On the Complexity of Probabilistic Trials for Hidden Satisfiability Problems*

Itai Arad[1], Adam Bouland[2], Daniel Grier[3], Miklos Santha[4], Aarthi Sundaram[5], and Shengyu Zhang[6]

1 **Center for Quantum Technologies, National University of Singapore, Singapore**
`arad.itai@fastmail.com`

2 **Massachusetts Institute of Technology, Cambridge, MA USA**
`adam@csail.mit.edu,`

3 **Massachusetts Institute of Technology, Cambridge, MA USA**
`grierd@mit.edu`

4 **Center for Quantum Technologies, National University of Singapore, Singapore and**
CNRS, IRIF, Université Paris Diderot 75205 Paris, France
`miklos.santha@gmail.com`

5 **Center for Quantum Technologies, National University of Singapore, Singapore**
`aarthims@gmail.com`

6 **Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong**
`syzhang@cse.cuhk.edu.hk`

Abstract

What is the minimum amount of information and time needed to solve 2SAT? When the instance is known, it can be solved in polynomial time, but is this also possible without knowing the instance? Bei, Chen and Zhang (STOC '13) considered a model where the input is accessed by proposing possible assignments to a special oracle. This oracle, on encountering some constraint unsatisfied by the proposal, returns only the constraint index. It turns out that, in this model, even 1SAT cannot be solved in polynomial time unless P=NP. Hence, we consider a model in which the input is accessed by proposing probability distributions over assignments to the variables. The oracle then returns the index of the constraint that is most likely to be violated by this distribution. We show that the information obtained this way is sufficient to solve 1SAT in polynomial time, even when the clauses can be repeated. For 2SAT, as long as there are no repeated clauses, in polynomial time we can even learn an equivalent formula for the hidden instance and hence also solve it. Furthermore, we extend these results to the quantum regime. We show that in this setting 1QSAT can be solved in polynomial time up to constant precision, and 2QSAT can be learnt in polynomial time up to inverse polynomial precision.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases computational complexity, satisfiability problems, trial and error, quantum computing, learning theory

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.12

* Research was supported by the Singapore Ministry of Education and the National Research Foundation by the Tier 3 Grant MOE2012-T3-1-009, by the European Commission IST STREP project Quantum Algorithms (QALGO) 600700, the French ANR Blanc Program Contract ANR-12-BS02-005. A.B. was supported in part by the NSF Graduate Research Fellowship under grant no. 1122374 and by the NSF Alan T. Waterman award under grant no. 1249349. S.Z.'s research was supported in part by RGC of Hong Kong (Project no. CUHK419413).

1 Introduction

SAT has been a pivotal problem in theoretical computer science ever since the advent of the Cook-Levin Theorem [7, 13] proving its NP-completeness. It has a wide array of applications in operations research, artificial intelligence and bioinformatics. Moreover, it continues to be studied under various specialized models such as as random-SAT and building efficient SAT solvers for real-life scenarios. In the complexity theoretic setting, we know that while 3SAT is NP-complete [7, 13], 2SAT can be solved in linear time [12, 9, 3]. Given the fundamental nature of 2SAT, in this paper, we consider the following question:

What is the minimum amount of information needed to solve 2SAT *in polynomial time?*

More precisely, what happens if there is no direct access to the problem instance? Are there settings where one can *solve* 2SAT without ever *learning* the instance under consideration? We can also pose the same question for the quantum setting where the quantum analogue of SAT (QSAT) can be seen as a central problem in condensed matter physics. Complexity theoretically, we know that 2QSAT can be solved in linear time [2, 8] while 3QSAT is hard for QMA_1 [10], where QMA_1 is a quantum complexity class analogous to NP. We approach these questions through the "trial and error" model. In this model, one guesses a solution to an unknown constraint satisfaction problem and tests if it is valid. If so, then the problem is solved. Otherwise, the trial fails, and some information about what was wrong with the trial is revealed. This type of problem arises in a number of natural scenarios, in which one has incomplete or limited information about the problem they are trying to solve [4]. For example, the CSP may be instantiated by a complex biological or physical process to which one does not have access.

This approach to problem solving was first formalized by Bei, Chen and Zhang [4]. They considered several types of CSPs and analyzed the computational complexity in the "unknown input" setting. Specifically, they consider an oracle model where one can propose solutions to the CSP, and if the solution is not satisfying, then the oracle reveals the identity of a constraint which was violated. For example, if the CSP is an instance of Boolean satisfiability (SAT), then after an unsuccessful trial, one may learn that "clause 7 was violated", but not anything further. In particular, literals present in clause 7 will not be revealed - only the label of violated clause is known. Furthermore, if there are several violated constraints, then the oracle reveals only one of them, in a possibly adversarial manner. In this paper, we will refer to this as the "arbitrary violated constraint" oracle.

This model gives extremely limited access to the instance. In fact, Ivanyos et al. [11] showed that even if the underlying CSP is a 1SAT instance, accessing it with the BCZ oracle, one cannot determine if it is satisfiable in polynomial time unless $\mathrm{P} = \mathrm{NP}$. This drastically increases the difficulty of deciding a trivial problem like 1SAT (assuming $\mathrm{P} \neq \mathrm{NP}$). Interestingly, if there is access to a SAT solver, then 1SAT (and even generic SAT) in this setting can be solved with polynomially many trials [4]. So in some sense, their model reveals a sufficient amount of *information* to solve the 1SAT instance. However, *decoding* this information requires superpolynomial time (assuming $\mathrm{P} \neq \mathrm{NP}$). In short the information needed to solve the problem is present, but it is not accessible to poly-time algorithms.

In this paper, we ask if there are any meaningful modifications of their model which allow us to solve simple CSPs like 1SAT and 2SAT in polynomial time. A natural starting point is to randomize the "arbitrary violated constraints" model. One obvious way to do that is to consider allowing randomized queries to the oracle. This however does not significantly decrease the complexity of the problems. A second approach to randomize is to let the oracle return a violated clause at random. Contrary to the previous approach, this model trivializes the problem, since by repeating the same trial many times the oracle will reveal all violated clause indices with high probability. This in turn allows one to learn the entire instance, and therefore trivially, to solve 1SAT and 2SAT.

