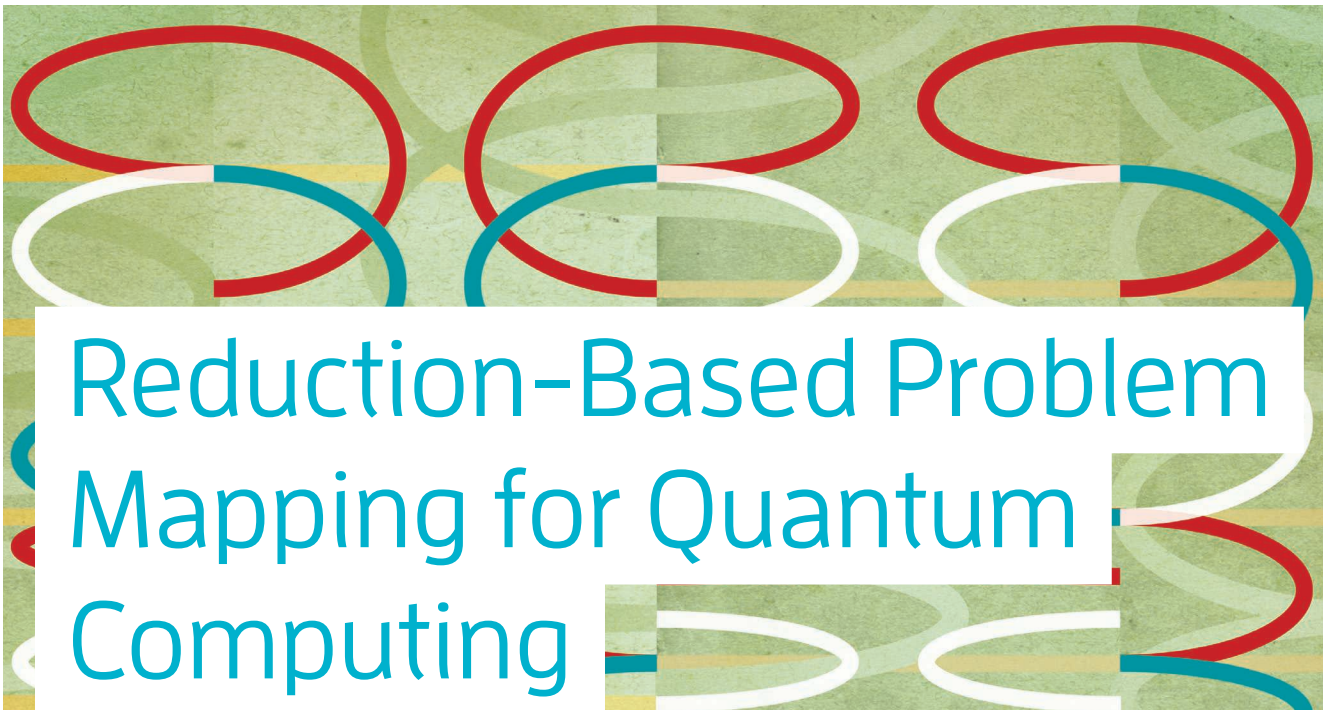

Reduction-Based Problem Mapping for Quantum Computing

Shaohan Hu, Peng Liu, Chun-Fu (Richard) Chen, Marco Pistoia, and Jay Gambetta, IBM Thomas J. Watson Research Center

In recent years, industry and academia have made tremendous research attempts to implement quantum computing technologies. But quantum computing is still grounded by numerous critical barriers, leading to its low accessibility and practicality. To overcome this problem, we propose an end-to-end framework for mapping computationally hard problems on a quantum computer via reduction.

Having dictated classical computing technology advances for decades, today, Moore's law has been slowed to a crawl by the limitations of physics. In recent years, a significant amount of time and resources from the industrial and academic communities has been devoted to making quantum computing technologies a reality.[13] D-Wave systems are claimed to be the first commercially available quantum computer since 2011.[6] They are, however, specifically designed for annealing computations[14] as opposed to being general circuit-model quantum computers,[9] which would provide general quantum computing capabilities. Multiple current efforts are devoted to building the first commercially available general circuit-model quantum computers,[3,13] with IBM making its latest systems publicly available.[13]

Although there is a wave of quantum computers on the horizon, quantum computing is not experiencing the same growth. Because quantum computing models behave in ways that are so different from their classical computing counterparts, much of our understanding of how computation systems work and how best to design algorithms and software programs is rendered unusable.[20] Thus, with a half-century of research efforts on theoretical quantum computing to rely on, there has been a relatively small set of quantum algorithms ever discovered. And from a software point of view, to be able to comprehend a simple

Digital Object Identifier 10.1109/MC.2019.2909709
Date of publication: 4 June 2019

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.

Looking at this disparity, we argue that it is imperative to bridge this accessibility and practicality gap surrounding quantum computing so that software engineering researchers and practitioners are better prepared to reap the benefits when quantum computers become increasingly more available and powerful. Toward this goal, we propose an end-to-end framework for mapping computationally hard problems on a general circuit-model quantum computer via reduction. This focus comes from the fact that easier problems are sufficiently handled by classical computers. It is, therefore, more meaningful that the potential speed-up brought about by quantum computers be used for investigating the harder problems.

The key feature of our framework is its exploitation of reduction.[15] With a core quantum solver designed to solve a single problem, many other problems may potentially benefit from its quantum speed-up by using a reduction wrapper around the core quantum solver. With this design, we circumvent the extreme difficulty of coming up with quantum models/encodings for a whole array of different problems; finding one potentially gives us immediate quantum accelerators for all. In particular, during the building of our initial prototype system, we picked unrestricted Boolean satisfiability (SAT) as the core problem and fully implemented our software toolkit based on quantum search.[12]

Our contribution in this article is threefold.

