Quantum logic synthesis for Satisfiability Problems

Shuai Yang,1,2 Wei Zi,1,2 Bujiao Wu,1,2 Cheng Guo,1,2 Jialin Zhang,1,2 and Xiaoming Sun1,2

1Institute of Computing Technology, Chinese Academy of Sciences
2University of Chinese Academy of Sciences

To demonstrate the advantage of quantum computation, many attempts have been made to attack classically intractable problems, such as the satisfiability problem (SAT), with quantum computer. To use quantum algorithms to solve these NP-hard problems, a quantum oracle with quantum circuit implementation is usually required. In this manuscript, we first introduce a novel algorithm to synthesize quantum logic in the Conjunctive Normal Form (CNF) model. Compared with linear ancillary qubits in the implementation of Qiskit open-source framework, our algorithm can synthesize an $m$ clauses $n$ variables $k$-CNF with $O(k^2m^2/n)$ quantum gates by only using three ancillary qubits. Both the size and depth of the circuit can be further compressed with the increase in the number of ancillary qubits. When the number of ancillary qubits is $\Omega(m^\epsilon)$ (for any $\epsilon > 0$), the size of the quantum circuit given by the algorithm is $O(km)$, which is asymptotically optimal. Furthermore, we design another algorithm to optimize the depth of the quantum circuit with only a small increase in the size of the quantum circuit. Experiments show that our algorithms have great improvement in size and depth compared with the previous algorithms.

INTRODUCTION

Quantum circuit synthesis, which constructs a quantum circuit for a specific unitary matrix, has been a research hotspot in recent years. The reasons for this attention are the limits in the current quantum devices. One is the fact that only 50 to 100 qubits can be used in current quantum devices, i.e. 53-qubit Google Quantum chip Sycamore [2] and 65-qubit IBM Quantum Hummingbird processor [22]. Another is only shallow circuits can be implemented on existing quantum devices due to decoherence [2, 4, 10, 20, 22]. There is plenty of work aiming to optimize the size of general quantum circuits [11, 17, 18]. Ancillary qubits are some extra qubits being used to achieve some specific goals in quantum computation, and are restored after executing the quantum circuits. We call an ancillary qubit clean, if the initial state of the ancillary qubit is $|0\rangle$ or $|1\rangle$, otherwise call it dirty. Without loss of generality, assume the initial state of a clean ancillary qubit is $|0\rangle$. Ancillary qubits are widely used in synthesis algorithm to reduce the size or depth of quantum circuit [6, 9, 13]. So, it is necessary to consider how to synthesize the quantum circuit with a small size/depth and limited ancillary qubits.

A Boolean formula over variables $x_1, x_2, \cdot \cdot \cdot, x_n$ consists of the variables and the logical operates $\text{AND} (\wedge), \text{OR} (\lor), \text{NOT}(\neg)$. If the Boolean formula is an AND of OR’s or their negations, we say that this Boolean formula is in conjunctive normal form (CNF). A clause in CNF is a disjunction of variables or negation of variables. If each clause contains at most $k$ variables, the formula is in $k$-CNF and the width of this CNF is $k$. In this manuscript, an $n$ variables $m$ clauses $k$-CNF is denoted by $\text{CNF}_n^k$. The satisfiability (SAT) problem is to determine whether a CNF formula is satisfiable. SAT problem is a basic NP-hard problem in computer science and is widely used in other fields. To solve the SAT problem with quantum computer, we need to construct a quantum circuit for quantum logic [1, 7]. Meanwhile, quantum logic is an indispensable part of quantum algorithm. Many researchers have designed algorithms to synthesize quantum logic in a different model [8, 12, 16, 19]. However, it is inefficient to use these methods to construct oracle in quantum algorithms for SAT problem, since it takes exponential time to convert a CNF formula to other forms. In previous algorithms for CNF formula, at least $m$ ancillary qubits are required to synthesize $\text{CNF}_n^k$. Each ancillary qubit is used to store the result of one clause. This idea is widely used in the implementation of CNF quantum circuits [14, 15, 21] and the Qiskit open-source framework. However, the number of ancillary qubits is limited in Noisy Intermediate-Scale Quantum (NISQ) devices and the number of clauses is usually huge. There is still no efficient algorithm for the CNF formula to synthesize the corresponding quantum circuit with arbitrary ancillary qubits.

This manuscript describes an algorithm to synthesize quantum logic in CNF formula. Due to the shortage of quantum resources, we first design an algorithm to synthesize $\text{CNF}_n^k$ with $\ell$ ancillary qubits and a few quantum gates, for all $\ell \geq 3$. With the growth of $\ell$, the size of the circuit decrease. When $\ell = \Omega(m^\epsilon)$, here $\epsilon$ is an arbitrary number greater than 0, the size of the circuit will be $O(km)$, which is asymptotically optimal. In order to further optimize the depth of the circuit, we just adjust our algorithm in the inner-most recursion. The depth of the circuit is greatly reduced and only a small increase in the size. Experiments are designed for all these algorithms to show their performances. For random 7-CNF, the depth of the quantum circuit generated by our algorithm is about $1\%$ of the other algorithm in CNF formula.