Motivated by these unfruitful approaches we consider a model which does not allow one to completely learn the underlying instance, but it still yields polynomial time algorithms for 1SAT and 2SAT. Specifically, in this model one can propose a probability distribution D over assignments, and the oracle reveals the index of the clause which is most likely to be violated by this trial. If there are multiple clauses with the same probability of violation under D, then the oracle can break ties arbitrarily. In particular, product distributions over the variables suffice for our application, so one merely specifies the probability p_i that each variable x_i is set to 1 in the assignment, to $1/\operatorname{poly}$ precision. We show that in this model, there exist cases where one cannot learn the underlying 1SAT or 2SAT instance. However, despite this limitation, one can still solve in polynomial time 1SAT and a restricted version of 2SAT where clauses are not repeated. In the course of the algorithm for the restricted version of 2SAT, we actually learn an equivalent formula with the same set of satisfying assignments. Furthermore, we are able extend this model to the quantum setting, and show that one can solve, in polynomial time, Quantum 1SAT (1QSAT) up to constant precision. We also show that in polynomial time we can learn Quantum 2SAT (2QSAT) up to inverse polynomial precision. This, however, seems insufficient to solve the hidden instance in polynomial time due to some subtle precision issues, which we discuss in Section 7.

Relation to prior work. As previously mentioned, Bei Chen and Zhang [4] introduced the trial and error model. They considered several examples of CSPs and analyzed their complexity under the unknown input model with the “arbitrary violated constraint” oracle. With regards to SAT, they showed an algorithm to solve hidden-SAT using polynomially many queries to the oracle (given access to a SAT oracle). Furthermore, they showed that one cannot efficiently learn generic SAT instances in this model, because it takes $\Omega(2^n)$ queries to the oracle to learn a clause involving all n variables of the instance.

Subsequently, Ivanyos *et al.* [11] characterized the complexity of classical CSPs in several hidden input models. In particular, they consider the “arbitrary violated constraint” model described above, as well as models which reveal more information such as the variables involved in the violated clause or the relation of the violated clause. They show a generic “transfer theorem” which classifies the complexity of hidden versions of CSPs given properties of the base CSP. In particular, their transfer theorem implies that the hidden version of 1SAT with arbitrary violated constraints cannot be solved in polynomial time unless P = NP. This indicates that the “arbitrary violated constraint” model is fairly restrictive.

In parallel, Bei, Chen and Zhang [5] considered a version of the trial and error model for linear programming. Suppose you have a linear program, and you are trying to determine whether or not it is feasible (By standard reductions this is as difficult as solving a generic LP). They consider a model in which one can propose a point, and the oracle will return the index of an arbitrary violated constraint (half-plane) in the linear program. They show that in this model, one requires exponentially many queries to the oracle to determine if an LP is feasible. However, they then consider a relaxation of this model, in which the oracle returns the index of the *worst-violated* constraint, i.e. the half-plane which is furthest (in Euclidean distance) from the proposed point. Surprisingly, they show (using a variant of the ellipsoid algorithm) that one can still solve linear programs in this model in polynomial time. Our model can be seen as an analogue of the "worst violated constraint" model of Bei, Chen and Zhang [5] for the case of hidden SAT (H–SAT).

Our Results. Our results can be broken into several sections. First, we consider a relaxation of the "arbitrary violated constraint" model of Bei, Chen and Zhang [4], in which the oracle reveals which subset of clauses are violated by each assignment[1]. We show that in some sense these models are almost "too easy"

▶ **Theorem** (Informal statement). *In the "all violated constraints" model, there is an algorithm which either learns an arbitrary* kSAT *instance on n variables and m clauses, or else finds a satisfying assignment to the instance, in time $O(mn^k)$.*

We then explore the "worst violated constraint" model for the rest of the paper. We provide an example for why this model is more powerful than the "arbitrary violated constraint" model of Bei, Chen and Zhang [4]. They showed that it requires $\Omega(2^n)$ time to learn a SAT clause involving all n variables. Our example states that

▶ **Proposition.** *Given a hidden* WIDESAT *instance on n variables and m distinct clauses where $m \leq n$, we can learn an equivalent instance in $O(\binom{n}{m-1}2^m + n)$ time.*

The proofs of the above results are omitted owing to space constraints[2]. Among our main results is the analysis of the computational complexity of H–1SAT and that of H–2SAT.

▶ **Theorem** (Informal statement). *Given a hidden* SAT *formula Φ on n variables and n clauses, it is possible to find a satisfying assignment for Φ in polynomial time if Φ is a*
(a) 1SAT *formula or*
(b) 2SAT *formula with no repeated clauses.*

Our algorithm for H–1SAT, in Section 3, works even when clauses are repeated multiple times in the instance, despite the fact that it's not possible to *learn* the instance in this setting. This is in sharp contrast to the "arbitrary violated constraint" model, where even H–1SAT cannot be solved in polynomial time unless P = NP [11]. The main difficulty in deriving our algorithm for H–1SAT comes from dealing with repeated clauses, which allow the oracle to obscure information about the instance. Unlike the H–1SAT case, the algorithm for H–2SAT discussed in Section 4, works by attempting to learn the instance; it either succeeds in learning an equivalent instance (in which case one can solve the problem using any 2SAT algorithm), or it accidentally stumbles upon a satisfying assignment in the meantime and aborts. The problem of solving H–2SAT with repeated clauses similar to H–1SAT is left for future work.

Following this we generalize these results to the quantum case. In this case the goal is to determine if a set of 1-qubit or a set of 2-qubit projectors is mutually satisfiable or not. We consider an analogue of this model in which one can propose a probability distribution over quantum states (i.e. a density matrix), and the oracle returns the index of the clause which is most likely to be violated. Our results for hidden QSAT (H–QSAT) show that

▶ **Theorem** (Informal statement). *Given a* H–QSAT *instance H defined on n qubits with m projectors and $\epsilon > 0$, it is possible to*

[1] This is equivalent to a model in which the oracle reveals a random violated clause - by repeating each query many times one can learn the set of violated clauses with high probability.
[2] Omitted details and proofs can be found in the full version of the paper at `http://arxiv.org/abs/1606.03585`.

