Quantum speedup of branch-and-bound algorithms

Ashley Montanaro[*]

School of Mathematics, University of Bristol, Bristol, United Kingdom

Branch-and-bound is a widely used technique for solving combinatorial optimization problems where one has access to two procedures: a branching procedure that splits a set of potential solutions into subsets, and a cost procedure that determines a lower bound on the cost of any solution in a given subset. Here we describe a quantum algorithm that can accelerate classical branch-and-bound algorithms near-quadratically in a very general setting. We show that the quantum algorithm can find exact ground states for most instances of the Sherrington-Kirkpatrick model in time $O(2^{0.226n})$, which is substantially more efficient than Grover's algorithm.

Quantum computers can solve certain problems, such as simulation of quantum-mechanical systems [1] and integer factorization [2], exponentially faster than the best classical algorithms known. As well as these special-purpose algorithms, there are general-purpose quantum algorithms which can outperform their classical counterparts more modestly for a wide range of problems within the domains of constraint satisfaction and optimization [3]. A famous example is Grover's algorithm for unstructured search [4], which can be applied to find the minimal value in a set of size N with $O(\sqrt{N})$ evaluations of values in the set [5]. This algorithm achieves a quadratic speedup over exhaustive classical search. However, for many problems encountered in practice, there are more efficient classical algorithms than exhaustive search, which take advantage of the structure of the problem. This can apply to NP-complete problems, which are expected not to have polynomial-time algorithms, yet which can sometimes be solved surprisingly efficiently in practice.

One of the most successful general approaches to solving constrained optimization problems is known as *branch-and-bound*. This approach can be applied to problems where the goal is to find a minimal-cost valid solution, in a setting where one has access to two functions: a bounding function *Cost* that, for a given subset of the set of possible solutions, returns a lower bound on the cost of any valid solution in that subset, and a branching rule *Branch* to be applied if a subset of possible solutions cannot yet be ruled out, which will divide that subset into two or more "live" subsets to be explored in later iterations. Then the branch-and-bound approach explores subsets of potential solutions, ruling out those where the cost of any valid solution is too high (e.g., higher than the lowest cost of a valid solution found so far). The goal is to use this additional information to avoid exploring every possible

solution. Clearly, this approach can equivalently be applied to problems where the goal is to maximize the value of a valid solution.

Although the number of subsets produced by the branching steps can grow exponentially, implying an exponential running time, algorithms based on this technique can sometimes find exact solutions to instances of hard optimization problems substantially beyond the reach of unstructured search. For example, integer linear programming problems can be solved using a branch-and-bound approach where the Cost function is based on relaxing to linear programming problems (see Appendix A for more details).

Here we describe a general quantum approach to accelerate classical branch-and-bound algorithms almost quadratically. The quantum branch-and-bound algorithm can be applied to speed up any classical algorithm that fits into the branch-and-bound paradigm (formally defined below). These include algorithms for integer linear programming, nonlinear programming, the traveling salesman problem, and more [6].

The quantum algorithm is based on quantum subroutines that speed up a related class of classical algorithms: backtracking algorithms [7,8]. Backtracking is an approach that solves constraint satisfaction problems given the ability to determine whether a partial solution to the problem could be extended to a full solution. Backtracking algorithms can be interpreted as the special case of branch-and-bound algorithms where the Cost function either returns 0 (for a valid solution to the problem) or ∞ (for an invalid potential solution, or a partial solution that cannot be extended to a full solution), so the algorithm described here can be seen as generalizing the results of [7,8].

This result contrasts with other quantum approaches to solve hard optimization problems, such as the adiabatic algorithm [9] and the quantum approximate optimization algorithm [10], in that the branch-and-bound algorithm guarantees to find the minimal-cost solution with arbitrarily high probability; however, for certain problems its running time can be long (e.g., exponential in the input size).

One area where branch-and-bound algorithms have been successfully applied classically is finding ground states of spin models [11–14]. For example, branch-and-bound has

_____
[*]ashley.montanaro@bristol.ac.uk

been used to find the largest exact ground states known of instances of the Bernasconi model [15], corresponding to binary sequences with minimal autocorrelation. The fastest known branch-and-bound algorithms for this model have run-time estimated numerically as approximately $O(2^{0.79n})$ [14]; the quantum branch-and-bound algorithm would improve this scaling to approximately $O(2^{0.4n})$.

Another spin model addressed using branch-and-bound is the well-studied Sherrington-Kirkpatrick (S-K) model [16], which is the family of classical Hamiltonians $H(x) = \sum_{1 \leqslant i < j \leqslant n} a_{ij} x_i x_j$ where $x \in \{\pm 1\}^n$ and a_{ij} are distributed according to the normal distribution $N(0, 1)$. Finding the lowest-energy state for such a Hamiltonian can be achieved in time $O(2^{n/2} \text{poly}(n))$ using Grover's algorithm. Recently, Callison *et al.* [17] have described an intriguing quantum algorithm that uses quantum walks to solve the S-K model, and gave evidence based on numerical experiments for small n that the runtime of the algorithm should be approximately $O(2^{0.41n})$. Here we apply the quantum branch-and-bound algorithm to speed up a simple classical branch-and-bound algorithm [12] for the S-K model, and show rigorously that the runtime of the quantum algorithm is $O(2^{0.226n})$ on most instances of size n, which is substantially more efficient than Grover search. Numerical evidence suggests that the runtime in practice could be as low as $O(2^{0.186n})$.

Branch-and-cut methods have also been used to find ground states of spin glasses [13]; these could be accelerated using the same quantum approach.

Model for branch-and-bound algorithms. For a problem to be accessible to branch-and-bound, we must have access to Cost and Branch procedures. Each takes as input a subset S of potential solutions to an optimization problem, perhaps chosen from a restricted family of subsets. Cost(S) returns a lower bound on the cost of any solution within S. Branch(S) either returns that S only contains one element, or splits S into two or more disjoint sets $S_1, \ldots, S_k$. For simplicity, here we assume that k is a fixed constant. For the behavior of Cost to be reasonable, we must have Cost(S) $\geqslant$ Cost(S') whenever $S \subseteq S'$. We assume that for all subsets S, either Cost(S) $\in$ $[0, c_{\max}]$ for some known $c_{\max}$, or Cost(S) $= \infty$, where the latter corresponds to S containing no valid solutions. We can assume essentially without loss of generality that Cost is integer-valued; real-valued cost functions can be handled by truncating their output to precision δ, and multiplying all costs by $1/\delta$. In this situation $c_{\max}$ effectively acts as a precision parameter.

An abstraction of the problem of finding a minimal-cost solution to a problem given access to the Cost and Branch procedures is the following model of search within trees [18], illustrated in Fig. 1. The search space is described by a rooted tree where each node v is either labeled with an integer $c(v)$ between 0 and $c_{\max}$ or with ∞, and satisfies the promise that if w is a child of v, then $c(w) \geqslant c(v)$. We are given query access to two oracles, each of which takes as input a node v. One oracle returns $c(v)$ and the other returns the children of v, if there are any. The goal is to find a leaf node v such that $c(v)$ is minimized, while making the minimal number of queries.

In this abstraction, a node in the tree represents a subset of possible solutions (with a leaf representing a single possible solution), and its label represents a lower bound on the cost
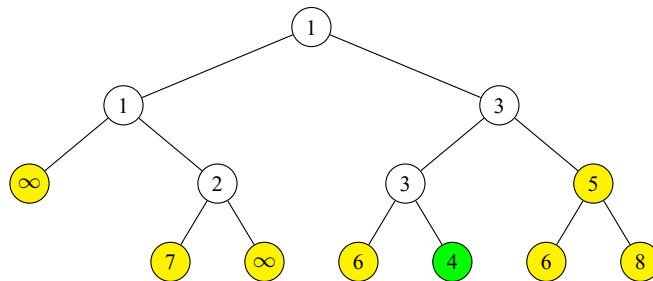

FIG. 1. An example tree corresponding to a branch-and-bound algorithm. Nodes are labeled with their cost bounds, which are nondecreasing on any path from the root to a leaf. The green node is the optimal solution; yellow nodes are removed if the tree is truncated at the optimal cost.

of any solution in that subset. A label of ∞ represents that there is no valid solution in that subset. Revealing the children of a node corresponds to splitting a set of potential solutions into subsets. Note that the backtracking approach for solving constraint satisfaction problems corresponds to exploring a tree of this form where each node is labeled either with 0 or ∞.

