

# Generalizing an Exactly-1 SAT Solver for Arbitrary Numbers of Variables, Clauses, and $K$

Francesco Piro<sup>a</sup>, Mehrnoosh Askarpour<sup>b</sup> and Elisabetta Di Nitto<sup>a</sup>

<sup>a</sup>DEIB, Politecnico di Milano, Italy

<sup>b</sup>McMaster University, Canada

## Abstract

Quantum computers promise to allow for great improvements in the solution of  $k$ -SAT problem, determining whether a set of clauses with  $k$  variables have a satisfiable boolean assignment, which is one of the fundamental problems of computational logic. Given the recent advancements of quantum computers, we argue that they allow for great improvements in solving the  $k$ -SAT problem. In order to evaluate this possibility, in this work, we generalized a pre-existing quantum 3-SAT solver [1] to the most general case for the number of variables, clauses, and  $k$ , using the IBM Qiskit library [2]. We extended basic gates and steps of the underlying algorithm to reduce the number of used qubits and gates, in order to deal with the decoherence problem [3]. We tested our solution on complex instances of  $k$ -SAT, which preserved the exponential speedup promised by Grover algorithm [4].

## Keywords

Quantum Computing, Satisfiability Problem, Quantum Logic, Grover Algorithm

## 1. Introduction

Quantum computing has demonstrated promising results in several areas of computer science and is changing our way of studying algorithms in the following years.

One of the areas that could drastically be touched by the effects of quantum computing is Logic, in particular, the question of equivalence of the  $NP$  and  $P$  classes. Answering this question will have a powerful impact on areas such as artificial intelligence and security.

$k$ -SAT problem—determining whether a given set of clauses have a satisfiable assignment—is one of the fundamental problems of computational logic, and the first problem proven to be  $NP$ -complete [5]. This result has brought big changes in theory of computation because it allows to prove a problem to be  $NP$  or  $NP$ -complete by demonstrating that it is reducible to an instance of the  $k$ -SAT. This means that the computational speedup that we can exploit with quantum algorithms can be also applied to all the problems that can be reduced to the  $k$ -SAT. An instance of the  $k$ -SAT problem is characterized by the number of variables ( $n$ ), the number of clauses ( $m$ ), and the maximal length of the clauses ( $k$ ).

This paper evaluates the application of quantum computing to solve arbitrary instances of the Exactly-1  $k$ -SAT problem and to show the enhancements provided by a quantum solver (an

---

1st Quantum Software Engineering and Technology Workshop


EMAIL: francesco.piro@mail.polimi.it (F. Piro); askarpom@mcmaster.ca (M. Askarpour);

elisabetta.dinitto@polimi.it (E. Di Nitto)

ORCID:



© 2020 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

exactly-1  $k$ -SAT problem is one where there exists a satisfying assignment with exactly one true literal in each of the  $m$  clauses). We discovered the quantum solver introduced by Nannicini [1] that exploits the quantum principle of *amplitude amplification* [6] using the Grover search algorithm [4] to find a solution up to an Exactly-1 3-SAT problem with  $n = 3$  and  $m = 3$ . We finally implemented a generalization of this solver, using the IBM Qiskit library [2], to solve problems to the most general case of arbitrary values for  $n$ ,  $m$ , and  $k$ . This paper describes our approach and derived conclusions. The complete implementation, graphics of the whole quantum circuits, and all the material we used to start approaching the quantum computing world are present at the following Git repository [7].

The rest of this paper is structured as follows: Section 2 reviews state of the art; Section 3 provides preliminary background; Section 4 presents our approach; Section 5 reports the evaluation procedure and its results; finally, Section 6 draws some conclusions and possible future works.

## 2. State of the art

As mentioned earlier, one of the most interesting questions of quantum computing is whether it can prove the equivalence of the  $NP$  and  $P$  classes.