(a) *solve H to a precision ϵ in time $O(n^{\log(1/\epsilon)})$ if H is a* 1QSAT *instance and*
(b) *learn each projector of H up to precision ϵ in time $O(n^4+n^2\log(1/\epsilon))$, if H is a* 2QSAT *instance as long as the interaction graph of H is not star-like.*

By star-like, we mean the interaction graph contains an edge that is incident to all other edges in the graph. At this point it is worth comparing the notions of *learning* and *solving* hidden instances both in the classical and quantum settings. The classical case is more straightforward where learning an instance means learning all the literals present in each clause, whereas solving means finding a satisfying assignment. For example, our algorithm for H–2SAT without repetitions learns the instance, while our algorithm for H–1SAT solves the instance without learning it. For hidden versions of 1SAT and 2SAT, learning the instance in polynomial time automatically triggers solving it in polynomial time as well.

However, in the quantum setting this simple relation between learning and solving breaks down. The continuous nature of QSAT means we can only learn a projector or find a satisfying assignment up to a specified precision ϵ. The latter is accomplished with our H–1QSAT algorithm in Section 6. However in the case of hidden 2QSAT learning the instance up to precision ϵ does not imply that one can solve the instance up to precision poly(n,ϵ) in polynomial time. This can be attributed to current algorithms for 2QSAT being very sensitive to precision errors. This issue of divergence between the notions of learning and solving H–2QSAT instances is further discussed in Section 7.

2 Notations and Preliminaries

Boolean Satisfiability. The Boolean satisfiability problem, generally referred to as SAT, is a constraint satisfaction problem defined on n variables $\mathbf{x} = \{x_1, \ldots, x_n\}$ where we are given a formula represented as a conjunction of m clauses and each clause is a disjunction of *literals* (variables, x_j, or negated variables, $\overline{x}_j$). The problem is solved if we can find an assignment to the variables (i.e. $\forall\ i,\ x_i \in \{0,1\}$) that sets the value of every clause to 1. In particular, if each clause involves at most k literals, then this problem is classified as kSAT. It is well known that while 2SAT can be solved in linear time [12, 9, 3], kSAT for $k \geq 3$ is NP-complete [7, 13]. A useful notion is that of *clause types* which is defined as the unordered set of literals present in the clause. Specifically, the clause type for $C_j = (x_a \vee \overline{x}_b \vee x_c)$ is denoted by $T(C_j) = \{x_a, \overline{x}_b, x_c\}$. So, all possible clause types for 2SAT would be $\{\{x_a, x_b\}, \{x_a, \overline{x}_b\}, \{\overline{x}_a, x_b\}, \{\overline{x}_a, \overline{x}_b\} \mid a, b \in [n]$ and $a \neq b\}$, where $[n]$ denotes the set $\{1, \ldots, n\}$. From this definition, it is clear that 2SAT has $O(n^2)$ clause types and similarly, kSAT would have $\binom{2n}{k} = O(n^k)$ clause types. . Given a SAT formula ϕ, we say that the SAT formula ϕ' is *equivalent* to ϕ if for all assignments $\mathbf{x} \in \{0,1\}^n$, $\mathbf{x}$ satisfies ϕ if and only if it satisfies ϕ'. For any formula ϕ, SAT$(\phi) := \{\mathbf{x} \in \{0,1\}^n \mid \phi(\mathbf{x}) = 1\}$.

Hidden SAT. While considering the unknown input version of SAT (resp. kSAT), the boolean formula is considered as hidden and accessible only via an oracle that accepts an assignment and reveals some form of violation information. In our case, this is the "worst violated oracle" which accepts a *probabilistic* assignment and reveals a clause that has the *highest probability* of being violated with ties being broken arbitrarily. A probabilistic assignment for a set of n variables is a function $\mathbf{a} : [n] \to [0,1]$ such that $Pr[x_i = 1] = \mathbf{a}(i)$ and $Pr[x_i = 0] = Pr[\overline{x}_i = 1] = 1 - \mathbf{a}(i)$. For the sake of concise notation, these are usually written as $x_i = \mathbf{a}(i)$ and $\overline{x}_i = 1 - \mathbf{a}(i)$. This naturally translates to the notion of the probability of a clause C_j being violated which is defined as $Pr[C_j = 0] := \prod_{\ell \in T(C_j)} Pr[\ell = 0] = \prod_{\ell \in T(C_j)}(1 - \ell)$ which allows the oracle to calculate the probability for each clause being violated. Here we are using ℓ to refer both to the identity of a literal as well as to the probability that literal ℓ is set to true. Now, the problem H–SAT (resp. H–kSAT) consists of finding a satisfying assignment for a hidden SAT (resp. kSAT) formula by proposing probabilistic assignments to the "worst violated oracle". One way we do this is also by *learning* an *equivalent formula* to the hidden instance and solve it to find a satisfying assignment. By learning we mean the process of using the information from a series of violations to determine what a clause in the hidden instance could be.

Note that it's possible for an instance to contain clauses which will never be returned by the oracle. For instance, given clauses C_i and C_j, if $T(C_i) \subset T(C_j)$, then clause C_i will always be at least as violated as C_j. Hence the oracle might never return clause C_j. For this reason we will say that C_i *obscures* C_j if $T(C_i) \subset T(C_j)$. An obscured clause might never be returned by the oracle.

The complexity of the algorithms in the following sections is in terms of the total running time where one query to the oracle takes unit time.

3 Hidden 1SAT

In this section, we will consider the problem of a hidden 1SAT instance Φ, possibly with repetitions. Our goal will be to determine whether or not Φ is satisfiable. A natural approach one might take to solve this problem would be to learn the identity of each clause in the instance Φ. Unfortunately, in the case that the 1SAT instance has repetitions, this is not possible.

▶ **Proposition 1.** *There is no algorithm which, given an instance Φ which is unsatisfiable, learns all the literals present in Φ (even granted arbitrary numbers of queries to the oracle).*

Here the difficulty in learning an unsatisfiable instance does not lie in the repetition of clauses, but rather in determining for which i do both x_i and $\overline{x}_i$ appear in Φ. This shows that no algorithm can learn the hidden 1SAT instance [3] (proof omitted owing to space constraints). Hence if there is an algorithm to solve 1SAT in this hidden setting, then it must solve the instance despite the fact that it cannot deduce the underlying instance. Surprisingly, this turns out to be possible.

