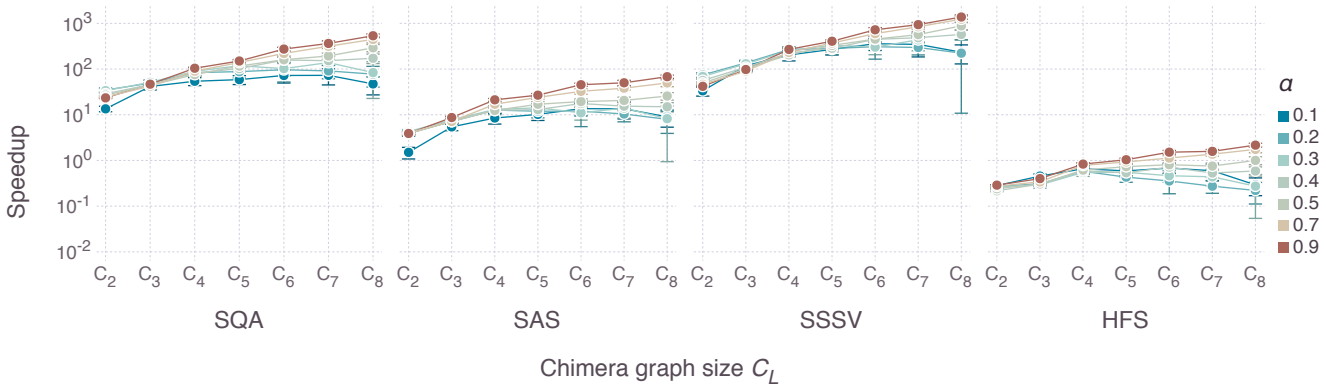


FIG. 6. **Speedup ratio.** Plotted is the median speedup ratio $S_X(L, \alpha, 0.5)$ (log scale) as defined in Eq. (3) for all algorithms tested. A negative slope indicates a definite slowdown for the DW2. A positive slope indicates the possibility of an advantage for the DW2 over the corresponding classical algorithm. This is observed for $\alpha > 0.4$ in the comparison to SAS, SQA~~and~~, SSSV, and ~~for $\alpha > 0.7$ against the~~ HFS ~~algorithm~~ (see Fig. 10 for a more detailed analysis). Error bars represent ~~1σ~~ 2σ confidence intervals.

indeed correlated.[8] The hardness peak is reminiscent of the analogous situation in SAT, where the clause density can be tuned through a phase transition between satisfiable and unsatisfiable phases [40]. The peak we observe may be interpreted as a finite-size precursor of a phase transition. This interpretation is corroborated below by time-to-solution results of all other tested algorithms which will also find problems near the critical point the hardest. Indeed, all the algorithms we studied exhibit qualitatively similar behavior to that seen in Fig. 2 (see Fig. 17 in Appendix B), with an easy-hard-easy pattern separated at $\alpha \approx 0.2$. This is in agreement with previous studies, e.g., for MAX 2-SAT problems on the DW1 [44], and for k-SAT with $k > 2$, where a similar pattern is found for backtracking solvers [45]. It is important to note that we do not claim that this easy-hard-easy

[8] Note that in our definition of frustration fraction, frustration is measured with respect to all edges of the graph from which clauses are chosen, similarly to the way clause density is defined in SAT problems.

transition coincides with a spin-glass phase transition; we have not studied which phases actually appear in our problem set as we tune the clause density.

A qualitative explanation for the easy-hard-easy pattern is that when the number of loops (and hence α) is small they do not overlap and thus each loop becomes an easy optimization problem. In the opposite limit many loops pass through each edge, thus tending to reduce frustration, since each loop contributes either a "frustrated" edge with small probability $1/\ell$ (where ℓ is the loop length) or an "unfrustrated" edge with probability $1-1/\ell$. The hard problems thus lie in between these two limits, where a constant fraction (bounded away from 0 and 1) of loops overlap.

B. General considerations concerning scaling and speedup

Of central interest is the question of whether there is any scaling advantage (with L) in using a quantum device to solve our problems. Therefore, we define a quantum speedup ratio for the DW2 relative to a given algorithm X as [15]

$$S_X(\lambda, t_a) = \frac{\mathrm{TTS}_X(\lambda)}{\mathrm{TTS}_{\mathrm{DW2}}(\lambda, t_a)} \ . \tag{3}$$

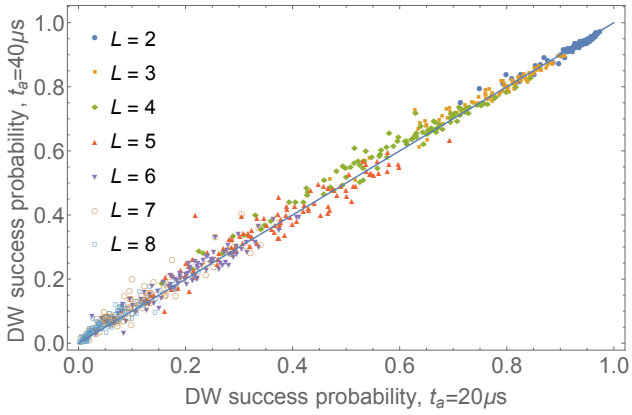
Since this quantity is specific to the DW2 we refer to it simply as the empirical "DW2 speedup ratio" from now on, though of course it generalizes straightforwardly for any other putative quantum annealer or processor against which algorithm X is compared.

We must be careful in using $S_X(\lambda, t_a)$ in assessing a speedup, since the annealing time must be optimized for each problem size L in order to avoid the pitfall of a fake speedup [15]. Let us denote the (unknown) optimal annealing time by $t_a^{\mathrm{opt}}(L)$. By definition, $\mathrm{TTS}_{\mathrm{DW2}}(\lambda, t_a^{\mathrm{opt}}(L)) \leq \mathrm{TTS}_{\mathrm{DW2}}(\lambda, t_a)$, where t_a is a fixed annealing time. We now define the *optimized speedup ratio* as

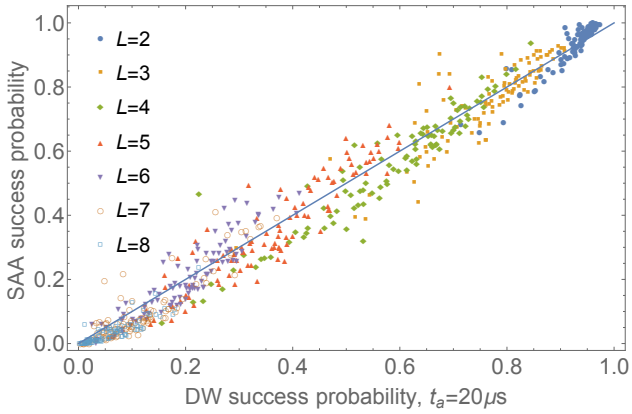
$$S_X^{\mathrm{opt}}(\lambda, t_a^{\mathrm{opt}}(L)) = \frac{\mathrm{TTS}_X(\lambda)}{\mathrm{TTS}_{\mathrm{DW2}}(\lambda, t_a^{\mathrm{opt}}(L))} \tag{4}$$

and clearly $S_X(\lambda, t_a) \leq S_X^{\mathrm{opt}}(\lambda, t_a^{\mathrm{opt}}(L))$, i.e., the speedup ratios computed using Eq. (3) are lower bounds on the optimized speedup. However, what matters for a speedup is the *scaling* of the speedup ratio with problem size. Thus, we are interested not in the numerical value of the speedup ratio but rather in the slope dS/dL (recognizing that this is a formal derivative since L is a discrete variable). A positive slope would indicate a DW2 speedup, while a negative speedup slope would indicate a slowdown.

We thus define the *DW2 speedup regime* as the set of problem sizes $\mathcal{L}^+$ where $\frac{d}{dL} S_X(\lambda, t_a^{\mathrm{opt}}(L)) > 0$ for all $L \in \mathcal{L}^+$. Likewise, the *DW2 slowdown regime* is the set of problem sizes $\mathcal{L}^-$ where $\frac{d}{dL} S_X(\lambda, t_a^{\mathrm{opt}}(L)) < 0$ for all $L \in \mathcal{L}^-$.



(a)



(b)

FIG. 7. **Success probability correlations.** The results for all instances at $\alpha = 0.35$ are shown. Each datapoint is the success probability for the same instance, (a) for DW2 at $t_a = 20\mu$s and $t_a = 40\mu$s, (b) for DW2 at $t_a = 20\mu$s and SAA at 50,000 sweeps and $\beta_f = 5$ [in dimensionless units, such that $\max(J_{ij}) = 1$]. Perfect correlation means that all data points would fall on the diagonal, and a strong correlation is observed in both cases. The data is colored by the problem size L and shows a clear progression from high success probabilities at small L to large success probabilities at small L. Qualitatively similar results are seen for $t_a = 20\mu$s *vs* $t_a = 40\mu$s at all α values, and for DW2 *vs* SAA at intermediate α values (see Figs. 22 and 23 in Appendix B).

From a computational complexity perspective one is ultimately interested in the asymptotic performance, i.e., the regime where L becomes arbitrarily large. In this sense a *true* speedup would correspond to the observation that $\mathcal{L}^+ = [L^+_{\mathrm{min}}, L^+_{\mathrm{max}}]$, with L^+_{min} a positive constant and $L^+_{\mathrm{max}} \to \infty$. Of course, such a definition is meaningless for a physical device such as the DW2, for which L^+_{max} is necessarily finite. Thus the best we can hope for is an observation that $\mathcal{L}^+$ is as large as is consistent with the device itself, which in our case would imply that $\mathcal{L}^+ = [1, 8]$. However, as we shall argue, we can in

fact only rule out a speedup, while we are unable to confirm one. I.e., we are able to identify $\mathcal{L}^- = [L^-_{\min}, L^-_{\max}]$, but not $\mathcal{L}^+$.

The culprit, as in earlier benchmarking work [15] and as we establish below, is the fact that the DW2 minimum annealing time of $t_a = 20\mu$s is too long (see also Appendix B). This means that the smaller the problem size the longer it takes to solve the corresponding instances compared to the case with an optimized annealing time, and hence the observed slope of the DW2 speedup ratio should be interpreted as a *lower bound* for the optimal scaling. This is illustrated in Fig. 4. Without the ability to identify $t_a^{\rm opt}(L)$ we do not know of a way to infer, or even estimate $\mathcal{L}^+$. However, as we now demonstrate, under a certain reasonable assumption we can still *bound* $\mathcal{L}^-$.

The assumption is that if $t_a > t_a^{\rm opt}$ then $\frac{d}{dL}\mathrm{TTS}_{\rm DW2}(\lambda, t_a) \leq \frac{d}{dL}\mathrm{TTS}_{\rm DW2}(\lambda, t_a^{\rm opt}(L))$ for all $L < L^*$, the problem size for which $t_a = t_a^{\rm opt}(L^*)$. This assumption is essentially a statement that the TTS is monotonic in L [9], as illustrated in Fig. 4. Next we consider the (formal) derivatives of Eqs. (3) and (4):

$$\frac{\frac{d}{dL}S_X(\lambda, t_a)}{S_X(\lambda, t_a)} = \frac{\partial_L \mathrm{TTS}_{\rm X}(\lambda)}{\mathrm{TTS}_{\rm X}(\lambda)} - \frac{\partial_L \mathrm{TTS}_{\rm DW2}(\lambda, t_a)}{\mathrm{TTS}_{\rm DW2}(\lambda, t_a)} \tag{5a}$$

$$\frac{\frac{d}{dL}S_X(\lambda, t_a^{\rm opt}(L))}{S_X(\lambda, t_a^{\rm opt}(L))} = \frac{\partial_L \mathrm{TTS}_{\rm X}(\lambda)}{\mathrm{TTS}_{\rm X}(\lambda)} - \frac{\frac{d}{dL}\mathrm{TTS}_{\rm DW2}(\lambda, t_a^{\rm opt}(L))}{\mathrm{TTS}_{\rm DW2}(\lambda, t_a^{\rm opt}(L))}\,. \tag{5b}$$

Collecting these results we have

$$\frac{S_X(\lambda, t_a)}{S_X(\lambda, t_a^{\rm opt})}\frac{d}{dL}S_X(\lambda, t_a^{\rm opt}) < \frac{d}{dL}S_X(\lambda, t_a) \text{ if } t_a > t_a^{\rm opt}(L). \tag{6}$$

Therefore, if we find that $\frac{d}{dL}S_X(\lambda, t_a) < 0$ in the suboptimal regime where $t_a > t_a^{\rm opt}(L)$, then it follows that $\frac{d}{dL}S_X(\lambda, t_a^{\rm opt}(L)) < 0$. In other words, *a DW2 speedup is ruled out if we observe a slowdown using a suboptimal annealing time.*

C. Scaling and speedup ratio results

In Fig. 5 we show the scaling of the median time-to-solution for all algorithms studied, for a representative set of clause densities. All curves appear to match the general dynamic programming scaling for $L \gtrsim 4$, i.e., $\mathrm{TTS}(\lambda) \sim \exp[b(\alpha)L]$, but the scaling coefficient $b(\alpha)$ clearly varies from solver to solver. This scaling is similar to that observed in previous benchmarking studies of random Ising instances [15, 20].

[9] Individual instances may not satisfy this assumption [46, 47], but we are not aware of any cases where averaging over an ensemble of instances violates this assumption.