Grover search algorithm is an important asset to approach this question because it is used to solve several  $NP$  or  $NP$ -complete problems with considerably reduced computational efforts. For example, *database search problem* has improved with Grover's quadratic speedup as reported in [4]; Gilles et al. [8] applied it to find the collisions of an  $r$ -to-*one* function and managed to reduce the temporal complexity to  $O(\sqrt[3]{N/r})$ ; Guodong et al. [9] exploited it indirectly by using the quantum counting algorithm, which relies on a combination of Grover and the Shor's factoring algorithm [10], to enhance the *polynomial root-finding problem*.

These examples are all indicators of how Grover could be useful to solve other logic problems, such as the one we are tackling. For instance, an interesting recent work by Porfiris [11] shows how to perform password cracking using quantum computation, based on the Exactly-1 3-SAT problem. Passwords can be visualized as a sequence of characters, hence solving the Exactly-1 SAT perfectly fits to model the search of that particular string with exactly one character that matches the one of the password. However, this approach is able to crack only passwords up to three characters, as it uses an Exactly-1 3-SAT solver. Another example is the work of Valentin Bura [12] on the kernelization of Exactly-1 3-SAT problems in order to show the power of Gaussian elimination. Thanks to kernelization, the author is able to prove a reduction in both time and space complexities for the corresponding counting problem. This results to a complexity which is still exponential but very near to a base of 1. At this point, applying a quantum algorithm on top of such kernelization method would bring us to an additional reduction of the complexity, affecting the exponent, always nearer to linearity.

## 3. Background

This paper relies on the work by Nannicini [1], where he proposes a solver that can find a solution to the Exactly-1 3-SAT problem by using Grover's search algorithm only for a maximal

number of three variables and clauses. Because of the quantum physics principle of amplitude amplification, this algorithm finds a satisfactory assignment by storing the formulation of an Exactly-1 3-SAT problem, iterating the following steps for a particular number of times, evaluating in the end the resulting state:

- a. *Initialization Step*: brings the problem state to the uniform superposition applying *Hadamard (H)* gates on all the qubits.
- b. *Problem Encoding Step*: encodes all the clauses of the problem inside the quantum circuit bringing for each clause one qubit that is updated to find the solution.
- c. *Inversion about the Average Step*: updates once more the coefficients of the variables that correspond to the correct solution.

## 4. Proposed Generalization

In this section, we explain our implemented generalization of Nanncini's solver to manage arbitrary numbers of variables and clauses for an Exactly-1 k-SAT problem, by maintaining the structure of the three steps explained in Section 3 and focusing on the following two aspects:

1. To allow for arbitrary numbers of clauses ( $m$ ) and variables ( $n$ )
2. To allow for arbitrary maximum length of the clauses ( $k$ )

Considering a problem with  $n$  variables and  $m$  clauses, the total required qubits in its representing circuit includes  $n$  for variables, one for the output register, and  $m$  for clauses. Additionally, the *Problem Encoding Step* and the *Inversion about the Average Step* need an  $n$  dimensional controlled NOT gate for the exploitation of the amplitude amplification principle. However, Qiskit does not allow us to build controlled-NOT gates with dimensions larger than two, and we had to solve this issue by using the smallest number of qubits possible. To realize such gate we concatenate the results of doubly controlled-NOT gates applied on two of the qubits at a time thus introducing additional  $n - 2$  ancillary qubits.

**Algorithm generalization** Provided the preliminaries on the number of qubits, we discuss our implementation in the rest of this section through an illustrative example of an Exactly-1 4-SAT problem with four variables and four clauses. To formalize an exactly-1 k-SAT problem we need to specify the variables on which it is defined, the clauses and the maximal length of the clauses. In the rest of this paper we call  $X$  the set of the variables of the problem,  $C$  the set of the clauses and for each clause  $C_i$  we define the variables that compose it with their respective polarity. The number of variables in the clauses will never exceed the number  $k$  decided, the cardinality of  $X$  is  $n$  and the cardinality of  $C$  is  $m$ . In the end remember that this formulation can also be seen as the *Conjunctive Normal Form (CNF)* of the clauses defined. This representation allows to visualize better the possible solution of the problem and understand a-priori if a solution actually exists.

