Performance of hybrid quantum/classical variational heuristics for combinatorial optimization

Giacomo Nannicini*
IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
(Dated: December 17, 2018)

The recent literature on near-term applications for quantum computers contains several examples of the applications of hybrid quantum/classical variational approaches. This methodology can be applied to a variety of optimization problems, but its practical performance is not well studied yet. This paper moves some steps in the direction of characterizing the practical performance of the methodology, in the context of finding solutions to classical combinatorial optimization problems. Our study is based on numerical results obtained applying several classical nonlinear optimization algorithms to Hamiltonians for six combinatorial optimization problems; the experiments are conducted via noise-free classical simulation of the quantum circuits implemented in Qiskit. We empirically verify that: (1) finding the ground state is harder for Hamiltonians with many Pauli terms; (2) classical global optimization methods are more successful than local methods due to their ability of avoiding the numerous local optima; (3) there does not seem to be a clear advantage in introducing entanglement in the variational form.

arXiv:1805.12037v3 [quant-ph] 13 Dec 2018

I. INTRODUCTION

The hybrid quantum/classical variational approach is an optimization algorithm devised for the early generation of universal quantum computers. Variational approaches have been applied to a variety of fields, e.g., chemistry [1, 2], machine learning [3, 4], optimization [5–7]. In broad terms, a variational approach works by choosing a parametrization of the quantum state that depends on a relatively small set of parameters, then using classical optimization routines to try to determine values of the parameters corresponding to a quantum state that maximizes or minimizes a given utility function. Typically, the utility function is a Hamiltonian encoding the total energy of the system, to be minimized. This directly relates to an optimization context: the same idea can be applied to classical combinatorial optimization problems, provided that we can construct a Hamiltonian encoding the objective function of the optimization problem.

This report summarizes our experience in using classical derivative-free optimization methods to try to find good solutions for these problems, and sheds some lights on limitations that should be overcome to increase the effectiveness of the variational approach. All our computational experiments are based on noise-free simulations of quantum hardware, and we therefore have access to the full quantum state with which we can exactly evaluate the Hamiltonians. The conclusions of the study might be considerably different if real hardware had been used, considering the inherent amount of noise that affects the computations. To the best of our knowledge, this is the most comprehensive numerical study of a hybrid quantum/classical variational method for combinatorial optimization. The most notable limitation of our study is the fact that we use only one type of variational form, and we do not test any problem-dependent variational form as advocated by some other works [7–9].

Our study leads to the following observations. Even if the variational form used in our experiments is guaranteed to span the ground state, the resulting optimization problems are difficult for classical local optimization methods, e.g., gradient descent: these methods often get stuck in local minima that may be very far from the optimal solution. Global optimization seems to be more reliable, but the performance of all algorithms is very problem-dependent. In particular, the variational approach has difficulties on the problem classes that also appear to be the hardest for a classical Branch-and-Bound solver. This difficulty is likely related to the concept of "density" of the problem representation. Indeed, the performance of the classical Branch-and-Bound solver can be explained by the fact that the classical representation of these difficult problems as binary quadratic optimization problems leads to dense matrices, which are known to be harder to deal with than sparse matrices. Similarly, for the variational approach our experiments show that the number of distinct eigenvalues of the Hamiltonian is a good indicator of the difficulty of a problem instance; this can be explained in light of the fact that eigenpairs represent stationary points of the optimization problem. The number of distinct eigenvalues is often related to the number of terms in the representation of the Hamiltonian as a weighted sum of Paulis, depending on the weights. This observation may provide an easy way to quickly estimate the difficulty of finding the ground state with the variational approach on a given Hamiltonian. Furthermore, for these classes of problems, which yield diagonal Hamiltonians, it is unclear if two-qubit entangling gates help accelerate convergence to the ground state. Finally, regardless of the method used and the problem class, attaining a good approximation ratio or attaining a good probability of sampling the optimal solution seems to require a large number of iterations of the classical optimization routine; the scaling of the performance with re-

* nannicini@us.ibm.com

spect to problem size (limited to what can be ascertained in a study that considers at most 18 qubits) indicates that as problem size increases, the number of necessary iterations grows more than linearly, which is expected when dealing with nonconvex problems. Considering the crucial role played by the variational form in this type of method, it is important that future research efforts carefully consider the choice of variational form and its effect on the performance of the optimization algorithm.

II. THE VARIATIONAL APPROACH

The hybrid quantum/classical variational approach aims to find the quantum state attaining minimum energy for a given Hamiltonian by varying a set of parameters that control the quantum state. The algorithm that varies the parameters is a classical optimization algorithm. Formally, let H be the Hamiltonian encoding the total energy of a system, let θ be a vector of parameters, and let $|\psi(\theta)\rangle = U(\theta)|0\rangle$ be the quantum state obtained by applying a given parametrized quantum circuit $U(\theta)$ to the initial state $|0\rangle$; for example, the quantum circuit $U(\theta)$ could include some rotations, and the vector θ encodes the rotation angles. The variational approach aims to determine:

$$\min_{\theta} \langle\psi(\theta)|H|\psi(\theta)\rangle. \tag{1}$$

It is well known that since $|\psi(\theta)\rangle$ is normalized, the minimum value of (1) is bounded below by the minimum eigenvalue $\lambda_{\min}$ of H, and in fact $\lambda_{\min} = \min_{|\psi\rangle}\langle\psi|H|\psi\rangle$. Determining such minimum value is in general NP-hard, as will be shown in the next section by encoding several NP-hard problems into this framework. The optimization of (1) can be performed in a hybrid setting that uses a classical computer running an iterative algorithm to select θ, and a quantum computer to compute information about $\langle\psi(\theta)|H|\psi(\theta)\rangle$ for given θ, for example the value of $\langle\psi(\theta)|H|\psi(\theta)\rangle$ itself or its derivatives with respect to θ. Since finding the minimum of (1) is an approximation of the problem of finding the minimum eigenvalue of H, this approach is typically called the variational quantum eigensolver (VQE) in the literature [10].

The unitary matrix $U(\theta)$ is typically called the variational form or ansatz. Clearly the choice of the variational form has a fundamental role. In certain settings, it is possible to show that a specific variational form spans the optimal solution to a class of problems, or that there exist efficient algorithms to optimize θ under some conditions [8]. However, appropriately choosing $U(\theta)$ is in general a difficult task. Determining an appropriate classical algorithm to optimize over θ is also difficult in general. Existing works in the literature typically employ iterative continuous optimization algorithm, e.g., various forms of gradient descent or direct search methods [2, 11].

III. HAMILTONIANS FOR BINARY OPTIMIZATION PROBLEMS

The most natural formulation of combinatorial optimization problems on a quantum computer is via an Ising spin glass model, which directly translates into a Hamiltonian. Indeed, we have:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad Z|0\rangle = |0\rangle \qquad Z|1\rangle = -|1\rangle,$$

and the two eigenvalues ± 1 of Z correspond to the positive and negative spin. The Ising spin glass model can be seen as a quadratic unconstrained binary optimization problem, and it inherits its hardness [12]. In general, computing the partition function of an Ising model is NP-complete [13]. Because of this, any problem in NP can be reduced to an Ising model; we are particularly interested in combinatorial problems that have a natural mapping to Ising spin glass models, i.e., problems with the property that if the original instance has size n, we need only n qubits for an Ising spin glass representation.

As we will see in the following, some problems are naturally formulated in terms of spin variables ± 1, whereas others have a more natural formulation in terms of 0-1 variables. The transformation between the two types of variables is straightforward, see e.g., [14] . The main idea is as follows. Consider a binary quadratic unconstrained optimization problem:

$$\min\{c^\top x + x^\top Q x : x \in \{0,1\}^n\}, \tag{2}$$

then transform (2) into an Ising model using the substitution $x_j = \frac{y_j^Z + 1}{2}$, where $x_j \in \{0,1\}$ and $y_j^Z \in \{-1,1\}$ for $j = 1, \ldots, n$; we use the superscript Z to distinguish ± 1 spins from 0-1 variables. Since (2) is a quadratic model, the substitution yields a summation of terms, each of which contains either one or two y_j^Z variables. The Hamiltonian is then a summation of weighted tensor products of Z Pauli operators, where each term of the summation contains at most two Zs. Furthermore, since Z is diagonal, the Hamiltonian resulting from this transformation is diagonal.

If the original binary quadratic optimization problem is constrained, the approach mentioned above can still be applied by adding appropriate (quadratic) penalties for constraint violations in the objective function. In the cases of relevance for this paper, the additional constraints for (2) can be expressed as the requirement $Ax = b$ for some choice of A, b; in this case, it is sufficient to add the term $\alpha\|Ax - b\|^2$ to the objective function (2) with a sufficiently large α, to ensure that the unconstrained formulation has the same optimum as the original constrained formulation.

We now describe the six classes of combinatorial optimization problem employed in our numerical study.