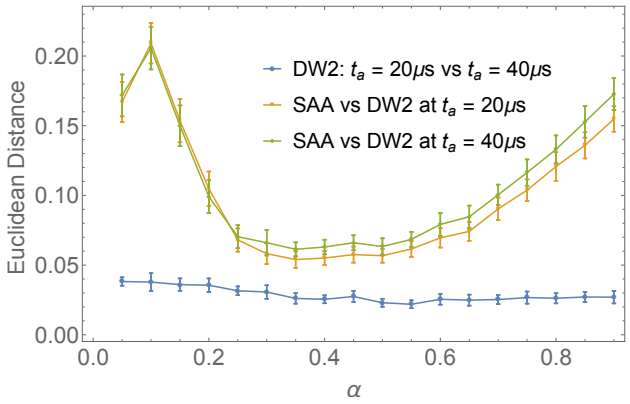


FIG. 8. **Correlation between the DW2 data for two different annealing times and SAA.** Plotted is the normalized Euclidean distance $D(\vec{p}_1, \vec{p}_2)$ for $\vec{p}_1$ and $\vec{p}_2$ being, respectively, the ordered success probability for DW2 at $t_a = 20\mu$s and $t_a = 40\mu$s (blue circles), DW2 at $t_a = 20\mu$s and SAA (yellow squares), DW2 at $t_a = 40\mu$s and SAA (green diamonds). For comparison, the Euclidean distance between two random vectors with elements $\in [0, 1]$ is ~ 0.4. SAA data is for 50,000 sweeps and $\beta_f = 5$. The correlation with SAA degrades slightly for $t_a = 40\mu$s. Error bars represent 2σ confidence intervals and were computed using bootstrapping (see Appendix A 3 for details). In each comparison, to construct $\vec{p}_1$ and $\vec{p}_2$ we fixed α and used half the instances (for bootstrapping purposes) for $L \in [2, 8]$.

In Fig. 6 we show the median scaling of S_X for the same set of clause densities as shown in Fig. 5. We observe that in all cases there is a strong dependence on the clause density α, with a negative slope of the DW2 speedup ratio for the lower clause densities, corresponding to the harder, more frustrated problems. In this regime the DW2 exhibits a scaling that is worse than the classical algorithms and by Eq. (6) there is no speedup. The possibility of a DW2 speedup remains open for the higher clause densities, where a positive slope is observed, i.e., the DW2 appears to find the easier, less frustrated problems easier than the classical solvers. This apparent advantage is most pronounced for ~~$\alpha > 0.7$~~$\alpha \geq 0.4$, where we observe the possibility of a potential speedup even against the highly fine-tuned HFS algorithm ~~.~~ (this is seen more clearly in Fig. 10). Moreover, the DW2 speedup ratio against HFS improves slightly at the higher percentiles (see Fig. 21 in Appendix B), which is encouraging from the perspective of a potential quantum speedup.

D. Scaling coefficient results

To test the dependence on t_a, we repeated our DW2 experiments for $t_a \in [20, 40]\mu$s, in intervals of 2μs. Figure 7(a) is a success probability correlation plot between

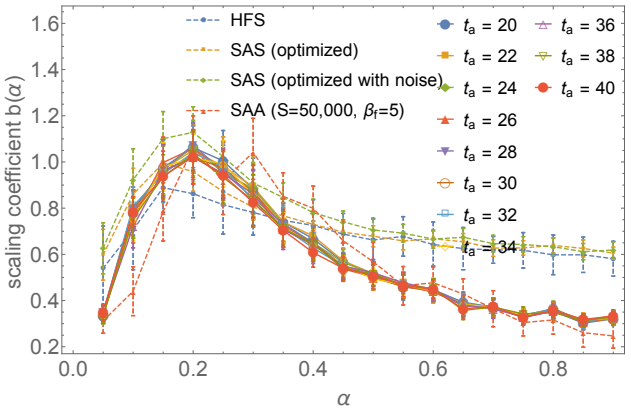


FIG. 9. **Scaling coefficients of the number of runs.** Plotted here is $b(\alpha)$ [Eq. (8)] for the DW2 at all annealing times (overlapping solid lines and large symbols), for the HFS algorithm, for SAS with an optimal number of sweeps, for SAS with noise and an optimal number of sweeps, and for SAA with a large enough number of sweeps that the asymptotic distribution has been reached at $\beta_f = 5$. The scaling coefficients of HFS and of optimized SAS each set an upper bound for a DW2 speedup against that particular algorithm. In terms of the scaling coefficient the DW2 result is statistically indistinguishable (except at $\alpha = 0.1$) from SAA run at $S = 50{,}000$ and $\beta_f = 5$. The coefficients shown here are extracted from fits with $L \geq 4$ (see Fig. 26(a) in Appendix B). Error bars represent 2σ confidence intervals.

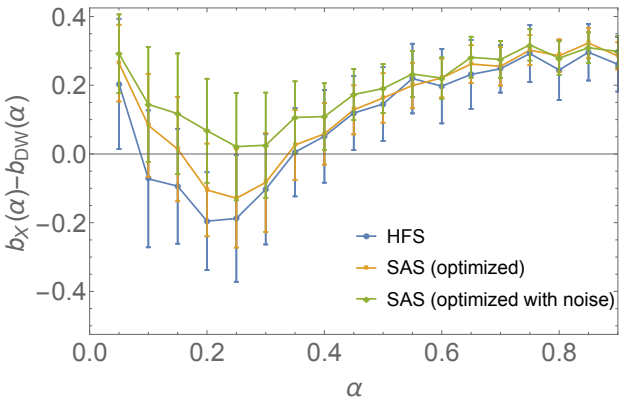


FIG. 10. **Difference between the scaling coefficients.** Plotted here is the difference between the scaling coefficients in Fig 9, $b_X(\alpha) - b_{DW2}(\alpha)$, where X denotes the HFS algorithm, SAS with an optimal number of sweeps, SAS with noise and an optimal number of sweeps, or SAA with $S = 50,000$ and $\beta_f = 5$. When the difference is non-positive there can be no speedup since optimizing t_a can only increase $b_{DW2}(\alpha)$; conversely, when the difference is positive a speedup is still possible, i.e., not accounting for the error bars, for ~~$\alpha \geq .55$~~ $\alpha \leq 0.05$ and $\alpha \geq 0.4$ for HFS, and for ~~$\alpha < 0.15$~~ $\alpha \leq 0.15$ and ~~$\alpha > 0.35$~~ $\alpha \geq 0.35$ for SAS without noise. These ranges shrink if the error bars are accounted for, but notably, for most α values SAS with noise does not disallow a limited speedup, suggesting that control noise may play an important factor in masking a DW2 speedup. Error bars represent 2σ confidence intervals.

$t_a = 20\mu$s and $t_a = 40\mu$s, at $\alpha = 0.35$. The correlation appears strong, suggesting that the device might already have approached the asymptotic regime where increasing t_a does not modify the success probabilities. To check this more carefully let us first define the normalized Euclidean distance between two length-M vectors of probabilities $\vec{p}_1$ and $\vec{p}_2$ as

$$D(\vec{p}_1, \vec{p}_2) := \frac{1}{M}\|\vec{p}_1 - \vec{p}_2\| \tag{7}$$

(where $\|\vec{p}\| = \sqrt{\vec{p}\cdot\vec{p}}$); clearly, $0 \leq D(\vec{p}_1, \vec{p}_2) \leq 1$. The result for the DW2 data with $\vec{p}_1$ and $\vec{p}_2$ being the ordered sets of success probabilities for all instances with given α at $t_a = 20\mu$s and $t_a = 40\mu$s respectively, is shown as the blue circles in Fig. 8. The small distance for all α suggests that for $t_a \geq 20\mu$s the distribution of ground state probabilities has indeed nearly reached its asymptotic value.

This result means that the number of runs $r(\lambda)$ [Eq. (2)] does not depend strongly on t_a either. To demonstrate this we first fit the number of runs to

$$r(L, \alpha, 0.5) = e^{a(\alpha)+b(\alpha)L}, \tag{8}$$

in accordance with the abovementioned expectation of the scaling with the treewidth, and find a good fit across the entire range of clause densities (see Fig. 26(a) in Appendix B). The corresponding scaling coefficient $b(\alpha)$ is plotted in Fig. 9 for all annealing times (the constant $a(\alpha)$ is shown in Fig. 26(b) in Appendix B and is well-behaved); the data collapses nicely, showing that the scaling coefficient $b(\alpha)$ has already nearly reached its asymptotic value. Also plotted in Fig. 9 is the scaling coefficient $b(\alpha)$ for HFS and for SAS with an optimized numbers of sweeps at each α and L, as extracted from the data shown in Fig. 5.

By Eq. (6), where $b_{DW2}(\alpha) \geq b_X(\alpha)$ there is no DW2 speedup ($\mathcal{L}^- = [4, 8]$), whereas where $b_{DW2}(\alpha) < b_X(\alpha)$, a DW2 speedup over algorithm X is still possible.

We thus plot the difference in the scaling coefficients in Fig. 10. Figures 9 and 10 do allow for the possibility of a speedup against both HFS and SAS, at sufficiently high clause densities. However, we stress once more that the smaller DW2 scaling coefficient may be a consequence of $t_a = 20\mu$s being excessively long, and that we cannot rule out that the observed regime of a possible speedup would have disappeared had we been able to optimize t_a by exploring annealing times shorter than 20μs. Note that an optimization of the SAS annealing schedule might further improve its scaling, but the same cannot be said of the HFS algorithm, and it seems unlikely that it could be outperformed even by the fully optimized version of SAS.

E. DW2 *vs* SAA

Earlier work has ruled out SAA as a model of the D-Wave devices for random Ising model problems [20], as well as for certain Hamiltonians with suppressed and enhanced ground states for which quantum annealing and SAA give opposite predictions [23, 27]. The observation made above, that the DW2 success probabilities have nearly reached their asymptotic values, suggests that for the problems studied here the DW2 device may perhaps be described as a thermal annealer. While we cannot conclude on the basis of ground state probabilities alone that the DW2 state distribution has reached the Gibbs state, we can compare the ground state distribution to that of SAA in the regime of an asymptotic number of sweeps, and attempt to identify an effective final temperature for the classical annealer that allows it to closely describe the DW2 distribution. We empirically determine $\beta_f = 5$ to be the final temperature for our SAA simulations that gives the closest match, and $S = 50{,}000$ (corresponding to 150ms, much larger than the DW2's $t_a = 20\mu$s) to be large enough for the SAA distribution to have become stationary (see Fig. 27(a) in Appendix B for results with additional sweep numbers confirming this). The result for the Euclidean distance measure is shown in Fig. 8 for the two extremal annealing times, and the distance is indeed small. To more closely assess the quality of the correlation we select $\alpha = 0.35$, the value that minimizes the Euclidean distance as seen in Fig. 8, and present the correlation plot in Fig. 7(b). Considering the results for each size L separately, it is apparent that the correlation is good but also systematically skewed, i.e., for each problem size the data points approximately lie on a line that is not the diagonal line. Additional correlation plots are shown in Fig. 23 of Appendix B.

As a final comparison we also extract the scaling coefficients $b(\alpha)$ and compare DW2 to SAA in Fig. 9. It can be seen that in terms of the scaling coefficients the DW2 and SAA results are essentially statistically indistinguishable. However, we stress that since the SAA number of sweeps is not optimized, one should refrain from concluding that the equal scaling observed for the DW2 and SAA rules out a DW2 speedup.

V. DISCUSSION

In this work we proposed and implemented a method for generating problems with a range of hardness, tuned by the clause density, mimicking the phase structure observed for SAT problems. By comparing the DW2 device to a number of classical algorithms we delineated where there is no DW2 speedup and where it might still be possible, for this problem set. No advantage is observed for the low clause densities corresponding to the hardest optimization problems, but a speedup remains possible for the higher clause densities. In this sense these results are more encouraging than for the random Ising problems studied in Ref. [15], where only the lowest percentiles of the success probability distribution allowed for the possibility of a DW2 speedup. In our case there is in fact a slight improvement for the higher percentiles (see Fig. 21 in Appendix B). In the same sense our findings are also more encouraging than very recent theoretical results showing that quantum annealing does not provide a speedup relative to simulated annealing on 2SAT problems with a unique ground state and a highly degenerate first excited state [48].

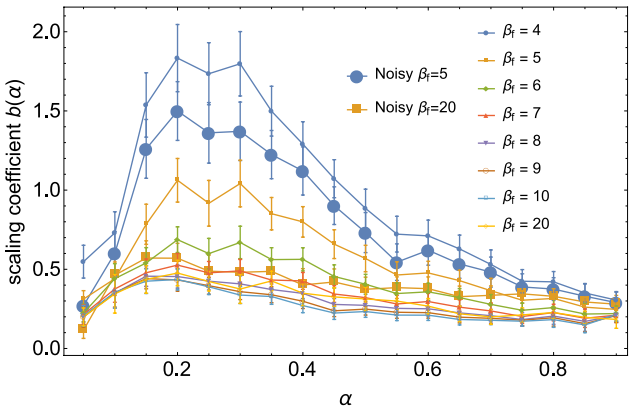


FIG. 11. **Scaling of SAA at different final temperatures.** SAA is run at $S = 50{,}000$ and various final inverse temperatures. The peak at $\alpha \sim 0.2$ remains a robust feature. The data marked "Noisy" is with 5% random noise added to the couplings J_{ij}.

The close match between the DW2 and SAA scaling coefficients seen in Fig. 9 suggests that SAA in the regime of an asymptotic number of sweeps can serve as a model for the expected performance of the D-Wave device as its temperature is lowered. Thus we plot in Fig. 11 the scaling coefficient for SAA at various final inverse temperatures, at a fixed (and still asymptotic) value of $S = 50{,}000$. Performance improves steadily as β_f increases, suggesting that SAA does not become trapped at 50,000 sweeps for the largest problem sizes we have studied (this may also indicate that even at the hardest clause densities these problems do not exhibit a positive-temperature spin-glass phase). We may infer that a similar behavior can be expected of the D-Wave device if its temperature were lowered.

An additional interesting feature of the fact that the asymptotic DW2 ground state probability observed here is in good agreement with that of a thermal annealer, is that it gives the ground state with a similar probability as expected from a Gibbs distribution. This result is consistent with the weak-coupling limit that underlies the derivation of the adiabatic Markovian master equation [49], i.e., it is consistent with the notion that the system-bath coupling is weak and *decoherence occurs in the energy eigenbasis* [50]. This rules out the possibility that decoherence occurs in the computational basis,

as this would have led to a singular-coupling limit master equation with a ground state probability drawn not from the Gibbs distribution but rather from a uniform distribution, if the single-qubit decoherence time is much shorter than the total annealing time [49]. This is important, as the weak-coupling limit is compatible with decoherence between eigenstates with different energies while maintaining ground state coherence, a necessary condition for a speedup via quantum annealing. In contrast, in the singular-coupling limit no quantum effects survive and no quantum speedup of any sort is possible.