The rest of the manuscript progresses as follows: we first show these two main algorithms and analyze the complexity of these algorithms in section algorithm. The experimental result will show in section performance and finally, the manuscript will be concluded in section conclusion.
ALGORITHM

In this section, we will first introduce two algorithms for quantum logic synthesis in CNF. And then we will give the lower bound of size and depth of the quantum circuit for CNF\(^k\)\(_{n,m}\) to show our algorithm is near to asymptotically optimal.

Firstly, we use arbitrary \(\ell (\geq 3)\) ancillary qubits to synthesize CNF formula. Similar to the circuit that synthesizes the \(k\)-Toffoli gate, we use the dirty ancillary qubits to store the information temporarily and then merge all the clauses into the target qubit. The circuit merges \(\ell /2 + 1\) clauses by leveraging \(\ell\) dirty ancillary qubits. Using Toffoli gate, we can easily synthesize one clause in CNF and the cost is \(O(k)\) \([21]\), which means Size\(_0\)(CNF\(^k\)\(_{n,1}\)) = \(O(k)\).

**Lemma 1.** Let \(g_1, g_2, \cdots, g_{\ell/(\ell + 1)/2}\) be boolean functions, then Size\(_x\)(\(g_1 \land g_2 \land \cdots \land g_{\ell/(\ell + 1)/2}\)) = \(O(\ell + \sum_1 \text{Size}_x(g_i))\).

**Proof.** In FIG. 1, we use a Toffoli-formula-Toffoli structure, which consists of two Toffoli gate and a formula gate \(g_i\), which merges \(|g_i(x)| \to |x| \land |g_i(x)|\). This structure can merge the Boolean formula and information in another control qubit to the target qubit. We use the structure similar to Toffoli gate to eliminate the influence of ancillary qubits. Then the circuit in FIG 1 can merge \((\ell + 1)/2\) formulae to the target qubit.

Some input qubits are free when we merge formulae, and we can use them to merge more clauses as dirty ancillary qubits temporarily. Then we can merge \(O(n/k + 2)\) clauses without ancillary qubits.

**Lemma 2.** Size\(_0\)(CNF\(^k\)\(_{n,n/(k+2)}\)) = \(O(n)\).

**Proof.** There are no more than \(kn/(k + 2)\) variables used in CNF\(^k\)\(_{n,n/(k+2)}\), so we can use other input qubits as dirty ancillary qubits. We use half of the ancillary qubits to store clauses and the rest to merge all of the clauses together.

With the Lemma 1 and Lemma 2, we now can synthesize CNF\(^k\)\(_{n,m}\) recursively.

**Theorem 1.** Size\(_x\)(CNF\(^k\)\(_{n,m}\)) = \(O(n(\frac{km}{n})^{1+\log(\ell+1)/2})\).

**Proof.** We recursively apply the circuit in Lemma 1 to synthesize CNF\(^k\)\(_{n,m}\), and regard input qubits as temporary ancillary qubits to merge \(n/(k + 2)\) clauses by using Lemma 2 in the inner-most recursion of the circuit in Lemma 1. Then we have Size\(_x\)(CNF\(^k\)\(_{n,m}\)) = 2Size\(_x\)(CNF\(^k\)\(_{n,2m/\ell}\)) + 4\(\ell\) and Size\(_0\)(CNF\(^k\)\(_{n,n/(k+2)}\)) = \(O(n)\), which can be further reduced to:

\[
\text{Size}_x(\text{CNF}^k_{n,m}) = O\left( n \left( \frac{km}{n} \right)^{1+\log(\ell+1)/2} \right).
\]

By leveraging Alg. 1, we can make full use of ancillary qubits to synthesize CNF. The former methods need \(O(m)\) ancillary qubits and \(O(km)\) gates, now we only need \(O(m^\epsilon)\) ancillary qubits, for all \(\epsilon > 0\) to synthesize with linear number of quantum gates. Furthermore, our method can be used to synthesize more complex formulae, such as the AND-OR tree.

**Corollary 1.** Size\(_m\)(CNF\(^k\)\(_{n,m}\)) = \(O(km), \epsilon > 0\).

In NISQ devices, a circuit with a long running time will suffer from decoherence, which can decline the accuracy of the algorithm results. So it is necessary to reduce the depth of the quantum circuit.

Alg. 1 just synthesize each clause one by one, a natural question is that can we use a little more quantum gate to synthesize some clauses parallel? The answer is yes. We just adjust the inner-most recursion in Alg. 1. First we divide the ancillary qubits into three parts, (1) \(q_{mem}\), which store the information of inputs; (2) \(q_{dirty}\), which is used for the recursion in Alg. 1; and (3) \(q_{clean}\), which merges \(|q_{dirty}|\) clauses together in parallel. When \(d = 1\), we first copy input to \(q_{mem}\). Then we can synthesize \(q_{mem}/k\) clauses in one layer, and next we use \(q_{clean}\) to merge all of these clauses in \(O(\log |q_{dirty}|)\) depth. Finally, we reset all the ancillary, the steps of the algorithm are shown in Alg. 2. The detail is shown in the supplementary materials. The outer recursion is running on \(q_{dirty}\).

**Theorem 2.** Let \(S = \max\{k/\log \ell, 1\}\). Using Alg. 1 and Alg. 2, Depth\(_{x}\)(CNF\(^k\)\(_{n,m}\)) = \(O(k\ell \log \ell (\frac{m}{\ell})^{1+\log(m/\ell)/2})\) and Size\(_x\)(CNF\(^k\)\(_{n,m}\)) = \(O(k(m)^{1+\log(m/\ell)/2})\).