▶ **Theorem 1.** *Given a hidden* 1SAT *instance Φ on n variables and m clauses, it is possible to determine if Φ is satisfiable in time $O(mn^2)$.*

Proof Consider an ordering of the variables $x_1...x_n$. The algorithm will work by inductively constructing a series of lists $L_1, L_2, \dots L_n$. Each list L_i will contain a list of partial assignments to the variables $x_1 \dots x_i$. Each list will be of size at most m, with the exception of L_n which will be of size at most $2m$. Let us call a partial assignment p to $x_1 \dots x_i$ *good* if there exists an assignment p' to the variables $x_{i+1} \dots x_n$ such that the assignment $p \cup p'$ satisfies Φ. Correspondingly, call p *bad* if it cannot be extended to a satisfying assignment of Φ. (Note in the case of 1SAT, every partial assignment is either good or bad.) Our algorithm will guarantee that, if Φ is satisfiable, then at least one assignment in each list is "good". Therefore, by constructing the list L_n, then trying all assignments in L_n, we will be guaranteed to find a satisfying assignment if one exists.

[3] Note, however, it is still possible that there exists an algorithm to learn the 1SAT instance when the instance is promised to be satisfiable.

We now describe how to construct the lists $\{L_i\}_{i\in[n-1]}$ by induction. The base case of L_1 is trivial - just add both $x_1 = 0$ and $x_1 = 1$ to the list. We now show how to construct L_{i+1} given L_i. First, let $\tilde{L}_{i+1}$ be all possible extensions of the assignments in L_i to the variable x_{i+1}. Clearly if one of the assignments in L_i was good, then one of the assignments in $\tilde{L}_{i+1}$ is good. However, when $i + 1 < n$, the size of $\tilde{L}_{i+1}$ could become too large - it is of size $2|L_i|$ which could at some point become larger than m. So we need to reduce the size of $\tilde{L}_{i+1}$ so that it contains at most m partial assignments. To decide which partial assignments to keep, we will perform the following oracle queries: for each partial assignment $y \in \tilde{L}_{i+1}$, propose the following query q_y to the oracle: set $x_1 ... x_{i+1}$ to 0 or 1 according to y, and set all other variables to value $1/2$. The oracle will return the identity of a clause C_j which is worst violated by this fractional assignment. Now partition the elements of $\tilde{L}_{i+1}$ according to which clause C_j was returned by the query. This divides the elements of $\tilde{L}_{i+1}$ into at most m equivalence classes. To construct L_{i+1}, simply pick one element from each equivalence class of $\tilde{L}_{i+1}$.

Clearly L_{i+1} has size at most m by construction. To complete the proof, we need to show that at least one element of L_{i+1} is good. First, by the induction hypothesis, at least one element of L_i is good. This implies at least one element $y^* \in \tilde{L}_{i+1}$ is good as well. Consider what happens when we perform the query q_{y^*}. Since y^* is good, q_{y^*} must satisfy all clauses involving the variables $x_1 \dots x_{i+1}$. If there are no clauses involving the remaining variables $x_{i+2} \dots x_n$, then q_{y^*} satisfies the instance, so the oracle will tell us this and we can terminate the algorithm. Otherwise, there is a clause involving some variable in $\{x_{i+2} \dots x_n\}$. When we query q_{y^*}, the worst violated clause will be some clause C_k involving a variable in $\{x_{i+2} \dots x_n\}$, which will be violated with probability $1/2$. So the equivalence class corresponding to C_k will contain a good assignment. Furthermore, since C_k involves one of the variables in $\{x_{i+2} \dots x_n\}$, it will never be returned as the worst violated clause for query $q_{y'}$ for any bad assignment $y' \in \tilde{L}_{i+1}$, because any bad assignment will violate a clause involving $\{x_1 \dots x_{i+1}\}$ by 1, while C_k will be violated only with probability $1/2$. Therefore the equivalence class corresponding to C_k will contain only good assignments. So by picking one assignment from each equivalence class, we will ensure L_{i+1} contains at least one good assignment, as claimed.

The time to construct each list is $O(mn)$, and the algorithm constructs n lists. Hence the algorithm runs in time $O(mn^2)$. $\square$

4 Hidden 2SAT without repetitions

In this section, we consider a hidden 2SAT formula Φ which is promised to contain no two clauses that are the same. Although Proposition 1 shows that we cannot always hope to learn Φ directly, it does not rule out the possibility of learning some Φ' such that $\mathrm{SAT}(\Phi') = \mathrm{SAT}(\Phi)$. In fact, this is exactly the approach we take. The full proof of this is omitted to conserve space, but the most interesting aspect is contained in the theorem below.

▶ **Theorem 2.** *Suppose Φ is a hidden repetition-free* 2SAT *instance on n variables. Then it is possible to generate a satisfying assignment in time $O(n^2)$.*

Proof The idea is to attempt to learn each clause present in the formula. Suppose we wish to determine if the clause $(x_i \vee x_j)$ is present in Φ (an analogous procedure works to determine if a 1SAT clause x_i is in Φ). We can assume that the clause is unobscured because the presence of an obscured clause does not affect the set of satisfying assignments. Run the following procedure:

Table 1 Violation of the clauses based on the fractional assignments of 1/4 and 3/4.

	$(x_i \vee x_j)$	$(x_i \vee x_k)$	$(x_j \vee x_k)$	$(x_{k_1} \vee x_{k_2})$	$(x_{k_1} \vee \bar{x}_{k_2})$	$(x_i \vee \bar{x}_k)$	$(x_j \vee \bar{x}_k)$	$(\bar{x}_{k_1} \vee \bar{x}_{k_2})$
1/4	1	3/4	3/4	9/16	3/16	1/4	1/4	1/16
3/4	1	1/4	1/4	1/16	3/16	3/4	3/4	9/16

1. First query the oracle with the assignment $x_i = 0$, $x_j = 0$, $x_k = 0$ for $k \neq i, j$. If this is a satisfying assignment, then we are done. Otherwise, we know that there must exist a clause of type:
  (a) $(x_i \vee x_j)$;
  (b) $(x_i \vee x_k)$ for $k \neq i, j$;
  (c) $(x_j \vee x_k)$ for $k \neq i, j$; or
  (d) $(x_{k_1} \vee x_{k_2})$ for $k_1, k_2 \neq i, j$.

5. Now query the oracle with the assignment $x_i = 0$, $x_j = 0$, $x_k = 1$ for $k \neq i, j$. As before, if this is satisfying, we are done. Otherwise, we know that there must exist a clause of type:
  (a) $(x_i \vee x_j)$;
  (b) $(x_i \vee \bar{x}_k)$ for $k \neq i, j$;
  (c) $(x_j \vee \bar{x}_k)$ for $k \neq i, j$; or
  (d) $(\bar{x}_{k_1} \vee \bar{x}_{k_2})$ for $k_1, k_2 \neq i, j$.