A. Maximum stable set

Given an undirected graph $G = (V, E)$, a stable set (also called independent set) is a set of mutually nonadjacent vertices. We are interested in determining a stable set of maximum cardinality: we label this problem StableSet. Assuming that $V = \{1, \ldots, n\}$, the problem can be formulated as:

$$\max \left\{ \sum_{j \in V} x_j - \sum_{(i,j) \in E} x_i x_j : x \in \{0,1\}^n \right\}.$$

Indeed, the first summation in the objective function represents the cardinality of the stable set, while the second part penalizes including two adjacent vertices. It is straightforward to check that the penalty always offsets the objective function increase derived from selecting a vertex that is adjacent to an already selected vertex. This problem is a specific case of the set packing problem [14]. It is one of six basic NP-complete problems discussed in the seminal work of [15] . Transforming the 0-1 binary variables into ± 1 spins gives the Hamiltonian.

B. Maximum 3-satisfiability

The maximum 3-satisfiability problem, Max3SAT in the following, tries to determine an assignment of Boolean variables that satisfies the largest number of clauses of a Boolean formula in conjunctive normal form, where each clause has exactly three literals. This is one of the six basic NP-complete problems in [15] . There are several approaches to construct a Hamiltonian for Max3SAT. The approach followed in this paper is to transform an instance of Max3SAT with m clauses into an instance of StableSet on a suitably constructed graph with $3m$ vertices. This transformation is well-known, and we refer the reader to [14, 15] for details. We remark that in principle we can model Max3SAT with n Boolean variables using n Ising spins and a 3-local Hamiltonian, i.e., a tensor product of three Pauli terms for each clause. However, we employ the transformation to StableSet for two reasons: first, it allows to study the behavior of VQE on random instances of StableSet versus structured instances of the same problem; second, the formulation with products of three Pauli terms cannot be directly translated into a quadratic unconstrained model with n 0-1 variables, and we use this direct transformation in Section IV C when assessing the difficulty of these problem instances with a classical Branch-and-Bound solver.

C. Number partitioning

Given a set of numbers $S := \{a_1, \ldots, a_n\}$, the problem of number partitioning (Partition) asks to determine $P_1, P_2 \subset \{1, \ldots, n\}, P_1 \cup P_2 = \{1, \ldots, n\}, P_1 \cap P_2 = \emptyset$ such that $|\sum_{j \in P_1} a_j - \sum_{j \in P_2} a_j|$ is minimum. To construct a Hamiltonian for this problem, notice that if we associate a Ising spin variable $y_j^Z \in \{-1, 1\}$ to each number $a_1, \ldots, a_n$, we have $\sum_{j=1,\ldots,n} y_j^Z = \sum_{j:y_j^Z=1} a_j - \sum_{j:y_j^Z=-1} a_j$. Furthermore, minimizing $|\sum_{j \in P_1} a_j - \sum_{j \in P_2} a_j|$ is equivalent to minimizing $|\sum_{j \in P_1} a_j - \sum_{j \in P_2} a_j|^2$, and we can thus write:

$$\min \left\{ \left(\sum_{j=1,\ldots,n} y_j^Z \right)^2 : y_j^Z \in \{-1, 1\} \right\}.$$

Expanding the square gives the Hamiltonian in the desired form. Partition is one of the six basic NP-complete problems in [15] .

D. Maximum cut

Given an undirected graph $G = (V, E)$ with weights w_{ij} on the edges, the maximum cut problem (MaxCut) calls for determining a partition of V into disjoint sets V_1, V_2 such that

$$\sum_{\substack{(i,j) \in E \\ i \in V_1, j \in V_2}} w_{ij}$$

is maximum, i.e., the sum of the weights of edges with endpoints on opposite sides of the partition. An Ising spin glass model without field is essentially a weighted MaxCut problem [12]. The problem can be formulated as:

$$\max \left\{ \sum_{(i,j) \in E} w_{ij} y_i^Z y_j^Z - \sum_{(i,j) \in E} \frac{w_{ij}}{2} \right\}.$$

E. Market split

The market split problem [16] can be described as the problem of assigning the n customers of a firm that sells m products to two subdivisions of the same firm, in such a way that the two subdivisions retain roughly an equal share of the market. Formally, we are given a matrix A with nonnegative entries a_{ij} that represent the amount of product i bought by customer j. We want to determine a 0-1 assignment x_j for each customer j so that for each product i, $\sum_{j=1}^n a_{ij} x_j \approx \sum_{j=1}^n a_{ij}$. If we let b be the vector with entries $b_i = \left\lfloor \sum_{j=1}^n a_{ij} \right\rfloor$, then this is simply the problem:

$$\min \left\{ \|Ax - b\|^2 : x \in \{0,1\}^n \right\}.$$

Epanding the square and transforming the 0-1 binary variables into ± 1 spins gives the Hamiltonian. This problem is known to be very difficult for classical algorithms based on Branch-and-Bound [17].

F. Traveling salesman problem

Given an undirected complete graph $G = (V, E)$ with weights w_{ij} on the edges, the traveling salesman problem (TSP) aims to find a Hamiltonian cycle of minimum weight, i.e., a cycle that visits all nodes of the graph and such that the sum of the edge weights is minimum. To formulate this problem we use the formulation given in [18] . Let n be the number of nodes. For $i, p = 1, \ldots, n$, let $x_{i,p}$ be 1 if node i appears in position p in the cycle, 0 otherwise. Fixing the first node of the cycle to be the node with index label 1, i.e., $x_{1,1} = 1$, TSP can be formulated as:

$$\left.\begin{array}{rrcl} \min & \sum_{i,j=1}^{n-1} w_{ij} \sum_{p=1}^{n-1} x_{i,p} x_{j,p+1} + & & \\ & \sum_{j=1}^{n-1} w_{j1} x_{j,n} & & \\ \forall i = 1, \ldots, n & \sum_{p=1}^{n} x_{i,p} & = & 1 \\ \forall p = 1, \ldots, n & \sum_{i=1}^{n} x_{i,p} & = & 1 \\ & x_{1,1} & = & 1 \\ \forall i, p = 1, \ldots, n & x & \in & \{0, 1\}. \end{array}\right\}$$

To derive a Hamiltonian for this problem, we penalize the violation of the constraints in the objective function inserting terms of the form $\alpha(\sum_{p=1}^{n} x_{i,p} - 1)^2$, where α is sufficiently large, e.g., $\alpha = n \max_{(i,j)\in E} w_{ij}$. TSP (rather, Hamiltonian cycle) is one of six basic NP-complete problems in [15] . Note that the formulation requires n^2 binary variables, so the number of qubits does not scale linearly in the problem size; this is the only problem in our test set with this property.

IV. DATA AND EXPERIMENTAL SETUP

This section describes the procedure used to generate random instances of each class, as well as the overall setup used for our experiments.

A. Generation of random instances

Given the desired number of qubits q, we generate instances as follows:

- StableSet: we generate a random Erdős-Rényi graph with q nodes and edge probability 0.3.
- Max3SAT: we generate a random formula in conjunctive normal form with $\lfloor q/3 \rfloor$ Boolean variables and $\lfloor q/2 \rfloor$ clauses. Clauses are generated sequentially, adding one literal (positive or negative) chosen uniformly at random among literals that do not appear in the same clause.
- Partition: we generate q integers in the interval $[1, q^2 + 1]$, chosen uniformly at random.
- MaxCut: we generate a complete graph with q nodes and integer weights selected uniformly at random in the interval $[-10, 10]$.
- Marketsplit: we follow the procedure described in [16] , for q binary variables.
- TSP: we generate a complete graph with $\sqrt{q} + 1$ nodes and integer weights selected uniformly at random in the interval $[0, 9]$. Note that the resulting problem instance is in general not symmetric.

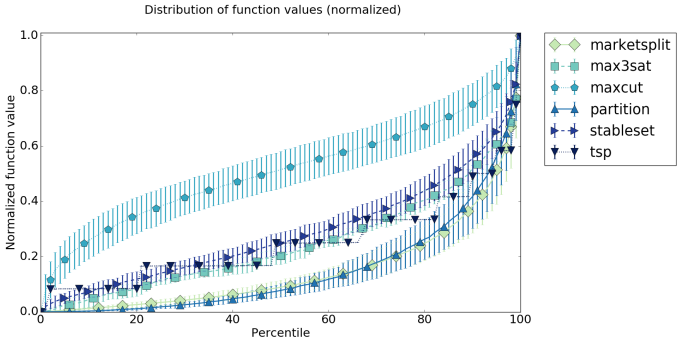
	# terms				# distinct eig.			
q	15	16	17	18	15	16	17	18
MaxCut	99	113	129	145	201	225	251	228
TSP	–	99	–	–	–	5291	–	–
Max3SAT	37	–	–	43	65	–	–	78
Partition	105	120	136	153	710	883	1081	1304
Marketsplit	120	136	153	171	22204	37564	57532	81151
Stableset	48	53	60	65	107	120	138	150

TABLE I. Average number of terms in the Hamiltonian (each term is a tensor product of Pauli Z), and average number of distinct eigenvalues. All numbers are rounded to the nearest integer.