We note that an important disadvantage the DW2 device has over all the classical algorithms we have compared it with, is control errors in the programming of the local fields and couplings [51–54]. As shown in Appendix B (Fig. 28), many of the rescaled couplings J_{ij} used in our instances are below the single-coupler control noise specification, meaning that with some probability the DW2 is giving the right solution to the wrong problem. We are unable to directly measure the effect of such errors on the DW2 device, but their effect is demonstrated in Figs. 10 and 11, where both SAS and SAA with $\beta_f = 5, 20$, respectively, are seen to have substantially increased scaling coefficients after the addition of noise that is comparable to the control noise in the DW2 device. In fact, the DW2 scaling coefficient is smaller than the scaling coefficient of optimized SAS with noise over almost the entire range of α, suggesting that a reduction in such errors will extend the upper bound for a DW2 speedup against SAS over a wider range of clause densities. The effect of such control errors can be mitigated by improved engineering, but also emphasizes the need for the implementation of error correction on putative quantum annealing devices. The beneficial effect of such error correction has already been demonstrated experimentally [25, 29] and theoretically [55], albeit at the cost of a reduction in the effective number of qubits and reduced problem sizes.

To summarize, we believe that at least three major improvements will be needed before it becomes possible to demonstrate a conclusive (limited or potential) quantum speedup using putative quantum annealing devices: (1) harder optimization problems must be designed that will allow the annealing time to be optimized, (2) decoherence and control noise must be further reduced, and (3) error correction techniques must be incorporated. Another outstanding challenge is to theoretically design optimization problems that can be unequivocally shown to benefit from quantum annealing dynamics.

Finally, the methods introduced here for creating frustrated problems with tunable hardness should serve as a general tool for the creation of suitable benchmarks for quantum annealers. Our study directly illustrates the important role that frustration plays in the optimization of spin-glass problems for classical algorithms as well as for putative quantum optimizers. It is plausible that different, perhaps more finely-tuned choices of clauses to create novel types of benchmarks, may be used to establish a clearer separation between the performance of quantum and classical devices. These may eventually lead to the demonstration of the coveted experimental annealing-based quantum speedup.

Note added. Work on problem instances similar to the ones we studied here, but with a range of couplings above the single-coupler control noise specification, appeared shortly after our preprint [56]. The DW2 scaling results were much improved, supporting our conclusion that such errors have a strong detrimental effect on the performance of the DW2.

ACKNOWLEDGMENTS

Part of the computing resources were provided by the USC Center for High Performance Computing and Communications. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. I.H. and D.A.L. acknowledge support under ARO grant number W911NF-12-1-0523. The work of J.J., T.A. and D.A.L. was supported under ARO MURI Grant No. W911NF-11-1-0268, and the Lockheed Martin Corporation. M.T. and T.F.R. acknowledges support by the Swiss National Science Foundation through the National Competence Center in Research NCCR QSIT, and by the European Research Council through ERC Advanced Grant SIMCOFE. We thank Mohammad Amin, Andrew King, Catherine McGeoch, and Alex Selby and for comments and discussions. I.H. would like to thank Gene Wagenbreth for assistance with the implementation of solution-enumeration algorithms. J.J. would like to thank the developers of the Julia programming language [57], which was used extensively for data gathering and analysis.

Appendix A: Methods

In this section we describe various technical details regarding out methods of data collection and analysis.

1. Experimental details

The DW2 is marketed by D-Wave Systems Inc. as a quantum annealer, which evolves a physical system of superconducting flux qubits according to the time-dependent Hamiltonian

$$H(t) = A(t)\sum_{i\in V}\sigma_i^x + B(t)H_{\rm Ising}\ , \quad t\in[0,t_a]\ , \tag{A1}$$

with $H_{\rm Ising}$ given in Eq. (1). The annealing schedules given by $A(t)$ and $B(t)$ are shown in Fig. 12. Our experiments used the DW2 device housed at the USC Information Sciences Institute, with an operating temper-

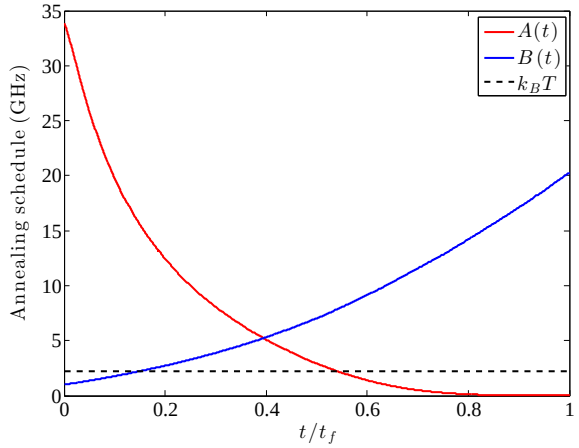


FIG. 12. **Annealing schedule of the DW2.** The annealing curves $A(t)$ and $B(t)$ are calculated using rf-SQUID models with independently calibrated qubit parameters. Units of $\hbar = 1$. The operating temperature of 17mK is also shown.

ature of 17mK. The Chimera graph of the DW2 used in our work is shown in Figure 13. Each unit cell is a balanced $K_{4,4}$ bipartite graph. In the ideal Chimera graph (of 512 qubits) the degree of each vertex is 6 (except for the corner unit cells). In the actual DW2 device we used 503 qubits were functional. For the scaling analysis we considered $L \times L$ square sub-lattices of the Chimera graph, and restricted our simulations and tests on the DW2 to the subset of functional qubits within these subgraphs, denoted C_L (see Figure 13). For each clause density α and problem size N (or C_L) we generated 100 instances, for a total of 12,600 instances. We performed approximately $990000\mu s/t_a$ annealing runs (experiments) for each problem instance and for each annealing time $t_a \in [20, 40]\mu$s, in steps of 2μs, for a total of more than 10^{10} experiments. No gauge averaging [20] was performed because we are not concerned with the timing data for a single instance but a collection of instances, and the variation over instances is larger than the variation over gauges.

2. Algorithms

a. *HFS algorithm*

The HFS algorithm is due to Hamze & Freitas [41] and Selby [43]. This is a tree-based optimization algorithm, which exploits the sparsity and local connections of the Chimera graph to construct very wide induced trees and repeatedly optimizes over such trees until no more improvement is likely.

We briefly discuss the tree construction of the HFS algorithm. It considers each part of the bipartite unit cell as a single 2^4-dimensional vertex instead of four distinct 2-dimensional vertices, resulting in a graph as depicted in Fig. 14. The tree represented by the dark vertices covers 78% of the graph, and such trees will cover 75% of the graph in the limit of infinitely large Chimera graphs. Finding the minimum energy configuration of such a tree conditioned on the state of the rest of the graph can be done in $\mathcal{O}(N)$ time where N is the number of vertices. Since the tree encodes so much of the graph, optimizing over such trees can quickly find a low-lying energy state. This is the strategy behind ProgQUBO [58]. ~~We solve a given problem instance once and record the time-to-solution t_i^{HFS} on the ith trial run. We repeat this 10^5 times. We then take the 99th percentile of the set $\{t_i^{\mathrm{HFS}}\}$ as our measure of the time to 99% confidence that we have found a solution at least once. Error bars are estimated by bootstrapping on the same set, as explained in more detail below~~Since each sample takes a variable number of trees, we estimate time to solution as as the average number of trees multiplied by the number of operations per tree, with a time constant of $0.5\mu s$ per operation.

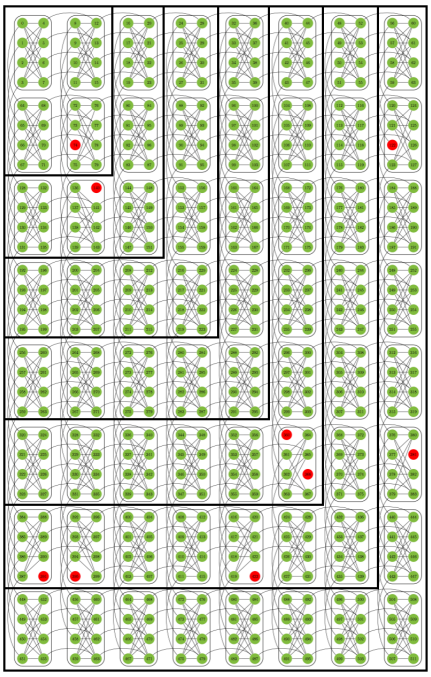


FIG. 13. **The DW2 Chimera graph.** The qubits or spin variables occupy the vertices (circles) and the couplings J_{ij} are along the edges. Of the 512 qubits, 503 were operative in our experiments (green circles) and 9 were not (red circles). We utilized subgraphs comprising $L \times L$ unit cells, denoted C_L, indicated by the solid black lines. There were $31, 70, 126, 198, 284, 385, 503$ qubits in our $C_2, \ldots, C_8$ graphs, respectively.

Prog-QUBO runs in a serial manner on a single core of a CPU. Since we allow the DW2 to use $\mathcal{O}(N)$ resources, we should also parallelize the HFS algorithm. In principle, HFS proceeds in a series of steps — at each step, one shears off each of the leaves of the tree (i.e., the outermost vertices) and then proceeds to the next step until the tree has collapsed to a point. Each of the leaves can be eliminated separately, however each elimination along a particular branch is dependent on the previous eliminations. As a result, we must perform between $L+2$ and $\frac{3}{2}L+2$ (average $\frac{5}{4}L+2$) irreducibly serial operations when reducing the trees on an $L \times L$ unit cell Chimera graph (depending on which of the trees we use and the specifics for how we do the reduction). Since we deal only with square graphs and are concerned with asymptotic scaling, we ignore the constant steps and ~~divide our serial TTS by the number of unit cells on the side ($L = \sqrt{N/8}$ for N-qubit problems)to account for the parallelism that we did not implement~~use $\frac{5}{4}L$ basic operations per tree (see Sec. A 3 b for additional details).

We ran Prog-QUBO in mode "-S3" [58], meaning that we used maximal induced trees (treewidth 1 in our case). Other options enable reduction over lines or treewidth 2 subgraphs, but we did not use those options here. In principle, one could define a new solver by optimizing the treewidth of subgraphs over which to reduce for each problem size, which will likely perform better than only using treewidth 1 for arbitrary problem sizes. However, this was not investigated in this work.

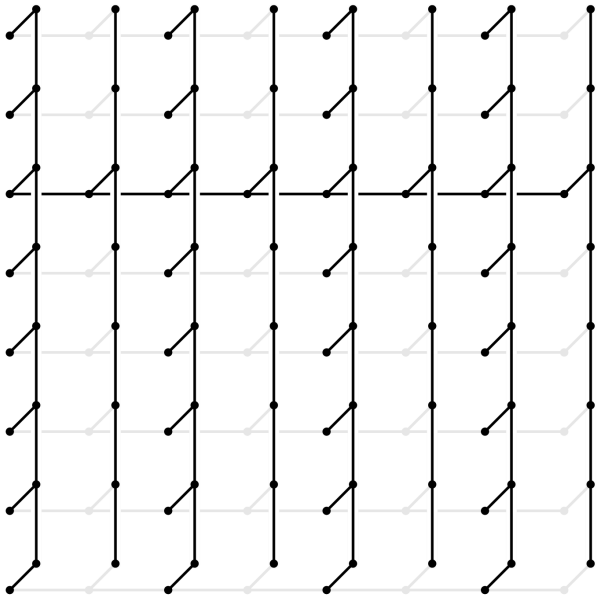


FIG. 14. An example of the view of the Chimera graph in the HFS algorithm. Each vertex is one half of a unit cell, and is 2^4-dimensional. The algorithm repeatedly finds the minimum energy configuration of the graph over trees (like the one highlighted) given the state of the remaining vertices (in gray).

b. *Simulated Annealing*

Our SA algorithm uses a single spin-flip Metropolis update method. In a single sweep, each spin is updated once according to the Metropolis rule: the spin is flipped, the change in energy ΔE is calculated. If the energy is lowered, the flip is accepted, and if not, it is accepted with a probability given by the Metropolis probability:

$$P_{\text{Met}} = \min\left(1, \exp(-\beta \Delta E)\right) \tag{A2}$$

A linear annealing schedule in β is used; Starting at $\beta_i = 0.01$, we increment β in steps of $\delta\beta = (\beta_f - \beta_i)/(S-1)$ where S is the number of sweeps, up to β_f. Each instance is repeated 10^4 times.

c. *SSSV*

The SSSV model was first proposed [26] as a classical model that reproduced the success probabilities of the DW1 device studied in Ref. [20], although there is growing evidence that this model fails to capture the behavior of the device for specific instances [27, 32, 34]. The model can be understood as describing coherent single qubits interacting incoherently by replacing qubits by O(2) rotors; the Hamiltonian can be generated by replacing $\sigma_i^x \mapsto \sin\theta_i$ and $\sigma_i^z \mapsto \cos\theta_i$. The system is then "evolved" by Monte Carlo updates on the angles $\theta_i \in [0, \pi]$. Although SSSV is not designed to be a fast solver, we studied it here as a potential classical limit of the DW2 and checked what the scaling of such a classical limit would be. The DW2 annealing schedule in Fig. 12 was used, and the temperature was kept constant at 10.56mK. Each instance was repeated 10^4 times.

d. *Simulated Quantum Annealing*

SQA [10, 11] is an annealing algorithm based on discrete-time path-integral quantum Monte Carlo simulations of the transverse field Ising model but using Monte Carlo dynamics instead of the open system evolution of a quantum system. This amounts to sampling the world line configurations of the quantum Hamiltonian (A1) while slowly changing the couplings. SQA has been shown to be consistent with the input/output behavior of the DW1 for random instances [20], and we accord-

ingly used a discrete-time quantum annealing algorithm. We always used 64 Trotter slices and the code was run at $\beta = 10$ (in dimensionless units, such that $\max(|J_{ij}|) = 1$) with linearly decreasing and increasing $A(t)$ and $B(t)$, respectively. Cluster updates were performed only along the imaginary time direction. A single sweep amounts to the following: for each space-like slice, a random spin along the imaginary time direction is picked. The neighbors of this spin are added to the cluster (assuming they are parallel) according to the Wolff algorithm [59] with probability $1 - e^{-2J_\perp}$, where $J_\perp = -0.5 \ln [\tanh A(t)]$ is the spin-spin coupling along the imaginary time direction. When the cluster construction terminates, the cluster is flipped according to the Metropolis probability using the change in energy along the space-like direction associated with flipping the cluster. Therefore a single sweep involves a single cluster update for each space-like slice. For every problem instance, we performed 10^3 repetitions in order to estimate the instance success rates.

e. *Constraint Solver Algorithm*

Here we outline the algorithm we developed and used to search for, and list, solutions to a specified planted-solution Hamiltonian, taking advantage of the fact that the total cost function is a sum of easily-solvable cost functions each defined on a finite number of bits. Since in our case each term in the Hamiltonian is a frustrated loop, it is easy to list the constraints each loop induces on any potential solution of the total Hamiltonian. The algorithm is an exhaustive search and is an implementation of the Bucket Elimination Algorithm (see, e.g., Ref. [60]).