1. We propose an end-to-end framework for bringing the potential power of circuit-model quantum computers to general software researchers and practitioners.
2. We use reduction to circumvent the difficulty—if not impossibility—of having to model and encode each different problem on circuit-model quantum computers.
3. We provide a fully implemented prototype software toolkit, with its effectiveness demonstrated.

REDUCTION-BASED QUANTUM PIPELINE

In this section, we assume the basic knowledge of quantum computing as a common ground. We first talk about the (im)possibility of finding direct quantum solutions to any input problems and then discuss our proposed reduction-based framework.

THE KEY FEATURE OF OUR FRAMEWORK IS ITS EXPLOITATION OF REDUCTION.

Direct quantum solutions?

As mentioned previously, our focus has been on computationally hard problems because the speed-up naturally brought about by the intrinsic properties of quantum computing makes them an intriguing candidate for the point of attack. Note that because computational complexity theory is a much more familiar topic to computer scientists in general, we will not include a dedicated overview for it.

After surveying quantum computing literature as well as communicating with quantum physicists, we feel that the biggest barrier preventing the software engineering and/or general computer science communities from having already deployed vast quantum computing systems, or implementing comprehensive software tool sets for solving classically hard problems, is that programming quantum computers is hard. This belief is evidenced by the following assertions:

1. The direct programming of general-purpose quantum computers requires a fair amount of background knowledge on quantum computing, which is not something that can be realistically expected from most software engineering professionals or computer scientists today.
2. It is not yet clear how best to encode in quantum computing data structures that are most commonly used in classical computing, such as a general graph, a doubly linked list, and so on. This greatly adds to the difficulties related to directly programming a quantum computer to solve problems modeled after the practical scenarios to which we are accustomed.

Putting aside the general computer science community, in the past half-century

(even from the much more focused quantum computing research community), only a limited set of quantum algorithms have been discovered. Shor said that one important possible reason why so few quantum algorithms have been discovered was that quantum computers simply behave too differently from classical machines, so much so that "our techniques for designing algorithms and our intuitions for understanding the process of computation no longer work,"[20] which also echoes the reasons we have presented thus far.

How to achieve a reduction-based quantum pipeline

Our goal is to make available, or at least more accessible, circuit-model quantum computers to the more general computer science professionals and researchers. Facing the realities of and responses from the quantum computing research community, we, as computer science researchers, think that our goal can be reached not from trying to fill the abyss surrounding quantum computing by attempting to quickly discover a new rich set of quantum algorithms but, rather, from bridging the gaps to the other side. This enables people with limited quantum computing backgrounds to tap into the power of quantum computers.

How do we actually achieve a general quantum computing pipeline that doesn't still come with the quantum barrier? From Cook and Karp's seminal work on computational complexity,[4,15] we know that NP-complete problems can be reduced to each other in polynomial time. Therefore, if we are able to program the quantum computer to find a solution to one particular problem, we then have a way of applying quantum speed-ups to all problems. As shown in Figure 1, at the core sits the solver, which implements the quantum algorithm that directly solves some particular problem **S**. Then, any general problem's input I is transformed via polynomial-time reduction to I', which the core quantum solution finder **S** can directly operate on. Afterward, the quantum algorithm's output O' can be converted back, depending on how the initial reduction was carried out, to the desired output O for the original input.

Given this blueprint, the two questions that would be asked next are as follows:

1. Which particular problem should be picked for the core quantum solver?
2. How can this particular problem be solved on a quantum computer?

Before answering these questions, we would like to refer back to our current goal of bridging the gap, rather than attempting to be the most efficient solution possible. Therefore, our current design choices (discussed in this section) focus more on achieving our desired end-to-end pipeline with general applicability and accessibility. With our general framework design in place, the specific choices of the particular problem for the core quantum solver and the actual quantum algorithm for the chosen problem can both be improved upon, as our (and other) continued exploration efforts uncover more efficient or optimal candidates.

Back to the two questions mentioned previously, regarding the specific problem, we chose an unrestricted Boolean SAT[4] for our current design. It is a well-studied problem for classical computing, and its formulation closely relates to Boolean logic. For finding solutions to SAT problems using circuit-model quantum computers, we propose using quantum search in our end-to-end pipeline, as illustrated in Figure 1. Any input problem is transformed via reduction to a SAT instance, for which we use quantum search to find a solution that can then be transformed back for the original input according to the reduction.

QUANTUM SEARCH FOR SAT

In this section, we first review how quantum search works and then detail how it is used to program a circuit-model quantum computer to find solutions to SAT problems.

Quantum Grover's search algorithm

Here, we present an overview of quantum search, also known as *Grover's search algorithm*. Readers who are familiar with this concept can skip this review and go directly to the "Quantum Search Circuit for SAT" section.

Before getting into the details, let's first establish what the corresponding search problem is. The setup for this search problem is quite straightforward: we have an unordered collection of

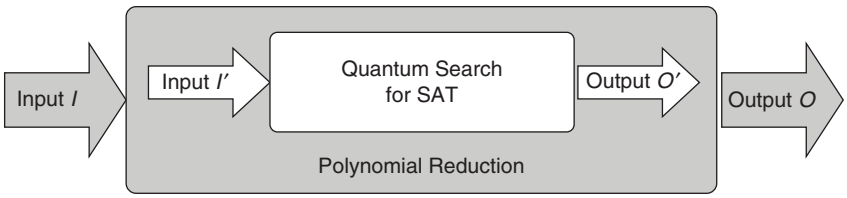

FIGURE 1. A quantum computing pipeline with a SAT-oriented quantum search core.