Hence, the formalization of the exactly-1 4-SAT problem with four variables ( $X = \{x_1, x_2, x_3, x_4\}$ ) and four clauses ( $C = \{C_1, C_2, C_3, C_4\}$ ) is:

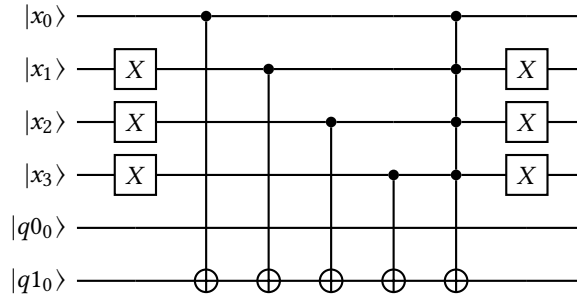
$$\begin{aligned}
C_1 &= \{x_1, \bar{x}_2, \bar{x}_3, \bar{x}_4\} \\
C_2 &= \{\bar{x}_1, x_2, \bar{x}_4\} \\
C_3 &= \{\bar{x}_1, \bar{x}_2, \bar{x}_3, x_4\} \\
C_4 &= \{\bar{x}_2, x_3, \bar{x}_4\}
\end{aligned}$$

That can be expressed in the CNF as:

$$CNF \equiv (x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_2 \vee x_3 \vee \bar{x}_4)$$

We can now list the steps of the generalized algorithm:

- a. *Initialization Step*: the initial state of the quantum circuit is set by applying Hadamard gates for each variable and for the output, in order to bring it to the uniform superposition.
- b. *Problem Encoding Step*: each clause is encoded in the circuit using gates that allow to bit-flip the qubit corresponding to the clause if and only if it has exactly one true literal (this is the definition of searching for a solution of an Exactly-1 k-SAT problem). Considering  $C_1$ , we want to flip the qubit  $q_{1_0}$  if and only if  $x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4$  has exactly one true literal in the problem. First we bring each variable to  $q_{1_0}$  with their respective polarity by using NOT gates and apply a CNOT of all the variables so that we obtain  $|q_{1_0}\rangle = |q_{1_0} \oplus x_1 \oplus \bar{x}_2 \oplus \bar{x}_3 \oplus \bar{x}_4\rangle$ . To complete the double implication of flipping  $q_{1_0}$  for the exactly one true variable, we need a quadruply controlled NOT gate between the four variables targeting  $q_{1_0}$ . Finally we have obtained  $|q_{1_0}\rangle = |q_{1_0} \oplus x_1 \oplus \bar{x}_2 \oplus \bar{x}_3 \oplus \bar{x}_4 \oplus (x_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4)\rangle$ . Hence, the theoretical circuit that we should realize is showed in the following Figure 1.



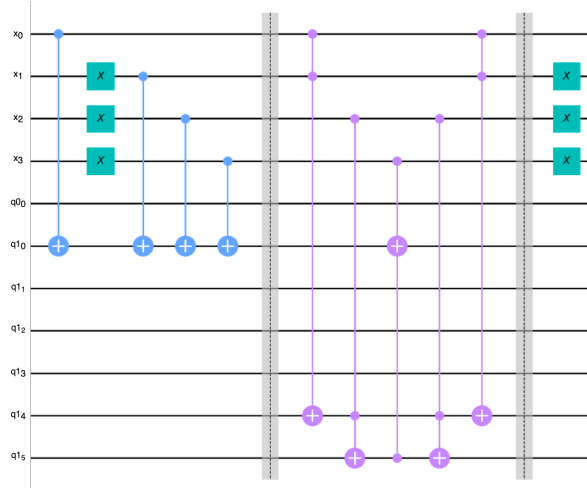
**Figure 1:** Theoretical encoding of clause  $C_1$  of our illustrative example.

However, as we mentioned before, we are not allowed to realize multiply controlled-NOT gates in Qiskit. The procedure that we adopted to encode such gate in the actual quantum circuit is shown in Figure 2. We can see here how ancillary qubits are used to concatenate the results of the doubly controlled-NOT gates and, after the result is stored in  $q_{1_0}$ , we reset the state of these qubits applying once more the same doubly controlled-NOT gates.