The problem structure consists of a set of values (or bits) where each value may be $+1$ or -1. A set of constraints restricts subsets of the bits to certain values. The problem is to assign values to all the bits so as to satisfy all the constraints.

Each constraint (in this case, the optimizing configurations of loop Hamiltonians) applies to a subset of the bits, and contains a list of allowed settings for that subset of bits. The constraint is met if the values of the bits in the subset matches one of the allowed settings.

The Bucket Elimination Algorithm as applied to this problem consists of the following steps. First is the constraint elimination stage: (i) Select a bit to eliminate; (ii) Save constraints which contain selected bit; (iii) Combine all constraints which contain selected bit; (iv) Generate new constraints without selected bit; (v) If combined constraints contain a contradiction, exit; (vi) Repeat until all bits have been addressed.

The second part of the algorithm tackles the enumeration of the solutions. It involves using the tables created in the constraint elimination stage to grow solution sets one bit at a time. The last table created in the constraint satisfaction stage contains tables for a single bit (the last to be eliminated). If it has no entries, no solutions are possible for the processed constraint set. If there is a single entry, it indicates the value that the bit must be -1, or $+1$. If there are two entries, then both -1 and $+1$ are possible values. A solution set is built listing the values of the final bit that was eliminated. The table created for the final two bits is processed next. It contains all the legal values for the final two bits. For each allowed value of the final bit, a solution is generated for the final two bits, resulting in a new solution set for the final two bits. The bit solution tables generated during the elimination phase are processed in reverse order, each time adding a single bit, combining the solutions from the previous step with the single bit, generating a new solution set with a bit added. Ultimately all of the bit solution tables generated during the elimination phase are processed and solutions with values for all bits are generated. The logic driving the elimination phase ensures that it is always possible to combine the set of solutions with the next bit.

This algorithm is guaranteed to find all solutions, but may exceed time and memory constraints. All steps are well defined except for the step to "select a bit to eliminate". The order in which bits are eliminated dramatically affects the time and memory required. Determination of the optimal order to eliminate bits is known to be NP complete. A more detailed description of the algorithm (including examples) will be published elsewhere as part of another study in the near future.

3. Error estimation

a. *Annealers*

For the annealers, the time to solution can be trivially found by applying a function $T(p)$ to the probability of success p. In the main text, the $T(p)$ used is the time required to find the ground state at least once with a probability 0.99:

$$T(p) = \frac{\log(1 - 0.99)}{\log(1 - p)} \,, \tag{A3}$$

For the purposes of the discussion below this function can in fact be totally arbitrary and may depend on system size.

We imagine our annealing algorithms as essentially a sequence of binary trials, where each run is either a success or a failure. Because we do not have infinitely many observations, there is some uncertainty in our estimate of the true probability of success of our binary trial. Since our data was collected as a series of r independent trials (which varies by solver) for each solver and instance, the annealers' results are described by a binomial distribution.

In order to determine what the probability of success is, we assumed a Bayesian stance — we do not know what the probability is currently and so must assume some prior distribution for our belief, and will update our belief based on the evidence. Which prior should we

choose? The obvious choice is a β distribution as it is the conjugate prior for the binomial distribution, giving us a closed form for our posterior distribution [61]. We have chosen the Jeffreys prior for the binomial distribution, $\beta(\frac{1}{2},\frac{1}{2})$ for two reasons: 1) It is invariant under reparameterization of the parameter space, 2) it is the prior which maximizes the mutual information between the sample and the parameter over all continuous, positive priors. In other words, it yields the same prior no matter how we parameterize our space and $\beta(\frac{1}{2},\frac{1}{2})$ maximizes the amount of information gained by learning the data. The other obvious prior is the uniform distribution or $\beta(1,1)$ but it is not invariant under reparameterization and learns less from the data than $\beta(\frac{1}{2},\frac{1}{2})$ [62]. For these reasons, $\beta(\frac{1}{2},\frac{1}{2})$ tends to be the standard choice of prior for binomial distributions [63].

After Bayesian updating by our prior, our probability distribution for p is $\beta(x+\frac{1}{2}, r-x+\frac{1}{2})$, where x is the number of successes and r the number of runs.

There may be a concern that our solvers do not fully conform to a binomial distribution: if there are correlations between successive runs then our empirical success probability $\frac{x}{r}$ would be inflated for hard problems. However, since we did not observe an advantage for the DW2 over the classical algorithms for the hardest problems, and the only potential advantage we observed is for the easier problems at high clause density, we do not believe this issue is affecting our conclusions.

Thus, for all instances $\mathcal{I}_i \in \{\mathcal{I}\}$ in a particular range of interest (for example, a particular problem size and clause density), we have some distribution β_i for the probability of success for that instance. To generate our error bars we used the bootstrap method, which we describe next.

If we have M instances in our set of interest, then we resample with replacement a "new" set of instances $\{\mathcal{I}_i\}_j$, also of length M, from our set $\mathcal{I}$. For each instance $\mathcal{I}_{i,j}$ from this new set we sample a value for its probability from its corresponding distribution $\beta_{\mathcal{I}_{i,j}}$ to get a set of probabilities $\{p_{i,j}\}$. We then calculate whatever function $F_j = f(\{p_{i,j}\})$ we wish on these probabilities, e.g., the median over the set of instances. We repeat this process a large number of times (in our case, 1000), to have many values of our function $\{F_j\}$. We then take the mean and standard deviation over the set $\{F_j\}$ to get a value $\bar{F}$ and a standard deviation σ_F, which form our value and error bar for that size and clause density.

When we take the ratio of two algorithms A and B, for each pair of corresponding data points for the two algorithms (i.e., each size and clause density) we characterize each point with a normal distributions $\mathcal{N}(\mu_A,\sigma_A)$ and $\mathcal{N}(\mu_B,\sigma_B)$. We then resample from each distribution a large number of times (1000) to get two sets of values for the function we wish to plot F: $\{F_A\}$ and $\{F_B\}$. We take the ratio of the corresponding elements in the two sets $S_i = F_{A,i}/F_{B,i}$ and take the mean and standard deviation of the set $\{S_i\}$ to get our data point and error bar for the ratio.

b. *HFS algorithm*

For the HFS algorithm, ~~as mentioned, above, the~~ time to solution for each problem ~~was calculated as the 99th percentile over the set $\{t_i^{\mathrm{HFS}}\}_{i=1}^{100,000}$ where a single t_i^{HFS} represents the time to solution on the ith trial run. To compute an error estimate, we performed a bootstrap over our set t_i^{HFS} 1000 times, resampling a new set of 10^5 t_i^{HFS}'s with replacement and computing its 99th percentile, and took the mean value over our resampled datasets as our data point and the standard deviation as its error bar (which is here an estimator for the standard error of the mean of our distribution~~is computed as the mean number of trees per sample multiplied by $0.625\mu s \times L$ for a C_L problem. The number 0.625 is chosen to approximate the times on a standard laptop, and the scaling with L is due to the fact that, in a parallel setting, the number of steps to reduce a tree is linear in L (exactly, it is $\frac{5}{4}L+2$, but the 2 serves only to mask asymptotic scaling and is thus not included in our TTS or speedup plots or the scaling analysis).

c. *Euclidean distance*

In order to generate the error bars in Fig. 8, instead of calculating the Euclidean distance over the total number of instances at a given α (700 total), we calculated the Euclidean distance over half the number of instances (350 total). We were then able to perform 100 bootstraps over the instances, i.e., we picked 350 instances at random for each Euclidean distance calculation. For each bootstrap, we calculated the Euclidean distance, and the data points in Fig. 8 are the mean of the Euclidean distances over the bootstraps, while the error bars are twice the standard deviation of the Euclidean distances.

d. *Difference in slope*

In order to generate the error bars in Fig. 10, we used the data points and the error bars in Fig. 9 as the mean and twice the standard deviation of a Gaussian distribution. We then took 1000 samples from this distribution and calculated their differences. The means of the differences are the data points in Fig. 10, and the error bars are twice the standard deviation of the differences.

Appendix B: Additional results

In this section we collect additional results in support of the main text.

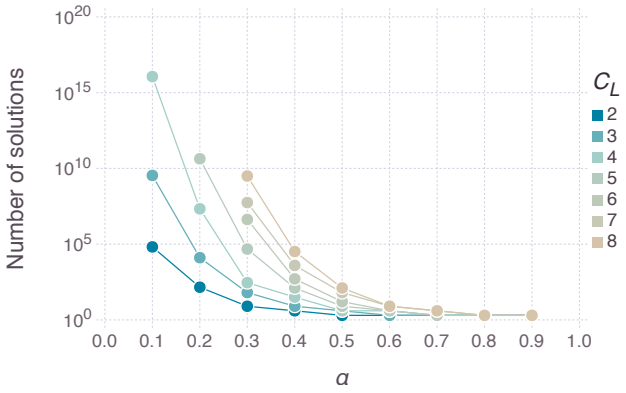


FIG. 15. **Ground state degeneracy**. Number of unique solutions as a function of clause density for different Chimera subgraph sizes C_L. As the clause density is increased, the number of unique solutions found decreases to one (up to the global bit flip symmetry). Shown is the median degeneracy, i.e., we sort the degeneracies of the 100 instances for each value of L and α, and find the median. Our procedure counts the degenerate solutions and stops when it reaches 10^5 solutions. If the median has 10^5 solutions then we assume that not all solutions were found and hence the degeneracy for that value of L and α is not plotted. These are solutions on the used qubits (e.g., there are many instances for each α at $L = 8$ that use < 503 qubits); to account for the n_{uq} unused qubits we multiply the degeneracy by $2^{n_{uq}}$.

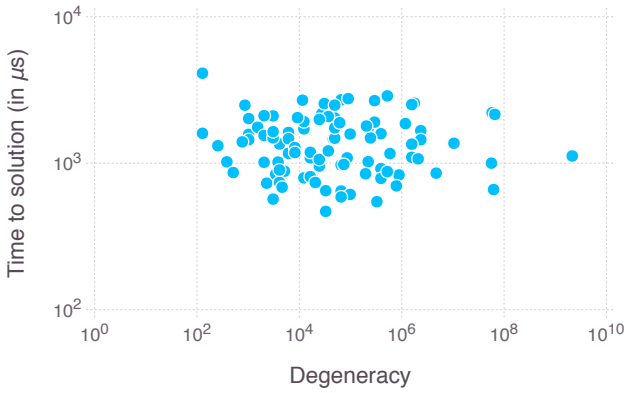


FIG. 16. Scatter plot of the TTS for the HFS algorithm and the degeneracy at $L = 8$ and $\alpha = 0.4$ (100 instances total). Even though there is a wide range of degeneracy over several orders of magnitude, we do not observe any trend in the TTS. The degeneracy accounts for the fact that some qubits are not coupled into the problem (e.g., if n qubits are not specified for that particular problem, then the degeneracy is 2^n times the directly counted degeneracy). The Pearson correlation coefficient is -0.046.

1. Degeneracy-hardness correlation

It is known that a non-degenerate ground state along with an exponentially (in problem size) degenerate first excited state leads to very hard SAT-type optimization problems [64]. Here we focus on the ground state degeneracy and ask whether it is correlated with hardness. We show the ground state degeneracy in Fig. 15. It decays rapidly as α grows, and for sufficiently large α (that depends on the problem size) the ground state found is unique (up to the global $\mathbf{Z}_2$ symmetry). This suggests that degeneracy is not necessarily correlated with hardness, since in the main text we found that hardness peaks at $\alpha \sim 0.25$. To this test directly we restrict ourselves to $L = 8$ and $\alpha = 0.4$. We then bin the 100 instances at this α according to their degeneracy and study their median TTS using the HFS algorithm. We show the results in Fig. 16, where we see no correlation between degeneracy and hardness for a fixed α. We find a similar result when we use the success probability of the DW2 as the metric for hardness.

2. Additional easy-hard-easy transition plots

The universal nature of the scaling behavior can be seen in Fig. 17, complementing Fig. 2 with results for the 25th and 75th percentiles respectively. The peak in the TTS near $\alpha = 0.2$ is a feature shared by all the solvers we considered.

3. Optimality plots

The absence of an optimal DW2 annealing time was discussed in detail in the main text, along with the optimality of the number of sweeps of the classical algorithms. Figure 18 illustrates this: a clear lower envelope is formed by the different curves plotted for SQA, SA, and SSSV, from which the optimal number of sweeps at each size can be easily extracted. Unfortunately no such envelope is seen for the DW2 results [Fig. 18(a)], leading to the conclusion that $t_a = 20\mu$s is suboptimal.