N items and are given some binary oracle function $f(\cdot)$ that tells whether an item meets the search criterion or not. Specifically, if we randomly pick some item i from the collection, $f(i) == 1$ means that we have a hit, whereas if $f(i) == 0$, then we need to keep searching. Obviously, in classical computing, an average of $N/2$ [or $\Omega(N)$] queries to the oracle function would be needed, as a single f lookup can check only on a single item. On a quantum computer, however, a single f query can have the effect of checking on multiple items at once because qubits can be put in superpositioned states. For example, if we have $n = \log_2 N$ qubits put into a uniform superposition, then a single application of f on them is like checking on all of the N items at once.

Rather than discussing mathematical formulations, we use simple diagrams to illustrate how quantum search works. Similar to many other quantum algorithms, quantum search starts by first putting all qubits into uniform superpositions using the quantum Hadamard gates; this process grants the same amplitude for all possible states, including the target, as illustrated in Figure 2(a), where all of the bars represent the amplitudes of every item in the collection, with the dark one indicating the search target. Obviously, if measurements are taken at this stage, any item has an equal probability—computed as the square of the amplitude—of being the outcome.

The oracle is then evaluated on the uniform superposition to mark the target by flipping its amplitude, while leaving all nontargets untouched. This is visualized in Figure 2(b), which shows the target's amplitude picking up a negative sign (flipped). If measurements are taken at this stage, it is, however, still the case that all of the items have equal probabilities of being the outcome because the negative sign of the target's amplitude will be squared away when computing its probability. But we observe that the target's flipped amplitude slightly brings down the mean of all amplitudes, as indicated by the dashed line in Figure 2(b).

The next step of quantum search is the application of what is known as the *diffusion operation*, which takes the mirror reflections of all amplitudes about their mean. As shown in Figure 2(c), the dashed bar outlines indicate the original amplitudes after the oracle marking from Figure 2(b), and the solid bars represent the resulting amplitudes after the diffusion's reflection-about-mean operation. Because the nontargets, which are the majority, were closer to the mean than the target was, their amplitudes were decreased slightly after the reflection. Conversely, the target's amplitude was much farther away from the mean, so after reflection, it increased by a greater amount. Therefore, the net effect of the reflection is that the target's amplitude was amplified while the nontargets' amplitude was shrunk.

This reflection process can also be illustrated by a toy numerical example. For instance, suppose we have five equal numbers with a single "oddball" that has a negative sign, i.e., 1, 1, 1, 1, and −1.

If we take the inverses (or mirror reflections) of all five numbers about their mean, i.e., 3/5, they become 1/5, 1/5, 1/5, 1/5, and 11/5 where $1/5 = (3/5) \times 2 - 1$ and $11/5 = (3/5) \times 2 - 1$. Because the mean is closer to the majority, inversion about the mean increases the magnitude of the "oddball" by a big margin, while simultaneously shrinking the magnitudes of others (the majority), exactly what Grover's diffusion operation does, i.e., to amplify the amplitude of the marked target state.

Coming back to our quantum algorithm discussion, in terms of actually realizing the diffusion operation, the corresponding unitary matrix M_n is

$$M_n = I_{2^n \times 2^n} - 2A_{2^n \times 2^n},$$

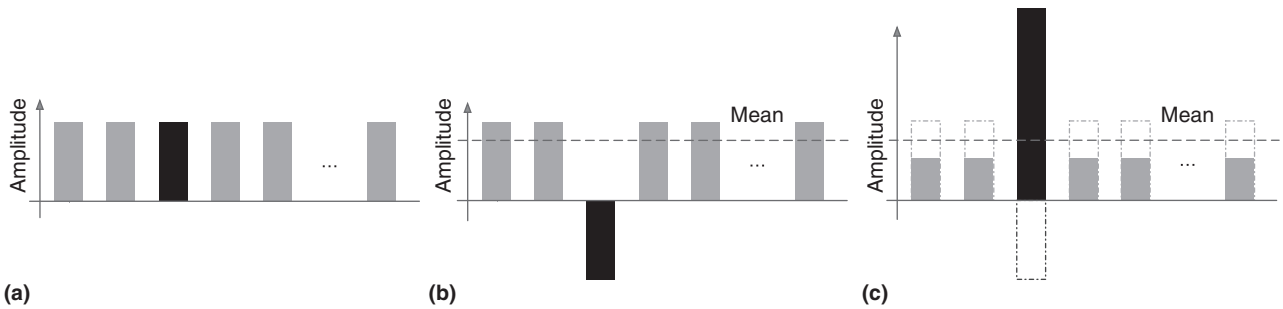

FIGURE 2. A simple illustration of how a quantum search works. (a) During initialization. (b) After oracle marking. (c) After diffusion operation.

where $A_{2^n \times 2^n}$ is a $2^n \times 2^n$ matrix filled with $1/2^n$, or more concisely,

$$A_{2^n \times 2^n} = \frac{1}{2^n} \mathbf{1}_{2^n \times 2^n},$$

if we use $\mathbf{1}_{2^n \times 2^n}$ to denote the $2^n \times 2^n$ matrix filled with all ones. Therefore, we have

$$M_n = \begin{bmatrix} 1-\frac{2}{2^n} & -\frac{2}{2^n} & \cdots & -\frac{2}{2^n} \\ -\frac{2}{2^n} & 1-\frac{2}{2^n} & \cdots & -\frac{2}{2^n} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{2}{2^n} & -\frac{2}{2^n} & \cdots & 1-\frac{2}{2^n} \end{bmatrix}.$$

With the diffusion matrix M_n, we can work out that an N-dimensional ($N = 2^n$) state vector $v_{(N,\, i)}$ with target index i, after a single iteration of $M_n \cdot v_{(N,\, i)}$, would have an amplitude ratio between its nontarget and target (i.e., the ith) elements being $(-N+4)/(-3N+4)$. We can also see that nontarget elements' amplitudes vanish when $N=4$, and as N grows, the ith element's amplitude asymptotically approaches three times that of nontarget elements' amplitudes. It is also worth pointing out that flipping the sign of M_n does not affect the amplification effect, i.e., the matrix $-M_n$ also suffices.