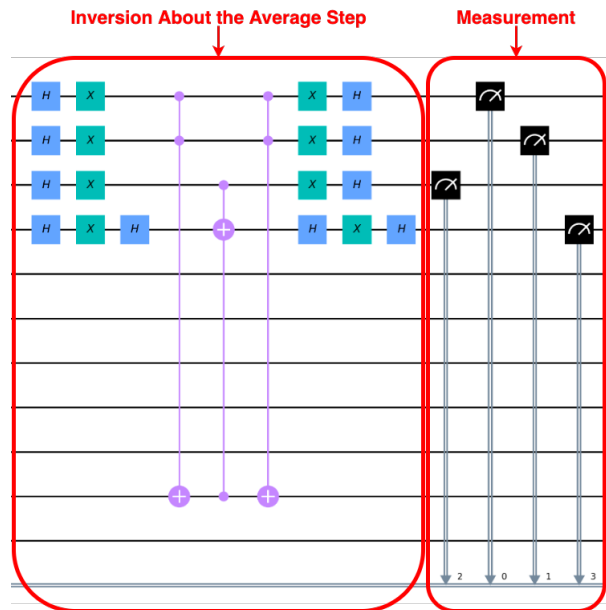
- c. *Inversion About the Average Step*: in this last step, we modify the coefficients corresponding to the correct solution by applying the unitary operation  $W$  defined as follows:

$$W = (-\bigotimes^n \mathcal{H})D(\bigotimes^n \mathcal{H})$$

where  $n$  is the number of variables and  $D$  is a diagonal matrix  $diag(-1, 1, \dots, 1)$  of size  $2^n$ . In particular, to realize  $D$  we needed once more to generalize the algorithm since it consists of a  $(n - 1)$ -controlled NOT gate between the first  $n - 1$  variables, targeting the last one. As shown in Figure 3, we exploited the same trick to deal with Qiskit's limitation. Moreover, we see the measurement of the four qubits representing the state, brought to the classical register  $c0$  which stores the final solution.

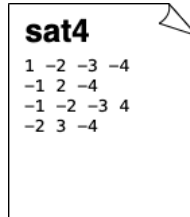


**Figure 2:** Encoding step of our illustrative example. The two barriers highlight the quadruply-controlled NOT gate between  $x_0, x_1, x_2$  and  $x_3$  targeting the qubit  $q1_0$  of clause  $C1$ .



**Figure 3:** Inversion About the Average step of our illustrative example. First we can see how the triply-controlled NOT gate conjuncts  $x_0, x_1$  and  $x_2$  targeting  $x_3$ . Then, at the end of the circuit the state is measured on a classical register of 4 bits.

**Algorithm specifics** The problems are fed to the solver in an input file when running the program. Variables have to be specified with the following notation:  $x$  followed by the integer  $i$  that identifies that variable (variable  $x_1$  will be  $x1$ ). If we want to use  $n$  variables, we can define  $x$ -es from  $i = 1$  up to  $i = n$ ; it is not allowed to define a number of  $n$  different variables with pedices not in the range  $[1, n]$ . Each line contains the definition of a clause, line 1 will define clause  $C_1$ , line 2 corresponds to  $C_2$  and so on. Variables in the clauses have to be specified in ascending order, to express the negative polarity a minus sign ( $-$ ) precedes the respective  $x$ . Considering the Exactly-1 4-SAT defined in this section, with four variables and four clauses, the file containing its definition (let us call it `sat4`) will have the following shape:



```

sat4
1 -2 -3 -4
-1 2 -4
-1 -2 -3 4
-2 3 -4

```

**Figure 4:** File `sat4` defining an Exactly-1  $k$ -SAT problem with 4 variables and 4 clauses.

Additionally, the solver also gets the iteration number of the *Problem Encoding* and the *Inversion About the Average* steps as input. The number of times that we run Grover's algorithm influences the results provided by the solver.