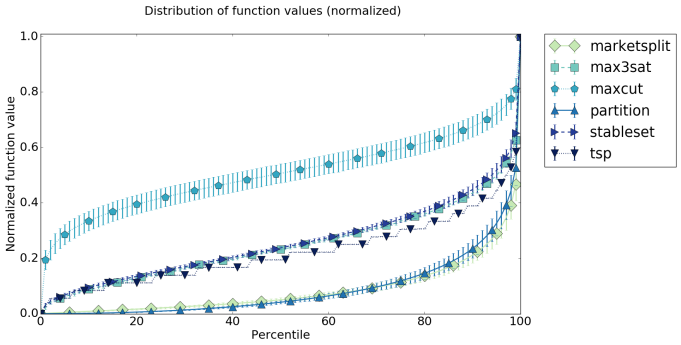
B. Experimental setup

We applied the VQE to all problems described in the previous subsection, testing all problem sizes from 6 to 18 qubits; because of the Hamiltonian formulation used, Max3SAT requires the number of qubits to be a multiple of 3, and TSP requires the number of qubits to be a perfect square, hence these two problems were only tested for sizes in the range $[6, 18]$ that satisfy the stated restrictions. For each problem type and size, we repeat the experiment 20 times with a different random seed; the random seed affects the instance itself (as it is randomly generated) and the starting point of the optimization algorithm. Note that we provide the same sequence of random seeds to all optimization algorithms, so that all optimization algorithms solve the same sequence of problems with the same sequence of starting points. The average number of terms in the Hamiltonian for some problem sizes, as well as the number of distinct eigenvalues, are reported in Table I, where we can see that a large number of terms does not necessarily correspond to more distinct eigenvalues, because of integer weights. The distribution of the objective function value of the feasible solutions for the randomly generated instances is plotted in Fig. 1. It shows that Partition and Marketsplit instances have many solutions that are close to the optimum in relative terms, MaxCut has very few, and the other problem classes are somewhere in between.

To evaluate the progress of VQE toward reaching an optimal solution for the problem at hand, we employ a methodology based on the data profiles described in [19] . More details regarding the meaning of each graph are given in the corresponding subsections. The optimum for each problem is computed using the classical solver IBM ILOG Cplex 12.7.1, which can certify optimality; we



(a) Instances between 6 and 12 qubits.



(b) Instances between 13 and 18 qubits.

FIG. 1. Distribution of the objective function value of all feasible solutions for the random instances used in the numerical experiments, averaged across all instances of a certain size. The y axis value is the approximation ratio. Error bars extend between $\pm$ the standard deviation.

use an optimal 0-1 solution obtained by Cplex to compute the value of the ground state of the corresponding Hamiltonian.

We tested five optimization algorithms:

1. Limited-memory BFGS (LBFGS) [20]: a quasi-Newton local optimization method. We use the SciPy implementation of this algorithm, in which the gradient is estimated numerically by finite differences.

2. Constrained Optimization By Linear Approximation (COBYLA) [21]: a model-based local optimization method that builds a linear approximation of the objective function over a simplex. We use the original FORTRAN implementation through its SciPy interface.

3. RBFOpt [22]: a model-based global search method that builds an adaptive radial basis function interpolant of the objective function. The algorithm alternates between a global search and a local search that follows a trust region framework [23].

4. Modified Powell's conjugate direction method (PCD) [24]: a pattern search local optimization method that searches along a given set of directions, which is updated at every iteration. We use the implementation in SciPy.

5. Simultaneous Perturbation Stochastic Approximation (SPSA) [25]: a model-based local optimization method that builds a gradient approximation using two function evaluations per iteration. We use the implementation found in QISKit [18].

In the brief description above, we classify as "local" algorithms those which converge to a (at least) first-order stationary point, and "global" algorithms those that do not employ convergence criteria based on first-order stationarity, but rather have a mechanism to escape any local optimum. In our tests, the performance of LBFGS and COBYLA is very similar across the board. PCD and SPSA performed considerably worse than the remaining algorithms in our experiments. This is not surprising: PCD was not designed to be parsimonious in the number of function evaluations, and therefore exhibits slower convergence; whereas SPSA was designed to be robust to noise, but this robustness comes at a price and is not exploited at all in our noise-free setting. Overall, numerical experiments using PCD and SPSA are simply slower and do not yield further insight with respect to looking at the first three optimization algorithms alone. Other popular algorithms such as Nelder-Mead and genetic algorithms were not considered as they usually yield inferior results in mathematical benchmarks [23]. For these reasons, in the following we only report results for COBYLA, RBFOpt and occasionally LBFGS.

C. Analysis of difficulty with classical Branch-and-Bound

To quickly assess the difficulty of instances in our test set on classical computers, we transformed each Hamiltonian into a quadratic unconstrained binary optimization problem, and we solved it to global optimality using using the Branch-and-Bound algorithm in the commercial integer programming solver IBM ILOG Cplex 12.7.1. The average solution times are given in Fig. 2. The graph shows that running times for Partition, Marketsplit and Maxcut scale exponentially with problem size, while the other problem classes appear considerably easier.

We remark that we did not make any attempt at fine-tuning the algorithm or improving the problem formulation: we directly translated the Hamiltonian into a quadratic form with binary variables, and let Cplex solve the instance with its default parameters [26]. It is well known that the performance of integer programming models depends heavily on the particular formulation of the optimization problem; our automatic translation of the Hamiltonian yields very weak formulations, therefore in practice one can expect significant improvements (potentially orders of magnitude) using better classical

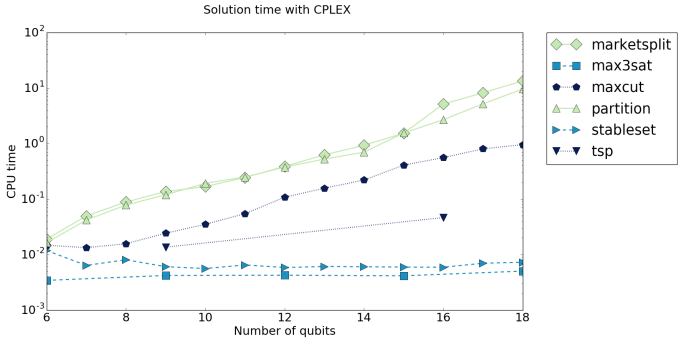


FIG. 2. Average solution times using IBM ILOG Cplex, single-threaded. The y axis is on a log scale.

	Density				Negative eigenvalues			
q	15	16	17	18	15	16	17	18
MaxCut	0.95	0.95	0.95	0.95	0.62	0.61	0.59	0.60
TSP	–	0.72	–	–	–	1.00	–	–
Max3SAT	0.31	–	–	0.25	1.00	–	–	1.00
Partition	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Marketsplit	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Stableset	0.40	0.39	0.39	0.38	1.00	1.00	1.00	1.00

TABLE II. Average density (expressed as the number of nonzero elements in Q over the total number of elements) and fraction of negative eigenvalues for the test instances.

methodologies. In other words, Fig. 2 is far from representing the state of the art of classical optimization for the problem classes under consideration.

To understand what causes the difficulty of certain problem classes for IBM ILOG Cplex, we looked at two properties that are known to affect the performance of solution methods for quadratic optimization problems, namely: the density of Q, and the distribution of its eigenvalues. The linear term c of the cost function was added to the diagonal of Q, since for binary variables $x^2 = x$. Statistics are reported in Table II. These data suggest that density strongly correlates with problem difficulty: most instances have many negative eigenvalues (all problems are expressed as minimization problems), but only some instances are dense (Marketsplit and Partition are fully dense, followed by MaxCut), and these appear to be the hardest to solve for Cplex.

D. Variational form

The choice of the variational form is crucial for the performance of VQE; in particular, if the ground state (minimum energy state) of the Hamiltonian cannot be attained by a given variational form, then VQE will never reach the optimum energy. For combinatorial optimization problems it is easy to construct variational forms that are guaranteed to contain the ground state in their span: it is sufficient to ensure that the variational form can generate any binary string on q qubits.

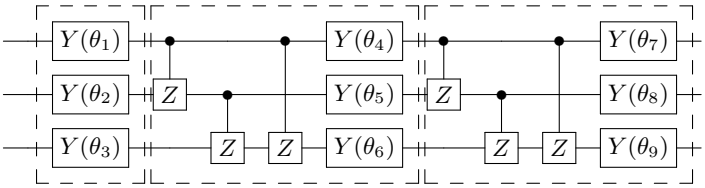


FIG. 3. Example of the variational form on three qubits. Each box represents a layer.

In our experiments, we use a variational form constructed in layers, see [2]. The first layer always consists in single-qubit Y rotations, with one variational parameter per qubit to determine the rotation angle. This ensures that any binary string can be obtained with just the first layer. Each additional layer after the first contains entangling gates, more specifically controlled-Z gates applied to all qubit pairs, followed by another set of single-qubit Y rotations with one variational parameter each to represent the angle. Thus, each layer has q variational parameters. The resulting circuit is exemplified in Fig. 3 for three qubits. We experimented with nearest-neighbor controlled-Z gates as well, i.e., between qubit j and $j+1$ for all $j = 1, \ldots, q$, but this does not change the conclusions of the study.