**Proof.** By leveraging the circuit shown in FIG. 1 recursively, then Depth\(_x\)(CNF\(^k\)\(_{n,m}\)) = \(2\ell\)Depth\(_x\)(CNF\(^k\)\(_{n,2m/\ell}\)) + 4\(\ell\), here \(\ell' = \ell/(S+1)\). Alg. 2 is used to synthesize in parallel in
There are 3. There exists a ε < L where CNF a specific set of CNF defined as follows, 

Lemma 3. Let \[ g(x) = g_1(x) \land g_2(x) \land \cdots \land g_{t/2+1}(x). \] For convenience, we use a single-qubit gate on target qubit to represent the circuit that synthesizes a Boolean formula on target qubit, \( g_i \mid g_j \rangle \rightarrow |g_j \oplus g_i(x)\rangle. \)

**Algorithm 2: Depth-considered synthesis algorithm**

1. Let \( S = \max \{ k/ \log \ell, 1 \} \), the ancillary qubits is divided into three parts, size of each part is \((S - 1)\ell/(S + 1), \ell/(S + 1), \ell/(S + 1)\), represented by \( q_{\text{mem}}, q_{\text{dirty}}, q_{\text{clean}} \).
2. Copy the input to \( q_{\text{mem}} \).
3. Use the input and \( q_{\text{mem}} \) to synthesize \( \ell/(S + 1) \) clauses in parallel in \( q_{\text{dirty}} \).
4. Use the \( q_{\text{clean}} \) to merge all the clauses in parallel.
5. Reset the ancillary qubits.

The inner-most recursion, which leads to \( \text{Depth}_{\ell}(\text{CNF}^k_{n, \ell'}) = O(\log n) \).

\[
\text{Depth}_{\ell}(\text{CNF}^k_{n, m}) = O\left(k \log \ell \left(\frac{mS}{\ell}\right)^{1+\log_{\ell/S^4}}\right). \tag{1}
\]

The analysis of size is similar to that of depth.

Some small optimizations, which do not change the complexity of our algorithm, will be shown in the supplementary materials. In the rest part of this section, we will show that our algorithm is asymptotically optimal for large \( \ell \). We prove that there exists a \( k \)-CNF with \( m \) clauses which needs \( \Omega(km) \) size of quantum circuits to approximate it with any error \( \varepsilon < \frac{\sqrt{\ell}}{2} \), as depicted in Theorem 3. To give this lower bound, we first need to use the following lemma to give a lower bound for the different instances for \( \text{CNF}^k_{n, m} \).

**Lemma 3.** There are \( \Omega\left(\binom{n}{m}\right) \) different instances for \( \text{CNF}^k_{n, m} \).

We prove this lemma by counting the different instances of a specific set of CNF defined as follows, \( \mathcal{A} := \{ \phi \mid \phi = l_1 \land \cdots \land l_m, l_i \neq l_j, \text{ for } i \neq j \in [m] \} \), where \( \mathcal{L} := \{ v_1 \lor \cdots \lor v_k \mid v_i \in \{ \overline{x_1}, \cdots, \overline{x_n} \}, v_i \neq v_j \text{ for } i, j \in [k] \} \) be the set of clauses with \( k \)-variables. We prove that the size of \( \mathcal{A} \) is \( \binom{n}{m} \), and all of instances in \( \mathcal{A} \) are different in Supplementary materials. Hence there are \( \Omega\left(\binom{n}{m}\right) \) different instances for \( k \)-CNF with \( m \)-clause.

**Theorem 3.** There exists a \( \text{CNF}^k_{n, m} \), any quantum circuits approximating it with error \( \varepsilon < \frac{\sqrt{\ell}}{2} \) needs size \( \Omega(km) \).

Let \( U \in \mathbb{C}^{4 \times 4} \) be a two-qubit gate, and the \( \delta \)-discretization of the \((j, k)\)-th element \( U_{jk} \) be \( U^\delta_{jk} = \delta \lfloor a/\delta \rfloor + i \delta \lfloor b/\delta \rfloor \), where \( U_{jk} = a + ib \). Then we have \( \| U - U^\delta \|_2 < 2\delta \). There are at most \( \left(\frac{\ell}{\delta}\right)^{32} \) different \( \delta \)-discretizations \( U^\delta \) for the infinite continuous \( U \) in the space by its definition. In the supplementary material, we prove that any two different instances in \( \mathcal{A} \) do not share any common \( \delta = \frac{\ell}{2\ell} \)-discretization, where \( s \) is the total number of gates for any \( \text{CNF}^k_{n, m} \). Here we suppose the quantum circuit only contains 2-qubit gates, since all of single-qubit gates can be expanded to 2-qubit gates by appending some identities. By the fact that the number of different instances is upper bounded by the number of \( \delta \)-discretization of quantum circuits, we can use counting method to give a lower bound of the circuit size,

\[
\binom{n}{k} \leq \left(\frac{\ell}{\delta}\right)^{32} s. \tag{2}
\]

It follows that the lower bound holds in Theorem 3. The proof detail is depicted in supplementary materials.