A complementary perspective is given by Fig. 19, where we plot the TTS as a function of the number of sweeps, for a fixed problem size $L = 8$. It can be seen that the classical algorithms all display a minimum for each clause density. The DW2 curves all slope upward, suggesting that the minimum lies to the left, i.e., is attained at $t_a < 20\mu$s. We note that in an attempt to extract an optimal annealing time we tried to fit the DW2 curves to various functions inspired by the shape of the classical curves, but this proved unsuccessful since the DW2 curves essentially differ merely by a factor of t_a, as discussed in the main text.

We can take this a step further and use these optimal number of sweeps results to demonstrate that the problems we are considering here really are hard. To this end we plot in Fig. 20 the optimal number of sweeps as a

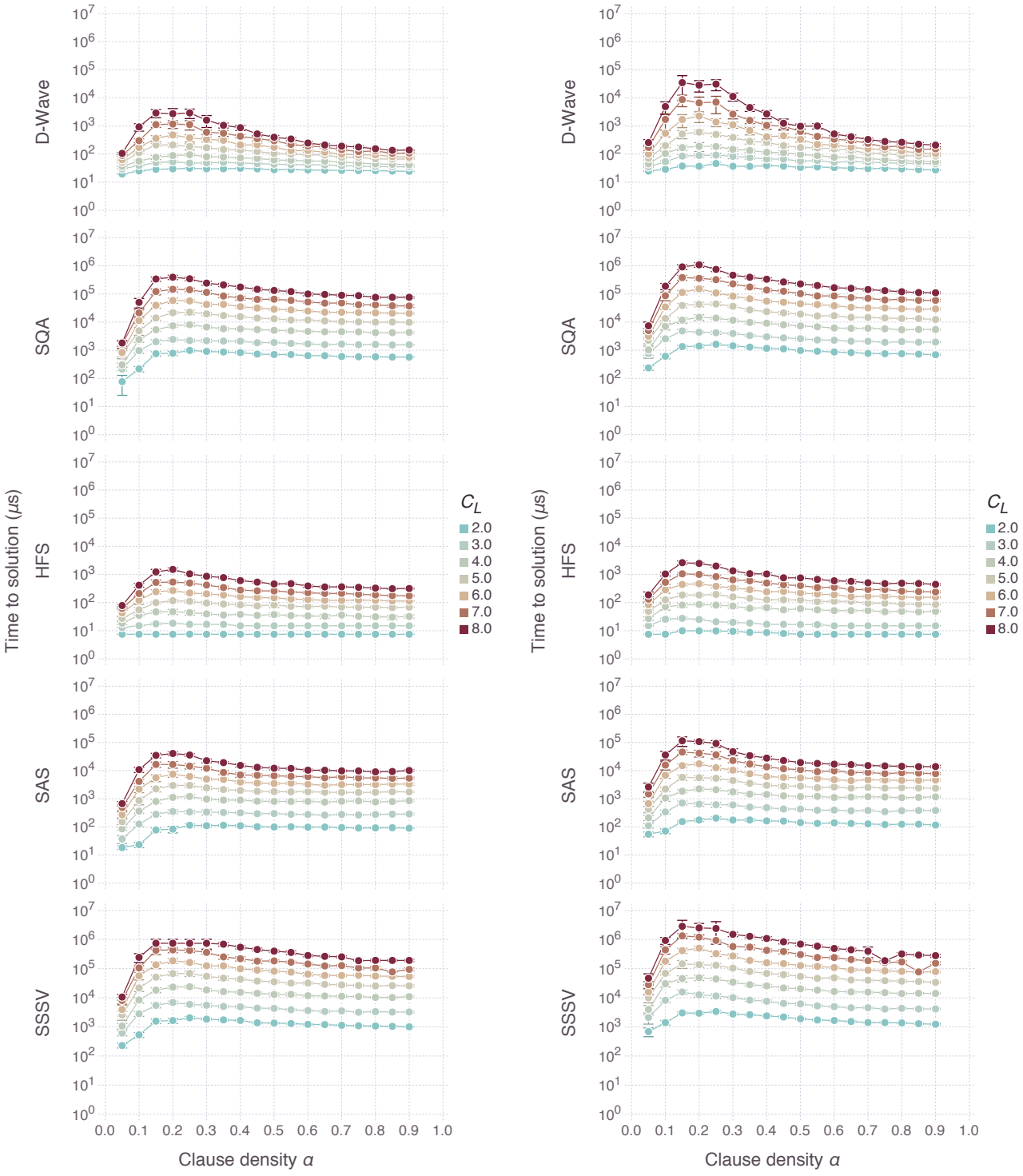


FIG. 17. Comparison of the 25th (left) and 75th (right) percentiles of the TTS (log scale) for all algorithms as a function of clause density α. The different colors represent the different Chimera sizes tested. All solvers show a peak at the same density value of $\alpha \approx 0.2$.

function of problems size for SAA. We observe that certainly for the smaller clause densities the optimal number of sweeps $s_{\rm opt}$ appears to scale exponentially in L, which indicates that the TTS (which is proportional to $s_{\rm opt}$) will also grow exponentially in L.

4. Additional speedup ratio plots

To test whether our speedup ratio results depend strongly on the percentile of the success probability distribution, we recreate Fig. 6 for the 25th and 75th percentiles in Fig. 21. The results are qualitatively similar, with a small improvement in the speedup ratio relative to HFS at the higher percentile.

5. Additional correlation plots

To complement Fig. 7, we provide correlation plots for both the DW2 against itself at $t_a = 20\mu$s and $t_a = 40\mu$s (Fig. 22), the DW2 *vs* SAA (Fig. 23), the DW2 *vs* SQAA (Fig. 24), and SAA *vs* SQAA (Fig. 25. The DW2 against itself displays an excellent correlation at all clause densities, while the DW2 *vs* SAA and DW2 *vs* SQAA continues to be skewed at low and high clause densities. Recall that Fig. reffig:Euclid-dist provides an objective Euclidean distance measure that is computed using all problem sizes and depends only on the clause density.

6. Additional scaling analysis plots

We provide a few additional plots in support of the scaling analysis presented in the main text.

Figure 26(a) shows the number of runs at different problem sizes and clause densities, and the corresponding least-squares fits. It can be seen that the straight lines fits are quite good. The slopes seen in this figure are the $b(\alpha)$ values for $t_a = 20\mu$s plotted in Fig. 9; the intercepts are plotted in Fig. 26(b) for all annealing times, and collapse nicely, just like the $b(\alpha)$.

Finally, Fig. 27(a) is a check of the convergence of SAA to its asymptotic scaling coefficient as the number of sweeps is increased from 5,000 to 50,000. Convergence is apparent within the 2σ error bars.

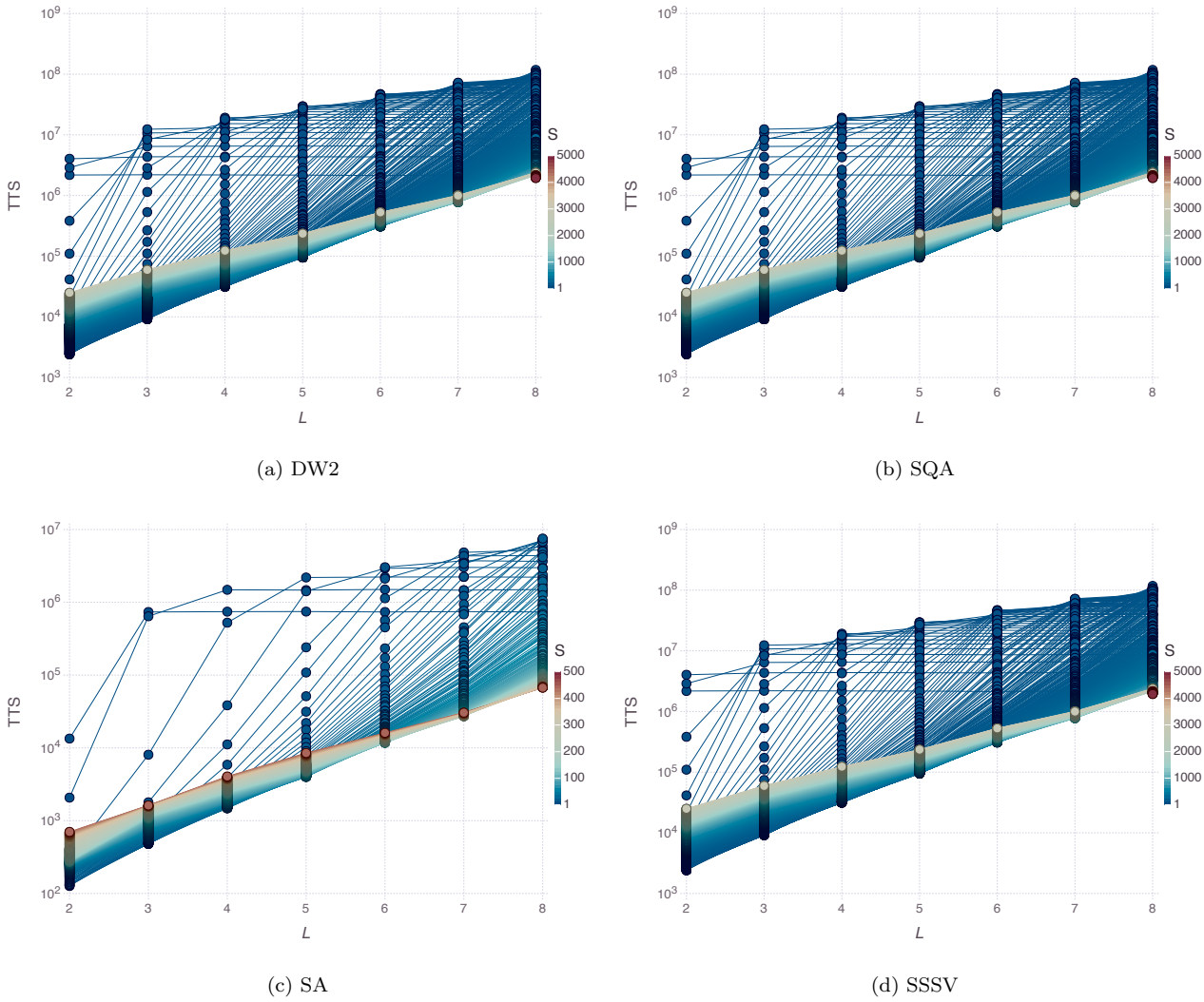
7. SQAS *vs* SAS

In the main text we only considered SQA as an annealer since that is a more faithful representation of the DW2. Here we present a comparison of SQA as a solver (SQAS), where we keep track of the lowest energy found during the entire anneal, with SAS. We present the scaling coefficient $b(\alpha)$ from Eq. (8) of these two solvers in Fig. 27(b). SAS has a smaller scaling coefficient than SQAS for the large α values, but at small α values we cannot make a conclusive determination because of the substantial overlap of the error bars. We note that Ref. [65] reported that discrete-time SQA (the version used here) can exhibit a scaling advantage over SA but that this advantage vanishes in the continuous-time limit. We have not explored this possibility here.

8. Scale factor histograms

We analyze the effect of increasing clause density and problem size on the required precision of the couplings. Fig. 28 shows a trend of scaling factor increasing as α increases for fixed L, and as L increases for fixed α. Increased scaling factor has the effect of relatively amplifying control error and thermal effects in DW2, and can therefore contribute to a decline in performance for the larger problems studied. However, recall that the region where a speedup is possible according to our results is in fact that of high clause densities. Thus, whatever the effect of precision errors is, it does not appear to heavily impact the DW2's performance in the context of our problems. The same is true for SAS, since as can be seen in Fig. 10, the scaling coefficient is unaffected by the addition of noise when $\alpha \gtrsim 0.6$.

[1] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. L. O'Brien, Nature **464**, 45 (2010).
[2] J. I. Cirac and P. Zoller, Nat Phys **8**, 264 (2012).
[3] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, Nature **473**, 194 (2011).
[4] S. Suzuki and A. Das (guest eds.), Eur. Phys. J. Spec. Top. **224**, 1 (2015).
[5] B. Apolloni, C. Carvalho, and D. de Falco, Stochastic Processes and their Applications **33**, 233 (1989).
[6] P. Ray, B. K. Chakrabarti, and A. Chakrabarti, Phys. Rev. B **39**, 11828 (1989).
[7] P. Amara, D. Hsu, and J. E. Straub, *The Journal of Physical Chemistry*, The Journal of Physical Chemistry **97**, 6715 (1993).
[8] A. B. Finnila, M. A. Gomez, C. Sebenik, C. Stenson, and J. D. Doll, Chemical Physics Letters **219**, 343 (1994).
[9] T. Kadowaki and H. Nishimori, Phys. Rev. E **58**, 5355 (1998).
[10] R. Martoňák, G. E. Santoro, and E. Tosatti, Phys. Rev. B **66**, 094203 (2002).



(a) DW2

(b) SQA

(c) SA

(d) SSSV

FIG. 18. **Suboptimal annealing time and optimal sweeps for $\alpha = 0.2$.** Plotted is the TTS (log scale) as a function of size L for (a) the DW2, with all available annealing times, (b) SQA, (c) SA, and (d) SSSV, for many different sweep numbers. The lower envelope gives the scaling curves shown in Fig. 5 for $\alpha = 0.2$. The TTS curves flatten at high L for the following reason: each classical annealer was run N_X times ($N_{\mathrm{SA}} = N_{\mathrm{SSSV}} = 10^4$, $N_{\mathrm{SQA}} = 10^3$), and our β distribution is $\beta(0.5, N_X + 0.5)$ for 0 successes, which has an average value of $\sim 1/(2N_X)$. This reflects the (Bayesian) information acquired after N_X runs with 0 successes (one would not expect the probability to be 0). The flattening has no impact on the scale of the optimal number of sweeps.

