Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/TQE.2020.DOI

Algorithm-Aware Qubit Mapping

YANJUN JI[1] (Graduate Student Member, IEEE) AND ILIA POLIAN[1] (Senior Member, IEEE)
[1]Institute of Computer Architecture and Computer Engineering, University of Stuttgart, 70569 Stuttgart, Germany

Corresponding author: Yanjun Ji (email: yanjun.ji@informatik.uni-stuttgart.de).

This work was supported in part by the project QORA within the Competence Center Quantum Computing Baden-Württemberg (funded by the Ministerium für Wirtschaft, Arbeit und Wohnungbau Baden-Württemberg).

ABSTRACT

Algorithm-Aware (AlgAw) qubit mapping aims at directly providing the solutions to qubit mapping of algorithms with regular structures based on the algorithm's features. Although the exact method provides a high-quality solution, its compilation time grows exponentially with the circuit size. To improve its scalability, we propose the AlgAw qubit mapping. The main idea is to first determine the subcircuits in an algorithm to be mapped, then analyze the optimal solutions of small-scale subcircuits found by exact methods to obtain solutions of large-scale subcircuits, and finally reconstruct the entire circuit and assign parameters. Applying AlgAw to the Quantum Approximate Optimization Algorithm (QAOA) on linear and T-shaped subtopologies produces optimal and scalable solutions for arbitrary numbers of qubits and depths, which is critical to the algorithm's performance on Noisy Intermediate-Scale Quantum (NISQ) computers. Compared to Qiskit, Tket, and SWAP-Network, AlgAw produces the least number of CNOT gates and the lowest circuit depth. Furthermore, AlgAw takes only a few seconds to obtain a circuit with a hundred qubits that satisfies the connectivity constraints. The benchmarking results on Quantum Processing Units (QPUs) show that AlgAw qubit mapping yields higher values of approximation ratio than others. AlgAw can also be applied to other algorithms such as the Variational Quantum Eigensolver (VQE).

INDEX TERMS Algorithm-Aware, Benchmarking, NISQ, QAOA, Quantum Optimization, Qubit Mapping, Superconducting Qubits, SWAP-Network, VQE

I. INTRODUCTION

VARIATIONAL Quantum Algorithms (VQAs) [1], [2] combining classical and quantum computers are promising to solve complex problems, such as combinatorial optimization and quantum simulation of material, more efficiently than classical algorithms. However, the present Quantum Processing Units (QPUs) support only a limited number of qubits, and many of them, including superconducting architectures, provide only restricted connectivity of qubits. The algorithms need to be transpiled, e.g., by inserting SWAP gates, to satisfy the connectivity constraints.

The qubits supported by QPUs are noisy, and the error rate of a two-qubit gate, e.g., CNOT or CX gate supported on IBM QPUs, is on average one order of magnitude higher than that of a single-qubit gate. The SWAP gate introduces more noise, as it needs to be implemented through 3 CX gates. Therefore, one of the important tasks to improve the algorithm's performance on the Noisy Intermediate-Scale Quantum (NISQ) computers is to map the algorithm efficiently to the physical qubits supported on the QPUs, such that a minimum number of SWAP or CX gates are introduced. This process is known as qubit mapping [3] or routing [4].

The qubit mapping can be formulated as a mathematical optimization/constraint-satisfaction problem and then solved using a specific solver. Such a procedure is referred to as an exact method and various approaches [5]–[9] have been proposed. Different objectives exist in the qubit mapping problem, such as minimizing the number of inserted SWAP gates or the circuit depth or maximizing the circuit fidelity. The qubit mapping problem has been shown to be NP-hard [10]. While a high-quality and stable solution calculated by exact methods improves the performance of algorithms on NISQ computers (e.g., [11]), the compilation time for finding the optimal solution grows exponentially with the circuit size. Many heuristic methods [7], [8], [12]–[15] have been developed to speed up this process. However, their solutions are of lower quality than those found with exact methods. In addition, SWAP-Network [16]–[18] or SWAP strategy [19], a scalable method, was proposed to solve qubit mapping problems by inserting SWAP layers. However, these