Our experiments use up to three layers of this variational form, using the labels 1L, 2L, 3L to indicate how many have been used. For the global classical optimization algorithm RBFOpt only 1L and 2L are tested, because the optimization for 3L is too time consuming. By construction, the variational form has low depth, as is the case for ansätze that have been implemented in hardware [2].

The specific choice of variational form used in this paper is justified by the fact that it is guaranteed to span the ground state. The entangling layers, coupled with Y rotations, allow us to control the level of entanglement, and experimentally verify whether entanglement helps speed up convergence, see Section V F. We remark that for 0-1 optimization problems there exists an optimal solution that is a computational basis state, hence entanglement is not necessary in principle, unlike, e.g., certain quantum chemistry problems in which the ground state is known to be an entangled state.

V. RESULTS AND ANALYSIS

We now report a summary of our findings, based on plots provided in this paper as well as further analysis not reported for space reasons.

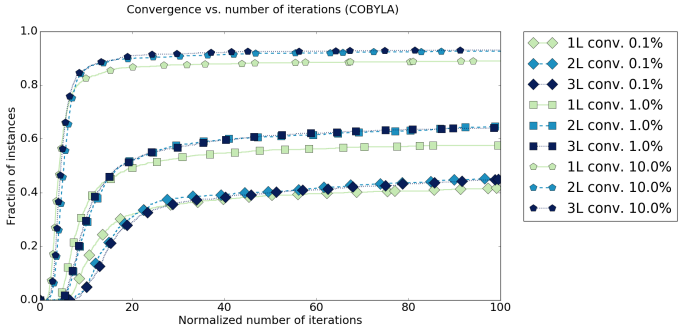
A. Convergence versus number of iterations

In the first set of graphs we study the convergence of each optimization algorithm as the number of iterations progresses, over the entire set of problem instances. Here and in the rest of the paper, convergence is defined in the following way. Let $\tilde{x}$ be the initial point given to the optimization algorithm [27], and let x^* be an optimal solution to the problem. Calling f the objective function, we say that an optimization algorithm converges to a precision of $\tau \in [0,1]$ if it determines a point x such that $\frac{f(x)-f(x^*)}{f(\tilde{x})-f(x^*)} \geq 1-\tau$. Notice that when $\tau = 0$ this implies determining an optimal solution, whereas $\tau = 1$ is trivially satisfied by any point x returned by the optimization algorithm.

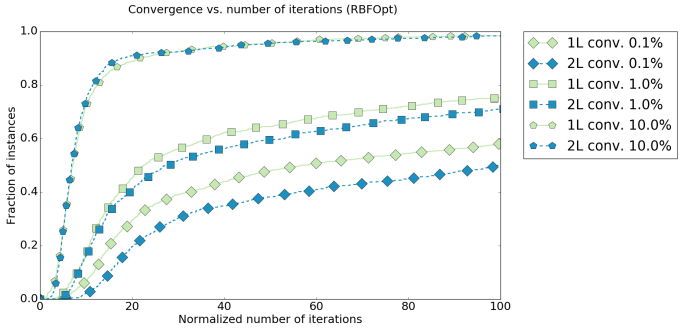
To account for different problem sizes and the dimension of the search space, we normalize the iteration number by reporting the "equivalent gradient iterations", where each gradient iteration performs $n+1$ function evaluations and n is the total number of parameters of the variational form that are being optimized. Using the variational form described in Section IV D, a problem instance on q qubits with a variational form with ℓ layers has $n = q\ell$ parameters. We remark that $n+1$ corresponds exactly to the number of function evaluations that are perfomed by a gradient-based method that estimates the gradient by finite differences along the coordinate axes (e.g., the LBFGS implementation used in our tests); for such methods, the normalization gives an accurate count of the major iterations of the optimization algorithm. Other methods, however, do not try to estimate the gradient at every major iteration, but we apply the same normalization in order to have a fair comparison. This normalization is also standard in the derivative-free optimization literature to account for varying problem sizes [19]. The maximum number of function evaluations is set to $100(n+1)$ for all optimization algorithms, in these and in all subsequent experiments.

In Fig. 4, we report aggregate results for COBYLA and RBFOpt over the entire test set. As mentioned earlier, LBFGS's performance is very close to that of COBYLA. The curves are drawn for three convergence levels: $\tau \in \{0.001, 0.01, 0.1\}$. The x-axis indicates the normalized number of iterations (i.e., equivalent gradient iterations), the y-axis reports the fraction of instances on which the optimization algorithm converges up to a specified tolerance.

We can see from Fig. 4 that the local optimization algorithm (COBYLA) plateaus after a relatively small number of normalized iterations, whereas the global optimization algorithm (RBFOpt) continues improving and in the long run achieves convergence on more instances. LBFGS shows the same behavior as COBYLA, and this suggests that the local optimization algorithms are stuck in a local minimum. However, all algorithms fail to converge to high accuracy in a large fraction of the instances. Interestingly, using two layers of the variational



(a) Constrained Optimization By Linear Approximation (COBYLA).

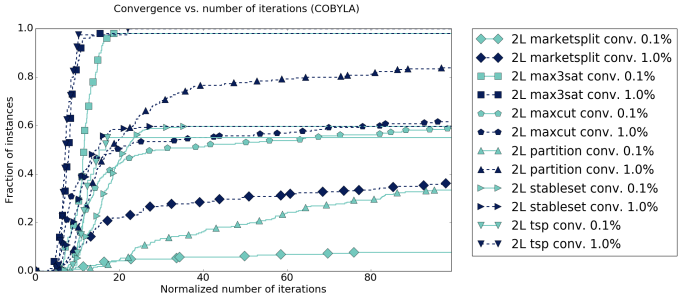


(b) RBFOpt.

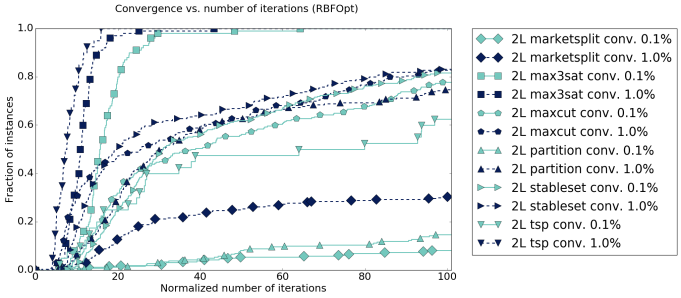
FIG. 4. Fraction of the instances on which a given algorithm converges to the specified tolerance, versus the normalized number of iterations.

form seems to be better when relying on a local optimization method, but worse when employing the global algorithm of RBFOpt. A possible explanation is that RBFOpt works better when the number of parameters to be optimized is small, and therefore does not benefit from the enlarged search space found in the case of several layers of the variational form. In other words, this fact may stem from properties of the optimization algorithm, rather than the variational form itself. It is therefore not clear whether the additional layers, which introduce entanglement, truly help. This will be discussed more in detail in Section V F.

To understand whether some of this behavior is problem-dependent or can be observed on the entire set of test instances, we report similar plots in Fig. 5, but now we plot curves for each instance class. For space reasons, we only provide plots for two layers of the variational form, but this is representative of the overall picture. The plots clearly show some problem-dependent behavior. In particular, some instances are considerably easier than others: all optimization algorithms excel on MAX3SAT, but struggle on MARKETSPLIT. The most difficult problem classes are MARKETSPLIT, PARTITION and (to a lesser extent) MAXCUT and TSP. This is the same ranking in terms of difficulty that was obtained when applying a classical Branch-and-Bound algorithm, which seems to indicate that problems that are hard for IBM ILOG Cplex are also hard for the VQE heuristic. In



(a) Constrained Optimization By Linear Approximation (COBYLA).



(b) RBFOpt.

FIG. 5. Fraction of the instances on which a given algorithm converges to the specified tolerance, versus the normalized number of iterations. In these plots we employed a variational form with two layers.

the next subsection we perform further numerical experiments to try to determine what makes certain problems harder.

To summarize the results presented in this subsection, our experiments indicate that different problem classes have different difficulty levels, independent of the optimization algorithm (PCG and SPSA, not reported here, exhibit similar behavior). This may depend on the specific procedure adopted to generate random instance, resulting in harder instances for some classes of problems. These remarks are consistent with the literature in classical combinatorial optimization, where the most successful methods to solve problems take advantage of problem-specific structure. The agnostic nature of VQE, coupled with classical derivative-free optimization algorithms, results in alternating performance with mixed results that seem to match the behavior of classical Branch-and-Bound.

B. Density and eigenvalues

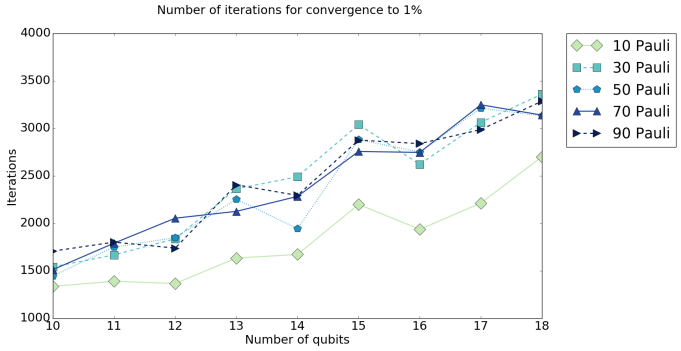
Table I shows that the hardest problems have more distinct eigenvalues than easier problems (TSP, that exhibits a large number of distinct eigenvalues, seems not too difficult in Fig. 5, but the analysis in subsequent sections indicates that the approximation ratio of the solution found by VQE is fairly large). These problems also seem to have Hamiltonians with more terms. It is important to investigate if these quantities correlate with difficulty.