We give two algorithms to synthesize quantum logic in this section, the former one can synthesize CNF with less ancillary qubits and quantum gates, the number of CNOT gate used in our algorithm is \( O\left( n \left( \frac{km}{\ell} \right)^{1+\log_{\ell/S^4}} \right) \). Then we try to parallel the inner-most recursion of Alg. 1, and the depth of the circuit is reduced to \( O\left( k \log \ell \left( \frac{mS}{\ell}\right)^{1+\log_{\ell/S^4}} \right) \). Furthermore, we can see that when \( \ell = \Omega(m^4) \), our algorithm is asymptotically optimal.

**Algorithm Performance**

In this section, we will design several experiments to verify the effectiveness of our algorithms. As represented on IBMQ [5], the error rate of a single qubit gate, such as H, X, Y, and Z gate, is much less than the error rate of a two-qubit gate. CNOT gate plays a key role in quantum circuits. Moreover, the duration of the CNOT gate is about ten times longer than
the single-qubit gate [10]. Hence, in this section we mainly consider the cost of CNOT gates in a quantum circuit.

We test our Alg. 1 and Alg. 2 with different input size, different number of ancillary qubits and different width of CNF. The input size is ranging from 200 to 1000. The number of ancillary qubits $\ell$ is $n, 2n, 3n, 4n, m$, corresponding to 5 points on each line in the figure. When the number of ancillary qubits is $m$, the circuit size generated by Alg. 1 is as well as the circuit size generated by the Ohya algorithm.

![Graph](image)

FIG. 2: The size of the circuit obtained by Alg. 1 for random 4-CNF with variable numbers of 200, 400, 600, 800, and 1000. Here, size means the number of CNOT gates, and the number of clauses $m = 9.931n$. The number of ancillary qubits $\ell$ is $n, 2n, 3n, 4n, m$, corresponding to 5 points on each line in the figure. When the number of ancillary qubits is $m$, the circuit size generated by Alg. 1 is as well as the circuit size generated by the Ohya algorithm.

We also have tested our Alg. 2 with different parameters. All the experiment data can also be found in the supplementary materials. Here we show the relationship between the number of ancillary qubits and the depth of the circuit generated by Alg. 2 first. The depth of the circuit reduces rapidly when the number of ancillary qubits increases since we make full use of each ancillary qubit. And then we compare our algorithm with the previous algorithms. Because the previous algorithm only works when the number of ancillaries is $m$ and $2m$, we just compare the performance when we use the same number of ancillary qubits. In FIG 3, we list a total of 4 methods to implement random CNF, two of which use Alg. 2 to optimize depth with $m$ and $2m$ ancillary qubits respectively. The third and forth type is using Ohya algorithm with $m$ and $2m$ ancillary qubits, respectively.

We can find the depth cost of our algorithm is related to the ratio of $m/n$. Our algorithm just uses no more than 1% layers circuit of Ohya algorithm when the ancillary is $m$ and shows significant advantages compared to the previous algorithm.

In this section, we design several experiments to show our performance of two algorithms. Both these two algorithms make full use of each ancillary qubit, we can find our algorithm can work for any number($\geq 3$) of ancillary qubits. Both size and depth decrease with the increase of ancillary qubits. Alg. 2 has significant advantages to the previous algorithm. Our algorithm generates the circuit with 1% layers compared to the previous algorithm. More data we will show in the sup-

![Graph](image)

FIG. 3: Fig (a) shows that when the number of ancillary qubits $\ell$ increases, the depth of the circuit for random 3-CNF generated by Alg. 2 decreases, here $n$ is the number of variables. Fig (b) is the result for randomly 7-CNF. $n$ is the number of variables, and $\ell$ is the number of ancillary qubits. The two lines of “Ohya” correspond to the depth of quantum circuit that implemented by Ohya algorithm, and the other two lines correspond to the depth of quantum circuit implemented by Alg. 2.

| $k$ | 3 | 4 | 5 | 6 | 7 |
|-----|---|---|---|---|---|
| $\alpha_k$ | 4.267 | 9.931 | 21.117 | 43.37 | 87.79 |

TABLE I: $\alpha_k$ used in our experiments.
CONCLUSION

In this manuscript, we present two algorithms to synthesize the Boolean formula represented by CNF. We first use the technology of synthesizing Toffoli gates with dirty ancillary qubits to merge different clauses together with any \( \ell(\geq 3) \) qubits. Later, we adjust the inner-most recursion of Alg. 1, which leads to much shallower circuits with only a little extra cost of the size of the circuit. Additionally, we prove the lower bound of the size and depth cost of the circuit for an \( n \) variables \( m \) clauses \( k\)-CNF instance with \( \ell \) ancillary qubits and our algorithm is asymptotically optimal when \( \ell = \Omega(m^2) \), \( \epsilon > 0 \). The performance of both Alg. 1 and Alg. 2 are verified by experiments. Our algorithm greatly reduces the depth of quantum circuits and the demand for ancillary qubits. When the number of ancillary qubits is \( m \), Alg. 2 use less than 1\% layers of the circuit constructed by previous algorithms. This work can directly be used to construct quantum oracles in the quantum algorithm.

In the future, we will further optimize our algorithm and try to reduce the quantum cost of an \( n \) variables \( m \) clauses \( k\)-CNF instance to \( O(km) \), which consist with the lower bound, with constant ancillary qubits.

