Automatically Solving NP-Complete Problems on a Quantum Computer

Shaohan Hu, Peng Liu, Chun-Fu (Richard) Chen, Marco Pistoia
IBM Research, Yorktown Heights, NY, 10598 USA
shaohan.hu@ibm.com,{liup,chenrich,pistoia}@us.ibm.com

ABSTRACT

In recent years, tremendous efforts from both the industrial and the academic research communities have been put into bringing forth quantum computing technologies. With the potential proliferation of universal quantum computers on the horizon, quantum *computing*, however, is still severely grounded by numerous grave barriers, which lead to its low accessibility and practicality. For example, the vastly different underlying computing models, combined with the steep background knowledge requirements, makes it extremely difficult, if possible at all, for most software engineering researchers and practitioners to even begin to design or implement quantum algorithms or softwares in practice. To overcome this problem, we, in this paper, propose a design that largely circumvents said accessibility and practicality barriers, by providing *an end-to-end quantum computing framework for solving NP-complete problems* via reduction. We fully implemented a toolkit under our design framework. With this toolkit, software engineering researchers and practitioners would be able to enjoy the speedup and scalability benefits of universal quantum computers without necessarily having to have prior knowledge on quantum computing.

KEYWORDS

Universal quantum computing, Grover's algorithm, NP-Complete, Reduction, Satisfiability

ACM Reference Format:
Shaohan Hu, Peng Liu, Chun-Fu (Richard) Chen, Marco Pistoia. 2018. Automatically Solving NP-Complete Problems on a Quantum Computer. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3194959

1 INTRODUCTION

Having dictated classical computing technology advances for decades, today Moore's Law has been dragged by the limit of physics to a crawl [1]. Thus, in recent years, heavy interests and investments from the industry as well as the academic community are poured into bringing quantum computers into reality [2, 3]. However, with half a century of research efforts on theoretical quantum computing, there has only been a small set of *quantum algorithms* ever discovered. And from a software point of view, to be able to even understand a simple piece of quantum program, let alone design and develop sophisticated tool stacks, requires a fair amount of background knowledge on quantum computing systems, which in general cannot be assumed for most software engineering researchers and practitioners.

Towards overcoming this *accessibility and practicality gap* surrounding quantum computing, we propose *an end-to-end quantum computing framework for solving NP-complete problems*. The focus on NP-complete problems comes from the simple fact that they are computationally hard. Compared to them, easier problems can already be handled with ease by classical computers. It is therefore more meaningful that the potential speedup brought about by quantum computers be used for solving the harder problems. Fig. 1 depicts our quantum NP-complete solution framework. As shown, one key feature is that our solution exploits reduction [4, 5], so that all NP-complete problems can potentially be solved by a core quantum solver that is directed towards a single NP-complete problem—all other NP-complete problems can be solved by using a reduction wrapper around the core solver. By designing our framework in this way, we circumvent the extreme difficulty of coming up with quantum models/encodings for a whole array of different NP-complete problems—finding one potentially gives us immediate quantum solutions for all. In particular, in building our initial prototype system, we picked Satisfiability, or SAT as the core problem, and fully implemented a streamlined Grover's search [6] algorithm-based quantum solution toolkit.

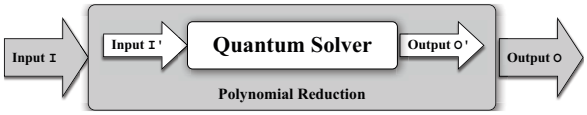

Figure 1: Quantum NP-Complete Solution Pipeline

2 FRAMEWORK OVERVIEW

We now discuss how to construct a SAT solver for universal quantum computers using Grover's search algorithm.

A SAT problem is a boolean feasibility test on logic conjunctive normal forms (CNFs), or *AND*s of *OR*s. Our plan is to basically use Grover's algorithm to search for the satisfactory variable assignment, for which a brute force search on classical computers takes $\Omega(2^n)$ lookups for an input problem with n variables, whereas our Grover solver should find an assignment under $O(2^{\frac{n}{2}})$. Since Grover's algorithm consists of the two steps, marking and amplification, we will need to implement them for SAT on a universal quantum computer.

The marking operation $U_f |\mathbf{x}, y\rangle = |\mathbf{x}, f(\mathbf{x}) \oplus y\rangle$ is completely determined by the boolean oracle function f, which takes as input a single quantum state, and spits out whether or not the state is a

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3194959

search target. This fits nicely with our intended logic satisfiability setting—actually, we can just make f the CNF expression itself, and each of the 2^n possible states naturally corresponds to a particular assignment to the n boolean variables.

Regarding how to implement the involved logical operators on a quantum computer, a *NOT* $\neg$ operator would just flip between the $|0\rangle$ and $|1\rangle$ states, or more generally, the $\alpha|0\rangle+\beta|1\rangle$ and $\beta|0\rangle+\alpha|1\rangle$ states to keep in mind quantum superposition. This is exactly what the quantum Pauli-X gate does. For the *OR* $\vee$ operator, since De Morgan's law tells us that $v_1 \vee v_2 \iff \neg(\neg v_1 \wedge \neg v_2)$, we can simply transform all $\vee$ operations into $\wedge$ operations with the help of $\neg$, which we already figured out how to do quantumly.