The best classical strategy for solving the search problem in such a tree is to maintain a list of "live" nodes (those whose children have not yet been explored), and always to choose to explore the live node with the lowest cost [18]. This strategy is known as best-first search. Although it is optimal in terms of query complexity, it may require a very large amount of space to store the list of live nodes, so in practice other strategies may be preferred (such as depth-first search); strategies which achieve near-optimal query complexity in limited space are known [18].

We will consider particular subtrees of the overall tree, obtained by truncating it at a particular cost c, i.e., deleting all nodes whose labels are greater than c. This is equivalent to changing the Cost function to a Cost$'$ function such that Cost$'(x) = \infty$ if Cost$(x) > c$. This transformation clearly preserves the tree structure and the monotonicity of the Cost function. We also have that truncating the tree at a cost $c_{\min}$ equal to the minimal cost of a valid solution preserves the presence of a minimal-cost valid solution in the tree.

As observed by Karp and Zhang [19], this kind of truncation controls the complexity of a class of classical search algorithms: Any algorithm which outputs all the minimal-cost leaves must explore the entire tree truncated at cost $c_{\min}$. Otherwise, it could not be sure that it had found all the minimal-cost leaves. In particular, if all solutions have distinct costs, any algorithm which outputs the minimal-cost solution must explore the whole tree truncated at that cost.

Statement of results. We can now state our main result. Let $\mathcal{T}$ be the tree corresponding to a branch-and-bound algorithm $\mathcal{A}$. Let d be the depth of $\mathcal{T}$, let $c_{\min}$ be the minimal cost of a valid solution, and let $T_{\min}$ be the size of the truncated tree with cost bound $c_{\min}$. (If there is no solution, $c_{\min} = \infty$, and $T_{\min}$ is the size of the whole tree $\mathcal{T}$.) Fix a constant $\epsilon > 0$. Then there is a quantum algorithm which uses

$$\widetilde{O}(\sqrt{T_{\min}} d^{3/2} \log c_{\max})$$

calls to Cost and Branch, and except with failure probability at most ϵ, returns a solution with minimal cost, if one exists, and otherwise returns "no solution." The $\widetilde{O}$ notation hides polylogarithmic factors in d, $1/\epsilon$, and $\log c_{\max}$.

Assuming that $T_{\min} \gg \text{poly}(d)$, $T_{\min} \gg \log c_{\max}$, this is roughly a quadratic speedup over any possible classical branch-and-bound search algorithm which finds *all* minimal-cost solutions in the tree corresponding to $\mathcal{A}$, whose complexity (as discussed above) is lower-bounded by $T_{\min}$. In the usual case where there is a unique minimal-cost solution, the speedup is approximately quadratic over any possible classical branch-and-bound algorithm (i.e., one that uses only the Cost and Branch procedures).

Quantum branch-and-bound algorithm. In order to state the quantum branch-and-bound algorithm, we will need two quantum-algorithmic ingredients. Both relate to determining properties of trees, in a model where only the root node is known in advance, and the structure of the tree can only be revealed via "oracle" queries to nodes. A query to a node reveals the neighbors of the node and whether the node is marked. This model disallows the straightforward use of methods such as Grover's algorithm to search in the tree.

The first ingredient is *quantum tree search*. It follows from [7,20,21] that there is a quantum algorithm which, given ϵ and oracle access to a tree with depth at most d, N nodes, and maximal degree $k = O(1)$, makes

$$O(\sqrt{N} d^{3/2} \log d \log(1/\epsilon)) \qquad (1)$$

queries and performs $O(1)$ other elementary operations per query, and except with failure probability at most ϵ,

(i) if the tree contains at least one marked node, the algorithm returns the label of a marked node;

(ii) if the tree does not contain any marked nodes, the algorithm returns "not found."

Jarret and Wan have described an algorithm [22] which solves the same tree search problem with complexity bounded by $O(\sqrt{N} d \log^4(md) \log(m/\epsilon))$, where m is the number of marked nodes. This is an improvement on (1) by up to a factor of almost d when m is small. However, if m is very large (e.g., exponential in d), this complexity bound can be larger than (1). Here, we will need to apply quantum tree search in a setting where we have no upper bound on m, which could be as large as N, which in turn can be exponentially large in n. So we will state complexity bounds based on (1), though in some cases (e.g., when m is small) the algorithm of Jarret and Wan may be more efficient.

When we use the quantum tree search algorithm we will have access to an upper bound on the size of the tree. Although, as stated above, having access to such an upper bound is not necessary, it can be seen by inspecting the proof of correctness in [7] that the algorithm is simplified somewhat given this additional information. However, this does not affect its asymptotic complexity.

The second ingredient we will need is *quantum tree size estimation* [8]. The quantum tree size estimation algorithm, given query access to a tree with depth at most d, N nodes, and maximal degree $k = O(1)$, and parameters T_0, ϵ, δ, makes

$$O\left(\frac{\sqrt{T_0} d}{\delta^{3/2}} \log^2(1/\epsilon) \right)$$

Algorithm 1. Quantum branch-and-bound algorithm with failure parameter ϵ.

---

1. Set $T \leftarrow 1$, $c_{\text{old}} \leftarrow 0$, $\epsilon' \leftarrow \epsilon/(Kd \log_2 c_{\max})$ for some sufficiently large constant K.

2. While $T \leqslant T_{\max}$:

    (a) If $T > T_{\max}/2$, $c_{\text{new}} \leftarrow c_{\max}$. Otherwise, set $c_{\text{new}} \leftarrow 0$ and for $i = 1$ to $\log_2 c_{\max}$:

        i. If $\text{Count}_{c_{\text{new}} + c_{\max}/2^i}(T, \epsilon', 1/2)$ does not return "contains more than T nodes," set

$$c_{\text{new}} \leftarrow c_{\text{new}} + c_{\max}/2^i.$$

    (b) Run $\text{Search}_{c_{\text{new}}}(\epsilon')$. If it returns the label of a solution, use binary search on c between c_{old} and c_{new} within $\text{Search}_c(\epsilon')$ to find the minimal c such that a solution with cost c exists, and return that solution.

    (c) Set $T \leftarrow 2T$, $c_{\text{old}} \leftarrow c_{\text{new}}$.

3. Return "no solution."

---

queries and performs $O(\log T_0)$ other elementary operations per query, and except with failure probability at most ϵ,

(i) if $N \leqslant T_0/(1+\delta)$, the algorithm outputs $\widetilde{N} \leqslant T_0$ such that $|\widetilde{N} - N| \leqslant \delta N$;

(ii) if $N > (1+\delta)T_0$, the algorithm outputs "contains more than T_0 nodes."

The restriction to $k = O(1)$ in these algorithms is not significant, as any node with degree k can be replaced with a binary tree of depth $O(\log k)$. Hence, if each node had degree at most k, the tree size N would increase by a constant factor and the depth d would increase by an $O(\log k)$ factor, corresponding to the complexity bounds increasing by a factor polylogarithmic in d.

Let $\text{Count}_c(T_0, \epsilon, \delta)$ denote the quantum tree size estimation algorithm applied to count the number of nodes in the truncated tree corresponding to the branch-and-bound algorithm with cost bound c, where d and k are fixed, and similarly let $\text{Search}_c(\epsilon)$ denote the quantum tree search algorithm applied to search in the truncated tree with cost bound c, where a node is marked if it corresponds to a valid solution to the original problem (i.e., was a leaf with finite cost in the original tree). Assume that we have an upper bound $c_{\max}$ such that all valid solutions have cost strictly less than $c_{\max}$, and an upper bound $T_{\max} \leqslant k^d$ on the size of the tree; further assume for simplicity that $c_{\max}$ is a power of 2.