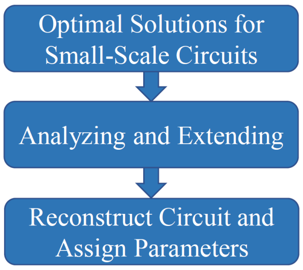
Optimal Solutions for Small-Scale Circuits
Analyzing and Extending
Reconstruct Circuit and Assign Parameters

FIGURE 1. Flowchart of Algorithm-Aware (AlgAw) qubit mapping.

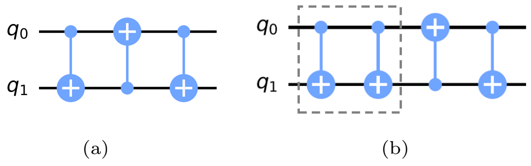

FIGURE 2. (a) SWAP gate. (b) A CX gate followed by a SWAP gate. The box represents the CX gate cancellation.

strategies do not guarantee optimality.

To achieve optimality and scalability of solutions, we present the Algorithm-Aware (AlgAw) qubit mapping for algorithms with a regular structure such as the Quantum Approximate Optimization Algorithm (QAOA) [20]–[22] with different depths. The AlgAw qubit mapping starts by investigating the features of the algorithms, which improves the scalability, e.g., in QAOA, only two-qubit gates need to be mapped instead of all gates. Then, we analyze the optimal solutions for small-scale circuits as they can be efficiently computed by exact methods and extend these solutions to large-scale circuits. Finally, we reconstruct the circuit based on the derived solutions and assign parameters to each gate at the algorithm level.

We apply AlgAw to the QAOA on dense portfolio optimization problems [23] where the qubits are required to have the maximum connectivity. An exact approach [9] is used to search solutions for qubit mapping of a subcircuit containing all two-qubit gates in a small-scale QAOA circuit on linear and T-shaped subtopologies. The solutions for larger-scale circuits are obtained by analyzing the solutions for small-scale circuits.

The remainder of this paper is organized as follows. Section II describes the methodology of AlgAw qubit mapping. Section III provides details on individual steps of applying AlgAw to QAOA. Section IV reports the benchmarking results of AlgAw in comparison with other approaches on several QPUs. Section V presents other applications of AlgAw, and section VI concludes.

II. ALGORITHM-AWARE (ALGAW) QUBIT MAPPING

The procedure for Algorithm-Aware qubit mapping is outlined in Fig. 1. In order to map an algorithm more efficiently, it is important to study its features first. Investigating the structure of the algorithm contributes to improving the scalability. For algorithms with the same structure, the solutions to the qubit mapping are potentially transferable. In addition, commuting gates, one of the key features, should be considered not only during the mapping process but also afterward. Moreover, the location of the SWAP gate can be used to improve the resilience of the algorithm. As shown in Fig. 2, in the case of a CX gate followed by a SWAP gate, two CX gates with the same direction can be canceled, thus reducing the number of CX gates. This phenomenon is known as the CX gate cancellation.

Through the analysis of the algorithm, specific subcircuits to be mapped in the algorithm can be identified. Compared to mapping the entire circuit, mapping only these subcircuits reduces the computational overhead. Exact methods are then employed to compute the optimal solutions of those circuits. For algorithms with a deterministic structure on symmetric subtopologies, these solutions can potentially be extended to large-scale circuits. Finally, the circuit satisfying the connectivity constraints is reconstructed and the parameters of the gates are assigned at the algorithm level.

No computation is required if the analyzed and deduced solutions are used. We call this approach an algorithm-aware qubit mapping because it depends individually on the features of the algorithm. Moreover, circuits satisfying the connectivity constraints are directly reconstructed at the algorithm level after the solutions are extended.