In the classical setting, the density of the quadratic objective function matrix is an important factor in determining the difficulty of an instance. When discussing quantum Hamiltonians, the most natural proxy is the number of Pauli terms that appear in the summation defining the Hamiltonian. Because we are using a nonlinear optimization tool to determine the ground state, it is also conceivable that the number of distinct eigenvalues of the Hamiltonian could be a good indicator of the difficulty of (1) (since every eigenvector is a stationary point of the optimization problem). To test these conjectures, we run some experiments on Hamiltonians consisting of a weighted summation of random pairs of Pauli Z. We fix the number of pairs in each Hamiltonian o $10, 20, \ldots, 100$, and the weights are chosen uniformly at random in $\{-1, 1\}$ in the first set of experiments, $[-1, 1]$ in the second set of experiments (notice that the first is a discrete set, the second is a real interval). The number of qubits varies between 10 and 18, and for each combination of parameters we generate 20 random Hamiltonians. The average number of unique eigenvalues of these Hamiltonians is given in Table III (for space reasons, we report only for q even). As expected, the Hamiltonians in the second set of experiments (random weights in $[-1, 1]$) have many more unique eigenvalues than in the first set of experiments, even if the number of Pauli terms is the same. Indeed, for large enough number of Pauli terms, the Hamiltonians with random weights in $[-1, 1]$ have the maximum number of distinct eigenvalues (2^{q-1}, since by construction for every eigenvalue λ, $-\lambda$ is also an eigenvalue).

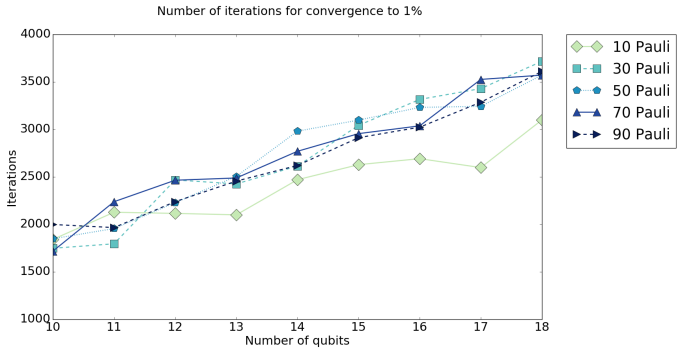
# Pauli terms	# of qubits q									
	weights in $\{-1, 1\}$					weights in $[-1, 1]$				
	10	12	14	16	18	10	12	14	16	18
10	9	9	10	10	11	336	390	633	755	896
20	13	14	16	17	18	499	1843	6246	19251	45056
30	16	18	20	20	22	512	1997	7680	30310	95027
40	19	21	22	24	26	512	2048	8192	31948	117964
50	21	23	25	27	29	512	2048	8192	32768	131072
60	23	26	28	30	33	512	2048	8192	32768	131072
70	24	28	31	32	35	512	2048	8192	32768	131072
80	26	30	32	34	37	512	2048	8192	32768	131072
90	14	32	34	36	39	512	2048	8192	32768	131072
100	14	33	35	38	41	512	2048	8192	32768	131072

TABLE III. Average number (rounded to the nearest integer) of unique eigenvalues in the Hamiltonian depending on the number of Pauli terms in the summation defining the Hamiltonian.

Running VQE on these Hamiltonians shows that density is not a good indicator of difficulty, but the number of eigenvalues is. In Fig. 6 we report convergence with $\tau = 0.01$ and a variational form with 2 layers using COBYLA; results with other optimizers or number of layers are similar. The graph suggests that problems with very small number of Pauli terms (e.g., 10) are easy



(a) Hamiltonians with weights in $\{-1,-1\}$.



(b) Hamiltonians with weights in $[-1,1]$.

FIG. 6. Average number of iterations for convergence to $\tau = 0.01$ for Hamiltonians with a different number of Pauli terms, using COBYLA and a variational form with 2 layers.

across all sizes but as soon as the number increases it is no longer possible to detect a strong correlation between number of terms and difficulty. However, the number of distinct eigenvalues affects difficulty (remember from Table III that there is saturation of the number of eigenvalues for more than ≈ 30 Pauli terms in the Hamiltonian with weights in $[-1,1]$, hence we cannot expect problems to get more difficult when they have more than 30 terms).

To support this conclusion, we compare the number of iterations for convergence for increasing number of Pauli terms using a nonparametric statistical test known as the Friedman test. The groups (algorithms) compared correspond to the number of Pauli terms in the Hamiltonian, for uniform weights in $\{-1,1\}$ and in $[-1,1]$. We use confidence $\alpha = 0.95$; the null-hypothesis (no differences between the variables) is rejected, p-value $1.11e^{-16}$, and we perform pairwise comparisons in the post-hoc analysis. The post-hoc analysis clearly indicates that for the same number of terms in the Hamiltonian, problems with uniform weights in $[-1,1]$, which have more distinct eigenvalues, take longer to converge. We report a subset of the results in Table IV.

To conclude, our experiments with random Hamiltonians obtained as sum of pairs of Zs indicate that we can expect the difficulty of a problem instance to increase with the number of distinct eigenvalues. This, in turn, can be related to the number of Pauli terms in the summation defining the Hamiltonian, depending on the distribution of the weights. The results are consistent with our observations in the preceding sections. From a practical standpoint, unfortunately computing the number of distinct eigenvalues is not an easy task unless the Hamiltonian is known in the full Hilbert space.

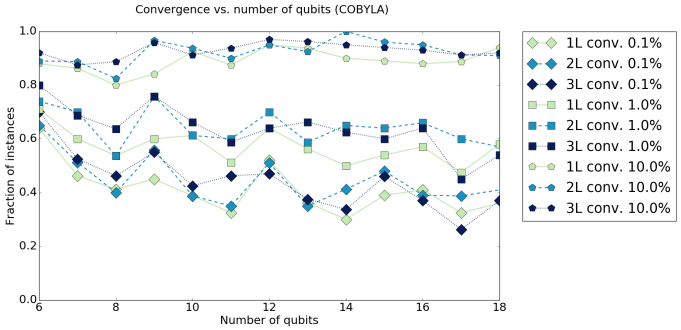
			Number of Pauli terms									
			Weights $\{-1,1\}$					Weights $[-1,1]$				
			10	30	50	70	90	10	30	50	70	90
Number of Pauli terms	Weights $\{-1,1\}$	10		-	-	-	-	-	-	-	-	-
		30	+		=	=	=	=	-	-	-	-
		50	+	=		=	=	=	-	-	-	-
		70	+	=	=		=	=	-	-	-	-
		90	+	=	=	=		=	-	-	-	-
	Weights $[-1,1]$	10	+	=	=	=	=		-	-	-	-
		30	+	+	+	+	+	+		=	=	=
		50	+	+	+	+	+	+	=		=	=
		70	+	+	+	+	+	+	=	=		=
		90	+	+	+	+	+	+	=	=	=	

TABLE IV. Pairwise comparison of the number of iterations for convergence. A "+" in row i and column j indicates that in experiment i the Friedman test detected more iterations than in experiment j; vice versa with a "−"; no difference is detected with a "=". The two-digit numbers labeling column and rows indicate the number of Pauli terms in the experiment.

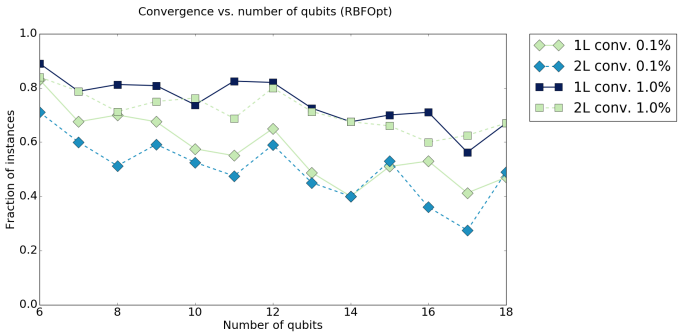
C. Impact of problem size

We now study the impact of problem size, i.e., number of qubits, on the performance of VQE. Fig. 7 reports the fraction of problem istances on which convergence to a specified level of tolerance is attained, with problem size varying from 6 to 18 qubits. Results are aggregated over all problem classes.