With the oracle marking and diffusion operations, quantum search is then carried out by repeated applications of these two steps. But how many iterations should be carried out? It is shown that $\mathcal{O}(\sqrt{N})$ (or $\mathcal{O}(2^{(n/2)})$ will suffice.[12] Compared to its classical counterpart, this brings a quadratic speed-up and is also proven to be optimal. So with proper polynomial-time reduction operations, our proposed quantum pipeline can still maintain its computational advantage.

In practice, the number of targets is usually, if not always, unknown beforehand. Therefore, the exact number of iterations cannot be determined in advance. In this case, the search can be carried out in an incremental fashion, starting with a single iteration and increasing the number of iterations during each successive round. For any particular round, the search result is verified against the classical version of the oracle (which can be done in polynomial time) to determine whether a true target has been found. If it has, then we return the found target and terminate the search; otherwise, the next round with an increased number of iterations is carried out. If the precomputed maximum number of iterations is reached with no targets found, we can terminate the search and claim that no targets exist.

Quantum search circuit for SAT

With an understanding of how quantum search works, we can now discuss how to construct a SAT solver for quantum computers.

A SAT problem is a Boolean feasibility test on a logic expression. Our plan is to use quantum search to find satisfying variable assignments, for which a brute-force search on classical computers takes $\Omega(2^n)$ lookups for an input problem with n variables, whereas quantum search should find an assignment under $\mathcal{O}(2^{(n/2)})$. Since quantum search consists of two steps, marking and diffusion, for the remainder of this section, we will discuss how to implement 1) the marking operation and 2) the diffusion operation.

Oracle-marking implementation

Recall from the "Quantum Grover's Search Algorithm" section that the marking operation 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 search target. This fits nicely with our intended logic SAT setting—actually, we can just make f the logic expression itself, and each of the 2^n possible states naturally corresponds to a particular assignment to the n Boolean variables.

With this convenient direct mapping between the SAT problems and oracle functions, we then need to work out the details of how to realize Boolean logic expressions. This means that we must be able to represent the Boolean logic operations *NOT* $\neg$, *OR* $\vee$, and *AND* $\wedge$ on a circuit-model 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 account for quantum superpositions; this is exactly what the quantum Pauli-*X* gate does. Therefore, we have the $\neg$ operator covered. For the *OR* $\vee$ operator, because De Morgan's law tells us that $v_1 \vee v_2 \Leftrightarrow \neg(\neg v_1 \wedge \neg v_2)$, we can simply transform all $\vee$ operations into $\wedge$ operations with the help of $\neg$, which we previously figured out how to do quantumly.

In essence, we need a quantum gate capable of carrying out the logic *AND* $\wedge$ operation. Inspired by the 3-qubit Toffoli gate (commonly denoted as *CCX*), which flips the state of the last qubit ($|0\rangle \leftrightarrow |1\rangle$) if the first two input control qubits are both $|1\rangle$, if we make the first two qubits $|q_0\rangle$ and $|q_1\rangle$ hold the problem variables and introduce an ancillary (helper) $|q_2\rangle = |0\rangle$ as the last qubit, then after the Toffoli operation, $|q_2\rangle$ will be in the state representing the *AND* of $|q_0\rangle$ and $|q_1\rangle$. However, this is only one part of the picture because only two variables can be *AND*ed together in this way, and in general, we must handle the case where

an arbitrarily large number of variables are *AND*ed together. For this we use the multiple-control Toffoli (MCT) gate, which is an extension to the traditional 3-qubit Toffoli and is capable of handling an arbitrary number of control inputs.

An n-qubit MCT gate's equivalent matrix is just a $2^n \times 2^n$ identity matrix with the 2×2 block at the bottom right corner rotated in-plane by $\pi / 2$ as

$$MCT_n = \begin{bmatrix} I_{(2^n-2)\times(2^n-2)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 0 & 1 \\ \mathbf{0} & 1 & 0 \end{bmatrix}.$$

In terms of the actual implementation of MCT, the textbook strategy[18] is to chain together multiple Toffoli gates in a V shape, where the left arm gradually computes and accumulates the intermediate result, and the right arm uncomputes to clean up all the ancillary qubits. Other strategies also exist, each with their own strengths and limitations. The choice should be made by taking into consideration the number of available ancillary qubits, the desired quantum circuit depths, the underlying hardware qubit connectivity, and so on. With all the tools in our quantum arsenal, we have successfully constructed our desired oracle-marking operation.

Grover diffusion implementation

Recall from the "Quantum Grover's Search Algorithm" section that the diffusion operation is equivalent to the matrix $M_n = I_{2^n \times 2^n} - 2A_{2^n \times 2^n}$. Therefore, we must establish how to realize this unitary operation on a quantum computer.

With a slight abuse of notation, we use the symbols for the single-qubit Hadamard H and Pauli X and Z gates paired with subscripts n to indicate their n-qubit analogs.