5. We can now construct an explicit test for the presence of the clause $(x_i \vee x_j)$. We will propose two fractional assignments to the oracle. If $(x_i \vee x_j)$ is present, then the clause returned each time will be the same. If it is not present, then the returned clause will be different. Formally, query the oracle with the assignment $x_i = 0$, $x_j = 0$, $x_k = \frac{1}{4}$ for $k \neq i, j$ and then with the assignment $x_i = 0$, $x_j = 0$, $x_k = \frac{3}{4}$ for $k \neq i, j$. Table 1 shows the accompanying violations.
It is clear that if $(x_i \vee x_j)$ is present in the formula, then it is returned on both assignments. If it is not present, then from the table we can also see that one of the clauses known to exist from our first query must be returned on the 1/4 fractional assignment. However, one of the clauses known to exist from our second query must be returned on the 3/4 fractional assignment. Thus, the clause returned by the oracle changes when $(x_i \vee x_j)$ is not present.

Notice that the above procedure also works to detect all 1SAT and 2SAT clause types. Therefore, if we complete the above procedure with all $O(n^2)$ clause types without finding a satisfying assignment, then we have identified all unobscured clauses in the formula. It is clear that the conjunction of these clauses forms a formula Φ' such that $\text{SAT}(\Phi') = \text{SAT}(\Phi)$. Therefore, we can use any 2SAT algorithm which runs in time $O(n^2)$ on Φ' to find some satisfying assignment of Φ. $\square$

While the above procedure may seem elementary, it acts as a stepping stone to tackle the harder problem of learning an unknown input instance of quantum 2SAT, which is introduced and discussed in the subsequent sections.

5 Quantum SAT Preliminaries

Notations. A quantum system of n qubits is described using a Hilbert space $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \ldots \otimes \mathcal{H}_n$ where each $\mathcal{H}_i$ is a two-dimensional Hilbert space of the i^{th} qubit. Vectors in $\mathcal{H}$ are called *pure states* and they describe a state of the system. By adding a subscript i to the vector $|\alpha\rangle$ we indicate that $|\alpha\rangle_i$ is defined in the local Hilbert space $\mathcal{H}_i$ of the i^{th}

qubit. Similarly, $|\psi\rangle_{ij}$ denotes a 2-qubit state $|\psi\rangle$ in $\mathcal{H}_i \otimes \mathcal{H}_j$. In any local qubit space $\mathcal{H}_i$, we pick an orthonormal basis $|0\rangle, |1\rangle$ so that every 1-qubit state $|\alpha\rangle$ can be expanded as $|\alpha\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$. We define its orthogonal state by $|\alpha^\perp\rangle := \alpha_1|0\rangle - \alpha_0|1\rangle$;[4] clearly, $\langle\alpha|\alpha^\perp\rangle = 0$. A standard geometrical representation of the state space of a single qubit is the Bloch sphere. The interested reader is referred to [14] for details on the exact correspondence between quantum states and points on the Bloch Sphere.

A more general way to describe a quantum state is by its *density matrix*. Density matrices can be viewed as statistical ensembles of pure states that are described by vectors. A density matrix representation a single pure state $|\psi\rangle$ is given by the matrix $\rho = |\psi\rangle\langle\psi|$. General density matrices are given as a convex sum of density matrices of the pure states with the coefficient summing up to 1: $\sigma = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ where $\forall i, p_i \geq 0$ and $\sum_i p_i = 1$. Alternatively, they are defined as semi-definite operators whose trace is equal to 1. For instance, the density matrix $\frac{1}{2}\mathbb{I}$ can be written as $\frac{1}{2}\mathbb{I} = \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|$. The state of a quantum system can always be fully specified by a density matrix.

Observables in quantum mechanics are associated with Hermitian operators. The eigenvalues of such an operator correspond to the possible outcomes of a measurement. Given such a Hermitian operator A and a pure state $|\psi\rangle$, the expression $\langle\psi|A|\psi\rangle$ is the *expectation value* of A. It is the result we get if we measure A over many copies of the same state $|\psi\rangle$ and average the result. One can use the Chernoff bound to deduce that, with high probability, if we measure A over $\operatorname{poly}(n)$ copies of a state $|\psi\rangle$, we obtain an approximation to $\langle\psi|A|\psi\rangle$ with an additive error of $1/\operatorname{poly}(n)$.

The expectation value of A with respect to a state which is described by a density matrix ρ is given as $\operatorname{Tr}(\rho A)$. Note that if ρ is given by $\rho = \sum_i p_i|\psi_i\rangle\langle\psi_i|$ with $\sum_i p_i = 1$, then $\operatorname{Tr}(\rho A) = \sum_i p_i \langle\psi_i|A|\psi_i\rangle$, which justifies the interpretation of ρ as a statistical ensemble of pure states. Like in the pure state case, using $\operatorname{poly}(n)$ identical copies of ρ, one can estimate the expectation value $\operatorname{Tr}(\rho A)$ up to an additive error of $1/\operatorname{poly}(n)$.

Local Hamiltonians and Quantum SAT. While classically SAT is given as a CSP, quantum kSAT (kQSAT) is defined as a special case of the k-local Hamiltonian problem. A k-local Hamiltonian on n qubits is a Hermitian operator $H = \sum_{e=1}^m h_e$, where each h_e is a *local* Hermitian operator acting non-trivially on at most k qubits. Formally, it is written as $h_e = \hat{h}_e \otimes \mathbb{I}_{rest}$, where $\hat{h}_e$ is defined on the Hilbert space of k qubits, and $\mathbb{I}_{rest}$ is the identity operator on the Hilbert space of the rest of the qubits. When it is clear from the context, we often use h_e instead of $\hat{h}_e$, even while referring to its action on the local Hilbert space.

In physics, k-local Hamiltonians model the local interactions between particles in a many-body system and are the central tool for describing the physics of such systems. The *energy* of the system for every state $|\psi\rangle$ is defined by $E_\psi(H) := \langle\psi|H|\psi\rangle = \sum_e \langle\psi|h_e|\psi\rangle$. The lowest possible energy of the system is called the *ground energy* and is denoted by $E_0(H)$. It is easy to verify that $E_0(H)$ is the lowest eigenvalue of H. The corresponding eigenspace is called the *ground space* of the system, and its eigenvectors are called *ground states*. A central task in condensed matter physics is to understand the properties of the ground space, as it determines the low-temperature physics of the system.