**Grover iterations** In his original work, Grover [4] described how difficult it is to know a priori the number of times to iterate his algorithm to find the best solution of a search problem. Nannicini [1] found that for instances of the Exactly-1 3-SAT, two iterations is the number that provides solutions with the highest probability. In our study, considering instances with more than just three variables and three clauses but also with arbitrary  $k$ , we were able to understand that even more iterations are needed to find the best solution. By doing more iterations, longer circuits are realized: this increases the number of gates and the execution time of the solver.

## 5. Evaluation

We evaluated our proposed quantum solver with arbitrary  $n$  and  $m$  values up to the 4-SAT problem and achieved correct solutions with very high probability. The results shown in this section are generated by the quantum simulator of the Qiskit library since the execution on real quantum devices is still not possible for such complex instances of the SAT problem. Quantum simulators execute the algorithm on a hardware where noise is minimal, hence allowing to use significant numbers of qubits and gates to implement the algorithms. We have to take this into account when consulting the results on the histograms, being conscious that very high probabilities make us expect that the correct results will be also displayed on the real quantum device since the additional noise introduced is not enough to compromise the execution. To show the results provided by our solver we decided also to highlight the number of iterations needed to find the best solution on each particular problem. As we have already discussed, the

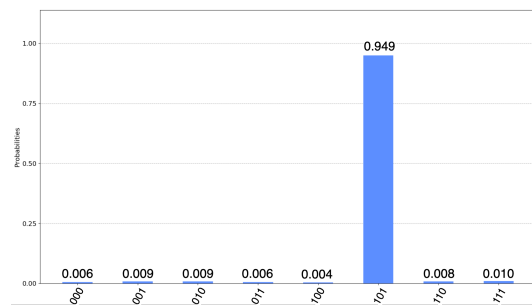
number of iterations affects significantly the results; in order to make this evident we will first present the solution obtained with the best number of iterations and then compare it with the results provided if we tried to increase by one this number. We will call the result with the best number of iterations *Best solution* while the one with one additional repetition as the *Next solution*. In the *Next solution* we see that the probability distribution changes a lot, in particular the correct solution drastically decreases its value and random other incorrect combinations increase their probability.

We start with the same problem described by Nannicini in his work [1] to prove the consistency of the generalized solver. Then, we continue with an unsolvable Exactly-1 3-SAT (which has not the comparison), with an Exactly-1 3-SAT problem of five variables and five clauses, and finally an Exactly-1 4-SAT with four variables and four clauses.

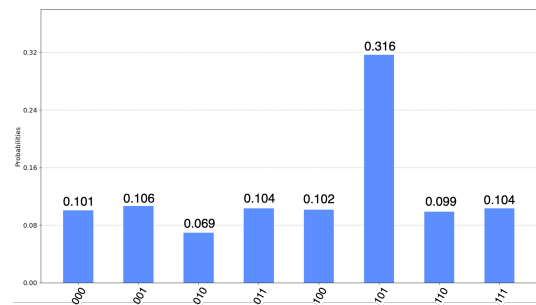
**Problem 1** Consider the 3-SAT problem defined over the set of variables  $X = \{x_1, x_2, x_3\}$  and the three clauses  $C = \{C_1, C_2, C_3\}$ , such that:

$$\begin{aligned} C_1 &= \{x_1, x_2, \bar{x}_3\} \\ C_2 &= \{\bar{x}_1, \bar{x}_2, \bar{x}_3\} \\ C_3 &= \{\bar{x}_1, x_2, x_3\} \end{aligned}$$

The formulation of the problem implies that the solution is  $S = \{x_1, \bar{x}_2, x_3\}$ .