- H_n: The n-qubit Hadamard transformation, which is simply an n-fold Kronecker product of the single-qubit Hadamard transformation. in terms of matrices,

$$H_n = \otimes_{i=1}^{n} H,$$

which is of the size $2^n \times 2^n$. Also, H_n has the recurrence property of

$$H_{k+1} = \frac{1}{\sqrt{2}} \begin{bmatrix} H_k & H_k \\ H_k & -H_k \end{bmatrix}.$$

- X_n: Similar to H_n, this denotes the n-fold Kronecker product of the single-qubit Pauli-X transformation

$$X_n = \otimes_{i=1}^{n} X.$$

Therefore, for any number n of qubits, X_n is simply a $2^n \times 2^n$ identity matrix rotated in-plane by $\pi / 2$ (i.e., the antidiagonal is filled with 1 and 0 everywhere else).

- Z_n: A $2^n \times 2^n$ identity matrix, with the bottom-right element flipped from 1 to –1, i.e.,

$$Z_n = \begin{bmatrix} I_{(2^n-2)\times(2^n-2)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & 0 \\ \mathbf{0} & 0 & -1 \end{bmatrix}.$$

Unlike the previous two equations, Z_n cannot be obtained by taking the n-fold Kronecker product of the single-qubit Pauli-Z transformation. However, compared to MCT_n toward the end of the "Oracle-Marking Implementation" section, we see that Z_n differs from it only in the bottom-right 2×2 block, which for MCT_n is an X and Z_n is a Z. Because we have the relation

$$HXH = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = Z,$$

linking together X and Z through H, we can therefore construct an n-qubit Z_n by sandwiching an n-qubit MCT_n transformation between two single-qubit Hadamard transformations on the last qubit. More precisely,

$$Z_n = [(\otimes_{i=1}^{n-1} I) \otimes H] \times MCT_n [(\otimes_{i=1}^{n-1} I) \otimes H],$$

where I is simply the 2×2 identity matrix.

With such a setup, using induction and simple matrix algebra, the general n-qubit Grover diffusion operator $M_n = I_{2^n \times 2^n} - 2A_{2^n \times 2^n}$ can be constructed as $M_n = H_n X_n Z_n X_n H_n$.

With both the oracle marking and the diffusion operator fully specified and implemented, we can construct a quantum search circuit for finding satisfying solutions to SAT problems on general circuit-model quantum computers, which, paired with classical reduction, gives us the general quantum computing pipeline.

IMPLEMENTATION

To validate our proposed pipeline, we implemented an end-to-end prototype of our proposed quantum solution pipeline using Qiskit[1] with the following modules:

- *A SAT quantum circuit generator*: This is the core component that takes as input an unrestricted SAT problem formulation, whose Boolean logic expression can be in any arbitrary format, and automatically generates its corresponding

quantum circuit by following the construction described in the "Quantum Search for SAT" section.

- *A problem parser*: This is an extensible wrapper around the core SAT quantum circuit generator; it defines an abstract interface for reducing other problems to SAT. Currently, for proof of concept, we provide the implementations for 3-coloring problems as well as pseudo-Boolean constraints problems. The capabilities of reducing other problems to SAT can be easily added by subclassing the abstract interface.
- *A back-end quantum processor*: We use the quantum back ends provided by Qiskit, including IBM's publicly available general circuit-model quantum computer[13] as well as multiple different simulators.

EVALUATION

For evaluation purposes, in this section we show several example runs to demonstrate the usage of our software toolkit to map various problems. For each example, we first show the input problem formulation, followed by the quantum code automatically generated by our toolkit, and finally the quantum execution results if the current quantum processor back end is powerful enough to run the code.

Let us first look at an example run with a toy SAT input string, "(w ^ x) & ~ (y ^ z) & (x & y & z)," involving Boolean variables w, x, y, and z, and the *AND*, *XOR*, and *NOT* operations. It is obvious that the satisfying assignment is (w, x, y, z) = (False, True, True, True).

With this toy SAT problem, the generated quantum code in OpenQASM format[5] is shown in "Listing 1: Generated SAT OpenQASM Quantum Code." Lines 3–7 declare the needed quantum

LISTING 1: GENERATED SAT OpenQASM QUANTUM CODE

```
1.  OPENQASM 2.0;
2.  include "qelib1.inc";
3.  qreg v[4];
4.  qreg o[1];
5.  qreg c[4];
6.  qreg a[2];
7.  creg m[4];
8.  h v[0];
9.  h v[1];
10. h v[2];
11. h v[3];
12. x o[0];
13. h o[0];
14. x c[0];
15. x v[1];
16. cx v[1],c[0];
17. x v[1];
18. x c[1];
19. x v[2];
20. cx v[2],c[1];
21. x v[2];
22. x c[2];
23. x v[3];
24. cx v[3],c[2];
25. x v[3];
26. x c[3];
27. ccx v[0],v[1],c[3];
28. ccx c[0],c[1],a[0];
29. ccx c[2],a[0],a[1];
30. ccx c[3],a[1],o[0];
31. ccx c[2],a[0],a[1];
32. ccx c[0],c[1],a[0];
33. x c[0];
34. x v[1];
35. cx v[1],c[0];
36. x v[1];
37. x c[1];
38. x v[2];
39. cx v[2],c[1];
40. x v[2];
41. x c[2];
42. x v[3];
43. cx v[3],c[2];
44. x v[3];
45. x c[3];
46. ccx v[0],v[1],c[3];
47. h v[3];
48. h v[2];
49. h v[1];
50. h v[0];
51. x v[0];
52. x v[1];
53. x v[2];
54. x v[3];
55. h v[3];
56. ccx v[0],v[1],a[0];
57. ccx v[2],a[0],v[3];
58. ccx v[0],v[1],a[0];
59. h v[3];
60. x v[0];
61. x v[1];
62. x v[2];
63. x v[3];
64. h v[0];
65. h v[1];
66. h v[2];
67. h v[3];
68. measure v[0] -> m[0];
69. measure v[1] -> m[1];
70. measure v[2] -> m[2];
71. measure v[3] -> m[3];
```