There is a deep connection between the problem of approximating the ground energy of a local Hamiltonian and the classical problem of finding an assignment with minimal violations in a local CSP. In both cases, one tries to minimize a global function that is given

[4] There are, of course, continuously many orthogonal states for every $|\alpha\rangle$, so here we simply choose one in a canonical way.

in terms of local constraints. This connection is evident if we consider the special case when the local Hermitian operators h_e are given as local *projectors* Π_e. Then for any state $|\psi\rangle$, the local energy $\langle\psi|\Pi_e|\psi\rangle$ is a number between 0 and 1 that can be viewed as a measure to how much the state is 'violating' the quantum clause Π_e. When the local energy is 0, the state is inside the null space of the projector Π_e and is said to satisfy the constraint. The total energy of the system, $E_\psi = \langle\psi|H|\psi\rangle = \sum_e\langle\psi|\Pi_e|\psi\rangle$ then corresponds to the total violation of the state $|\psi\rangle$. When the ground energy of the system is 0, necessarily the ground space is the non-vanishing intersection of all the null spaces of the local projectors, and we say that the system is satisfiable. From a physical point of view, such a system is called *frustration-free*, since any ground state of the global system also minimizes the energy of every local term Π_e.

The quantum kQSAT problem is analogous to the classical kSAT problem. Whereas in the kSAT case we are asked to decide whether a k-local CSP is satisfiable or not, in the kQSAT problem we are asked to determine whether the ground energy of a k-local Hamiltonian made of projectors is 0 or not. Unlike the truth values of SAT clauses, however, the ground energy of a k-local Hamiltonian is a continuous function that is sensitive to any infinitesimal change in the form of the local projectors. To make the kQSAT problem more physically relevant, we define it using a promise: Given a k-local Hamiltonian of projectors over n qubits and a value $b > \frac{1}{n^\alpha}$ for some constant α, decide if the ground energy of H is 0 (the YES case) or the ground energy of H is at least b (the NO case). Bravyi [6] showed that kQSAT for $k \geq 4$ is QMA$_1$-complete while Gosset and Nagaj [10] showed that 3QSAT is also QMA$_1$-complete. The class QMA$_1$ stands for 'Quantum Merlin Arthur' with one-way error, and is the quantum generalization of the classical MA$_1$ class with one-way error. The differences are that the witness can be a quantum state over poly(n) qubits, and the verifier can be an efficient quantum machine. In Ref. [6] it was known that 2QSAT has an $O(n^4)$ classical algorithm, and is therefore in P. More recently linear time algorithms for the same problem have been constructed [2, 8].

As the Hamiltonian in a 2QSAT instance is a sum of 2-qubit projectors, every local projector is defined on a 4-dimensional Hilbert space and is of rank $1, 2$ or 3. The non-zero subspace of each projector (the subspace on which it projects) is commonly referred to as the *forbidden space* of that projector and the orthogonal subspace is its *solution space*. Finally, we say that H has *no repetitions* if there does not exist any pair of different projectors $\Pi_e, \Pi_{e'}$ which act non-trivially on the same set of qubits. In the case of repetition free 2QSAT, each projector can also be indexed by the qubit pairs it acts on and the instance can be written as $H = \sum_{(u,v)\in S}\Pi_{uv}$, where $S \subseteq [n] \times [n]$ and each Π_{uv} is non-zero. For any projector Π and a state $|\psi\rangle$, we say that $|\psi\rangle$ *satisfies* Π up to ϵ if $E_\psi(\Pi) := \langle\psi|\Pi|\psi\rangle \leq \epsilon^2$. The energy $E_\psi(\Pi)$ is the *violation energy* of $|\psi\rangle$ with respect to the projector Π. Notice that when the state of the system is described by a density matrix ρ, its violation energy with respect to Π is given by $E_\rho(\Pi) := \mathrm{Tr}(\rho\Pi)$

Finally, a 2QSAT Hamiltonian H is said to have a *Star-like* configuration if there exists a pair of qubits u, v with $\Pi_{u,v} \neq 0$ such that *all* projectors involve either u or v.

Hidden QSAT. The hidden version of QSAT is defined analogously to the classical case. Our task is to decide whether a k-local Hamiltonian $H = \sum_e \Pi_e$ that is made of m k-local projectors over n qubits is frustration-free with $E_0 = 0$ (YES instance) or $E_0 > m \cdot 2\epsilon^2$ (NO instance). Here, $\epsilon > 0$ is some threshold parameter that can be assumed to be inverse polynomially small in n. Moreover, as in H–SAT, here we do not know the Hamiltonian itself; instead we can only send *quantum* states to a "worst violated oracle", which will return the index e of the projector Π_e with the highest violation energy. Since we want to generalize the notion of a probabilistic assignment that is used in H–SAT, we allow ourselves to send the oracle qubits that hold a general quantum state ρ, which can only be described by a density matrix. Recall from the previous section that this can be regarded as an ensemble of pure quantum states. Then the oracle will return the the index e for which $\mathrm{Tr}(\Pi_e \rho)$ is maximized. If the total energy of the proposed state is $\leq m \cdot \epsilon^2$ then the oracle will indicate that a satisfying assignment has been found.

6 Hidden Quantum 1SAT

The algorithm used to solve H–1SAT can be extended to solve the H–1QSAT problem as well. A 1-local projector defined on $\mathbb{C}^2$ is satisfiable if it is of rank at most 1 and can be viewed as setting the direction of the qubit on the Bloch sphere. Unlike the classical case, where we may view the 1SAT clauses as either the $|0\rangle\langle 0|$ or $|1\rangle\langle 1|$ projectors, here the projectors can point in any direction in the Bloch sphere. To handle the continuous nature of the Bloch Sphere, we consider discretizing it by using an ϵ-net that covers the whole sphere. This allows us to generalize the lists of $0-1$ strings used in H–1SAT into lists of n-qubit product states where each qubit is assigned an element of the ϵ-net.