[11] G. E. Santoro, R. Martoňák, E. Tosatti, and R. Car, Science **295**, 2427 (2002).

[12] J. Brooke, D. Bitko, T. F., Rosenbaum, and G. Aeppli, Science **284**, 779 (1999).

[13] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, Science **292**, 472 (2001).

[14] B. W. Reichardt, in *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04 (ACM, New York, NY, USA, 2004) pp. 502–510.

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

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Science **220**, 671 (1983).

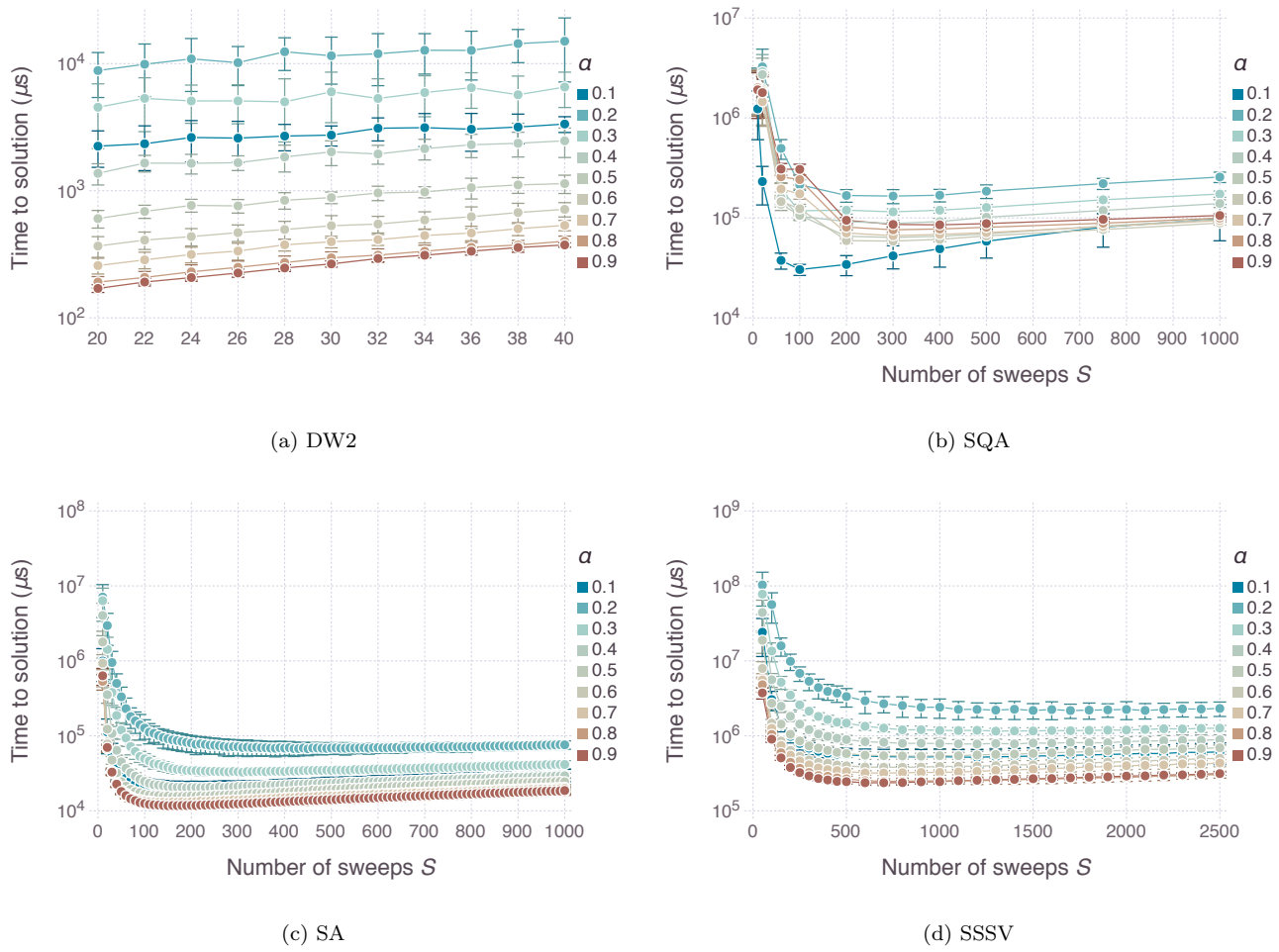
[17] A. Das and B. K. Chakrabarti, Rev. Mod. Phys. **80**, 1061 (2008).

[18] K. Binder and A. P. Young, Reviews of Modern Physics **58**, 801 (1986).

[19] H. Nishimori, *Statistical Physics of Spin Glasses and Information Processing: An Introduction* (Oxford University Press, Oxford, UK, 2001).

[20] S. Boixo, T. F. Ronnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, Nat. Phys. **10**, 218 (2014).

[21] P. I. Bunyk, E. M. Hoskinson, M. W. Johnson, E. Tolkacheva, F. Altomare, A. Berkley, R. Harris, J. P. Hilton, T. Lanting, A. Przybysz, and J. Whittaker, *Applied Superconductivity, IEEE Transactions on*, Applied Superconductivity, IEEE Transactions on **24**, 1 (Aug. 2014).



(a) DW2

(b) SQA

(c) SA

(d) SSSV

FIG. 19. TTS (log scale) for $L = 8$ as a function of number of sweeps for DW2, SQA, SA, and SSSV used to identify the optimal number of sweeps.

[22] V. Choi, Quant. Inf. Proc. **10**, 343 (2011).

[23] S. Boixo, T. Albash, F. M. Spedalieri, N. Chancellor, and D. A. Lidar, Nat. Commun. **4**, 2067 (2013).

[24] N. G. Dickson, M. W. Johnson, M. H. Amin, R. Harris, F. Altomare, A. J. Berkley, P. Bunyk, J. Cai, E. M. Chapple, P. Chavez, F. Cioata, T. Cirip, P. deBuen, M. Drew-Brook, C. Enderud, S. Gildert, F. Hamze, J. P. Hilton, E. Hoskinson, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Lanting, T. Mahon, R. Neufeld, T. Oh, I. Perminov, C. Petroff, A. Przybysz, C. Rich, P. Spear, A. Tcaciuc, M. C. Thom, E. Tolkacheva, S. Uchaikin, J. Wang, A. B. Wilson, Z. Merali, and G. Rose, Nat. Commun. **4**, 1903 (2013).

[25] K. L. Pudenz, T. Albash, and D. A. Lidar, Nat. Commun. **5**, 3243 (2014).

[26] S. W. Shin, G. Smith, J. A. Smolin, and U. Vazirani, arXiv:1401.7087 (2014).

[27] T. Albash, W. Vinci, A. Mishra, P. A. Warburton, and D. A. Lidar, Physical Review A **91**, 042314 (2015).

[28] S. W. Shin, G. Smith, J. A. Smolin, and U. Vazirani, arXiv:1404.6499 (2014).

[29] K. L. Pudenz, T. Albash, and D. A. Lidar, Phys. Rev. A **91**, 042302 (2015).

[30] P. J. D. Crowley, T. urić, W. Vinci, P. A. Warburton, and A. G. Green, Physical Review A **90**, 042317 (2014).

[31] D. Venturelli, S. Mandrà, S. Knysh, B. O'Gorman, R. Biswas, and V. Smelyanskiy, arXiv:1406.7553 (2014).

[32] T. Albash, T. F. Rønnow, M. Troyer, and D. A. Lidar, Eur. Phys. J. Spec. Top. **224**, 111 (2015).

[33] T. Lanting, A. J. Przybysz, A. Y. Smirnov, F. M. Spedalieri, M. H. Amin, A. J. Berkley, R. Harris, F. Altomare, S. Boixo, P. Bunyk, N. Dickson, C. Enderud, J. P. Hilton, E. Hoskinson, M. W. Johnson, E. Ladizinsky, N. Ladizinsky, R. Neufeld, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, S. Uchaikin, A. B. Wilson, and G. Rose, Phys. Rev. X **4**, 021041 (2014).

[34] S. Boixo, V. N. Smelyanskiy, A. Shabani, S. V. Isakov, M. Dykman, V. S. Denchev, M. Amin, A. Smirnov, M. Mohseni, and H. Neven, arXiv:1411.4036 (2014).

[35] H. G. Katzgraber, F. Hamze, and R. S. Andrist, Phys. Rev. X **4**, 021008 (2014).

[36] M. Mezard, G. Parisi and M.A. Virasoro, *Spin Glass Theory and Beyond*, World Scientific Lecture Notes in

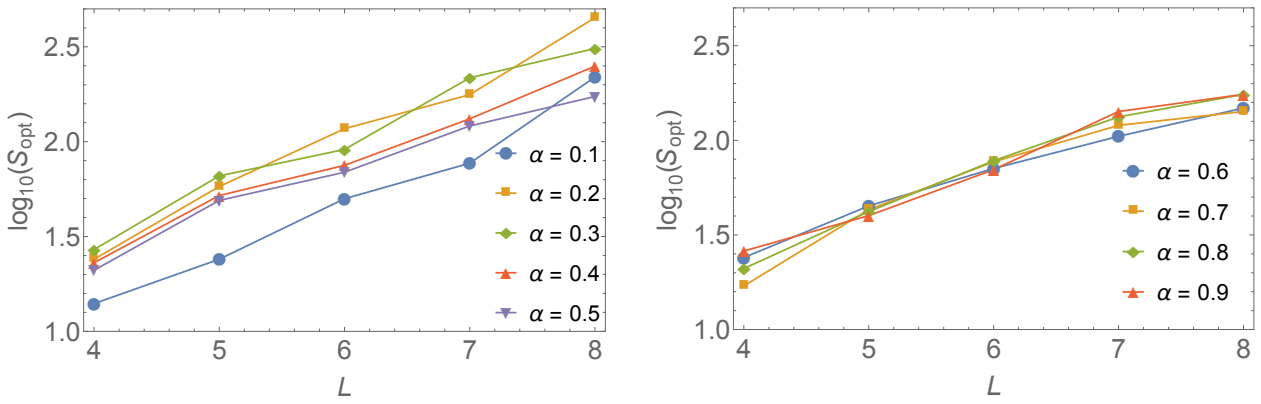


FIG. 20. **Scaling of the SAA optimal number of sweeps.** The optimal number of sweeps is extracted for each L from Fig. 19. The scaling is roughly exponential for the smaller α values, and appears to be close to exponential for the larger α values. Lines are guides to the eye.

Physics (World Scientific, Singapore, 1987).

[37] W. Barthel, A. K. Hartmann, M. Leone, F. Ricci-Tersenghi, M. Weigt, and R. Zecchina, Phys. Rev. Lett. **88**, 188701 (2002).

[38] F. Krzakala and L. Zdeborová, Phys. Rev. Lett. **102**, 238701 (2009).

[39] S. Bravyi and B. Terhal, *SIAM Journal on Computing*, SIAM Journal on Computing **39**, 1462 (2009).

[40] B. Bollobás, C. Borgs, J. T. Chayes, J. H. Kim, and D. B. Wilson, Random Struct. Algorithms **18**, 201 (2001).

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

[42] A. Selby, arXiv:1409.3934 (2014).

[43] A. Selby, "D-wave: comment on comparison with classical computers," `http://tinyurl.com/dwave-vs-classical` (2013).

[44] S. Santra, G. Quiroz, G. V. Steeg, and D. A. Lidar, New J. of Phys. **16**, 045006 (2014).

[45] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, Nature **400**, 133 (1999).

[46] M. Amin, arXiv:1503.04216 (2015).

[47] T. Albash and D. A. Lidar, Physical Review A **91**, 062320 (2015).

[48] T. Neuhaus, arXiv:1412.5460 (2014).

[49] T. Albash, S. Boixo, D. A. Lidar, and P. Zanardi, New J. of Phys. **14**, 123016 (2012).

[50] A. M. Childs, E. Farhi, and J. Preskill, Phys. Rev. A **65**, 012322 (2001).

[51] A. D. King and C. C. McGeoch, arXiv:1410.2628 (2014).

[52] A. Perdomo-Ortiz, J. Fluegemann, R. Biswas, and V. N. Smelyanskiy, arXiv:1503.01083 (2015).

[53] A. Perdomo-Ortiz, B. O'Gorman, J. Fluegemann, R. Biswas, and V. N. Smelyanskiy, arXiv:1503.05679 (2015).

[54] V. Martin-Mayor and I. Hen, arXiv:1502.02494 (2015).

[55] K. C. Young, R. Blume-Kohout, and D. A. Lidar, Phys. Rev. A **88**, 062314 (2013).

[56] A. D. King, arXiv:1502.02098 (2015).

[57] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, arXiv:1411.1607 (2014).

[58] A. Selby, "Prog-qubo," `https://github.com/alex1770/QUBO-Chimera` (2013).

[59] U. Wolff, Phys. Rev. Lett. **62**, 361 (1989).

[60] R. Dechter, Artificial Intelligence **113**, 41 (1999).

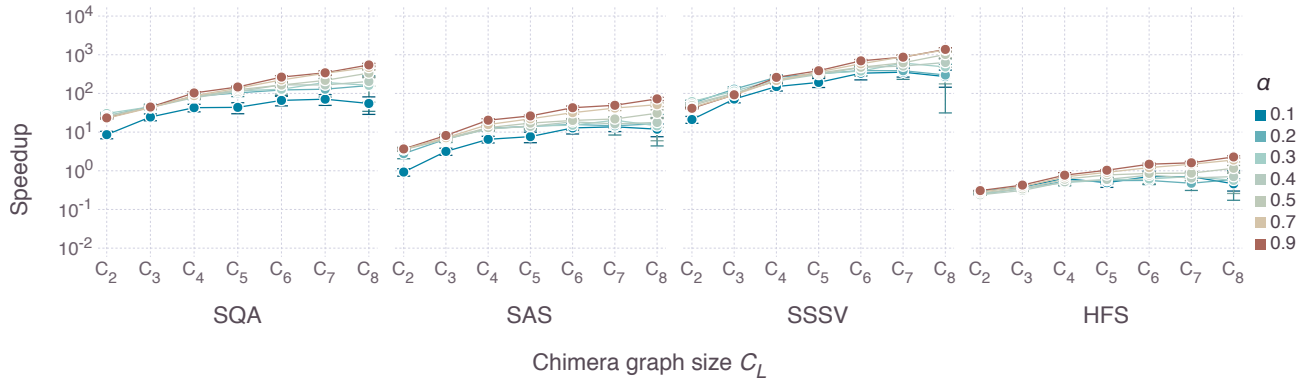
[61] D. Fink, *A compendium of conjugate priors*, Tech. Rep. (1997).