Then the quantum branch-and-bound algorithm is stated formally as Algorithm 1. The intuitive idea behind it is as follows: we want to find a cost $c \geqslant c_{\min}$ such that the size T_c of the tree truncated at cost c is not much greater than $T_{\min}$. Given such a c, we can find a minimal-cost solution within the tree using Search_c, with a complexity of $O(\sqrt{T_{\min}} \text{poly}(d))$ queries. And to find such a c efficiently, we can perform a binary search on c using Count_c, to find the maximal c such that the tree size is smaller than some upper bound, which is not much greater than $T_{\min}$. There is the technicality that $\text{Count}_c(T, \epsilon, \delta)$ may not return the correct answer for some choices of c and T. However, this turns out not to affect the correctness or performance of the binary search, because $\text{Count}_c(T, \epsilon, \delta)$ does always return the correct answer when

the size of the tree truncated at c is sufficiently small with respect to T. The formal proof of correctness and runtime of the algorithm is deferred to Appendix B.

Sherrington-Kirkpatrick spin glass. The Sherrington-Kirkpatrick (S-K) model [16] is the family of classical Hamiltonians $H(x) = \sum_{1 \leqslant i < j \leqslant n} a_{ij} x_i x_j$ where $x \in \{\pm 1\}^n$ and a_{ij} are distributed according to the normal distribution $N(0, 1)$. Let $A = (a_{ij})$ denote the corresponding square matrix where $a_{ij} = 0$ for $i \geqslant j$. Given a Hamiltonian H described by the matrix A, our computational task is to determine $E_{\min}(H) := \min_x H(x)$.

Determination of the limiting form of the expected ground state energy $E_{\min} := \mathbb{E}_H[E_{\min}(H)]$ as $n \to \infty$ has been a question of extensive interest within the theory of spin glasses. A precise limiting expression is known for this quantity, which evaluates to $E_{\min} = [-0.763167 \cdots + o(1)] n^{3/2}$. This formula was conjectured by Parisi [23] and later proven correct by Talagrand [24]. Although an explicit expression, evaluating it numerically is nontrivial; however, the constant factor is now known to many digits of precision [25,26].

Finding the ground state energy of general Ising model Hamiltonians (of the form of H with arbitrary coefficients a_{ij}) is NP-hard, which holds even given locality restrictions on the pairs i, j such that $a_{ij} \neq 0$ [27]. The S-K model is a natural family of Ising model Hamiltonians, and has been a target for a number of different algorithmic approaches, both heuristic (e.g., [28,29]) and exact [11,12,30]. Although it was recently proven that ground state energies of the S-K model can be approximated efficiently [31], there is no known efficient (i.e., polynomial time in n) method to compute them exactly.

There is a straightforward exact approach to computing $E_{\min}(H)$ which fits into the branch-and-bound paradigm and was proposed in [12]. Variables $x_1, \ldots, x_n \in \{\pm 1\}$ are assigned values sequentially. To determine a lower bound on the cost of any assignment beginning with a partial assignment $x_1, \ldots, x_\ell$, we observe that

$$H(x_1, \ldots, x_\ell, z_{\ell+1}, \ldots, z_n)$$

$$= \sum_{1 \leqslant i < j \leqslant \ell} a_{ij} x_i x_j + \sum_{i=1}^{\ell} \sum_{j=\ell+1}^{n} a_{ij} x_i z_j + \sum_{\ell+1 \leqslant i < j \leqslant n} a_{ij} z_i z_j$$

and hence

$$\min_z H(x_1, \ldots, x_\ell, z_{\ell+1}, \ldots, z_n)$$

$$= \sum_{1 \leqslant i < j \leqslant \ell} a_{ij} x_i x_j + \min_z \left(\sum_{i=1}^{\ell} \sum_{j=\ell+1}^{n} a_{ij} x_i z_j + \sum_{\ell+1 \leqslant i < j \leqslant n} a_{ij} z_i z_j \right)$$

$$\geqslant \sum_{1 \leqslant i < j \leqslant \ell} a_{ij} x_i x_j - \sum_{j=\ell+1}^{n} \left| \sum_{i=1}^{\ell} a_{ij} x_i \right| + \min_z \sum_{\ell+1 \leqslant i < j \leqslant n} a_{ij} z_i z_j$$

$$=: \text{Bound}_A(x).$$

The first and second components of $\text{Bound}_A(x)$ can be computed from x in time $O(n^2)$. The third component is equal to $\min_x H'(x)$, where H' is formed from H by deleting the first ℓ rows and columns of the matrix $A = (a_{ij})$. Thus, if we have already solved the $n - 1$ minimization problems obtained by restricting A to the corresponding submatrices,

we can compute $\text{Bound}_A(x)$ efficiently. As the runtime of the overall algorithm is exponential in n, this additional cost does not significantly affect its overall complexity.

This algorithm fits into the formal model for branch-and-bound discussed above in a straightforward way: the branch-and-bound tree is a binary tree, where nodes at depth ℓ correspond to ℓ-bit strings, for $0 \leqslant \ell \leqslant n$, and the Cost function is Bound_A. So the quantum branch-and-bound algorithm can immediately be applied to accelerate it quadratically. Note that Cost should be non-negative and integer-valued; this can be achieved by truncating the real values a_{ij} after p binary digits to produce a new Hamiltonian $\widetilde{H}$, rescaling by 2^p, and performing an overall energy shift. It is sufficient to take $p = O(\log n)$ to achieve $E_{\min}(\widetilde{H}) = E_{\min}(H) + o(1)$. This corresponds to a cost bound for the integer-valued problem $c_{\max} = \text{poly}(n)$.

Let T_A denote the size of the branch-and-bound tree corresponding to the Hamiltonian H with matrix A, using cost function Bound_A, truncated optimally at cost $E_{\min}(H)$. In Appendix C we prove that for sufficiently large n, $\Pr_A[T_A \geqslant 2^{0.451n}] \leqslant 0.01$. So for the vast majority of instances of size n, the runtime of the classical branch-and-bound algorithm is $O(2^{0.451n})$ steps, which is already faster than Grover's algorithm. The runtime of the quantum branch-and-bound algorithm on these instances is $O(2^{0.226n})$ steps, observing that the depth d of the branch-and-bound tree is equal to n, and $\log c_{\max} = O(\log n)$, so these only contribute lower-order terms to the complexity. The constant 0.01 could be made arbitrarily small.

This rigorous result is only an upper bound on the tree size of the classical algorithm (and by extension the runtime of the quantum algorithm), which may not be tight. To investigate this, a depth-first variant of the classical algorithm was implemented and run on instances of the S-K model for $n \leqslant 50$. Extrapolation of the tree sizes obtained suggests that the true scaling of T_A for random A is $O(2^{0.371n})$, which would correspond to a quantum runtime of $O(2^{0.186n})$. See Appendix E for further details.

Conclusions and further work. We have described a quantum algorithm which can accelerate classical branch-and-bound algorithms in a very general setting. We finish by discussing potential routes to improving the results of this work. First, in some cases the square-root dependence of the runtime on $T_{\min}$ cannot be improved; even in the special case where the Cost function either evaluates to 0 or ∞, there are depth-d trees for which the quantum algorithm requires $\Omega(\sqrt{T_{\min} d})$ queries to determine whether there is a leaf of cost 0 [32]. However, it might be possible to improve the depth dependence by a factor of up to d, e.g., by extending ideas of [22].

Ambainis and Kokainis [8] have given a quantum algorithm which, given a deterministic classical algorithm that explores a search tree to find a marked node and uses Q queries to do so, can find a marked node using $\widetilde{O}(\sqrt{Q} d^{3/2})$ queries. This improves on the quantum tree search algorithm used here in that its complexity depends on Q, rather than the size T of the whole tree. It seems unclear whether this algorithm could be applied to accelerate Algorithm 1 directly, for at least two reasons. First, the algorithm as designed also uses the Count subroutine, whose runtime depends on the entire

tree size. Second, in practice any classical or quantum branch-and-bound algorithm is likely to explore the whole tree of potential solutions, in order to ensure that it has not omitted any solutions with lower cost than the current best solution.

Finally, in order to determine whether the quantum branch-and-bound algorithm will genuinely outperform the best classical methods for problems of practical interest, once all overheads are taken into account, a more detailed analysis of the algorithm's runtime should be undertaken, extending previous analysis for backtracking [33].