III. APPLYING ALGAW TO QAOA

We consider the QAOA on dense portfolio optimization problems, where each qubit needs to interact once with all other qubits. Fig. 3 shows the circuit of a 5-qubit QAOA (5Q-QAOA) with depth $p = 1$. For n qubits, there are n Hadamard, $n(n-1)/2$ ZZ, n RZ, and n RX gates. The corresponding rotation angles of ZZ, RZ, and RX gates acting on the qubit pair (i, j), the qubit k, and the qubit l are $\gamma[i, j]$, $\alpha[k]$, and $\beta[l]$, respectively, where $i, j, k, l \in \{0, ..., n-1\}$ and $i < j$. We observe that in the QAOA circuit, only two-qubit gates need to be mapped since the single-qubit gates can be assigned at the algorithm level. Compared to mapping the entire circuit, mapping only ZZ gates reduces the computation effort and thus speeds up the compilation time.

In the topology of the QPU, e.g., Fig. 4 shows the topology of IBM QUPs with 27 qubits, there are several possibilities regarding the connectivity between qubits. Compared to finding solutions in the whole topology, especially for a topology containing hundreds of qubits, finding solutions on a specific type of subtopology can significantly reduce the computational complexity and thus improve the scalability. In the following, we discuss two types of subtopologies, namely linear and T-shaped.

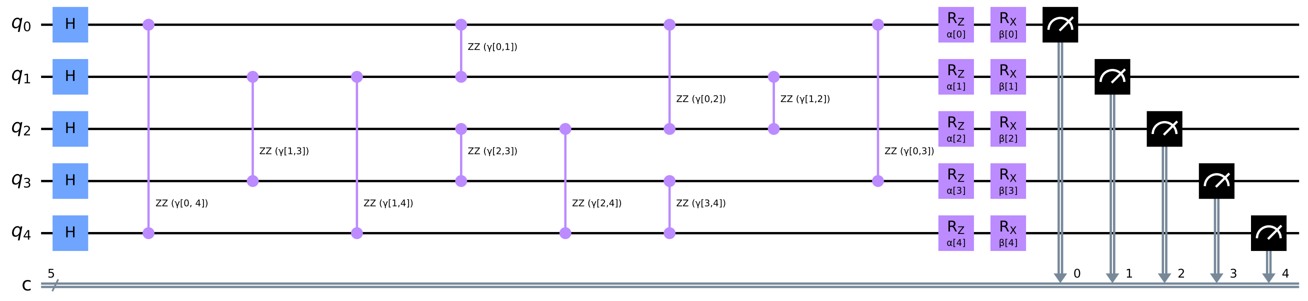

FIGURE 3. 5Q-QAOA circuit with depth $p = 1$. A Hadamard gate acts on each qubit to generate the initial state. ZZ($\gamma[i, j]$) denotes that a ZZ gate with the corresponding rotation angle acts on the qubit pair (i, j). RZ and RX act on the qubit i with corresponding parameters $\alpha[i]$ and $\beta[i]$, respectively, followed by the measurement of qubits.

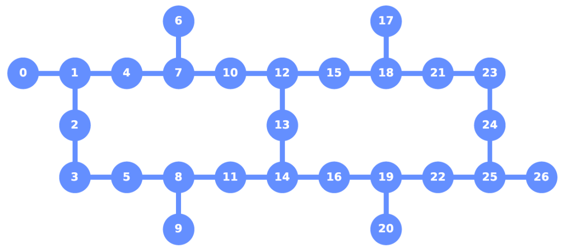

FIGURE 4. Topology of IBM QPUs with 27 qubits.