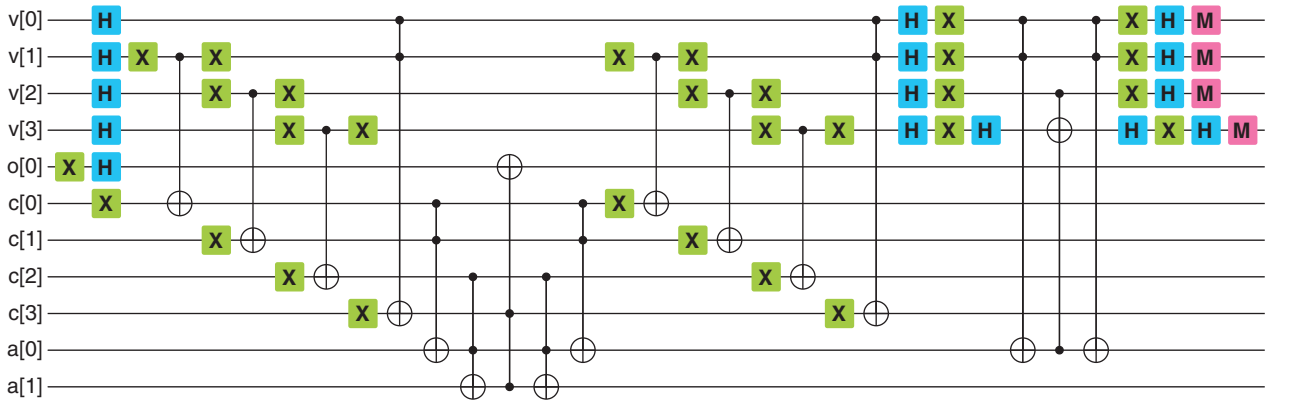

FIGURE 3. The generated quantum circuit for a small SAT. *H*, *X*, and *M* represent the Hadamard gate, Pauli-**X** gate, and measurements, respectively.

and classical registers. In particular: o holds the qubit for oracle output, v holds the qubits corresponding to the actual problem variables, the qubits in c and a are ancillae for intermediate results, and m holds classical bits for the final measurement of v.

Lines 8–11 put the variable qubits in uniform superposition. Lines 12–46 implement the oracle marking. Lines 47–67 carry out the diffusion operation. Finally, the variable qubits are measured as expressed in lines 68–71. The equivalent quantum circuit is depicted in Figure 3. *H*, *X*, and *M* represent the Hadamard gate, Pauli-*X* gate, and measurements, respectively.

Please note that, as opposed to the $\mathcal{O}(\sqrt{N})$ repetitions mentioned in the "Quantum Search for SAT" section, we carried out only a single iteration of marking and diffusion. We take 1,024 shots on the simulator and collect the measurement statistics, as shown in Figure 4. The prominent measurement outcome, when reading from right to left, gives the assignments for the variables w, x, y, and z. And it agrees with our expected satisfying solution, i.e., (False, True, True, True).

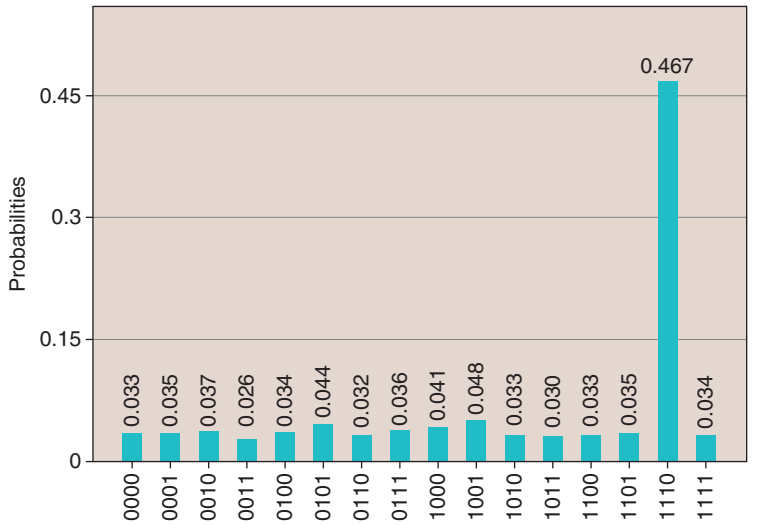

FIGURE 4. The execution measurement results for the Toy SAT.

LISTING 2: AN EXAMPLE 3-COLORING PROBLEM INSTANCE

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

Next, to demonstrate problem mapping via reduction, we show an example run on a small 3-coloring problem input. 3-coloring is the decision problem of whether or not a graph's nodes can be colored using only three colors without any edge connecting two nodes of the same color. The input formulation of the 3-coloring problem instance is shown in "Listing 2: An Example 3-Coloring Problem Instance." Line 1 specifies that this problem formulation file lists the edges for a graph containing three nodes and three edges. Lines 2–4 list the node pairs for each of the three edges.

Due to space limitations, we omit listing the generated OpenQASM code and show only the generated quantum circuit, as shown in Figure 5.