Supporting data are available at the University of Bristol data repository, data.bris [34].

APPENDIX A: EXAMPLE: INTEGER LINEAR PROGRAMMING

To gain some intuition for how the results presented here could be applied, in this Appendix we describe one simple and well-known application of branch-and-bound techniques: integer linear programming. An integer linear program (ILP) is a problem of the form

$$\text{minimize } c^T x$$
$$\text{subject to } Ax \geqslant b,$$
$$x \geqslant 0,$$
$$x \in \mathbb{Z}^n,$$

where b and c are vectors, A is a matrix, and inequalities are interpreted componentwise. Integer linear programming problems have many applications, including production planning, scheduling, capital budgeting, and depot location [35].

We can solve ILPs using branch-and-bound as follows. We begin by finding a lower bound on the optimal solution to the ILP, by relaxing it to a standard linear program (LP) and solving the LP, that is, removing the constraint $x \in \mathbb{Z}^n$. This corresponds to the Cost function. If the solution s is integer-valued, we are done, as it corresponds to a valid solution to the ILP. Otherwise, consider an index i such that the found solution value s_i is not an integer. To implement branching, we consider the two ILPs formed by introducing the constraints $x_i \leqslant \lfloor s_i \rfloor$, $x_i \geqslant \lceil s_i \rceil$. At least one of these must have the same optimal solution as the original ILP. We then repeat with these new ILPs. An appealing aspect of this method is that the solution to the relaxation simultaneously tells us a lower bound on the cost, and a good variable to branch on.

The sequences of additional constraints specify subsets of potential solutions to the overall ILP. The Branch and Cost

functions take this sequence as input and solve the resulting LP, to make a decision about which variable to branch on next, and compute a lower bound on cost, respectively. The complexity of the LP-solving step is polynomial in the input size, so the primary contribution to the overall runtime will in general be the exponential scaling in terms of the number of branching steps. A standard classical method could be used (e.g., the simplex algorithm), or one of the recently developed quantum algorithms for linear programming [36–40].

A particularly simple and elegant special case of this approach is the knapsack problem. Here we are given a list of n items, each with weights w_i and values v_i, and an overall weight upper bound W. We seek to find a subset S of the items that maximizes $\sum_{i \in S} v_i$, given that $\sum_{i \in S} w_i \leqslant W$. We can write this as an integer linear program as follows:

$$\text{maximize } \sum_{i=1}^{n} v_i x_i$$
$$\text{subject to } \sum_{i=1}^{n} w_i x_i \leqslant W,$$
$$x_i \in \{0, 1\} \text{ for all } i.$$

Each variable x_i corresponds to whether the ith item is included in the knapsack. Then the LP relaxation is simply to replace the constraint $x_i \in \{0, 1\}$ with the constraint $0 \leqslant x_i \leqslant 1$ for all i. This is equivalent to allowing fractional amounts of each item to be included.

The branch-and-bound approach to solving ILPs can immediately also be applied to the generalization to mixed integer linear programming, where only certain variables are constrained. Now we only branch on those variables which are forced to be integers. One can also apply it to "branch and cut" algorithms. In this approach, when the LP relaxation returns a non-integer-valued solution, one may also add a new constraint (hyperplane) which separates that solution from all integer-valued feasible solutions.

APPENDIX B: ANALYSIS OF ALGORITHM 1

In this Appendix we prove the correctness and the claimed runtime bound of Algorithm 1.

Theorem 1. Let $c_{\min}$ be the minimal cost of a valid solution, and let $T_{\min}$ be the size of the truncated tree with cost bound $c_{\min}$. (If there is no solution, $c_{\min} = \infty$, and $T_{\min}$ is the size of the whole tree.) Algorithm 1 uses

$$O\left(\sqrt{T_{\min} d} \log c_{\max} \log \left(\frac{d \log c_{\max}}{\epsilon} \right) \right.$$
$$\left. \times \left[\log \left(\frac{d \log c_{\max}}{\epsilon} \right) + d \log d \right] \right)$$

oracle calls, and except with failure probability at most ϵ, returns a solution with minimal cost, if one exists, and otherwise "no solution."

Proof. We first show that the algorithm succeeds with probability at least $1 - \epsilon$. The loop executes at most $O(d)$ times, so each of Count and Search is used at most $O(d \log c_{\max})$ times. By a union bound, it is sufficient to pick $\epsilon' = O(\epsilon/(d \log c_{\max}))$ to ensure that all the uses of Count and

Search succeed, except with total probability at most ϵ. So we henceforth assume that Count and Search do always succeed.

If this is the case, we first observe that the algorithm always correctly outputs a minimal-cost solution, if one exists, or otherwise "no solution." This is because at the final iteration (when $T > T_{\max}/2$), if no solution has previously been found then Search will explore the entire tree and find a solution if one exists. To see that it outputs a *minimal-cost* solution, note that the binary search on c using Search is over the range $[c_{\text{old}}, c_{\text{new}}]$, and c_{old} is no larger than the largest value of c_{new} previously computed, so any solution with cost smaller than c_{old} would have been found in a previous iteration.

It remains to prove the runtime bound. Let T_c denote the size of the truncated tree with cost bound c (so $T_{\min} = T_{c_{\min}}$). The first binary search (in part 2a) executes Count_c $O(\log c_{\max})$ times, each iteration using $O(\sqrt{Td}\log^2(1/\epsilon'))$ queries, and the second binary search executes $\text{Search}_{c_{\text{new}}}$ $O(\log c_{\max})$ times, where each iteration uses $O(\sqrt{T_{c_{\text{new}}}}d^{3/2}\log d \log(1/\epsilon'))$ queries. At each iteration of the loop, after the binary search using Count_c, $T_{c_{\text{new}}} \leqslant 3T/2$ by correctness of the quantum tree size estimation algorithm. Further, at the first iteration when $T \geqslant 3T_{\min}/2$ (if such an iteration occurs), for all $c \leqslant c_{\min}$, Count_c does not return "contains more than T nodes." This implies that $c_{\text{new}} \geqslant c_{\min}$, because as the binary search terminated at cost c_{new}, $\text{Count}_{c_{\text{new}}+1}$ must have returned "contains more than T nodes." Note that this holds even though Count_c can return an arbitrary outcome when $T_c \in (2T/3, 3T/2)$.

Therefore, at this iteration the tree truncated at cost c_{new} contains a minimal-cost solution, which will be found by the binary search on c using Search_c, and the algorithm will terminate. On the other hand, if there is no iteration such that $T \geqslant 3T_{\min}/2$, we must have $T_{\min} > 2T_{\max}/3$. Combining these two claims, we have $T \leqslant 3T_{\min}$ throughout the algorithm. The loop over exponentially increasing values of T does not affect the overall complexity bound, so the overall complexity is

$$O\left(\sqrt{T_{\min}d}\log c_{\max}\log\left(\frac{d\log c_{\max}}{\epsilon}\right)\right.$$
$$\left.\times\left[\log\left(\frac{d\log c_{\max}}{\epsilon}\right) + d\log d\right]\right)$$

queries, as claimed. ∎

We remark that it seems that, in general, Algorithm 1 could not be replaced with simply using the Search subroutine with exponentially increasing values of the cost parameter (an approach taken in [41] for the special case of accelerating backtracking algorithms for the traveling salesman problem). This is because increasing the cost at which the tree is truncated by a constant factor could increase the size of the truncated tree substantially beyond $T_{\min}$.

APPENDIX C: TRUNCATED TREE SIZE BOUND FOR SHERRINGTON-KIRKPATRICK MODEL

In this Appendix, let T_A denote the size of the tree corresponding to the classical branch-and-bound algorithm applied

to find the ground state energy of an Ising Hamiltonian H corresponding to an $n \times n$ matrix A, using the bounding function Bound_A described in the main text, where the tree is truncated at the optimal value $c_{\min}$.

We will prove the following result:

Theorem 2. Let A be an $n \times n$ matrix corresponding to a Sherrington-Kirkpatrick model instance on n spins. For all sufficiently large n,

$$\Pr_A[T_A \geqslant 2^{0.451n}] \leqslant 0.01.$$

The dominant term in the quantum complexity is the square root of the classical complexity, which is determined by T_A, so Theorem 2 implies that the quantum branch-and-bound algorithm has an $O(2^{0.226n})$ running time on 99% of Sherrington-Kirkpatrick model instances.