TABLE 1. Number of SWAP Gates introduced by Qubit Mapping of Only Two-Qubit Gates in QAOA using Exact Approach [9] on a Linear Subtopology

p	1	2	3	4	5
3Q	1	2	4	5	7
4Q	3	8	12	15	19
5Q	6	16	25	36	40

A. LINEAR SUBTOPOLOGY

Fig. 5 (a) shows a linear subtopology, while Fig. 5 (b) and (c) show the solutions of qubit mapping for two-qubit gates in 5Q- and 6Q-QAOA, respectively. Following the AlgAw flow illustrated in Fig. 1, we derive the solutions for QAOA with arbitrary numbers of qubits and depths on a linear subtopology.

1) Optimal Solutions for Small-Scale Circuits

We use an exact approach [9] aiming to minimize the circuit depth to solve the qubit mapping problem of two-qubit gates in small-scale QAOA circuits. The inserted number of SWAP gates is shown in Table 1. Although the QAOA with depth p contains p subcircuits with the same structure, the number of SWAP gates C does not grow linearly with p, but satisfies $C \geq C_0 \times p$, where C_0 is the number of SWAP gates when $p = 1$.

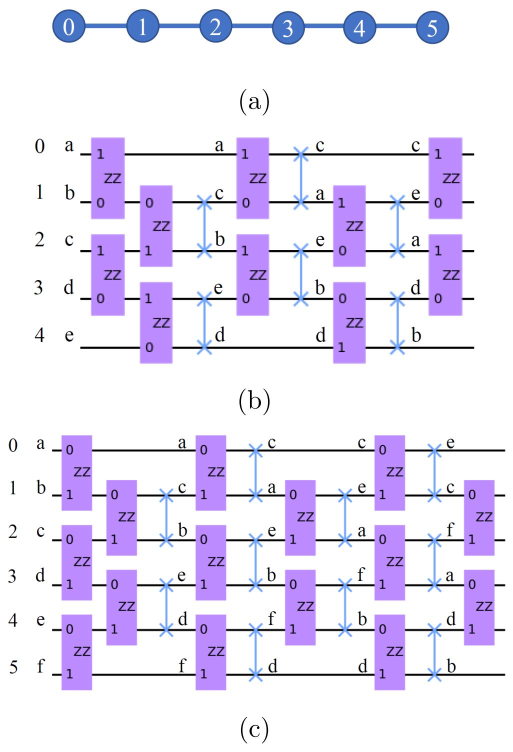

FIGURE 5. (a) Linear subtopology. Analyzed qubit mapping solutions of only two-qubit gates in (b) 5Q- and (c) 6Q-QAOA. Independent of the initial qubit order, all the required ZZ gates can be executed on a linear subtopology. The parameters in ZZ gates are then assigned at the algorithm level.

2) Analyzing and Extending

The core step in AlgAw is the analyzing and extending. The solution to qubit mapping of two-qubit gates in QAOA should be independent of the initial qubit order, and all required two-qubit gates can be implemented by parameter assignment. The commutation rule that all ZZ gates with arbitrary rotation angles commute with each other can be used to combine ZZ and SWAP gates on the same qubit pair to take the advantage of CX gate cancellation, as illustrated in Fig. 2 (b).

Consider an arbitrary initial qubit order $[a, b, c, d, e]$ in the solution of 5Q-QAOA qubit mapping shown in Fig. 5 (b). After each SWAP gate, the qubit order is exchanged between both gates, so that the orders after each SWAP layer are $[a, c, b, e, d]$, $[c, a, e, b, d]$, and $[c, e, a, d, b]$. Each qubit needs to interact with all other qubits, so for qubit a, the ZZ gate acts on qubit pairs (a, b), (a, c), (a, d), and (a, e); for qubit b, it acts on (b, c), (b, d), and (b, e); for qubit c, it acts on (c, d) and (c, e); and for qubit d, it acts on (d, e). Note that ZZ, SWAP, and ZZ-SWAP gates are all undirected, i.e., they do not distinguish between control and target qubits. We observe that all the required 10 ZZ gates in 5Q-QAOA can be executed on a linear subtopology, independent of the initial order of qubits. This finding is convenient for the qubit mapping of the algorithm, since this scheme can then be automatically extended to a higher QAOA-depth. Similarly, as shown in Fig. 5 (c), all the required 15 ZZ gates for 6Q-QAOA can also be executed by 4 SWAP layers.