The plot indicates a slight downward trend in the convergence rate as problem size increases. We remark that in these experiments the scaling of the number of iterations that each algorithm is allowed to run is the same as in the previous section, namely $100(q\ell+1)$ iterations where q is the number of qubits. Hence, the plot suggests that increasing the number of iterations linearly in the number of qubits is not sufficient to maintain constant the fraction of instances on which convergence to high accuracy is attained. Indeed, as remarked in the previous section, Fig. 4 shows diminishing returns for increased iteration number, and some algorithims reach a plateau after a certain number of iterations; Fig. 7 provides the additional information that problem instances become harder to solve as the qubit count increases. The data gathered from our experiments is not sufficient to extract an overall trend and determine what is the correct scaling of the number of iterations to maintain a constant

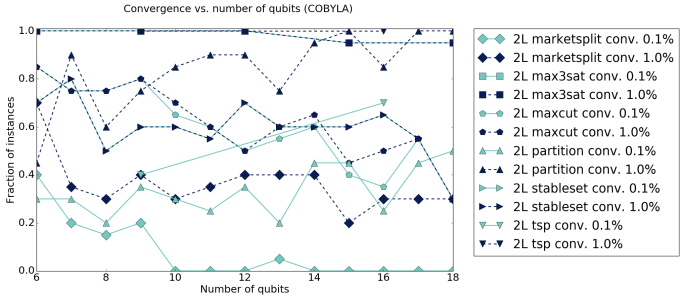


(a) Constrained Optimization By Linear Approximation (COBYLA).

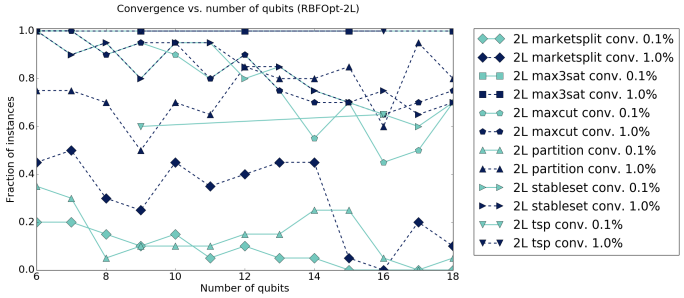


(b) RBFOpt.

FIG. 7. Fraction of the instances on which a given algorithm converges to the specified tolerance, versus the number of qubits.



(a) Constrained Optimization By Linear Approximation (COBYLA).



(b) RBFOpt.

FIG. 8. Fraction of the instances on which a given algorithm converges to the specified tolerance, versus the number of qubits. In these plots we employed a variational form with two layers.

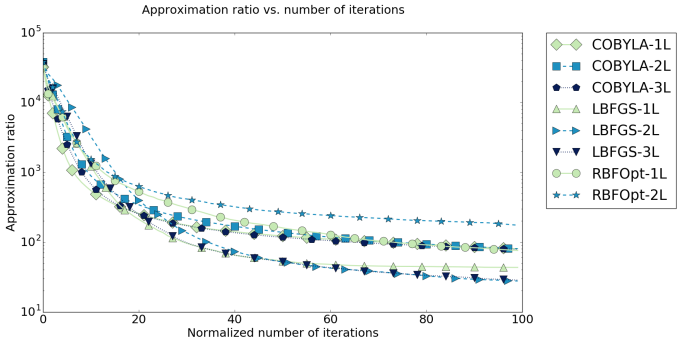
fraction of instances on which converge is achieved.

Similar plots that provide a curve for each problem class (Fig. 8) indicate that the downward trend is not observed among all problem classes: specifically, the performance of all optimization algorithms on Partition and TSP does not seem to deteriorate for increasing problem size. However, this is the case for the remaining problems, and this leads to our observations for the average case.

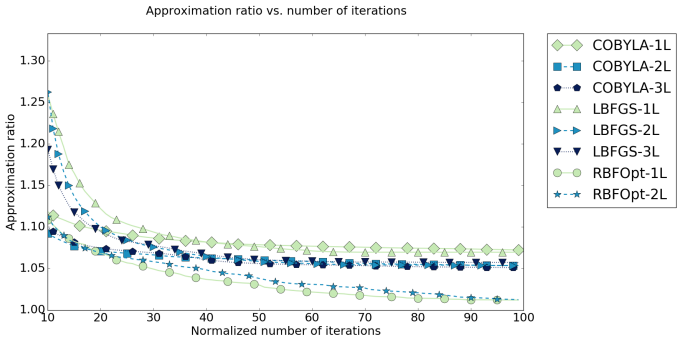
D. Approximation ratio

The convergence profiles shown in previous sections have the useful properties of being normalized with respect to the initial point and invariant to constant shifts to the diagonal of the Hamiltonian. A more commonly used metric to assess the performance of optimization methods (especially from a theoretical point of view) is the approximation ratio, defined as the value $(1+\epsilon)$ such that the algorithm attains a solution of value at most $f(x^*)(1+\epsilon)$, where $f(x^*)$ is the optimum value. We report the evolution of the approximation ratio for Partition and MaxCut in Fig. 9. We remark that some of the test problems have energy values unrestricted in sign, i.e., the classical optimization algorithm explores points with positive and negative value. The approximation ratio only makes sense for nonnegative values. Notice that the minimum eigenvalue of the Hamiltonian can be negative, in particular for all max optimization problems converted to a min problem by taking the negative of the objective function. Therefore, in our graphs we only report iterations for which the energy value of the quantum state has the same sign as the minimum eigenvalue of the Hamiltonian, and for problems with negative minimum eigenvalue we plot the ratio $f(x^*)/f(x)$ rather than $f(x)/f(x^*)$; this way, all graphs are decreasing and lower bounded by 1. The average across all instances is taken using the geometric average, which is more suitable in this context since the approximation ratio is a multiplicative quantity rather than additive.

On Partition the approximation ratio is very large, consistent with the observation that this problem class appears to be difficult. The performance on Marketsplit is very similar (additional graphs are given in the Appendix). On MaxCut, the global optimization algorithm eventually reaches an approximation ratio close to 1, but convergence is slow; the local optimization algorithms are stuck in local optima with ratio ≥ 1.05. StableSet is similar to MaxCut, whereas on Max3SAT all algorithms quickly attain ratio very close to 1, consistent with our previous observations. TSP starts with approximation ratios $\approx 10^3$ but quickly reaches values ≤ 10, even though no algorithm attains 1. Since the original formulation for TSP is heavily constrained (transformed to unconstrained by penalizing constraint violations), the initial point given to the classical optimization



(a) Partition.



(b) MaxCut

FIG. 9. Average approximation ratio (geometric average) versus the normalized number of iterations. The y axis in the graph for Partition is on log scale.

algorithm is likely infeasible (i.e., it does not satisfy the constraints) and incurs heavy penalties that affect its energy value; the optimization then moves towards feasibility, which explains the large initial approximation ratios that quickly decrease as the feasible region is reached.

E. Probability of sampling the optimal solution

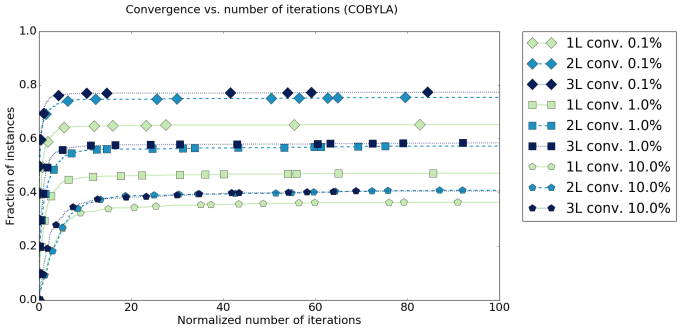
So far, we have been concerned with studying the speed with which optimization algorithm find a quantum state with an optimal or close-to-optimal energy value, as evaluated according to a given Hamiltonian. We remark that the energy of a quantum state corresponds to the expected objective function value of the binary solutions that can can be sampled from that quantum state. This can be easily verified: let H be Hamiltonian encoding of a combinatorial problem with objective function $f : \{0,1\}^q \to \mathbb{R}$. Then if $|\psi\rangle$ is a basis state, it corresponds to a binary string z, and we must have $\langle\psi|H|\psi\rangle = H_{z,z} = f(z)$, where by $H_{z,z}$ we denote the element of H whose row and column are indexed by z. Furthermore, recall that Hamiltonians for combinatorial problems encoded as binary problems use only Pauli Z operators, resulting in a diagonal Hamiltonian. Therefore, for a general state $|\psi\rangle$, we can write:

$$\begin{aligned}\langle\psi|H|\psi\rangle = \langle\psi|\mathrm{diag}(H)|\psi\rangle &= \sum_{z\in\{0,1\}^q} \langle\psi_z|\psi_z\rangle H_{z,z} \\ &= \sum_{z\in\{0,1\}^q} \Pr(|\psi\rangle = z) H_{z,z} \\ &= \sum_{z\in\{0,1\}^q} \Pr(|\psi\rangle = z) f(z).\end{aligned}$$

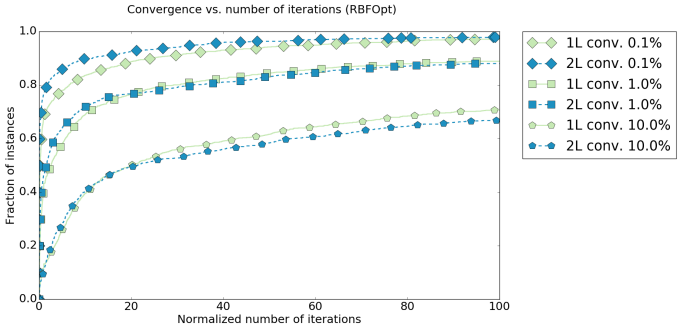
In the above expression, $\Pr(|\psi\rangle = z)$ is the probability of sampling z when performing a measurement of all the q qubits from state $|\psi\rangle$.