(a) **Problem 1** best solution



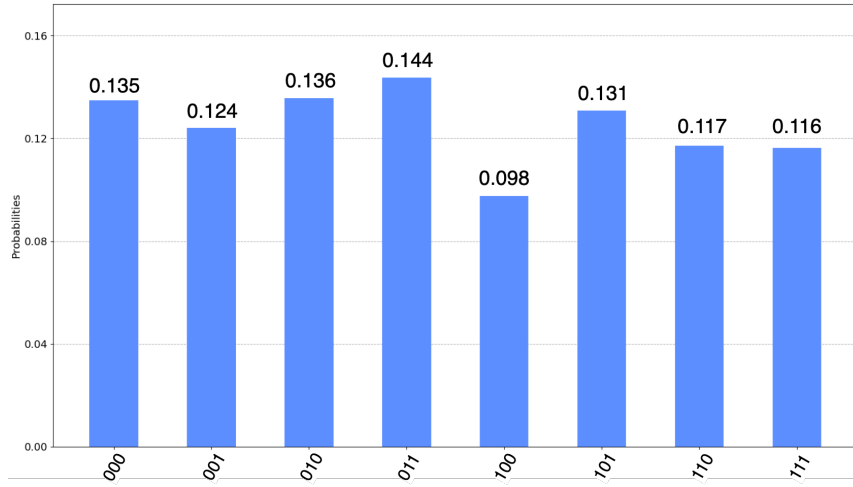
(b) **Problem 1** next solution

The result of our solver, shown in Figure 5a, provides the same solution as  $S$  with a probability near 95%, which is a very promising result and consistent with the one by Nannicini. Figure 5a shows the best solution obtained with 2 iterations of Grover's algorithm. Comparing it with Figure 5b, obtained with 3 iterations, we see that the probability of the correct solution decreases of around the 60%. In addition, all the other combinations of variables have increased their probability rounding all near the 10% which is very close to the 30% of 101.

**Problem 2** Consider the 3-SAT problem defined over the set of variables  $X = \{x_1, x_2, x_3\}$  and the three clauses  $C = \{C_1, C_2, C_3\}$ , such that:

$$\begin{aligned} C_1 &= \{x_1, x_2, \bar{x}_3\} \\ C_2 &= \{\bar{x}_1, x_2, \bar{x}_3\} \\ C_3 &= \{x_1, x_2, x_3\} \end{aligned}$$

This problem seems to be trivially satisfiable, for example, with  $S = \{\bar{x}_1, x_2, \bar{x}_3\}$ . However, this is not a solution for the Exactly-1 formulation since both  $\bar{x}_1$  and  $\bar{x}_3$  in the second clause make the disjunction true. Thus, the Exactly-1 **Problem 2** has no solution.



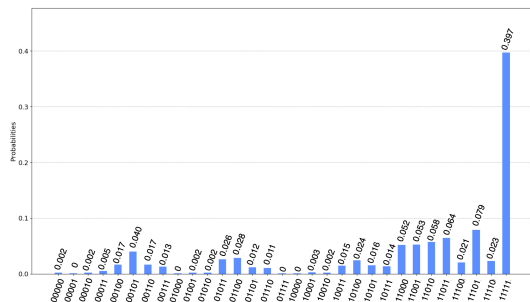
**Figure 6:** **Problem 2** outcome has no solution for the Exactly-1 formulation

The result of our solver, shown in Figure 6, confirms the lack of an acceptable answer as the probabilities of all the possible solutions are very close (i.e. around 10% each), which does not allow to choose between one of them.

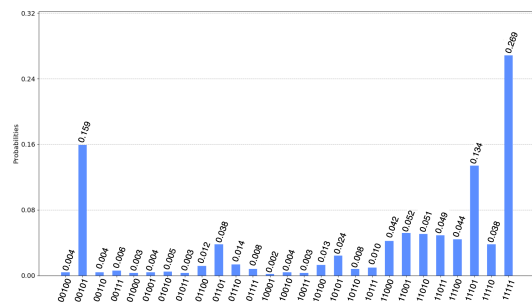
**Problem 3** Consider the 3-SAT problem defined over the set of variables  $X = \{x_1, x_2, x_3, x_4, x_5\}$  and the five clauses  $C = \{C_1, C_2, C_3, C_4, C_5\}$ , such that:

$$\begin{aligned}
 C_1 &= \{x_1, \bar{x}_2, \bar{x}_3\} \\
 C_2 &= \{x_2, \bar{x}_3, \bar{x}_4\} \\
 C_3 &= \{x_3, \bar{x}_4, \bar{x}_5\} \\
 C_4 &= \{x_4, \bar{x}_5, \bar{x}_1\} \\
 C_5 &= \{x_5, \bar{x}_3, \bar{x}_4\}
 \end{aligned}$$

Again, the formulation of the problem suggests the solution to be  $S = \{x_1, x_2, x_3, x_4, x_5\}$ .