The qubit mapping solution of two-qubit gates in n-qubit QAOA contains n layers of ZZ gates. A SWAP gate is performed after each ZZ gate, except for the ZZ gates on the first and last layer. Due to the CX gate cancellation, each SWAP gate behind the ZZ gate introduces only one additional CX gate. From Fig. 5 we observe that the optimal solutions of QAOA on the linear subtopology have the same structure as the SWAP-Network in [16] without the first and last SWAP layer, which provides the method to extend the solution to arbitrary numbers of qubits.

To compare scalability, we explore the compilation time of finding the optimal solutions by the exact approach aiming to minimize the circuit depth [9]. We consider the entire QAOA circuit consisting of all required single- and two-qubit gates with different numbers of qubits and depths on a 27-qubit IBM QPU with the topology shown in Fig. 4.

As shown in Table 2, the compilation time of 3Q-QAOA grows with the QAOA-depth, from a few seconds to several days. For 4Q-QAOA, the compilation time is 13 seconds for $p = 1$, whereas it takes more than 15 hours for $p = 2$ and more than a week for $p = 3$. In comparison, using AlgAw the solution for $p = 1$ can be automatically extended to higher p without any computation, and for QAOA with hundreds of qubits, it takes only a few seconds.

The data show that the compilation time using the exact method increases from 3 seconds for 3Q- to 41 hours for 9Q-QAOA, meaning that it is difficult to scale the solution to a larger number of qubits. As we have observed that the solution of two-qubit gates in QAOA on a linear subtopology has the same structure as the SWAP-Network in [16] without SWAP gates on the first and last layer, using AlgAw we can extend the solution to an arbitrary number of qubits without any computation.

3) Reconstruct Circuit and Assign Parameters

The circuit that satisfies the connectivity constraints is finally reconstructed at the algorithm level. The structure of the mapped two-qubit gates has been determined and only the corresponding parameters need to be assigned. Algorithm 1 shows the pseudocode of applying AlgAw to QAOA on the linear subtopology. The reconstructing starts with an initialized qubit order $O = \{0, 1, ...i, ..., j, n-1\}$ for n qubits. The order of qubits i and j is exchanged only when a SWAP gate exists on the qubit pair (i, j), otherwise it remains the same. The parameters are assigned according to the current qubit order.

The mapped circuit of 5Q-QAOA with $p = 1$ is shown in Fig. 6. There are $n = 5$ ZZ layers. If the ZZ gate is not on the first or last layer, a SWAP gate is located directly behind it. All single rotation gates RZ and RX are assigned according to the current qubit order, followed by the measurement of qubits. For higher p, the gates that implement the cost function and the mixer are repeated p times as a unit. The parameters are changed accordingly.

B. T-SHAPED SUBTOPOLOGY

The previous results are based on a linear subtopology. In the following, we discuss another symmetric subtopology, the T-shaped, as shown in Fig. 7 (a). Note that the minimum number of qubits on a T-shaped subtopology is 4.

The qubit mapping solutions of two-qubit gates in 5Q- and 6Q-QAOA on a T-shaped subtopology are shown in Fig. 7 (b) and (c), respectively. The solutions can also be automatically extended to higher QAOA-depths, similar to the solutions on linear subtopology in Fig. 5. Consider again an arbitrary initial order of qubits $[a, b, c, d, e]$ for the solution of 5Q-QAOA in Fig. 7 (b). After each SWAP gate, the order of the corresponding qubits is exchanged, e.g., the first SWAP gate on the qubit pair (b, d) leads to a new qubit order of $[a, d, c, b, e]$. The results show that all the required 10 ZZ gates in 5Q-QAOA can be executed on the T-shaped subtopology by assigning the corresponding parameters, thus returning the solutions to arbitrary QAOA-depths. We observe the same result for 6Q-QAOA in Fig. 7 (c), i.e., for an arbitrary initial order of qubits $[a, b, c, d, e, f]$, these SWAP gates enable all the required 15 ZZ gates to be executed.