In order to prove Theorem 2, we will need two technical lemmas, proven in Appendix D.

Lemma 1. Let $x_1, \ldots, x_N \sim N(0, 1)$. Let $f : \mathbb{R}^N \to \mathbb{R}$ be continuous and 1-Lipschitz in each coordinate separately, i.e., $|f(x_1, \ldots, x_N) - f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_N)| \leqslant |x_i - x_i'|$ for all $i \in [N]$. Then

$$\Pr[f(x) \geqslant \mathbb{E}_x[f(x)] + t] \leqslant e^{-t^2/(2N)},$$
$$\Pr[f(x) \leqslant \mathbb{E}_x[f(x)] - t] \leqslant e^{-t^2/(2N)}.$$

Lemma 1 was shown in [42] but with an incorrect constant. We will also need a bound on the expectation $\mathbb{E}_A[\min_{z \in \{\pm 1\}^n} z^T A z]$. A precise value for this is known as $n \to \infty$, but we will need a bound that holds for arbitrary n:

Lemma 2. Let A be an $n \times n$ matrix corresponding to a Sherrington-Kirkpatrick model instance on n spins. For all n, $\mathbb{E}_A[\min_{z \in \{\pm 1\}^n} z^T A z] \geqslant -0.601\sqrt{n} - 0.833n^{3/2}$.

We are now able to prove Theorem 2. The basic strategy is to upper-bound the expected value of T_A, using that (by linearity of expectation) this can be expressed as a sum over all bit strings $x \in \{\pm 1\}^\ell$, $0 \leqslant \ell \leqslant n$, of the probability that the node corresponding to x is contained within the truncated branch-and-bound tree. These bit strings x are precisely those such that $\text{Bound}_A(x) \leqslant \min_{z \in \{\pm 1\}^n} z^T A z$, and a tail bound can be used to upper-bound the probability that this event occurs.

There are two technical difficulties which need to be handled. First, this approach does not give a good upper bound in the case where $\min_{z \in \{\pm 1\}^n} z^T A z$ is high, which can occur with non-negligible probability, leading to $\mathbb{E}_A[T_A]$ becoming large. We therefore handle this case separately and show that it occurs with low probability. Next, to find a tail bound on $\text{Bound}_A(x)$, we need to compute expressions of the form $\mathbb{E}_A[\min_{z \in \{\pm 1\}^{n-\ell}} z^T A z]$; although a limiting form for this is known [23,24,26], we will additionally need relatively tight bounds in the case $\ell \approx n$. We therefore split into cases $\ell \leqslant 0.9n$ [where $(n - \ell) \to \infty$ and we can use the precise limiting result] and $\ell > 0.9n$ (where we use Lemma 2).

Proof of Theorem 2. Write $\mu = \mathbb{E}_A[\min_{z \in \{\pm 1\}^n} z^T A z]$, and let $\gamma > 0$ be an arbitrary value to be determined. We will upper-bound the probability that $T_A \geqslant B$ for some B as follows, where we use the notation $[Y]$ for the indicator random

variable which evaluates to 1 if Y is true, and 0 if Y is false:

$$
\begin{aligned}
\Pr_A[T_A \geqslant B] &= \Pr_A\left[T_A \geqslant B \wedge \min_z z^T A z \leqslant \mu + \gamma n^{3/2}\right] + \Pr_A\left[T_A \geqslant B \wedge \min_z z^T A z > \mu + \gamma n^{3/2}\right] \\
&\leqslant \Pr_A\left[T_A\left[\min_z z^T A z \leqslant \mu + \gamma n^{3/2}\right] \geqslant B\right] + \Pr_A\left[\min_z z^T A z > \mu + \gamma n^{3/2}\right] \\
&\leqslant \frac{1}{B}\mathbb{E}_A\left[T_A\left[\min_z z^T A z \leqslant \mu + \gamma n^{3/2}\right]\right] + \Pr_A\left[\min_z z^T A z > \mu + \gamma n^{3/2}\right] \\
&= \frac{1}{B}\sum_{\ell=0}^n \sum_{x \in \{\pm 1\}^\ell} \mathbb{E}_A\left[\left[\mathrm{Bound}_A(x) \leqslant \min_z z^T A z\right]\left[\min_z z^T A z \leqslant \mu + \gamma n^{3/2}\right]\right] + \Pr_A\left[\min_z z^T A z > \mu + \gamma n^{3/2}\right] \\
&\leqslant \frac{1}{B}\sum_{\ell=0}^n \sum_{x \in \{\pm 1\}^\ell} \Pr_A\left[\mathrm{Bound}_A(x) \leqslant \mu + \gamma n^{3/2}\right] + \Pr_A\left[\min_z z^T A z > \mu + \gamma n^{3/2}\right] \\
&\leqslant \frac{n+1}{B}\max_{\ell, x \in \{\pm 1\}^\ell} 2^\ell \Pr_A\left[\mathrm{Bound}_A(x) \leqslant \mu + \gamma n^{3/2}\right] + \Pr_A\left[\min_z z^T A z > \mu + \gamma n^{3/2}\right],
\end{aligned}
$$

where the second inequality is Markov's inequality and we use linearity of expectation in the second equality.

To upper-bound the last term, we use Lemma 1. We first observe that $f(A) = \min_z z^T A z$ is 1-Lipschitz in each variable, as if we modify A to produce A' by changing a_{pq} to a'_{pq} for some pair $p < q$,

$$
\min_{z \in \{\pm 1\}^n} \sum_{i<j} a'_{ij} z_i z_j = \min_{z \in \{\pm 1\}^n}\left((a'_{pq} - a_{pq})z_p z_q + \sum_{i<j} a_{ij} z_i z_j\right) \geqslant -|a'_{pq} - a_{pq}| + \min_{z \in \{\pm 1\}^n}\sum_{i<j} a_{ij} z_i z_j, \tag{C1}
$$

and by a similar argument $\min_{z \in \{\pm 1\}^n}\sum_{i<j} a'_{ij} z_i z_j \leqslant |a'_{pq} - a_{pq}| + \min_{z \in \{\pm 1\}^n}\sum_{i<j} a_{ij} z_i z_j$. So Lemma 1 implies that

$$
\Pr_A\left[\min_z z^T A z > \mu + \gamma n^{3/2}\right] \leqslant e^{-(\gamma n^{3/2})^2/[2\binom{n}{2}]} \leqslant e^{-\gamma^2 n}.
$$

For this to be upper-bounded by a small constant (e.g., 0.005) we can take $\gamma = O(1/\sqrt{n})$.

We next upper-bound the first term by bounding $\Pr_A[\mathrm{Bound}_A(x) \leqslant \mu + \gamma n^{3/2}]$. We only need to consider $\ell \geqslant 0.4n$ in the maximization, because when $\ell \leqslant 0.4n$, trivially upper-bounding this probability by 1 already gives a sufficiently strong bound. Recall that

$$
\mathrm{Bound}_A(x) = \sum_{1 \leqslant i < j \leqslant \ell} a_{ij} x_i x_j - \sum_{j=\ell+1}^n \left|\sum_{i=1}^\ell a_{ij} x_i\right| + \min_z \sum_{\ell+1 \leqslant i < j \leqslant n} a_{ij} z_i z_j.
$$

The function $f_x(A) = \mathrm{Bound}_A(x)$ is 1-Lipschitz in each variable by a similar argument to (C1). Thus, for any $\eta \geqslant 0$,

$$
\Pr_A[\mathrm{Bound}_A(x) \leqslant \mathbb{E}_A[\mathrm{Bound}_A(x)] - \eta n^{3/2}] \leqslant e^{-\eta^2 n}. \tag{C2}
$$