[62] B. S. Clarke and A. R. Barron, Journal of Statistical Planning and Inference **41**, 37 (1994).

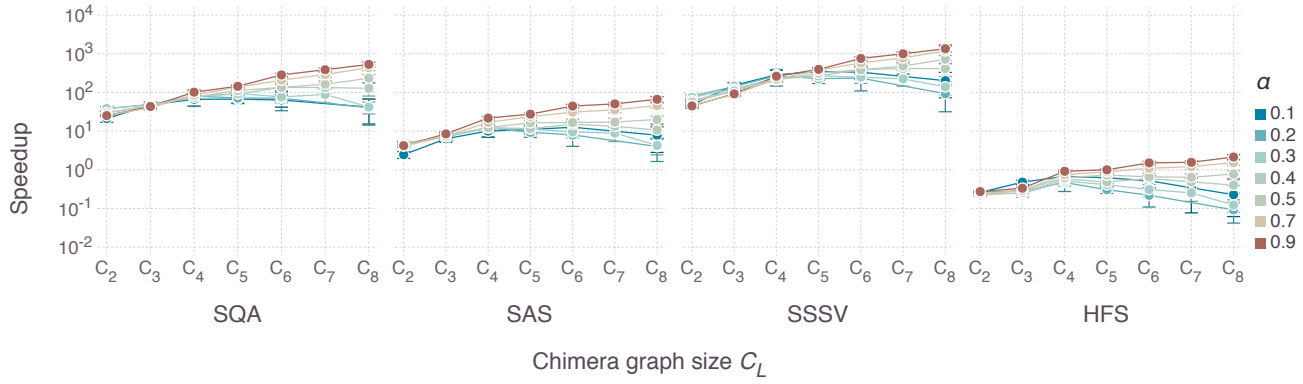
[63] J. M. Bernardo, in *International Encyclopedia of Statistical Science*, edited by M. Lovric (Springer, 2011) pp. 107–133.

[64] T. Neuhaus, arXiv:1412.5361 (2014).

[65] B. Heim, T. F. Rønnow, S. V. Isakov, and M. Troyer, Science **348**, 215 (2015).



(a) 25th percentile



(b) 75th percentile

FIG. 21. The speedup ratios (log scale) for the 25th and 75th percentile of time-to-solution as a function of system size for various α. The different colors denote a representative sample of clause densities.

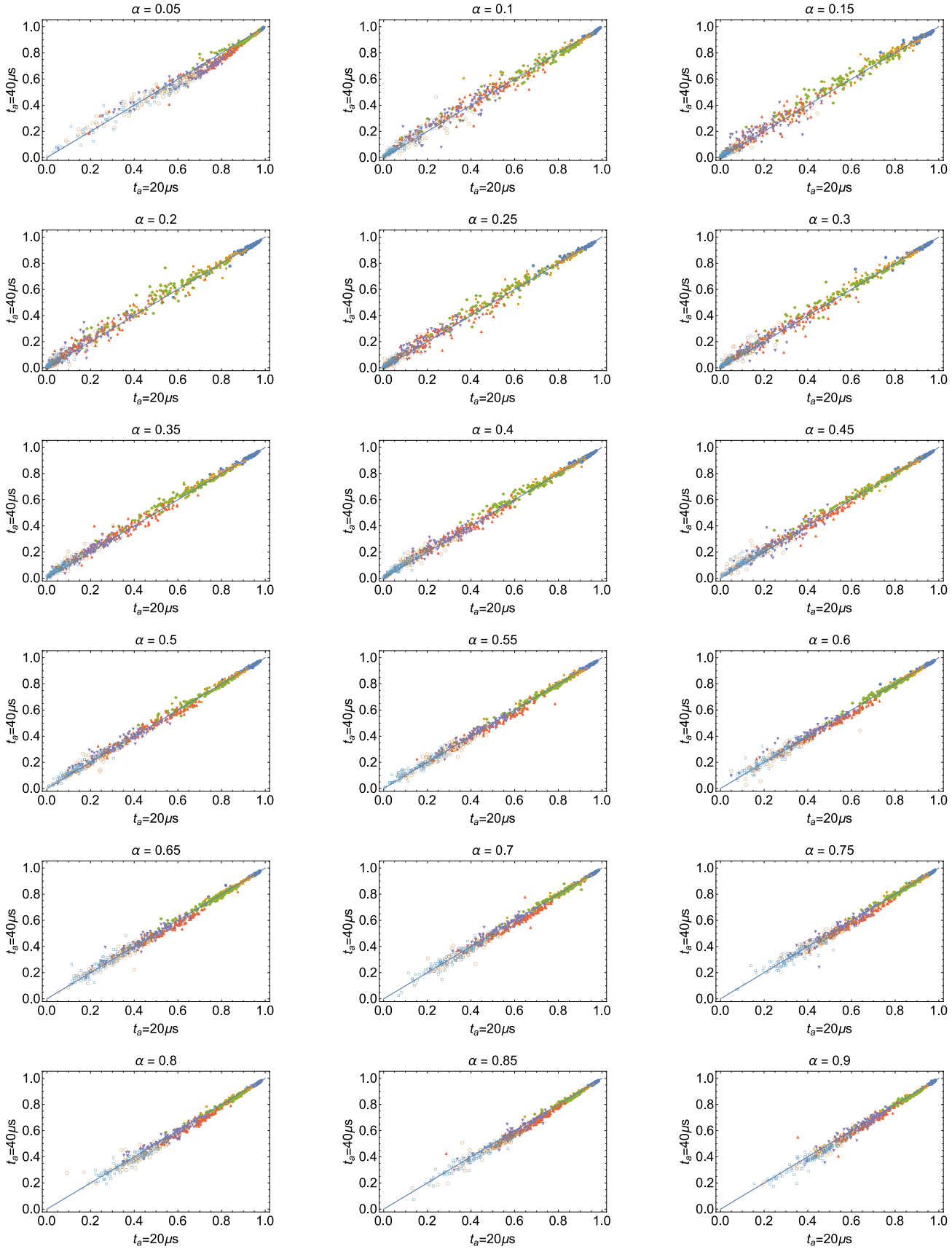


FIG. 22. **Success probability correlations.** The results for all instances at all α values are shown for the DW2 data at $t_a = 20\mu$s and $t_a = 40\mu$s. This complements Fig. 7; the color scheme corresponds to different sizes L as in that figure.

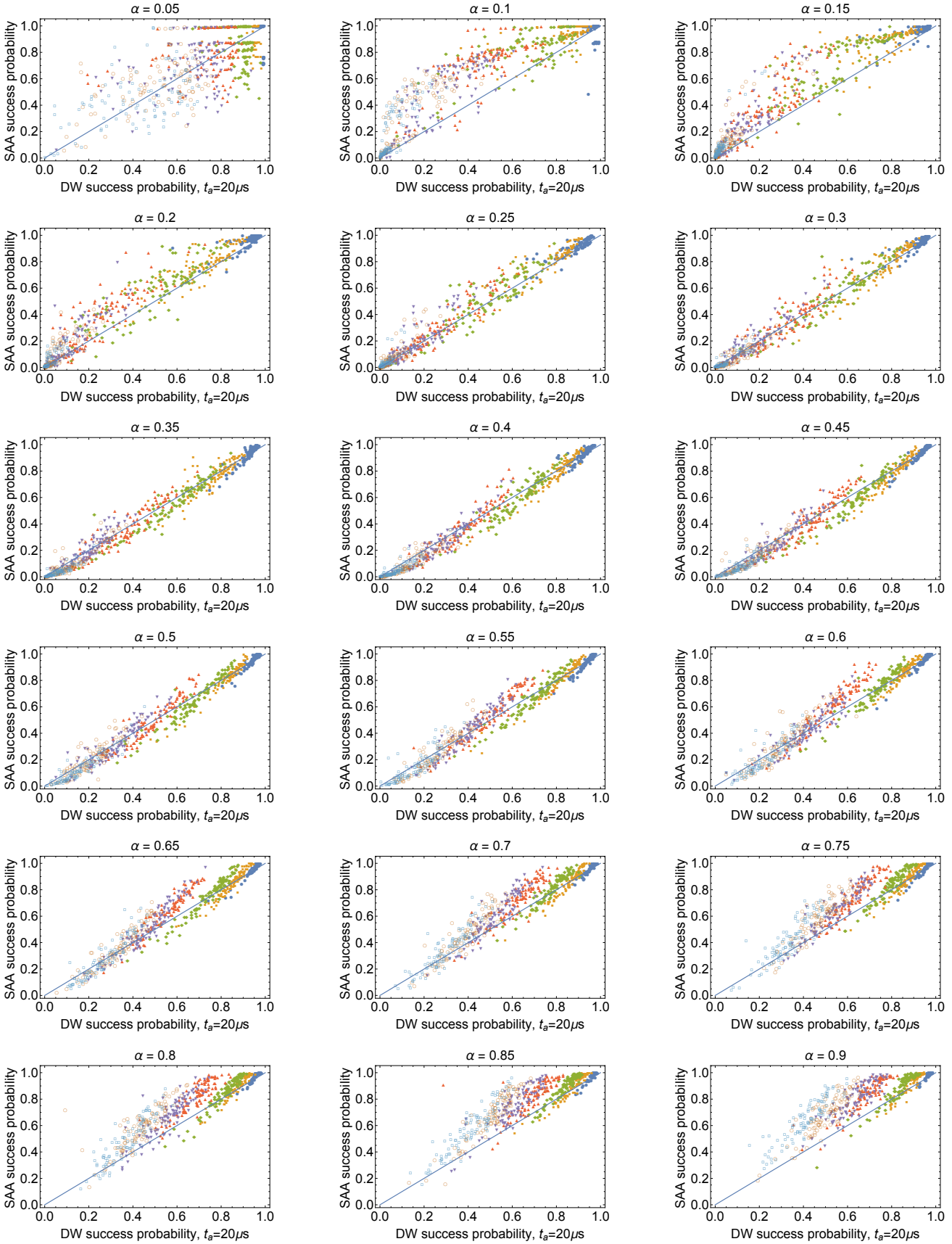


FIG. 23. **Success probability correlations.** The results for all instances at all α values are shown for the DW2 data at $t_a = 20\mu$s and SAA with $S = 50{,}000$ and $\beta_f = 5$. This complements Fig. 7; the color scheme corresponds to different sizes L as in that figure. The correlation gradually improves from poor for the lowest clause densities to strong at $\alpha = 0.35$, then deteriorates again.

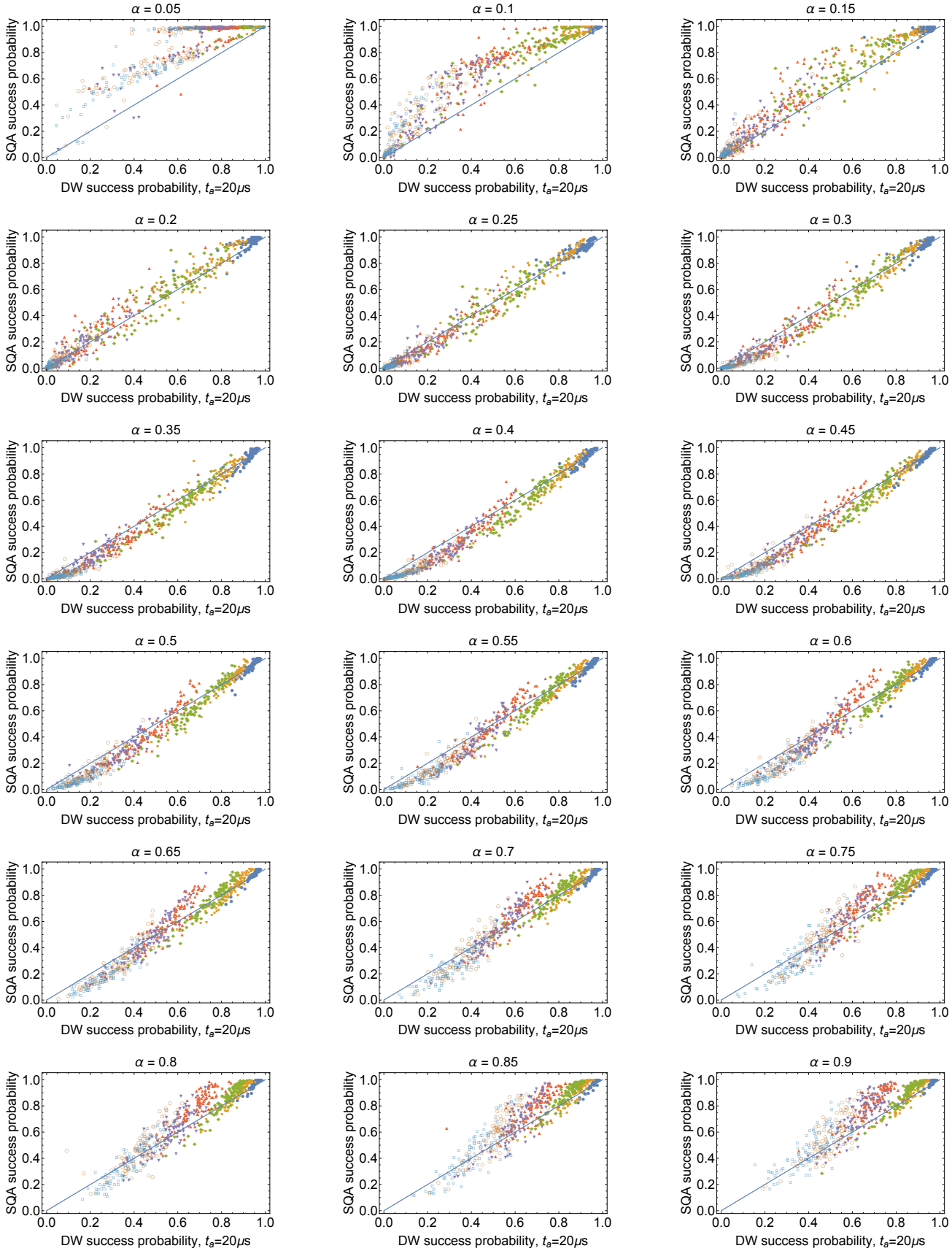


FIG. 24. **Success probability correlations.** The results for all instances at all α values are shown for the DW2 data at $t_a = 20\mu$s and SQAA with $S = 10{,}000$ and $\beta = 5$. This complements Fig. 7; the color scheme corresponds to different sizes L as in that figure. The correlation gradually improves from poor for the lowest clause densities to strong at $\alpha = 0.35$, then deteriorates again.