Note that the shown quantum circuit directly solves the SAT reduction of the supplied 3-coloring problem. The reduction under the hood is straightforward: for each edge e connecting node pair (d_1 and d_2), we introduce the following variables: R_{d_1}, G_{d_1}, and B_{d_1} and R_{d_2}, G_{d_2}, and B_{d_2}, representing the binary status of whether each of the nodes, i.e., d_1 and d_2, is of the colors red, green, or blue. Then, the 3-coloring problem can be easily encoded as a SAT. For example, "the two neighbor nodes (d_1 and d_2) cannot both be red" can be encoded as $\neg R_{d_1} \vee \neg R_{d_2}$; and "node d_2 must be of some color" is represented as $R_{d_2} \vee G_{d_2} \vee B_{d_2}$. The SAT reduction is the collective conjunction of all the disjunctive clauses from all of these constraints on all of the edges and all of the nodes.

RELATED WORK

On the classical computing side, we are motivated by seminal works from Cook[4] and Karp.[15] We are also actively exploring additional transformation and translation routes to make our framework more inclusive for different types of problems, e.g., converting integer programs or other constrained problems to SAT problems.[16]

On the other hand, the detailed design and construction of our proposed core quantum solver have been inspired by the various pioneering work from the quantum computing research community, especially the quantum Grover's search algorithm[12], which we were able to dissect and tailor to fit our particular setting.

Aside from the various well-known quantum algorithms such as quantum search, other, more recent, approximation quantum computing models have also

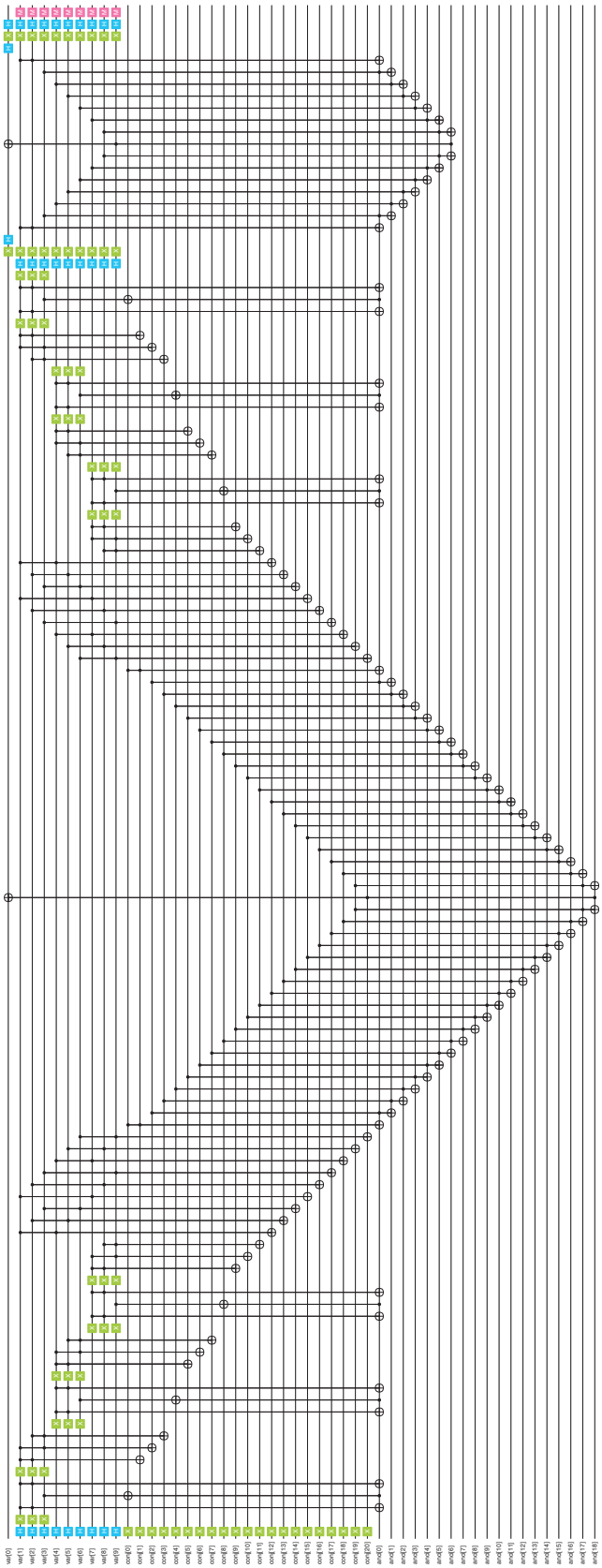

FIGURE 5. A generated quantum circuit for the 3-coloring problem.

been explored, including adiabatic computing[10] and quantum annealing.[8] Quantum solutions to various constraint satisfaction problems[17] have also been studied. Conversion-based approaches have been examined in the context of a quantum annealer, with various efforts toward converting different problems to D-Wave's QUBO formats, including binary constraint satisfaction problems[2], Prolog programs[19], and NP-hard graph problems.[7]

Using quantum search to solve NP-complete problems has also been studied.[11] Here, the focus is on the theoretical side only, targeting problems with known bounds on recursive decomposition solutions. In our work, we target a design to enable quantum accessibility through reductions, providing an immediate realization on general circuit-model quantum computers with a fully implemented toolkit. Our proposed framework builds on top of this work and focuses more on providing an accessible end-to-end framework.

In this article, we tackle the low accessibility and practicality problem of quantum computing by proposing an end-to-end framework for mapping computationally hard problems for general circuit-model quantum computers. Our framework takes advantage of reductions, and therefore, does not require quantum-specific modeling and encoding techniques for different problems. We implement a complete prototype system with unrestricted Boolean SAT as the core problem and use quantum Grover's search to find satisfying solutions. The effectiveness of our system is demonstrated through workflows on different-sized/-typed problems, including actual execution results. With our novel framework, software engineering researchers as well as practitioners are able to tap into the power of the general circuit-model quantum computers without having prior knowledge of quantum computing.