Compared to the solution of 5Q-QAOA on the linear subtopology that has the optimal circuit depth of 5 in the gate set {ZZ, ZZ-SWAP} and contains 6 SWAP gates, the solution on the T-shaped subtopology saves 2 SWAP gates, whereas increases the circuit depth by 3. For 6Q-QAOA, T-shaped subtopology saves 2 SWAP gates and increases the circuit depth by 4. The results show that a T-shaped subtopology provides more connectivity for qubits and requires fewer SWAP gates, whereas a linear subtopology yields a shorter circuit depth.

Based on these solutions, we can extend it to an arbitrary number of qubits. As shown in Fig. 7 (c), the solution of 6Q-QAOA on T-shaped subtopology has a similar structure to that of 5Q-QAOA on linear subtopology. We observe that the

TABLE 2. Compilation Time (hh:mm:ss) of the Exact Approach [9] for QAOA with different numbers of qubits and depths on an IBM QPU with 27 Qubits

	3qp1	3qp2	3qp3	3qp4	3qp5	3qp6	3qp7	4qp1	4qp2	5qp1	6qp1	7qp1	9qp1
time	0:00:03	0:01:13	0:22:47	1:48:17	12:48:27	31:07:12	129:59:04	0:00:13	15:26:30	0:01:19	4:42:10	1:12:40	41:47:33

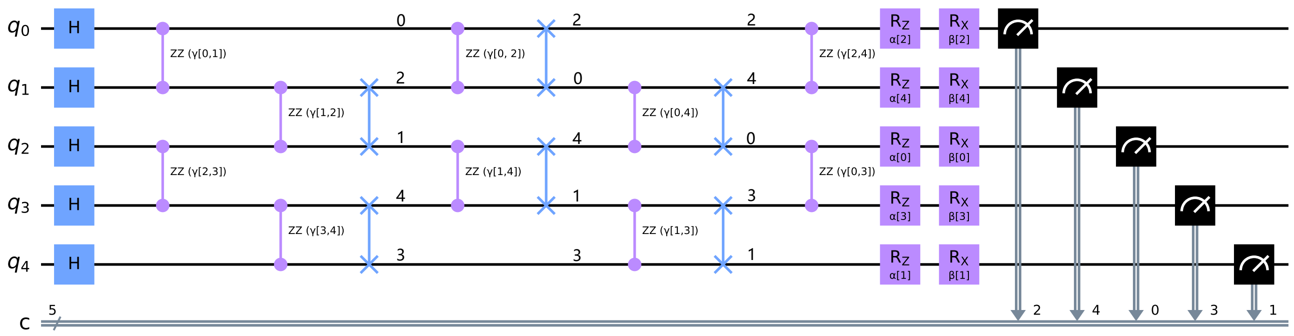

FIGURE 6. The Mapped 5Q-QAOA circuit with $p = 1$ on a linear subtopology. Each SWAP layer introduces a new qubit order that enables the gates satisfying the connectivity constraints in the remaining ZZ gates to be executed. Single-qubit gates RZ and RX, as well as the measurement operators, are assigned according to the current qubit order.