First assume that $0.4n \leqslant \ell \leqslant 0.9n$, so $(n - \ell) \to \infty$ as $n \to \infty$. For any $x \in \{\pm 1\}^\ell$, we have

$$
\mathbb{E}_A[\mathrm{Bound}_A(x)] = \sum_{1 \leqslant i < j \leqslant \ell} \mathbb{E}_A[a_{ij}] x_i x_j - \sum_{j=\ell+1}^n \mathbb{E}_A\left[\left|\sum_{i=1}^\ell a_{ij} x_i\right|\right] + \mathbb{E}_A\left[\min_z \sum_{\ell+1 \leqslant i < j \leqslant n} a_{ij} z_i z_j\right] \tag{C3}
$$

$$
= -(n - \ell)\mathbb{E}_A\left[\left|\sum_{i=1}^\ell a_{ij}\right|\right] + \mathbb{E}_A\left[\min_z \sum_{\ell+1 \leqslant i < j \leqslant n} a_{ij} z_i z_j\right] \tag{C4}
$$

$$
= -(n - \ell)\sqrt{\frac{2}{\pi}}\sqrt{\ell} - [0.763\cdots + o(1)](n - \ell)^{3/2}, \tag{C5}
$$

where we use linearity of expectation to obtain the first expression, that $\sum_{i=1}^{\ell} a_{ij} \sim \sqrt{\ell} N(0,1)$, and the known limiting result $\mathbb{E}_A[\min_z \sum_{\ell+1 \leqslant i < j \leqslant n} a_{ij} z_i z_j] = [-0.763167 \cdots + o(1)](n-\ell)^{3/2}$ [23,24,26].

Writing $\alpha = \ell/n$, we have that

$$\mathbb{E}_A[\text{Bound}_A(x)] = \left\{ -(1-\alpha)\sqrt{\alpha}\sqrt{\frac{2}{\pi}} - [0.763 \cdots + o(1)](1-\alpha)^{3/2} \right\} n^{3/2} =: g_1(\alpha)n^{3/2}.$$

On the other hand, for $\ell \geqslant 0.9n$, we follow a similar argument but apply the nonasymptotic result of Lemma 2 to bound $\mathbb{E}_A[\min_z \sum_{\ell+1 \leqslant i < j \leqslant n} a_{ij} z_i z_j]$, which implies that

$$\mathbb{E}_A[\text{Bound}_A(x)] \geqslant -(n-\ell)\sqrt{\frac{2}{\pi}}\sqrt{\ell} - 0.601\sqrt{n-\ell} - 0.833(n-\ell)^{3/2} \tag{C6}$$

$$\geqslant -(n-\ell)\sqrt{\frac{2}{\pi}}\sqrt{\ell} - 1.434(n-\ell)^{3/2} \tag{C7}$$

$$= \left[-(1-\alpha)\sqrt{\alpha}\sqrt{\frac{2}{\pi}} - 1.434(1-\alpha)^{3/2} \right] n^{3/2} =: g_2(\alpha)n^{3/2}. \tag{C8}$$

In either case, we have

$$\Pr_A[\text{Bound}_A(x) \leqslant \mu + \gamma n^{3/2}] = \Pr_A[\text{Bound}_A(x) - \mathbb{E}_A[\text{Bound}_A(x)] \leqslant \mu + \gamma n^{3/2} - \mathbb{E}_A[\text{Bound}_A(x)]].$$

By (C2), using $\mu = [-0.763167 \cdots + o(1)]n^{3/2}$ and observing (see Fig. 2) that $\mathbb{E}_A[\text{Bound}_A(x)] > \mu + \gamma n^{3/2}$ for sufficiently large n, so the right-hand side is negative as required, we have

$$\Pr_A[\text{Bound}_A(x) \leqslant \mu + \gamma n^{3/2}] \leqslant \begin{cases} e^{-\{g_1(\alpha) + [0.763 \cdots + o(1)] - \gamma\}^2 n}, & \text{if } 0.4 \leqslant \alpha \leqslant 0.9, \\ e^{-\{g_2(\alpha) + [0.763 \cdots + o(1)] - \gamma\}^2 n}, & \text{if } \alpha \geqslant 0.9. \end{cases}$$

So

$$\max_{\ell \geqslant 0.4n, x \in \{\pm 1\}^{\ell}} 2^{\ell} \Pr_A[\text{Bound}_A(x) \leqslant \mu + \gamma n^{3/2}]$$

$$\leqslant \max \left\{ \max_{\alpha \in [0.4, 0.9]} 2^{\alpha n} e^{-[g_1(\alpha) + 0.763 \cdots + o(1)]^2 n}, \max_{\alpha \in [0.9, 1]} 2^{\alpha n} e^{-[g_2(\alpha) + 0.763 \cdots + o(1)]^2 n} \right\}$$

$$= \max \left\{ \max_{\alpha \in [0.4, 0.9]} 2^{n\{\alpha - [g_1(\alpha) + 0.763 \cdots + o(1)]^2 / \ln 2\}}, \max_{\alpha \in [0.9, 1]} 2^{n\{\alpha - [g_2(\alpha) + 0.763 \cdots + o(1)]^2 / \ln 2\}} \right\},$$

observing that $\gamma = o(1)$. It remains to determine upper bounds on the functions

$$\alpha - \frac{[g_1(\alpha) + 0.763 \cdots + o(1)]^2}{\ln 2} = \alpha - \frac{\left\{ -(1-\alpha)\sqrt{\alpha}\sqrt{\frac{2}{\pi}} + 0.763 \ldots [1 - (1-\alpha)^{3/2}] \right\}^2}{\ln 2} + o(1) =: h_1(\alpha) + o(1), \tag{C9}$$

$$\alpha - \frac{[g_2(\alpha) + 0.763 \cdots + o(1)]^2}{\ln 2} = \alpha - \frac{\left[-(1-\alpha)\sqrt{\alpha}\sqrt{\frac{2}{\pi}} + 0.763 \cdots - 1.434(1-\alpha)^{3/2} \right]^2}{\ln 2} + o(1) =: h_2(\alpha) + o(1). \tag{C10}$$

This can easily be achieved numerically, giving (see Fig. 3) the result that $h_1(\alpha) < 0.45003$ for $0 \leqslant \alpha \leqslant 1$ and $h_2(\alpha) < 0.45003$ for $\alpha \geqslant 0.9$. Hence

$$\Pr_A[T_A \geqslant B] \leqslant \frac{(n+1)2^{[0.45003+o(1)]n}}{B} + 0.005,$$

and to upper-bound the first term by 0.005, for sufficiently large n one can take $B = 2^{0.451n}$. This completes the proof. ∎

APPENDIX D: PROOFS OF TECHNICAL LEMMAS

In this Appendix we prove Lemmas 1 and 2. We say that $f(x_1, \ldots, x_N)$ satisfies the bounded differences condition with

constants d_i, $i \in [N]$, if $|f(x) - f(x')| \leqslant d_i$ whenever x and x' differ only in the ith coordinate.

Lemma 3 (McDiarmid's inequality or method of bounded differences [[43], Corollary 5.2]). If $f(x_1, \ldots, x_N)$ satisfies the bounded differences condition with constants d_i, and $x_1, \ldots, x_N$ are independent random variables, then

$$\Pr[f(x) \geqslant \mathbb{E}_x[f(x)] + t] \leqslant e^{-2t^2/d},$$

$$\Pr[f(x) \leqslant \mathbb{E}_x[f(x)] - t] \leqslant e^{-2t^2/d},$$

where $d = \sum_{i=1}^{N} d_i^2$.