[1] Andris Ambainis. Quantum search algorithms. *ACM SIGACT News*, 35(2):22–35, 2004.
[2] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.
[3] Adrian Balint, Anton Belov, Matti Järvisalo, and Carsten Sinz. Sat challenge 2012 random sat track: Description of benchmark generation. *Proceedings of SAT Challenge*, pages 72–73, 2012.
[4] Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J Bremmer, John M Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14(6):595, 2018.
[5] Sergey Bravyi, Sarah Sheldon, Abhinav Kandala, David C Mckay, and Jay M Gambetta. Mitigating measurement errors in multi-qubit experiments. *arXiv preprint arXiv:2006.14044*, 2020.
[6] Richard Cleve and John Watrous. Fast parallel circuits for the quantum fourier transform. *arXiv preprint quant-ph/0006004*, 2000.
[7] Vedran Dunjko, Yimin Ge, and Ignacio Cirac. Computational speedups using small quantum devices. *Physical Review Letters*, 121(25):250501, 2018.
[8] Kenneth Fazel, Mitchell A Thornton, and JE Rice. Esop-based toffoli gate cascade generation. In 2007 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pages 206–209. IEEE, 2007.
[9] Jiaqing Jiang, Xiaoming Sun, Shang-Hua Teng, Bujiao Wu, Kewen Wu, and Jialin Zhang. Optimal space-depth trade-off of cnot circuits in quantum logic synthesis. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 213–229. SIAM, 2020.
[10] Petar Jurcevic, Ali Javadi-Abhari, Lev S Bishop, Isaac Lauer, Daniela F Bogorin, Markus Brink, Lauren Capelluto, Oktay Günlük, Toshinaro Itoko, Naoki Kanazawa, et al. Demonstration of quantum volume 64 on a superconducting quantum computing system. *arXiv preprint arXiv:2008.08571*, 2020.
[11] Emanuel Knill. Approximation by quantum circuits. *arXiv preprint quant-ph/9508006*, 1995.
[12] D Michael Miller, Dmitri Maslov, and Gerhard W Dueck. A transformation based algorithm for reversible logic synthesis. In *Proceedings of the 40th annual Design Automation Conference*, pages 318–323, 2003.
[13] Cristopher Moore and Martin Nilsson. Parallel quantum computation and quantum codes. *SIAM Journal on Computing*, 31(3):799–815, 2001.
[14] Masanori Ohya. Quantum algorithm for sat problem and quantum mutual entropy. In *Selected Papers Of M Ohya*, pages 51–67. World Scientific, 2008.
[15] Masanori Ohya and Natsuki Masuda. Np problem in quantum algorithm. *Open Systems & Information Dynamics*, 7(1):33–39, 2000.
[16] Vivek V Shende, Aditya K Prasad, Igor L Markov, and John P Hayes. Synthesis of reversible logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(6):710–722, 2003.
[17] Vivek V Shende, Stephen S Bullock, and Igor L Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, 2006.
[18] Juha J Vartiainen, Mikko Mottonen, and Martti M Salomaa. Efficient decomposition of quantum gates. *Physical review letters*, 92(17):177902, 2004.
[19] Robert Wille and Rolf Drechsler. Bdd-based synthesis of reversible logic for large functions. In *Proceedings of the 46th Annual Design Automation Conference*, pages 270–275, 2009.
[20] Yangsen Ye, Zi-Yong Ge, Yulin Wu, Shiyu Wang, Ming Gong, Yu-Ran Zhang, Qingling Zhu, Rui Yang, Shaowei Li, Futian Liang, et al. Propagation and localization of collective excitations on a 24-qubit superconducting processor. *Physical Review Letters*, 123(5):050502, 2019.
[21] Ahmed Younes and Jonathan E Rowe. A polynomial time bounded-error quantum algorithm for boolean satisfiability. *arXiv preprint arXiv:1507.05061*, 2015.
[22] Guanyu Zhu and Andrew Cross. Hardware-aware approach for fault-tolerant quantum computation. Website, 2020. https://www.ibm.com/blogs/research/2020/09/hardware-aware-quantum-computation.html.
Supplementary materials for “Quantum logic synthesis for Satisfiability Problems”

Shuai Yang, Wei Zi, Bujiao Wu, Cheng Guo, Jialin Zhang, and Xiaoming Sun
Institute of Computing Technology, Chinese Academy of Sciences and University of Chinese Academy of Sciences

Detail of algorithm 2

In this section, we will introduce the detail of algorithm 2 to reduce the depth of the circuit. An \( n \) variables \( m \) clauses \( k\)-CNF is denoted by \( \text{CNF}^k_{n,m} \). Different from algorithm 1, we need to use the lemma shown as follows.

\[
\text{Lemma 1.} \quad \text{Depth}_t(\text{CNF}^k_{n,t/(S+2)}) = O(k \log \ell), \text{ here } S = \max\{k/\log \ell, 1\}.
\]

**Proof.** We first copy the input into \( q_{\text{mem}} \). Since we will synthesize \( \ell/(S+2) \) clauses in parallel, we copy corresponding variables in \( q_{\text{mem}} \). The size of \( q_{\text{mem}} \) is \( S\ell/(S+2) \) and the cost of this step is at most \( \log(k\ell/\log \ell) = O(\log(k\ell)) \). Then we synthesize each clause in parallel in the \( q_{\text{dirty}} \). The size of \( q_{\text{mem}} \) is \( S\ell/(S+2) \), so we can synthesize \( \ell/(S+2) \log \ell \) clauses in \( k \) layers. The cost of step 3 is \( \frac{\ell/(S+2) \log \ell}{k\ell/(S+2)} = O(k \log \ell) \). Using the circuit in FIG. 1 we can merge 2 clauses to the qubits in the \( q_{\text{clean}} \) in step 3. Later, we merge all these clauses together in the \( q_{\text{clean}} \), the detail is in FIG. 4. The cost of this step is \( \log(\ell/(S+2)) = O(\log \ell) \). Finally, we reset all the memory by performing the inverse process of algorithm 2. The total CNOT cost is \( O(\log(k\ell)) + O(k \log \ell) + O(\log \ell) = O(k \log \ell) \).