Because of this relationship, it is in principle possible (although unlikely) that VQE produces quantum states that have a high probability of sampling an optimal binary string z, while the energy of the quantum state is larger than the optimum value; see [8] for a discussion of concentration of the probability distribution. We analyze this possibility in our next set of experiments. More specifically, we look at how the probability of sampling an optimal solution increases as the optimization algorithm progresses. In the spirit of the plots reported in previous section, given a convergence level ρ, we compute the fraction of instances in which a quantum state that has probability at least ρ of sampling the optimal binary string has been observed within a certain number of normalized iterations. We remark that this yields "optimistic" graphs, because it yields nondecreasing curves: in practice it is possible that the optimization algorithm explores a quantum state with high probability of sampling the optimal string, but the algorithm does not stop there and in subsequent iterations such probability decreases.

Plots generated as discussed above are reported in Fig. 10 and 11. It is evident from the plots in Fig. 10 that for the local optimization algorithm (COBYLA in this graph, but LBFGS is similar), the fraction of instances in which the probability of sampling the optimal solution is greater than a given constant plateaus very quickly. Further iterations of the optimization algorithm may increase such probability (typically, the energy value goes down, as observed in previous sections), but the plots show that this is an improvement on instances for which the optimization algorithm has already found "good" quantum states. The situation is different for the global algorithm RBFOpt: we observe a steady increase in the curves, implying that the algorithm explores quantum states with sufficiently large probability of sampling the optimal string on more and more instances. However, we remark that for a global algorithm the "optimistic" way of generating these graphs may have a significant impact: indeed, a global optimization algorithm will often explore quantum states from unknown regions of the search space, and for this reason it is more likely to encounter quantum states that satisfy the convergence criterion, but it may not be able to detect when one of these states has been found. In other words, even though RBFOpt explores better quantum states as the

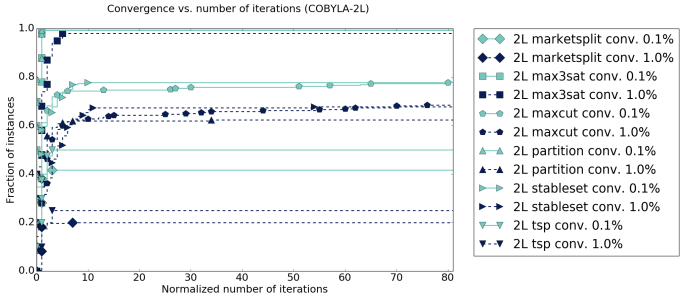


(a) Constrained Optimization By Linear Approximation (COBYLA).

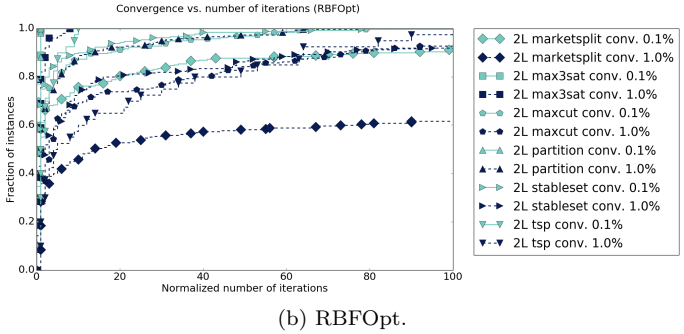


(b) RBFOpt.

FIG. 10. Fraction of the instances on which a given algorithm explores at least one quantum state with probability of sampling the optimal solution greater than a given threshold, versus the normalized number of iterations.



(a) Constrained Optimization By Linear Approximation (COBYLA).



(b) RBFOpt.

FIG. 11. Fraction of the instances on which a given algorithm explores at least one quantum state with probability of sampling the optimal solution greater than a given threshold, versus the normalized number of iterations. These plots are generated with a variational form with two layers.

iteration count increases, it may not be able to indicate from which quantum state the optimal string can be sampled. In any case, these plots indicate that the local optimization algorithms can quickly get stuck in local minima that do not contain quantum states likely to yield the optimal binary string. This is not suprising: due to their hardness, we expect NP-hard combinatorial optimization problem to give rise to Hamiltonians associated with highly nonconvex energy landscapes.

Fig. 11 highlights once more the large discrepancy between problem instance classes: while for some classes any optimization algorithm quickly determines a quantum state that has high probability of yielding the optimal string, other classes of problems appear out of reach, especially for the local optimization algorithms. Global optimization looks more promising, but it is affected by a different set of issues which may limit its practical usefulness. We remark that optimizing the expected objective function value of the binary strings may not be the best possible approach, if the end goal is simply to reach a certain probability of sampling an optimum (rather than aiming for the ground state, that has probability 1 of sampling an optimum).

F. Entanglement vs. no entanglement

The variational form used throughout the paper introduces entanglement after the first layer of Y rotations. Experiments discussed in Section V A do not show any clear advantage for the variational forms 2L and 3L that use entanglers, as compared to 1L that generates product states only. Indeed, Fig. 4 reports a marginal improvement with 2L and 3L using COBYLA, but this comes at the cost of several additional iterations of the optimization algorithm (recall that the x-axis is normalized by $q\ell+1$, where ℓ is the number of layers); with RBFOpt, the variational form 1L achieves the best results. Other local solvers yield results consistent with COBYLA. We now try to understand whether 2L, 3L can truly improve performance of the local solvers because of entanglement, or if the reason for such improvement could be attributed to other factors, e.g., better chance to escape local minima. To do so, we repeat the experiments and the analysis using a variational form that mimicks the one described in Sec. IV D, but does not use two-qubit gates, i.e., the CZ gates of Fig. 3. Having multiple adjacent Y rotations on the same qubit would amount to introducing copies of the same variational parameter, which is undesirable from an optimization standpoint. Hence, after each layer of Y rotations we apply a T gate on each qubit. This way, each variational parameter on the same qubit has a different effect. We obtain a variational form exemplified

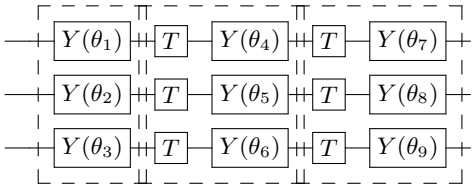
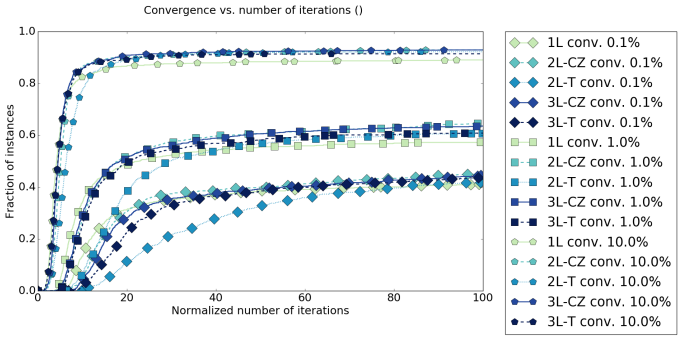
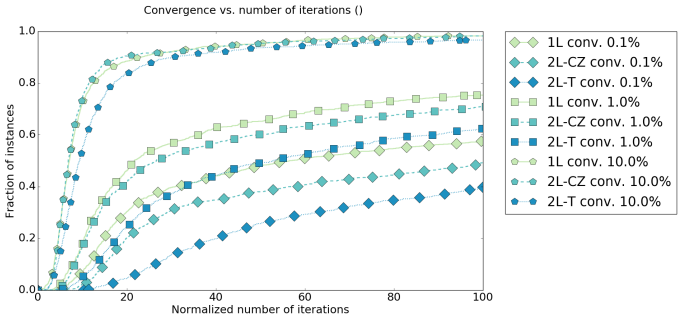


FIG. 12. Example of the variational form without entanglement on three qubits. Each box represents a layer.

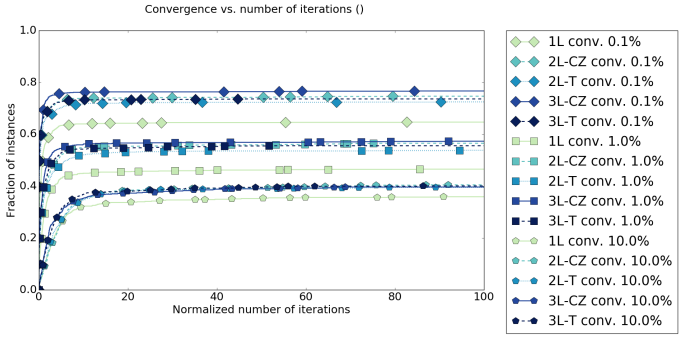


(a) Constrained Optimization By Linear Approximation (COBYLA).