OUR FRAMEWORK TAKES ADVANTAGE OF REDUCTIONS, AND THEREFORE, DOES NOT REQUIRE QUANTUM-SPECIFIC MODELING AND ENCODING TECHNIQUES FOR DIFFERENT PROBLEMS.

REFERENCES

1. G. Aleksandrowicz et al., "Qiskit: An open-source framework for quantum computing," Accessed on: Mar. 16, 2019. [Online]. Available: http://dx.doi.org/10.5281/zenodo.2562110
2. Z. Bian, F. Chudak, W. Macready, A. Roy, R. Sebastiani, and S. Varotti, "Solving SAT and MaxSAT with a quantum annealer: Foundations and a preliminary report," in *Proc. Int. Symp. Frontiers of Combining Systems*, 2017, pp. 153–171.
3. T. Commissariat, "Google gains new ground on universal quantum computer," *Physics World*, June 10, 2016. [Online]. Available: https://physicsworld.com/a/google-gains-new-ground-on-universal-quantum-computer/
4. S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. 3rd Annu. ACM Symp. Theory of Computing*, 1971, pp. 151–158.
5. A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, "Open quantum assembly language." 2017. [Online]. Available: https://arxiv.org/abs/1707.03429
6. D-Wave Systems Inc. Accessed on: Mar. 21, 2019. [Online]. Available: https://www.dwavesys.com/
7. GitHub, Inc., "Dwavesystems/dwave_networkx." Accessed on: Mar. 16, 2019. [Online]. Available: https://github.com/dwavesystems/dwave_networkx
8. A. Das and B. K. Chakrabarti, *Quantum Annealing and Related Optimization Methods*. New York: Springer-Verlag, 2005.
9. D Deutsch. "Quantum theory, the church-turing principle and the universal quantum computer," *Proc. R. Soc. Lond. A, Math. Phys. Eng. Sci.*, vol. 400, pp. 97–117. June 1985.
10. E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem," *Science*, vol. 292, no. 5516, pp. 472–475, 2001.
11. M. Fürer, "Solving NP-complete problems with quantum search," in *LATIN 2008: Theoretical Informatics*, (Lecture Notes in Computer Science, vol. 4957), E. S. Laber, C. Bornstein, L. T. Nogueira, and L. Faria, Eds. Berlin, Germany: Springer, 2008, pp. 784–792.
12. L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory of Computing*, 1996, pp. 212–219.

ABOUT THE AUTHORS

SHAOHAN HU is a research staff member at the IBM Thomas J. Watson Research Center. His research interests include quantum computing, cyberphysical systems, mobile ubiquitous computing, crowd and social sensing, big data analytics, and cloud computing. Hu received a Ph.D. in computer science from the University of Illinois at Urbana-Champaign. He is a Member of the IEEE. Contact him at shaohan.hu@ibm.com.

PENG LIU is a research staff member at the IBM Thomas J. Watson Research Center. He has published extensively in several areas of computer science, including compiler theory, artificial intelligence, static and dynamic program analysis, security, and quantum computing. Liu received a Ph.D. in computer science from the Hong Kong University of Science and Technology. Contact him at liup@us.ibm.com.

CHUN-FU (RICHARD) CHEN is a senior software engineer at the IBM Thomas J. Watson Research Center. His research interests include quantum computing for chemistry, machine learning, and optimization, as well as computer vision and graph computing. Chen received an M.S. in electrical engineering from National Cheng Kung University. He is a Member of the IEEE and ACM. Contact him at chenrich@us.ibm.com.

MARCO PISTOIA is a distinguished research staff member and senior manager at the IBM Thomas J. Watson Research Center. His research interests include linear algebra, invariant theory, and quantum computing. Pistoia received a Ph.D. in mathematics from New York University. For his publications, he received four Association for Computing Machinery distinguished paper awards and one IEEE honorable mention. Contact him at pistoia@us.ibm.com.

JAY GAMBETTA is the IBM Global lead of quantum theory, applications, and software at the IBM Thomas J. Watson Research Center. His research interests include quantum information science. Gambetta received a Ph.D. (Hons.) in physics from Griffith University. He is a Senior Member of the IEEE and fellow of IBM and the American Physical Society. Contact him at jay.gambetta@us.ibm.com.

13. IBM, "IBM Q." Accessed on: Mar. 21, 2019. [Online]. Available: https://www.research.ibm.com/ibm-q/
14. M. W. Johnson, et al., "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.
15. R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, (The IBM Research Symposia Series), R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, Eds. Boston: Springer, 1972, pp. 85–103.
16. R. Li, D. Zhou, and D. Du, "Satisfiability and integer programming as complementary tools," in *Proc. 2004 Asia and South Pacific Design Automation Conf.*, 2004, pp. 879–882.
17. S. Mandrà, G. G. Guerreschi, and A. Aspuru-Guzik, "Faster than classical quantum algorithm for dense formulas of exact satisfiability and occupation problems," *New J. Phys.*, vol. 18, no. 7, pp. 073003, 2016.
18. M. A. Nielsen and I. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge, U.K.: Cambridge Univ. Press, 2010. doi: 10.1017/CBO9780511976667.
19. S. Pakin, "Performing fully parallel constraint logic programming on a quantum annealer," *Theory Practice Logic Programming*, vol. 18, no. 5–6, pp. 928–949, 2018.
20. P. W. Shor, "Why haven't more quantum algorithms been found?" *J. ACM*, vol. 50, no. 1, pp. 87–90, 2003.

IEEE COMPUTER SOCIETY
DIGITAL LIBRARY
Access all your IEEE Computer Society subscriptions at
computer.org /mysubscriptions