The Toffoli used in FIG. 1 is not the general Toffoli gated but approximate Toffoli, which is denoted by the triangle. By using this we can reduce the quantum cost to implement a \( m \)-control Toffoli with \( m \) clean ancillary qubits.

Some small optimizations

In this section, we will show some optimization to reduce the quantum cost. In algorithm 1, if the ancillary qubits are clean at the beginning, the outer-most recursive circuit is shown in FIG. 4. When the number of clauses is close to the number of ancillary, this optimization will reduce the quantum cost by about half. Fur-
in the algorithm, to about 25%. The circuit is similar to the circuit shown in FIG. 3.

When the number of ancillary qubits is small, we can use the circuit shown in FIG. 5 to replace the circuit in FIG. 1 of article. All these optimizations will not change the complexity of algorithm 1 but can reduce the quantum cost in constant. By choosing these optimizations reasonably can help us synthesize CNF with less quantum cost.

The lower bound for the number of k-CNF with m-clause

Lemma 2. There are \( \Omega\left(\binom{n}{m}\right) \) different instances for CNF\(_{n,m}\).

Proof. For two functions \( \phi_1(x), \phi_2(x) \), we say \( \phi_1 \equiv \phi_2 \) if \( \phi_1(x_1, \ldots, x_n) = \phi_2(x_1, \ldots, x_n) \) for any legal input \((x_1, \ldots, x_n)\). Let

\[
\mathcal{L} := \{v_1 \lor \cdots \lor v_k | v_i \in \{\overline{x_1}, \ldots, \overline{x_n}\}, v_i \neq v_j \text{ for } i, j \in [k]\}
\]

be the set of all clauses with k-variables. Let k-CNF set

\[
\mathcal{A} := \{\phi | \phi = l_1 \land \cdots \land l_m, l_i \in \mathcal{L}, l_i \neq l_j, \text{ for } i \neq j \in [m]\}.
\]

Observe that the size of \( \mathcal{A} \) is \( s(\mathcal{A}) = \binom{n}{m} \). Let

\[
\phi = (v_1 \lor \cdots \lor v_k) \land \cdots \land (v_{km-2} \lor \cdots \lor v_{km}).
\]

Then it can be represented as a Boolean function

\[
f_\phi(x) = (1 - \bar{v}_1 \cdots \bar{v}_k) \cdots (1 - \bar{v}_{km-2} \cdots \bar{v}_{km}), \tag{1}
\]

where \( \bar{v}_k = 0 \) iff \( v_i = T \) and \( \bar{v}_i = 1 \) iff \( v_i = F \). (Here we still use symbol \( x \) as the input in \( f \) and simply change the scope of inputs from \( \{T, F\}^n \) to \( \{0, 1\}^n \).

In the following, we prove that for any two different instances \( \phi_1, \phi_2 \in \mathcal{A} \). And we have \( \phi_1 \not\equiv \phi_2 \) by contradiction. Suppose \( \phi_1 \equiv \phi_2 \), then \( f_{\phi_1}(x) \equiv f_{\phi_2}(x) \). Let \( g_\phi(x) \) satisfies \( \text{deg}(g) = k \) be the summations of all degree-k terms of \( f_\phi(x) \). By the definition of \( f_\phi(x) \) in Equation 1,

\[
g_\phi(x) = \sum_{j=0}^{m-1} \bar{v}_{jk+1} \cdots \bar{v}_{(j+1)}.
\]

Since \( f_{\phi_1}(x) \equiv f_{\phi_2}(x) \), then \( \text{deg}(f_{\phi_1} - f_{\phi_2}) = 0 \), i.e., \( g_{\phi_1}(x) \equiv g_{\phi_2}(x) \). Let \( l_1 := \bar{v}_1 \cdots \bar{v}_k \) be one term of \( f_{\phi_1}(x) \). Let \( x_i := 1 \) when \( x_i \in \{\bar{v}_1, \ldots, \bar{v}_k\} \) and \( x_i := 0 \) otherwise. Then \( g_{\phi_2}(x) = 1 \) iff \( x_1 \cdots x_k \) is one term of it. Hence \( l_1 \) is also a term of \( g_{\phi_2}(x) \). Without loss of generality, each term \( l_j := \bar{v}_{jk+1} \cdots \bar{v}_{(j+1)} \) \( \in g_{\phi_1}, l_j \in g_{\phi_2} \) at the same time. Therefore \( \phi_1 = \phi_2 \), contrary with the fact that \( \phi_1, \phi_2 \) are two different instances in \( \mathcal{A} \).

\( \square \)

Theorem 1. There exists a CNF\(_{n,m}^k\), any quantum circuits approximating it with error \( \varepsilon \in \frac{2\delta}{2} \) needs size \( \Omega(km) \).