Lemma 1 (restated). Let $x_1, \ldots, x_N \sim N(0,1)$. Let $f: \mathbb{R}^N \to \mathbb{R}$ be continuous and 1-Lipschitz in each coordinate separately, i.e., $|f(x_1, \ldots, x_N) - f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_N)| \leqslant |x_i - x_i'|$ for all $i \in [N]$.
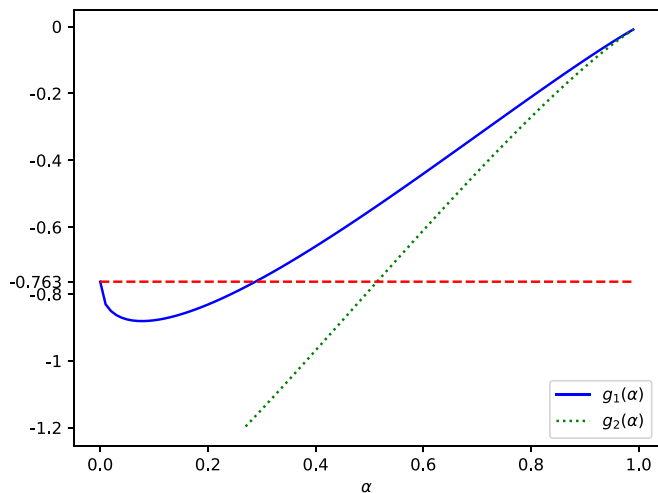
FIG. 2. The functions $g_1(\alpha)$, $g_2(\alpha)$ defined in (C5), (C8). $g_1(\alpha) \geqslant -0.763$ for all $\alpha \geqslant 0.4$, while $g_2(\alpha) \geqslant -0.763$ for all $\alpha \geqslant 0.9$.

Then

$$\Pr\left[f(x) \geqslant \mathbb{E}_x[f(x)] + t\right] \leqslant e^{-t^2/(2N)},$$
$$\Pr\left[f(x) \leqslant \mathbb{E}_x[f(x)] - t\right] \leqslant e^{-t^2/(2N)}.$$

Proof. For $i \in [N]$, $j \in [M]$, let y_i^j be a Rademacher random variable, taking values ± 1 with equal probability. Then define the sequence $x(y)$ by $x(y)_i = \frac{1}{\sqrt{M}} \sum_{j=1}^M y_i^j$. Let $g : \{\pm 1\}^{MN} \to \mathbb{R}$ be defined by setting $g(y) = f(x(y))$. Then changing one entry of y can change $x(y)_i$ by at most $2/\sqrt{M}$, so we can apply Lemma 3 with $d_i = 2/\sqrt{M}$ to obtain

$$\Pr\left[f(x(y)) \geqslant \mathbb{E}_y[f(x(y))] + t\right] \leqslant e^{-t^2/(2N)},$$
$$\Pr\left[f(x(y)) \leqslant \mathbb{E}_y[f(x(y))] - t\right] \leqslant e^{-t^2/(2N)}.$$

As $M \to \infty$, the distribution of $x(y)_i$ approaches a standard normal distribution for all i. The lemma follows. ∎
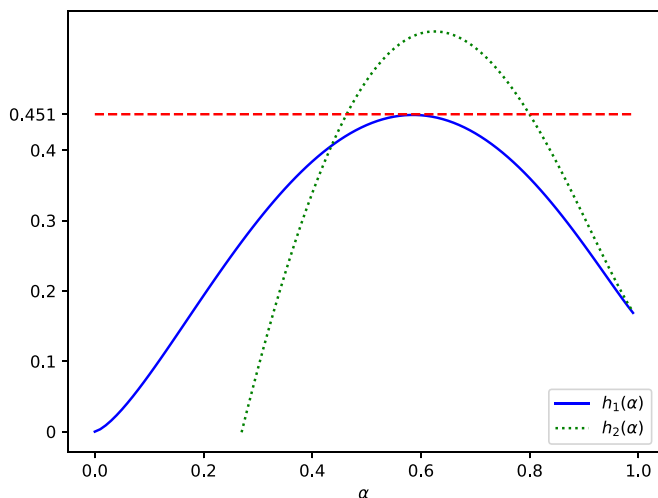

FIG. 3. The functions $h_1(\alpha)$, $h_2(\alpha)$ defined in (C9), (C10). $h_1(\alpha) < 0.45003$ for all α, while $h_2(\alpha) < 0.45003$ for all $\alpha \geqslant 0.9$.

Lemma 2 (restated). Let A be an $n \times n$ matrix corresponding to a Sherrington-Kirkpatrick model instance on n spins. For all n, $\mathbb{E}_A[\min_{z \in \{\pm 1\}^n} z^T A z] \geqslant -0.601\sqrt{n} - 0.833n^{3/2}$.

Proof of Lemma 2. We start with

$$\mathbb{E}_A\left[\min_{z \in \{\pm 1\}^n} z^T A z\right] = -\int_{-\infty}^0 \Pr\left[\min_{z \in \{\pm 1\}^n} z^T A z \leqslant t\right] dt,$$

valid as $\min_{z \in \{\pm 1\}^n} z^T A z$ is nonpositive. Next, for any $t \leqslant 0$ we have

$$\Pr\left[\min_{z \in \{\pm 1\}^n} z^T A z \leqslant t\right] \leqslant 2^n \Pr\left[\sum_{i<j} a_{ij} \leqslant t\right]$$

using a union bound over z and symmetry of the distribution of a_{ij}. By a tail bound on the normal distribution, we have

$$\Pr\left[\sum_{i<j} a_{ij} \leqslant t\right] \leqslant e^{-t^2/[2\binom{n}{2}]} \leqslant e^{-t^2/n^2}$$

for all $t \leqslant 0$. So

$$\mathbb{E}_A\left[\min_{z \in \{\pm 1\}^n} z^T A z\right]$$
$$\geqslant -\int_{-\infty}^0 \min\{1, 2^n e^{-t^2/n^2}\} dt$$
$$= -\int_{-\infty}^{-n^{3/2}\sqrt{\ln 2}} 2^n e^{-t^2/n^2} dt - \int_{-n^{3/2}\sqrt{\ln 2}}^0 1\, dt$$
$$= -n 2^{n-1/2} \int_{-\infty}^{-\sqrt{2n \ln 2}} e^{-t^2/2} dt - \sqrt{\ln 2}\, n^{3/2}$$
$$\geqslant -\frac{\sqrt{n}}{2\sqrt{\ln 2}} - \sqrt{\ln 2}\, n^{3/2}$$
$$= -0.600561\ldots \sqrt{n} - 0.832555\ldots n^{3/2},$$

where we use the bound $\int_a^\infty e^{-x^2/2} dx \leqslant \frac{1}{a} e^{-a^2/2}$ for any $a > 0$ in the second inequality. ∎

APPENDIX E: CLASSICAL NUMERICAL BRANCH-AND-BOUND RESULTS

We implemented the classical branch-and-bound algorithm described in the main text, with cost function Bound_A, using a simple depth-first search procedure within the branch-and-bound tree which backtracks on nodes corresponding to partial solutions with an energy bound worse than the lowest energy seen thus far (see Figs. 4 and 5). For an S-K model instance described by a matrix A, this gives an upper bound on the size T_A of an optimally truncated branch-and-bound tree (equivalently, on the runtime of the best-first search algorithm applied to find the ground state energy, with cost function Bound_A).

This algorithm enabled instances on more than 50 spins to be solved within minutes on a standard laptop computer. We then carried out a least-squares fit on the log of the number of
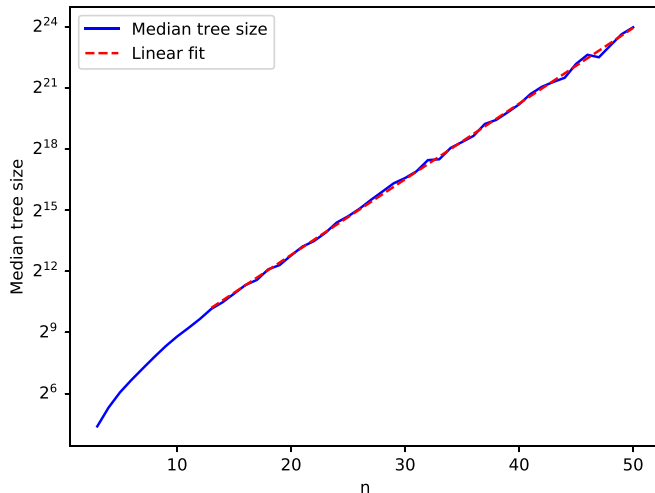
FIG. 4. Median tree size explored by classical branch-and-bound algorithm with depth-first strategy. 99 random instances generated for each n. Fit is line $y = 2^{0.371n+5.380}$.