(a) **Problem 3** best solution



(b) **Problem 3** next solution

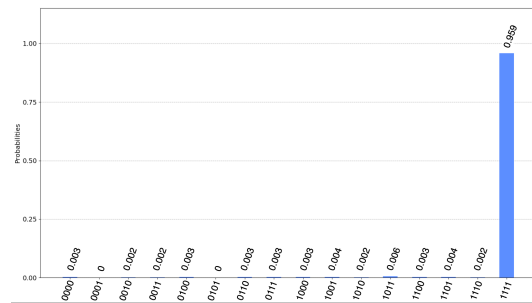


The *Best solution* of our solver (obtained with 3 iterations), shown in Figure 7a, is 11111 which is the same as the expected result  $S$ . As the figure shows, the answer is provided with a probability of near the 40% which is significantly higher than all the other 5 qubits combinations. In this case, the probability of the *Next solution* (Figure 7b obtained with 4 iterations) decreases of 15% for the correct result and for the other combinations it distributes the remaining probability; in particular for 00100 that reaches a value around the 16%.

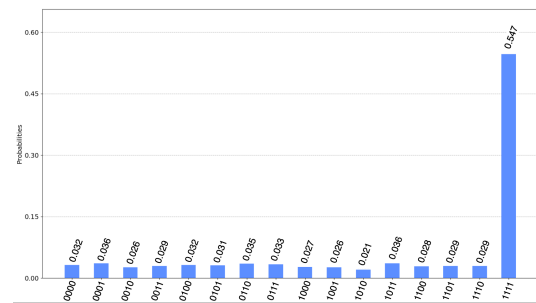
**Problem 4** Consider the problem presented in Section 4 which is a 4-SAT problem defined over the set of variables  $X = \{x_1, x_2, x_3, x_4\}$  and the four clauses  $C = \{C_1, C_2, C_3, C_4\}$ , such that:

$$\begin{aligned} C_1 &= \{x_1, \bar{x}_2, \bar{x}_3, \bar{x}_4\} \\ C_2 &= \{\bar{x}_1, x_2, \bar{x}_4\} \\ C_3 &= \{\bar{x}_1, \bar{x}_2, \bar{x}_3, x_4\} \\ C_4 &= \{\bar{x}_2, x_3, \bar{x}_4\} \end{aligned}$$

The formulation of the problem points to  $S = \{x_1, x_2, x_3, x_4\}$  as the expected solution.



(a) **Problem 4** best solution



(b) **Problem 4** next solution

The result from our solver, as shown in figure 8a, is the string 1111 with a probability near 90%, significantly higher than all the other probabilities and conforms to the expected solution. Comparing the *Best solution* in Figure 8a (obtained with 3 cycles) with the *Next* one (obtained with 4 cycles) in Figure 8b we see that the probability distribution maintains the same shape and equally spreads on all the other incorrect combinations increasing for each up to around the 3%. The correct solution is still higher than all the other probabilities but decreased of 35% with respect to the one obtained with the best number of iterations.

## 6. Conclusions and Future Work

The Exactly-1 k-SAT problem has a fundamental importance in computational theory. The proposed solver in this paper can tackle any instance of this problem with arbitrary numbers of variables, clauses, and  $k$  with quadratic speedup, thanks to Grover's search algorithm.