Proof. Let \( U \in \mathbb{C}^{4 \times 4} \) be a two qubit gate, and the \( \delta \)-discretization of the \((j,k)\)-th element \( U_{jk} \) be \( U^\delta_{jk} = \delta[a/\delta] + ib/\delta \), where \( U_{jk} = a + ib \). Then we have \( \|U - U^\delta\| < 2\delta \). There are at most \( \left(\frac{\delta}{\varepsilon}\right)^{32} \) different \( \delta \)-discretizations \( U^\delta \) for the infinite continuous \( U \) in the space by its definition.

In the following, we prove that any two different instances in \( \mathcal{A} \) do not share any common \( \delta \)-discretization. Hence, we can use the counting method to give a lower bound of the circuit size.

Let \( A_G, A_H \) be the quantum circuit representations of two different instances in \( \mathcal{A} \). Let \( s \) be the maximum size of all the unitaries related to \( A_G \) and \( A_H \). By the fact that the unitary \( U \in \mathbb{C}^{2n \times 2n} \) has a s-size quantum circuit, the following inequalities

\[
\|A_G - A_G^\delta\| < 2s\delta \leq \varepsilon,
\]

\[
\|A_H - A_H^\delta\| < 2s\delta \leq \varepsilon,
\]

hold when \( \delta = \frac{\varepsilon}{2s} \), and \( \varepsilon < \frac{\sqrt{2}}{2} \). Combined with the fact that \( \|A_G - A_H\| = \sqrt{2} \), we have \( A_G^\delta \neq A_H^\delta \).

Hence, any two different instances in \( \mathcal{A} \) have different \( \delta \)-discretization. There are \( \Omega\left(\binom{n}{m}\right) \) different instances for k-CNF with m-clause by Lemma 2. Combined with the fact that the number of different instances is upper bounded by the number of \( \delta \)-discretization of quantum circuits,

\[
\binom{n}{m} \leq \left(\frac{2}{\varepsilon \delta} \cdot n\right)^s.
\]
Since \( \binom{n}{k} = \Omega((n/k)^k) \) for any \( k \). Then
\[
\binom{n}{m} = \Omega\left(\left(\frac{n}{m}\right)^m\right) = \Omega\left(\left(\frac{n^k}{k^m}\right)^m\right).
\]

By inequality \( 2 \) we have \( s = \Omega(km) \) when \( k = o(n) \). When \( k = cn \) for constant \( c < 1 \), by Stirling’s formula, \( \binom{n}{k} = \Omega(2^{an}) \) for some constant \( a < 1 \). Hence,
\[
\binom{n}{m} = \Omega\left(\left(\frac{2^{an}}{m}\right)^m\right),
\]
combined with inequality \( 2 \) give us \( s = \Omega(km) \) when \( k = cn \) for constant \( c < 1 \). This implies the lower bound also holds for any \( k < n \) for general \( k \)-CNF.

**Detail of experiment**

In this section, we show all the data in our experiment. For random CNFs with different \( k \)-CNFs, different numbers of variables, and different numbers of ancillary qubits, we show in Table \( \text{II} \) and Table \( \text{III} \) the comparison results of the circuit parameters implemented using the size optimization algorithm, depth optimization algorithm, and the Ohya algorithm.

The implementation of random \( k \)-CNF used here is to independently generate \( m \) clauses for the input \( n \) and \( m \). Each clause first randomly selects \( k \) variables from the \( n \) variables, and then each selected variable is set randomly not or no change. For CNFs with all parameters determined, we randomly generate 10 CNFs as input to let the algorithm run and take the average of size and depth as the final result.

The table \( \text{II} \) is the performance of algorithm 1. We test our algorithm in 3, 4, 5, 6, 7-CNF with the different number of input qubits and ancillary qubits and record the CNOT gate used under different conditions. We can find that with the growth of the number of ancillary qubits, the size of the circuit generated by our algorithm decrease. When we use the same number of ancillary qubits as other algorithms, our algorithm is not worse than other algorithms.

The table \( \text{III} \) is the performance of algorithm 2. We compare our algorithm with Ohya algorithms in 3, 4, 5, 6, 7-CNフ with the different number of input qubits and ancillary qubits and then record the depth of these circuits under different conditions. The circuit generated by our algorithm 2 uses less than 1% layers circuit of previous algorithms when \( \ell = 2n \), which will help us to design more large experiment in NISQ.

| \( \ell \) | \( \text{3-CNF-CNOT} \) | \( \text{4-CNF-CNOT} \) | \( \text{5-CNF-CNOT} \) | \( \text{6-CNF-CNOT} \) | \( \text{7-CNF-CNOT} \) |
|---|---|---|---|---|---|
| 200 | 400 | 600 | 800 | 1000 | 000 |
| \( \ell = n \) | 133596 | 270840 | 406224 | 541668 | 677156 |
| \( \ell = 2n \) | 111664 | 223228 | 334984 | 446548 | 558304 |
| \( \ell = 3n \) | 76142 | 152278 | 228812 | 304702 | 380936 |
| \( \ell = 4n \) | 81342 | 162678 | 244112 | 325502 | 406936 |
| \( \ell = 5n \) | 40896 | 81840 | 122832 | 163776 | 204768 |
| \( \ell = 6n \) | 25584 | 51174 | 76794 | 102384 | 128004 |
| \( \ell = 7n \) | 25584 | 51174 | 76794 | 102384 | 128004 |