nodes explored, omitting small n, to estimate the scaling of the algorithm with n. Note that, due to finite-size effects, this may not be accurate for large n; however, it gives an indication of tree size scaling. The median normalized ground state energy
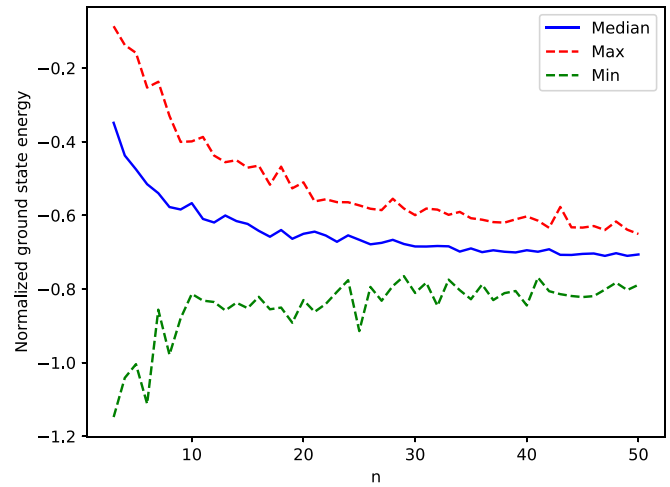

FIG. 5. Normalized ground state energy $E_{\min}n^{-3/2}$ of instances of the S-K model. 99 random instances generated for each n.

found for the larger values of n (e.g., ≈ -0.71 for $n = 50$) seems to approach the limiting value $-0.763167\ldots$ relatively slowly. These results are consistent with heuristic finite-size results reported in [28] and were validated using exhaustive search for small n.

[1] I. Georgescu, S. Ashhab, and F. Nori, Quantum simulation, Rev. Mod. Phys. **86**, 153 (2014).

[2] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. **26**, 1484 (1997).

[3] A. Montanaro, Quantum algorithms: An overview, npj Quantum Inf. **2**, 15023 (2016).

[4] L. Grover, Quantum Mechanics Helps in Searching for a Needle in a Haystack, Phys. Rev. Lett. **79**, 325 (1997).

[5] C. Dürr and P. Høyer, A quantum algorithm for finding the minimum, arXiv:quant-ph/9607014.

[6] E. Lawler and D. Wood, Branch-and-bound methods: A survey, Oper. Res. **14**, 699 (1966).

[7] A. Montanaro, Quantum-walk speedup of backtracking algorithms, Theory Comput. **14**, 1 (2018).

[8] A. Ambainis and M. Kokainis, Quantum algorithm for tree size estimation, with applications to backtracking and 2-player games, in *Proceedings of the 49th Annual ACM Symposium on the Theory of Computing* (ACM, New York, 2017), pp. 989–1002.

[9] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Quantum computation by adiabatic evolution, arXiv:quant-ph/0001106.

[10] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv:1411.4028.

[11] S. Kobe and A. Hartwig, Exact ground state of amorphous Ising systems, Comput. Phys. Commun. **16**, 1 (1978).

[12] A. Hartwig, F. Daske, and S. Kobe, A recursive branch-and-bound algorithm for the exact ground state of Ising spin-glass models, Comput. Phys. Commun. **32**, 133 (1984).

[13] M. Palassini, F. Liers, M. Juenger, and A. Young, Low-energy excitations in spin glasses from exact ground states, Phys. Rev. B **68**, 064413 (2003).

[14] T. Packebusch and S. Mertens, Low autocorrelation binary sequences, J. Phys. A: Math. Gen. **49**, 165001 (2016).

[15] J. Bernasconi, Low autocorrelation binary sequences: Statistical mechanics and configuration space analysis, J. Phys. (France) **48**, 559 (1987).

[16] D. Sherrington and S. Kirkpatrick, Solvable Model of a Spin-Glass, Phys. Rev. Lett. **35**, 1792 (1975).

[17] A. Callison, N. Chancellor, F. Mintert, and V. Kendon, Finding spin-glass ground states using quantum walks, New J. Phys. **21**, 123022 (2019).

[18] R. Karp, M. Saks, and A. Wigderson, On a search problem related to branch-and-bound procedures, in *Proceedings of the 27th Annual Symposium on the Foundations of Computer Science* (IEEE Computer Society, New York, 1986), pp. 19–28.

[19] R. Karp and Y. Zhang, Randomized parallel algorithms for backtrack search and branch-and-bound computation, J. Assoc. Comput. Mach. **40**, 765 (1993).

[20] A. Belovs, Quantum walks and electric networks, arXiv:1302.3143.

[21] A. Belovs, A. Childs, S. Jeffery, R. Kothari, and F. Magniez, Time-efficient quantum walks for 3-distinctness, in *Automata, Languages, and Programming: 40th International Colloquium (ICALP 2013)* (Springer, Berlin, 2013), pp. 105–122.

[22] M. Jarret and K. Wan, Improved quantum backtracking algorithms using effective resistance estimates, Phys. Rev. A **97**, 022337 (2018).

[23] G. Parisi, A sequence of approximated solutions to the S-K model for spin glasses, J. Phys. A: Math. Gen. **13**, L115 (1980).

[24] M. Talagrand, The Parisi formula, Ann. Math. **163**, 221 (2006).

[25] A. Crisanti and T. Rizzo, Analysis of the ∞-replica symmetry breaking solution of the Sherrington-Kirkpatrick model, Phys. Rev. E **65**, 046137 (2002).

[26] M. Schmidt, Replica symmetry breaking at low temperatures, Ph.D. thesis, Universität Würzburg, 2008.

[27] F. Barahona, On the computational complexity of Ising spin glass models, J. Phys. A: Math. Gen. **15**, 3241 (1982).

[28] S. Boettcher, Extremal optimization for Sherrington-Kirkpatrick spin glasses, Eur. Phys. J. B **46**, 501 (2005).

[29] M. Pelikan, H. Katzgraber, and S. Kobe, Finding ground states of Sherrington-Kirkpatrick spin glasses with hierarchical BOA and genetic algorithms, in *GECCO 2008: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation* (ACM, New York, 2008), pp. 447–454.

[30] S. Kobe, Ground-state energy and frustration of the Sherrington-Kirkpatrick model and related models, arXiv:cond-mat/0311657.

[31] A. Montanari, Optimization of the Sherrington-Kirkpatrick Hamiltonian, arXiv:1812.10897.

[32] S. Aaronson and A. Ambainis, Quantum search of spatial regions, Theory Comput. **1**, 47 (2005).

[33] E. Campbell, A. Khurana, and A. Montanaro, Applying quantum algorithms to constraint satisfaction problems, Quantum **3**, 167 (2019).

[34] See https://doi.org/10.5523/bris.2lp4ih2gs5o2w1yntjca3e3il7.

[35] D. Chen, R. Batson, and Y. Dang, *Applied Integer Programming: Modeling and Solution* (Wiley, Hoboken, NJ, 2010).

[36] F. Brandão and K. Svore, Quantum speed-ups for semidefinite programming, in *58th Annual IEEE Symposium on Foundations of Computer Science* (IEEE, Piscataway, 2017), pp. 415–426.

[37] J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf, Quantum SDP-solvers: Better upper and lower bounds, in *58th Annual IEEE Symposium on Foundations of Computer Science* (IEEE, Piscataway, 2017), pp. 403–414.

[38] J. van Apeldoorn and A. Gilyén, Improvements in quantum SDP-solving with applications, arXiv:1804.05058.

[39] I. Kerenidis and A. Prakash, A quantum interior point method for LPs and SDPs, arXiv:1808.09266.

[40] F. Brandão, A. Kalev, T. Li, C. Y.-Y. Lin, K. Svore, and X. Wu, Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning, in *Automata, Languages, and Programming: 46th International Colloquium (ICALP 2019)* (Springer, Berlin, 2019).

[41] D. Moylett, N. Linden, and A. Montanaro, Quantum speedup of the traveling-salesman problem for bounded-degree graphs, Phys. Rev. A **95**, 032323 (2017).

[42] S. Lalley, Concentration inequalities, 2013, https://galton.uchicago.edu/~lalley/Courses/386/Concentration.pdf.

[43] D. Dubhashi and A. Panconesi, *Concentration of Measure for the Analysis of Randomized Algorithms* (Cambridge University Press, Cambridge, UK, 2009).