Given a 1-local projector $|\psi\rangle\langle\psi|$, its zero space is spanned by $|\psi^{\perp}\rangle$. We can divide the Bloch sphere into two hemispheres, one hemisphere containing states $|\phi\rangle$ having $|\langle\psi|\phi\rangle| \leq \frac{1}{2}$ and the other with states having $|\langle\psi|\phi\rangle| > \frac{1}{2}$. An n-qubit state $a = |a_1\rangle|a_2\rangle\dots|a_n\rangle$ is called *good* if for each qubit i, where $|\psi_i\rangle$ is its forbidden state, $|\langle\psi_i|a_i\rangle| \leq \frac{1}{2}$ and *bad* if $\forall i, |\langle\psi_i|a_i\rangle| > \frac{1}{2}$. For the n-qubit state $a = |a_1\rangle|a_2\rangle\dots|a_n\rangle$, let $a' := |a_1^{\perp}\rangle|a_2^{\perp}\rangle\dots|a_n^{\perp}\rangle$.

Now, we can sketch the H–1QSAT algorithm. Adapting the process described in Theorem 1 for an arbitrary n-qubit state a gives a list of n-qubit states, $L_{a/a'}$, where at least one state is *good*. This is formally stated in Lemma 3 and the proof is omitted to owing to space constraints.

▶ **Lemma 3.** *Let $a = |a_1\rangle \otimes \dots \otimes |a_n\rangle$ where $|a_i\rangle, |a_i^{\perp}\rangle$ is a basis for qubits i, for $i = 1, \dots, n$. Then one can produce a list, $L_{a/a'} \subset \bigotimes_{i=1}^{n} \{|a_i\rangle, |a_i^{\perp}\rangle\}$ of at most $2mn$ states such that, if the instance is satisfiable, there is at least one* good *n-qubit state in the list. The time taken to produce this list is $O(n^2 m)$.*

However, this only gives us an assignment that violates each projector by $\leq \frac{1}{4}$ while we require assignments that violate each projector by $\leq \epsilon^2$. The key observation involves constructing two lists $L_{a/a'}$ and $L_{b/b'}$ where $b \neq a, a'$ and picking a state from each list. Consider the case when both states are good. Let the states on qubit i from each list be $|a_i\rangle$ and $|b_i\rangle$ respectively. Each state defines a hemisphere R_{i,a_i} and R_{i,b_i} containing all the states that are bad with respect to the forbidden state for qubit i, $|\psi_i\rangle$. Then, $|\psi_i\rangle$, should be contained in $R_{i,a_i b_i} := R_{i,a_i} \cap R_{i,b_i}$. The optimal choice for b_i, given a_i, would be one where $|R_{i,a_i b_i}| \leq \frac{|R_{i,a_i}|}{2}$. Then, similar to performing a binary search on the Bloch Sphere, repeating this process $\log_2\left(\frac{1}{\epsilon}\right)$ times, will give a region consisting of good approximations to the forbidden state.

▶ **Theorem 4.** *Let $\epsilon > 0$. Given a* H–1QSAT *on n qubits containing m projectors, there exists an an $O((2mn)^{2\log\frac{1}{\epsilon}} \cdot mn^2)$ time algorithm, with the property that*

(a) *for a frustration free instance, it outputs an assignment where for each projector, the forbidden state is violated with probability $\leq \epsilon^2$ and*

(b) *for a* NO *instance, the algorithm outputs* UNSAT.

Proof Initially, with no information, for each qubit i, R_i = Bloch sphere. Now the algorithm executes the following steps:

- Start by picking an arbitrary state, say $\bar{a} = |0\rangle^{\otimes n}$, and construct $L_{|0\rangle^{\otimes n}/|1\rangle^{\otimes n}}$ as per the procedure in Lemma 3. For each $a \in L_{|0\rangle^{\otimes n}/|1\rangle^{\otimes n}}$:
  - a defines the region R_{i,a_i} in this branch of the iteration.
  - For $i = 1, \dots, n$ pick a basis $\{|b_i\rangle, |b_i^{\perp}\rangle\}$ such that their equator bisects R_{i,a_i}.
  - Set $\bar{b} = b_1 \dots b_n$, construct $L_{\bar{b}/\bar{b}'}$ and for each $b \in L_{\bar{b}/\bar{b}'}$:
    - The tuples (a, b) define the region $R_{i,a_i b_i}$ in this branch.
    - Repeat the process to find $\bar{c}$ to bisect each $R_{i,a_i b_i}$;
    - Find a new region $R_{i,a_i b_i c_i}$ for each $c \in L_{\bar{c}/\bar{c}'}$.
    - Continue the recursion up to $\log_2\left(\frac{1}{\epsilon}\right)$ depth and let the last list be $L_{z/z^{\perp}}$.
    - Propose $|\phi^{\perp}\rangle = \bigotimes_{i=0}^{n} |\varphi_i^{\perp}\rangle$ where $\forall i, |\varphi_i\rangle \in R_{i,a_i b_i \dots z_i}$ to the oracle. Output $|\phi^{\perp}\rangle$ if the oracle returns YES, otherwise continue.
- Output UNSAT if none of the trials satisfy the instance.

This algorithm essentially creates a recursion tree with each new string created where the width of the recursion at each point is $2mn$ and the depth is $\log_2\left(\frac{1}{\epsilon}\right)$. This leads to $(2mn)^{\log_2 \frac{1}{\epsilon}}$ trials to be proposed at the end and the number of lists created is also $(2mn)^{\log_2 \frac{1}{\epsilon}}$, each at a cost of $O(mn^2)$. Hence, the total running time of the algorithm is $O((2mn)^{\log \frac{1}{\epsilon}} \cdot mn^2)$.