The four instances reported here are the most significant to show the behavior of the solver. The first three show how it works with more than just three variables and clauses, on the Exactly-1 3-SAT problem which is the most popular in applications. The obtained results seem very promising, as the probability of the correct solution is significantly higher than all the others; we believe that the same results will be obtained also on a real quantum device where

the noise is not mitigated as in a simulated computation. The generalization has been tested also on **Problem 4** that considers four variables and clauses with a maximal length  $k = 4$ . Also in this case, the correct solution has probability around 90%, definitely better than all the other combinations.

We compared the *Best* and the *Next* solutions and deduced that the correct iteration number for the second and third steps of Grover's algorithm depends on the complexity of the problem. It is coherent with what these steps do, which is increasing the coefficients of the correct solution. Hence, more complex instances correspond to more possible solutions, and therefore, the probability spreads on a larger set of qubits combinations; more iterations allow for a larger detach between the probability of the correct solution and others. The simple instances studied by Nannicini in his work required at most two iterations while in our study we discovered that already an Exactly-1 3-SAT with five variables and five clauses as well as **Problem 4** executes at best with 3 iterations. As future work, we plan to determine the relationship between the complexity of an Exactly-1 k-SAT problem and the best number of iterations, hence a formula that finds the cycles once we provide  $n$ ,  $m$  and  $k$ .

The reported histograms are the result of the execution on the `qasm_simulator` provided by Qiskit which allows for high number of qubits. As we mentioned before, we can conclude that our work will also provide correct results on real quantum devices, given to the high probabilities that we have obtained. We tried to execute the problems also on a real quantum device, in particular `ibmq_16_melbourne` provided by *IBM-Quantum experience*. All the executions fail because of the transpilation function of Qiskit. It optimizes the encoding of the quantum algorithm on the real device, trying to reduce the number of qubits needed, adding gates to perform the same quantum operations. For complex algorithms like the one that we implemented, the transpilation has two main issues: (i) adding more gates causes the decoherence problem, and (ii) the huge number of gates which leads to the error reported by the machine. We obtain a too long circuit that needs an execution time greater than the circuit repetition rate.

The current state of quantum technology does not tackle this issue, as the most powerful quantum machine provided by the library that we used is the `ibmq_16_melbourne`.

## References

- [1] G. Nannicini, An introduction to quantum computing, without the physics (2017).
- [2] Qiskit textbook, <https://qiskit.org/textbook/>, Last Accessed in 2020.
- [3] M. A. Schlosshauer, Decoherence: and the quantum-to-classical transition, Springer Science & Business Media, 2007.
- [4] L. K. Grover, A fast quantum mechanical algorithm for database search, in: G. L. Miller (Ed.), Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996, ACM, 1996, pp. 212–219. URL: <https://doi.org/10.1145/237814.237866>. doi:10.1145/237814.237866.
- [5] S. A. Cook, The complexity of theorem-proving procedures, in: M. A. Harrison, R. B. Banerji, J. D. Ullman (Eds.), Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA, ACM, 1971, pp. 151–158. URL: <https://doi.org/10.1145/800157.805047>. doi:10.1145/800157.805047.
- [6] L. K. Grover, Quantum computers can search rapidly by using almost any transformation, Physical Review Letters 80 (1998) 4329.
- [7] Repository of our implementation and experiments, [github/repository](https://github.com/qiskit/qiskit), 2020.
- [8] G. Brassard, P. Høyer, A. Tapp, Quantum cryptanalysis of hash and claw-free functions, in: C. L. Lucchesi, A. V. Moura (Eds.), LATIN'98: Theoretical Informatics, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 163–169.
- [9] S. Guodong, S. Shenghui, X. Maozhi, Quantum algorithm for polynomial root finding problem, in: 2014 Tenth International Conference on Computational Intelligence and Security, 2014, pp. 469–473.
- [10] P. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. 26 (1994) 1484–1509.
- [11] T. Porfiris, Exactly-1 3-sat problem and grover's algorithm: Breaking the rules of classical systems, [linkedin/Porfiris/grover-in-sat-problem](https://www.linkedin.com/company/porfiris/), Accessed August 2019.
- [12] V. Bura, A kernel method for positive 1-in-3-sat (2018).