| \( \text{Ohya} : \ell = m \) | \( \ell = m \) | \( \ell = m \) | \( \ell = m \) | \( \ell = m \) | \( \ell = m \) |
|---|---|---|---|---|---|
| \( \ell = m \) | 142944 | 285936 | 428928 | 51920 | 714984 |
| \( \ell = 2m \) | 83406 | 166818 | 250230 | 333642 | 417096 |
| \( \text{Ohya} : \ell = 2m \) | 228036 | 456078 | 681474 | 912216 | 1140312 |
| \( \ell = 3m \) | 405360 | 810768 | 1216272 | 1621680 | 2027184 |
| \( \ell = 4m \) | 1040832 | 2081712 | 3122592 | 4163472 | 5207452 |
| \( \ell = 5m \) | 228036 | 456078 | 681474 | 912216 | 1140312 |



| \( \text{Ohya} : \ell = 2m \) | \( \ell = 2m \) | \( \ell = 2m \) | \( \ell = 2m \) | \( \ell = 2m \) | \( \ell = 2m \) |
|---|---|---|---|---|---|
| \( \text{Ohya} : \ell = 3m \) | 142944 | 285936 | 428928 | 51920 | 714984 |
| \( \text{Ohya} : \ell = 4m \) | 83406 | 166818 | 250230 | 333642 | 417096 |
| \( \text{Ohya} : \ell = 5m \) | 228036 | 456078 | 681474 | 912216 | 1140312 |
| \( \text{Ohya} : \ell = 6m \) | 405360 | 810768 | 1216272 | 1621680 | 2027184 |
| \( \text{Ohya} : \ell = 7m \) | 1040832 | 2081712 | 3122592 | 4163472 | 5207452 |
| \( \text{Ohya} : \ell = 8m \) | 228036 | 456078 | 681474 | 912216 | 1140312 |
| \( \text{Ohya} : \ell = 9m \) | 83406 | 166818 | 250230 | 333642 | 417096 |
| \( \text{Ohya} : \ell = 10m \) | 228036 | 456078 | 681474 | 912216 | 1140312 |

**TABLE I.** For \( k \) is 3 to 7, the number of variables is 200 to 1000, \( m/n \) is the threshold value, and the number of ancillary qubits \( \ell \) is shown in the table. The number of CNOT gates of the quantum circuit generated by our algorithm is as follows as shown in this table. The representatives with the word “Ohya” use the Ohya algorithm.
| Depth | Variables | Ohya: \( \ell = m \) | Ohya: \( \ell = 2n \) |
|-------|-----------|----------------|----------------|
| 3-CNF | \( \ell = n/2 \) | 20109 | 54191 |
|       | \( \ell = n \) | 4790 | 1871 |
|       | \( \ell = 3n/2 \) | 1430 | 1232 |
|       | \( \ell = 2n \) | 1212 | 1122 |
|       | \( \ell = m \) | 705 | 406 |
|       | \( \ell = 2m \) | 351 | 305 |
| 4-CNF | \( \ell = n/2 \) | 91925 | 64349 |
|       | \( \ell = n \) | 144954 | 12511 |
|       | \( \ell = 3n/2 \) | 1430 | 1232 |
|       | \( \ell = 2n \) | 1212 | 1122 |
|       | \( \ell = m \) | 705 | 406 |
|       | \( \ell = 2m \) | 351 | 305 |
| 5-CNF | \( \ell = n/2 \) | 4957 | 4987 |
|       | \( \ell = n \) | 28805 | 28573 |
|       | \( \ell = 3n/2 \) | 1430 | 1232 |
|       | \( \ell = 2n \) | 1212 | 1122 |
|       | \( \ell = m \) | 705 | 406 |
|       | \( \ell = 2m \) | 351 | 305 |
| 6-CNF | \( \ell = n/2 \) | 15988 | 16188 |
|       | \( \ell = n \) | 503796 | 390335 |
|       | \( \ell = 3n/2 \) | 280204 | 103492 |
|       | \( \ell = 2n \) | 23140 | 103492 |
|       | \( \ell = m \) | 3075 | 3079 |
|       | \( \ell = 2m \) | 1565 | 1568 |
| Ohya:\( \ell = m \) | 112471.2 | 213895.2 |
|       | \( \ell = 2n \) | 5661.6 | 5703.6 |
| Ohya: \( \ell = 2m \) | 5516.6 | 5736.6 |
| 7-CNF | \( \ell = n/2 \) | 9230629 | 6030103 |
|       | \( \ell = n \) | 2022000 | 2059392 |
|       | \( \ell = 3n/2 \) | 1222208 | 932602 |
|       | \( \ell = 2n \) | 1032280 | 260107 |
|       | \( \ell = m \) | 6550 | 6561 |
|       | \( \ell = 2m \) | 3054 | 3058 |
| Ohya: \( \ell = m \) | 529020 | 950448 |
| Ohya: \( \ell = 2m \) | 32392.8 | 32409.6 |

**TABLE II.** For \( k \) is 3 to 7, the number of variables is 200 to 1000, \( m/n \) is the threshold value, and the number of ancillary qubits \( \ell \) is shown in the table. The depth of the quantum circuit generated by our algorithm is as follows as shown in this table. The representatives with the word “Ohya” use the Ohya algorithm.