To argue the correctness of this algorithm, we analyze region $R_{i,a_i b_i \dots z_i}$ obtained at the leaf of the recursion tree. At the beginning, let $\forall\ i$, $|R_i| = 1$ (the complete Bloch sphere) and the only guarantee for each list is that there is at least one good string in it. Tracing the path in the recursion tree to the leaf, let us assume that each step of the recursion picks a good string i.e. $a, b, \dots, z$ are all good strings. For a and $\forall i$, the forbidden state $|\psi_i\rangle$ is in the opposite hemisphere to $|a_i\rangle$ which reduces the size of the region to $|R_{i,a_i}| = 1/2$. Taking (a, b) at the next iteration, the region for each qubit is the over lap of two hemispheres $R_{i,a_i} \cap R_{i,b_i}$ and by construction, since b_i bisects R_{i,a_i}, the overlaps of the hemispheres also bisect R_{i,a_i} setting $|R_{i,a_i b_i}| = 1/4$. As this pattern continues, each step of the iteration halves the region for qubit i and we are left with regions of size at most ϵ at the end of the branch. If the instance is satisfiable, the state proposed will satisfy each projector up to ϵ resulting in the oracle to return YES. Of course, when one of the strings chosen is bad, the proposal $|\phi_j^{\perp}\rangle$ for some qubit j will end up having a large inner product with the forbidden state $|\psi_j\rangle$ and will result in the oracle returning the ID of the projector involving j. This concludes the proof. ☐

7 Hidden Quantum 2SAT

This section deals with a 2QSAT instance that is hidden and can only be accessed by a worst-violation oracle. We show how learn the underlying local Hamiltonian to precision ϵ by finding 2-local projectors Π'_e such that $\|\Pi_e - \Pi'_e\| \leq \epsilon$ for every projector Π_e. This yields an approximate local Hamiltonian $H' = \sum_e \Pi'_e$ whose ground energy is at most $m\epsilon$ away from the ground energy of the original Hamiltonian $H = \sum_e \Pi_e$. If ϵ is set such that $m\epsilon$ is much smaller than the promise gap of the initial Hamiltonian H (which merely requires $\epsilon < 1/\operatorname{poly}$), then the Hamiltonian H' will have a promise gap as well. This is stated in the theorem below with the proof and algorithm details omitted owing to space constraints.

▶ **Theorem 5.** *Given a* H–2QSAT *problem* $H = \sum_{(u,v)} \Pi_{uv}$ *on* n *qubits, and precision* ϵ*. If the interaction graph for* H *is not Star-like, then there is an* $O(n^4 + n^2 \log\left(\frac{1}{\epsilon}\right))$ *algorithm that can find an approximation* $H' = \sum_{(u,v)} \Pi'_{uv}$ *where* $\forall\ (u,v), \|\Pi'_{uv} - \Pi_{uv}\| \leq \epsilon$

The algorithm proceeds by:

1. Identifying two pairs of qubits $(i,j) \neq (k,\ell)$ on which two projectors are defined, Π_{ij} and $\Pi_{k\ell}$, and finding a constant approximation for these projectors;
2. Improving the constant approximation of the two projectors recursively so that the approximation improves by a factor of 2 in each iteration and
3. Using the ϵ-approximation of a projector to identify the rest of the independent projectors and approximating them to ϵ-precision.

Using the H' output by the above algorithm in a procedure which could find a good approximation to the ground energy of H' would completely solve H–2QSAT. At this time, though, existing 2QSAT algorithms [2, 6, 8] are not robust to such errors and seem to require $\frac{1}{\exp(n)}$ precision. Our algorithm for H–2QSAT does allow one to learn the projectors to exponential precision, since the dependence on ϵ in Theorem 5 is merely logarithmic. However, in this parameter regime our algorithm is somewhat unrealistic, as this would require the oracle to be able to distinguish between values that are exponentially close together[5]. If our oracle were constrained to be implementable in polynomial time by an experimenter, acting on polynomially many copies of the proposed state ρ, then one could only learn the instance up to error $\epsilon = \frac{1}{\text{poly}(n)}$. A natural open question is to determine whether one can still solve 2QSAT when one only knows the individual clauses to inverse polynomial precision; we believe this is a fundamental question about the nature of 2QSAT, which is left for future work.

References

1 Daniel S. Abrams and Seth Lloyd. Nonlinear quantum mechanics implies polynomial-time solution for NP-complete and #P problems. *Phys. Rev. Lett.*, 81:3992–3995, 1998.

2 Itai Arad, Miklos Santha, Aarthi Sundaram, and Shengyu Zhang. Linear time algorithm for quantum 2SAT. *CoRR*, abs/1508.06340, 2015. To appear in the 43rd International Colloquium on Automata, Languages and Programming. URL: `http://arxiv.org/abs/1508.06340`.

3 Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett.*, 8(3):121–123, 1979. Erratum: Information Processing Letters 14(4): 195 (1982).

4 Xiaohui Bei, Ning Chen, and Shengyu Zhang. On the complexity of trial and error. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *STOC*, pages 31–40. ACM, 2013. `doi:10.1145/2488608.2488613`.

5 Xiaohui Bei, Ning Chen, and Shengyu Zhang. Solving linear programming with constraints unknown. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP (1)*, volume 9134 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 2015. `doi:10.1007/978-3-662-47672-7_11`.

6 Sergey Bravyi. Efficient algorithm for a quantum analogue of 2-SAT. In Kazem Mahdavi, Deborah Koslover, and Leonard L. Brown, editors, *Contemporary Mathematics*, volume

[5] This seems to give the oracle too much power - because having the ability to distinguish exponentially close quantum states would allow one to solve PP-hard problems [1]. In contrast all of the problems considered are in NP due to the presence of classical, poly(n) size witnesses.

536. American Mathematical Society, 2011. URL: http://arxiv.org/abs/quant-ph/0602108.

7 S. A. Cook. The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium*, pages 151–158, New York, 1971. ACM.

8 Niel de Beaudrap and Sevag Gharibian. A linear time algorithm for quantum 2-SAT. *CoRR*, abs/1508.07338, 2015. To appear in 31st Conference on Computational Complexity. URL: http://arxiv.org/abs/1508.07338.

9 Shimon Even, Alon Itai, and Adi Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. Comput.*, 5(4):691–703, 1976.

10 David Gosset and Daniel Nagaj. Quantum 3-SAT is QMA1-complete. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 0:756–765, 2013. doi:10.1109/FOCS.2013.86.

11 Gábor Ivanyos, Raghav Kulkarni, Youming Qiao, Miklos Santha, and Aarthi Sundaram. On the complexity of trial and error for constraint satisfaction problems. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP*, volume 8572 of *Lecture Notes in Computer Science*, pages 663–675. Springer, 2014. doi:10.1007/978-3-662-43948-7_55.

12 M. R. Krom. The decision problem for a class of first-order formulas in which all disjunctions are binary. *Mathematical Logic Quarterly*, 13(1-2):15–20, 1967. doi:10.1002/malq.19670130104.

13 L. A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.

14 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.