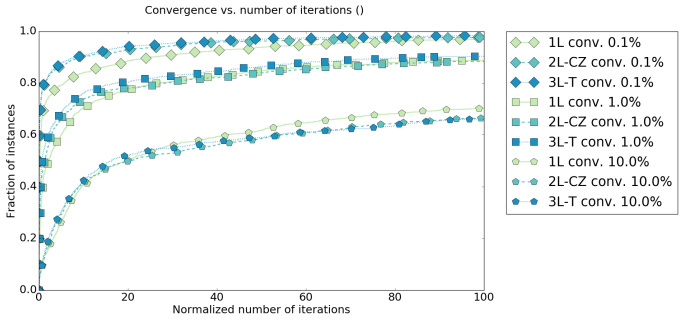


(b) RBFOpt.

FIG. 13. Fraction of the instances on which a given algorithm converges to the specified tolerance, versus the normalized number of iterations. We compare variational forms with and without entanglement.



(a) Constrained Optimization By Linear Approximation (COBYLA).



(b) RBFOpt.

FIG. 14. Fraction of the instances on which a given algorithm explores at least one quantum state with probability of sampling the optimal solution greater than a given threshold, versus the normalized number of iterations. We compare variational forms with and without entanglement.

in Fig. 12.

We compare the performance of the optimization algorithms using the previous variational form (which we label 2L-CZ, 3L-CZ) and the new variational form without entanglement (labeled 2L-T, 3L-T). A summary of the results is given in Fig. 13, reporting convergence versus number of iterations as in Section V A, and Fig. 14, reporting probability of sampling an optimal solution versus number of iterations as in Section V E. Fig. 13 shows that for COBYLA, the performance of 2L-CZ, 2L-T, 3L-CZ, 3L-T is essentially indistinguishable. This is confirmed by Fig. 14, looking at the probability of sampling an optimal solution. The variational form with two and three layers still appear to be better than 1L, but there does not seem to be any significant difference between using CZ gates or T gates. A possible explanation is that the local optimization algorithm finds better solutions (in a larger number of iterations) when there are additional variational parameters that can be used to avoid being trapped in a local minimum. Similar results are obtained with LBFGS. The results with RBFOpt tell a different story: here 2L-CZ is sinificantly better than 2L-T in Fig. 13, even though attaining a better objective function value does not seem to affect the probability of sampling an optimal solution, see Fig. 14. The variational form 1L is still better than any 2L variational form with RBFOpt, as remarked in Section V A.

The results reported in this section paint a mixed picture. They suggest that there may be nothing special about variational form with entangling CZ gates, because we can achieve similar results with a different variational form that does not introduce entanglement. The variation in performance of the various algorithm may be attributed to characteristics of the algorithms themselves, rather than of the variational form: some algorithms seem to benefit from having additional variational parameters to optimize, while others do not. Overall, while the experiments are not conclusive, we are unable to observe any advantage in using two-qubit gates in our setting, as compared to variational forms with single-qubit gates only.

VI. CONCLUSIONS

Our study of VQE on classical combinatorial optimization problems highlights several obstacles that need to be overcome in order to make it a viable method with potential to be used in practice. The first obstacle is that local optimization algorithms that aim at first-order stationary points appear to get stuck in local minima very frequently on certain problem classes. A possible solution is to use different classical optimization algorithms, more suited for nonconvex problems, or a different variational form. A second obstacle is that convergence to the optimum is slow, requiring more and more iterations of the classical optimization routine as the problem size increases. This difficulty may be alleviated by computing derivatives on the quantum computer, rather than relying on derivative-free optimization methods or finite-difference estimations of the gradient; the computation of derivatives is possible for certain types of variational forms. A different choice of variational form may also help. The third obstacle is that the performance of VQE on classes of problems correlates with the performance of a classical Branch-and-Bound solver on a naive translation of the Hamiltonian to a quadratic form over binary variables: since VQE performs well only on problems for which the classical Branch-and-Bound also performs well, there is little advantage to be gained. This is likely due to the fact that these hard problems have dense quadratic matrices in the classical representation, and the corresponding Hamiltonians have many distinct eigenvalues. A possible way forward is to start exploiting problem structure in a systematic way, moving away from the problem-agnostic nature of VQE; this may also alleviate some of the other issues.

Our experiments indicate that there does not seem to be a significant gain in performance, if any, by using two-qubit gates in the variational form, as compared to single-qubit gates. Of course, if the variational form yields a product state the computation could be performed efficiently on a classical computer, hence suggesting that VQE does not yield any quantum speedup on this class of problems. However, two important remarks are in order: first, binary optimization problems by construction admit a ground state that is a basis state, therefore they are very poor candidates to showcase the benefits of entanglement; second, from a theoretical point of view we know that two-qubit gates can be useful even for binary optimization problems (e.g., [8] shows that we can essentially simulate adiabatic optimization with a problem-dependent variational form with a sufficient number of gates). Hence, while two-qubit gates do not seem to yield benefits in the setting of this paper, this conclusion would change on a different class of problems or with a different optimization approach than a problem-agnostic VQE.

Ultimately, the most important question is to understand whether a VQE-like approach has potential to be competitive with classical combinatorial optimization methodologies. At the scale at which we are able to simulate, which is approximately the scale of existing superconducting qubit devices, this question cannot be answered: algorithms and software for binary optimization on classical computers are very refined, and optimal solutions for the problems discussed in this paper can typically be found in fractions of a second [28]. On the other hand, the VQE implementation tested in this paper requires the exploration of hundreds or thousands of trial states as well as iterations of a classical optimization algorithm), and may fail to converge anyway. Leaving aside considerations on hardware efficiency, this suggests that the performance of VQE must be greatly increased before it can be considered competitive. Our study provides some suggestions on possible directions of improvement.

[1] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. Obrien, Nature communications **5**, 4213 (2014).

[2] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Nature **549**, 242 (2017).

[3] J. Romero, J. P. Olson, and A. Aspuru-Guzik, Quantum Science and Technology **2**, 045001 (2017).

[4] E. Farhi and H. Neven, arXiv preprint arXiv:1802.06002 (2018).

[5] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, *et al.*, arXiv preprint arXiv:1710.01022 (2017).

[6] E. S. Fried, N. P. Sawaya, Y. Cao, I. D. Kivlichan, J. Romero, and A. Aspuru-Guzik, arXiv preprint arXiv:1709.03636 (2017).

[7] E. Farhi, J. Goldstone, S. Gutmann, and H. Neven, arXiv preprint arXiv:1703.06199 (2017).

[8] E. Farhi, J. Goldstone, and S. Gutmann, arXiv preprint arXiv:1411.4028 (2014).

[9] E. Farhi and A. W. Harrow, arXiv preprint arXiv:1602.07674 (2016).

[10] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, New Journal of Physics **18**, 023023 (2016).

[11] J. Romero, R. Babbush, J. McClean, C. Hempel, P. Love, and A. Aspuru-Guzik, Quantum Science and Technology (2018).

[12] F. Barahona, Journal of Physics A: Mathematical and General **15**, 3241 (1982).

[13] S. Istrail, in *Proceedings of the thirty-second annual ACM symposium on Theory of computing* (ACM, 2000) pp. 87–96.

[14] A. Lucas, Frontiers in Physics **2**, 5 (2014).

[15] M. R. Garey and D. S. Johnson, *Computers and intractability*, Vol. 29 (W. H. Freeman and Company, New York, NY, 1972).

[16] G. Cornuéjols and M. Dawande, INFORMS Journal on Computing **11**, 205 (1999).

[17] K. Aardal, R. E. Bixby, C. A. J. Hurkens, A. K. Lenstra,

and J. W. Smeltink, INFORMS Journal on Computing **12**, 192 (2000).
[18] Qiskit, "Quantum information software kit," `https://www.qiskit.org/` (2017), accessed Feb 2018.
[19] J. Moré and S. M. Wild, SIAM Journal on Optimization **20**, 172 (2009).
[20] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, SIAM Journal on Scientific Computing **16**, 1190 (1995).
[21] M. Powell, in *Advances in Optimization and Numerical Analysis*, edited by S. Gomez and J.-P. Hennart (Kluwer, Dordrecht, 1994) pp. 51–67.
[22] A. Costa and G. Nannicini, Mathematical Programming Computation **10**, 597 (2018).
[23] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*, MPS-SIAM Series on Optimization (Society for Industrial and Applied Mathematics, 2009).
[24] M. J. Powell, The computer journal **7**, 155 (1964).
[25] J. C. Spall, IEEE transactions on automatic control **37**, 332 (1992).
[26] Because the quadratic form is in general nonconvex, to solve the instance to global optimality we set the parameter `optimalitytarget` to `optimalglobal`, we set `absmipgap` to 0.1 and `mipgap` to 0.
[27] This assume that the algorithm accepts and uses an initial point, e.g., to start a gradient descent; all optimization algorithms tested in this paper accept an initial point.
[28] The times reported in Fig.2 are largely due to the effort of *proving* optimality of the solution, rather than *finding* the solution.