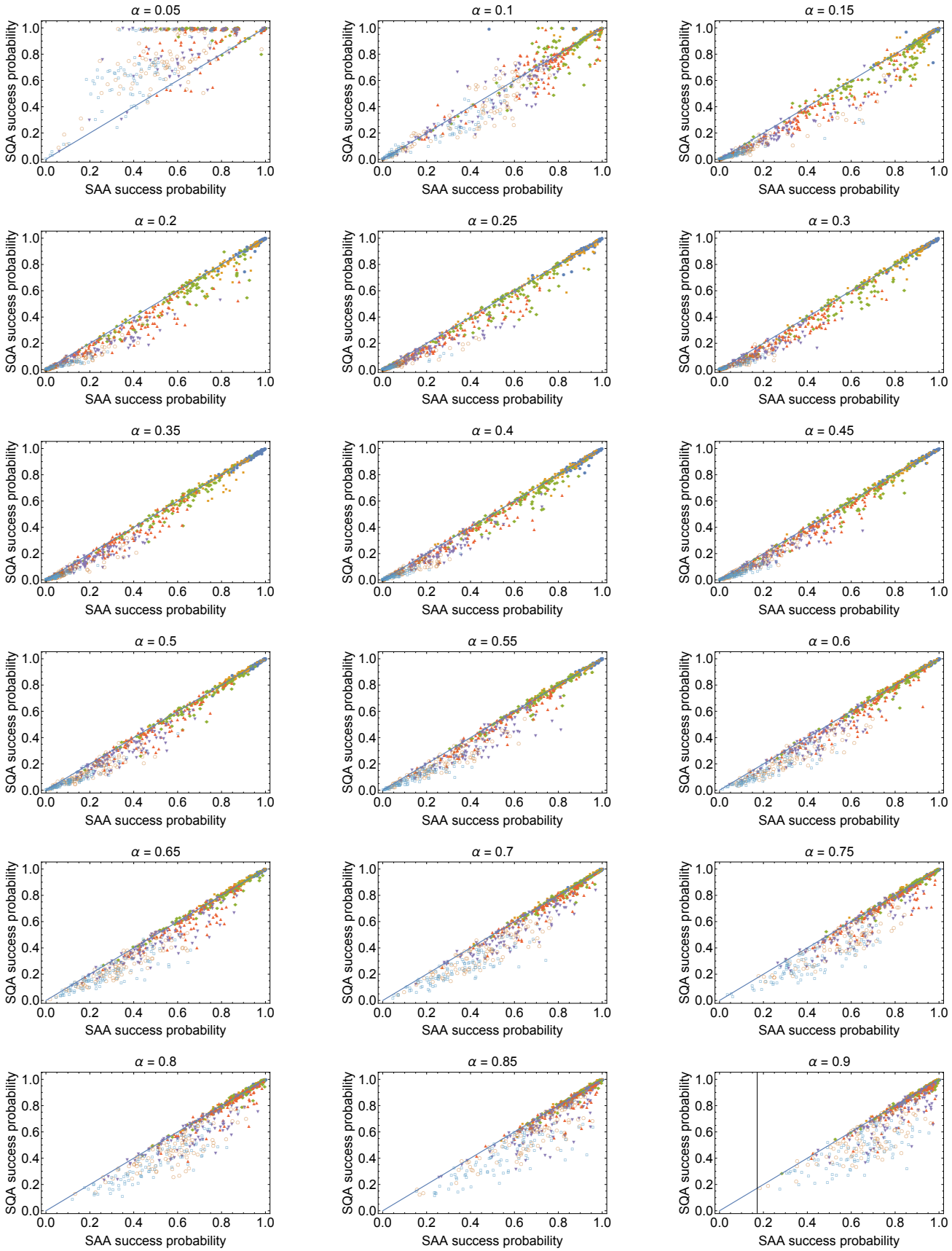
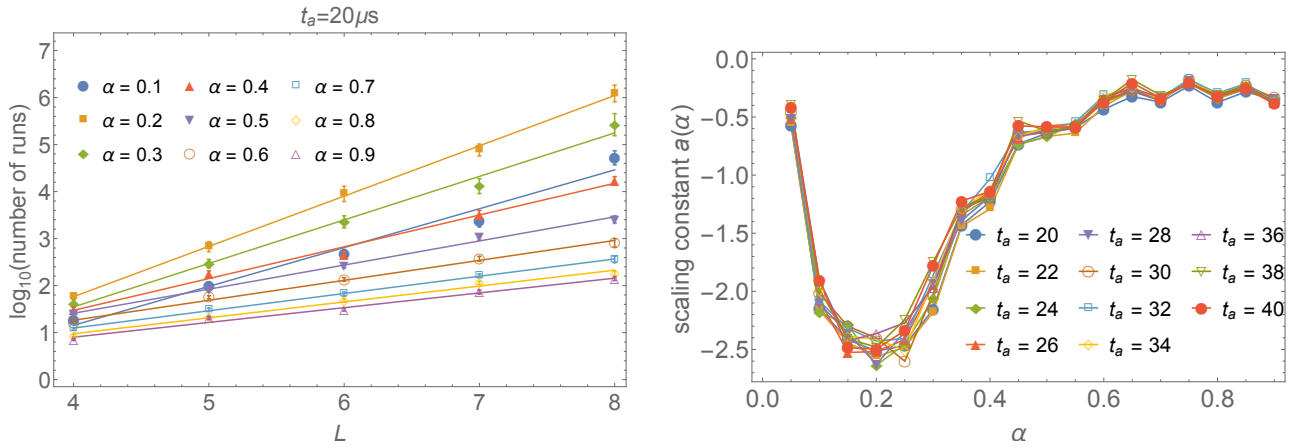
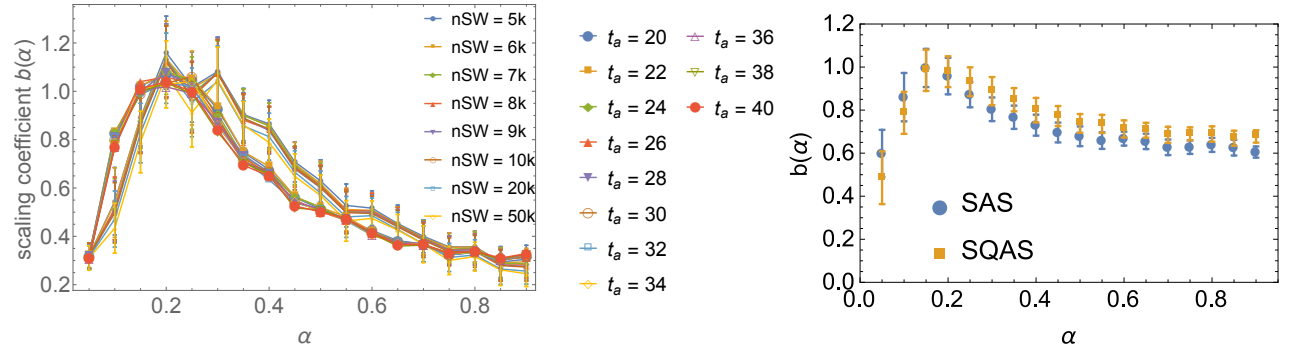


FIG. 25. **Success probability correlations.** The results for all instances at all α values are shown for the SQAA data at $S = 10{,}000$ and $\beta = 5$ and SAA with $S = 50{,}000$ and $\beta_f = 5$. This complements Fig. 7; the color scheme corresponds to different sizes L as in that figure. The correlation gradually improves from poor for the lowest clause densities to strong at $\alpha = 0.35$, then deteriorates again. Note that we did not attempt to optimize the correlations between the two methods.



(a) (b)

FIG. 26. **Exponential fits to the DW2 number of runs.** In accordance with Eq. (8), the least-squares linear fits to $\log[r(L, \alpha, 0.5)]$ are plotted in (a) for $t_a = 20\mu$s. To reduce finite-size scaling effects we exclude $L = 2, 3$ and perform the fit for $L \geq 4$. The intercept of the linear fits is the the DW2 scaling constant $a(\alpha)$ from Eq. (8), and is shown in (b). Error bars represent ~~1σ~~ 2σ confidence intervals.



(a) (b)

FIG. 27. (a) The DW2 scaling coefficients $b(\alpha)$ from Eq. (8) for SAA at various sweep numbers and $\beta_f = 5$, along with the DW2 scaling coefficients for all t_as we tried. (b) The SQAS and SAS scaling coefficient $b(\alpha)$ at the optimal number of sweeps for each. SAS had a final temperature of $\beta_f = 5$ and SQAS was operated at $\beta = 5$.

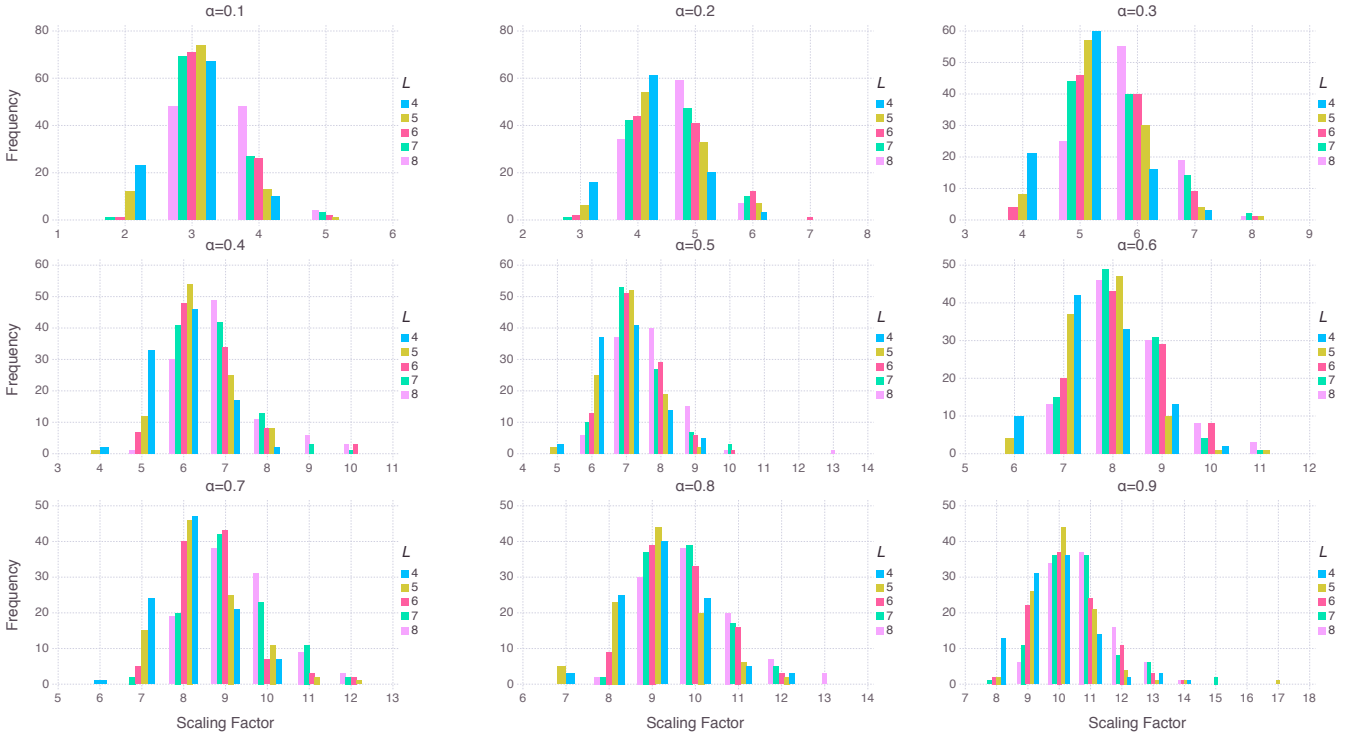


FIG. 28. Histograms of the scale factor of the instances as a function of system size for various values of α. All problems passed to DW2 must have all coupler in the range $[-1, 1]$, so all couplers in a problem are rescaled down by a factor equal to the maximum absolute value of the couplers in the problem (and hence this quantity is called the scale factor). Since internal control error (ICE) is largely instance-independent, the larger the scaling factor of an instance, the worse the relative impact of ICE will be. We see a drift to larger scaling factors for with increasing size L and increasing clause density α. Larger values of α obviously will have larger scale factors as there are on average more loops per qubit (and thus, per edge) and thus a larger maximum potential coupler strength. Since the edges included in a loop are generated randomly, the more edges available at a fixed clause density, the more opportunities for a single edge to be included in many loops by chance, resulting the average scale factor to drift upward as function of problem size.