In essence, then, we need a quantum gate capable of carrying out the logic *AND* $\wedge$ operation. A 3-qubit quantum Toffoli (or CCX) gate [7] can be used to carry out an *AND* for 2 qubits, if we make the first two qubits $|q_0\rangle$ and $|q_1\rangle$ hold the problem variables, and introduce an ancillary $|q_2\rangle = |0\rangle$ as the last qubit. In order to handle general CNF expressions, which potentially has a large number of clauses that need to be *AND*ed together, we extend *CCX* to achieve *CNX* which would then be able to handle the *AND*ing of an arbitrary number of variables, by piecing together multiple *CCX* gates and introducing additional ancillary qubits, where each single *CCX* brings a new variable into the collective *AND*, and the ancillas help hold intermediate states.

The amplitude amplification operation for Grover's algorithm is equivalent to the matrix $M_n = I_{2^n\times 2^n} - 2A_{2^n\times 2^n}$, which, on a quantum computer, can be implemented as, and mathematically proven to be $M_n = H_n X_n Z_n X_n H_n$, where we have the corresponding n-qubit Hadamard [8] transformation $H_n = \otimes_{i=1}^{n} H$, Pauli-$X$ transformation $X_n = \otimes_{i=1}^{n} X$, and $Z_n = [(\otimes_{i=1}^{n-1} I) \otimes H]CNX_n[(\otimes_{i=1}^{n-1} I) \otimes H]$.

3 EVALUATION

We fully implemented an prototype of our Quantum NPC Solver in python. In this section we demonstrate its usage and capabilities with a couple of example runs.

Listing 1 shows the input of a small 3-SAT problem instance. Line 1 indicates that this is a CNF problem, containing 2 variables and 3 clauses. Lines 2 through 4, where the 0's denote line terminations, specify the 3 clauses: $(v_1 \vee v_2)$, $(v_1 \vee \neg v_2)$, and $(\neg v_1 \vee v_2)$, the *AND* of which is the complete 3-SAT formulation. It can be easily worked out that the unique satisfying solution is $v_1 = v_2 = True$.

```
1 p cnf 2 3
2 1 2 0
3 1 -2 0
4 -1 2 0
```

Listing 1: A Small 3-SAT Problem Instance

With this input, the automatically generated quantum circuit is depicted in Fig. 2. Please note that, we only carried out a single round of marking and amplification transformations, for the purpose of better demonstrating the probability nature of quantum computing. Fig. 3 shows the execution results from IBM Quantum Platform. As can be clearly seen, the most prominent measurement outcome state is $|11\rangle$, which corresponds to our expected solution $v_1 = v_2 = True$ to our input 3-SAT problem.

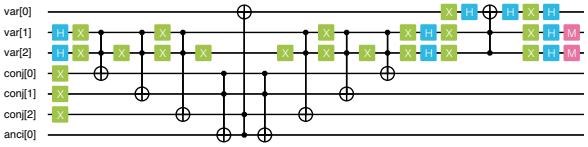

Figure 2: Generated Quantum Circuit for the 3-SAT

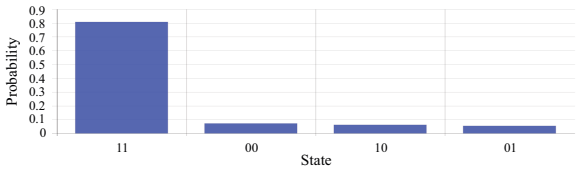

Figure 3: Execution Measurements Results for the 3-SAT

Next, we show an example run on a small 3-Coloring problem input, as shown in Listing 2, where Line 1 specifies that this problem formulation file lists the edges for the graph containing 3 nodes and 3 edges. Line 2 through 4 list the node pairs for each of the 3 edges.

```
1 p edge 3 3
2 e 1 2
3 e 1 3
4 e 2 3
```

Listing 2: An Example 3-Coloring Problem Instance

The auto-generated quantum circuit is depicted in Fig. 4. A total of 50 qubits (9 variables and 41 ancillas) are involved in this quantum circuit, where the current backend IBM quantum processor we use is only equipped with 16 qubits, and is thus not able to execute our generated quantum code, yet.

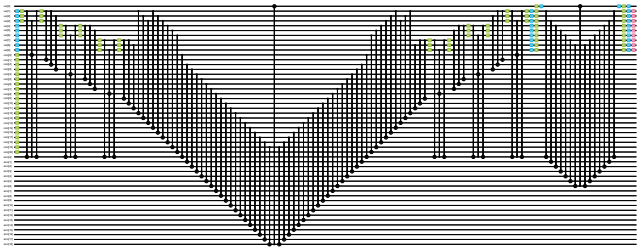

Figure 4: Generated Quantum Circuit for the 3-Coloring Problem

REFERENCES

[1] Suhas Kumar. 2015. Fundamental Limits to Moore's Law. *arXiv preprint arXiv:1511.05956* (2015).
[2] IBM. 2018. IBM Q. https://www.research.ibm.com/ibm-q/. (2018). Accessed: 2018-01-20.
[3] Davide Castelvecchi. 2017. Quantum computers ready to leap out of the lab in 2017. *Nature* 541 (2017), 9–10.
[4] Stephen A Cook. 1971. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*. ACM, 151–158.
[5] Richard M Karp. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer, 85–103.
[6] Lov K Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. ACM, 212–219.
[7] Tommaso Toffoli. 1980. Reversible computing. *Automata, Languages and Programming* (1980), 632–644.
[8] Noson S Yanofsky and Mirco A Mannucci. 2008. *Quantum computing for computer scientists*. Vol. 20. Cambridge University Press Cambridge.