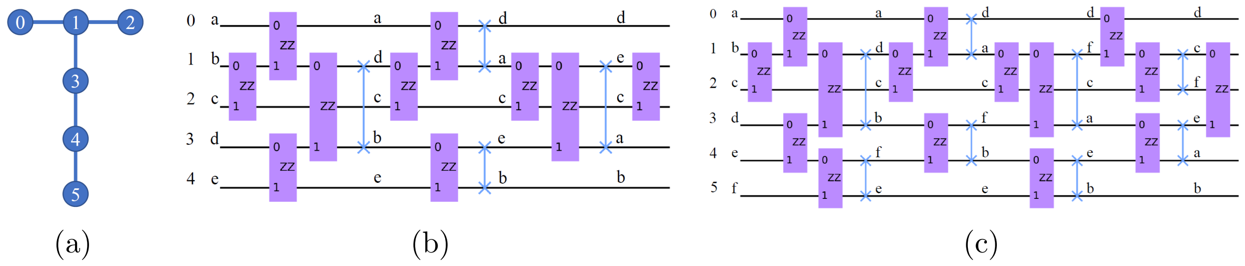

FIGURE 7. (a) T-shaped subtopology. Analyzed qubit mapping solutions of only two-qubit gates in (b) 5Q- and (c) 6Q-QAOA on a T-shaped subtopology. Independent of the initial order of qubits, all the required ZZ gates can be executed.

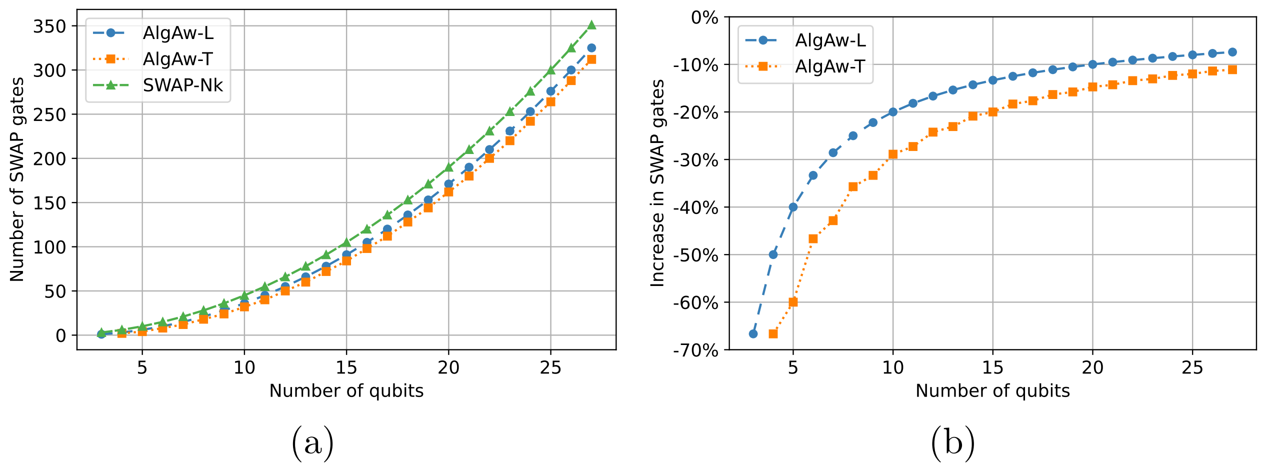

FIGURE 8. (a) Number of SWAP gates for AlgAw on a linear subtopology (AlgAw-L, blue), AlgAw on a T-shaped subtopology (AlgAw-T, orange), and SWAP-Network (SWAP-Nk, green). (b) Increase in the number of SWAP gates with AlgAw-L (blue) and AlgAw-T (orange) compared to SWAP-Network.

Algorithm 1 AlgAw Qubit Mapping for QAOA on Linear Subtopology

```
Input: Number of qubits $n$, QAOA-depth $p$, Parameters
    $\gamma[i, j]$, $\alpha[k]$, and $\beta[k]$ of gates $Z_iZ_j$, $RZ_k$, and $RX_k$,
    respectively, where $i, j, k \in \{0, ..., n-1\}$ and $i < j$
Output: Circuit satisfying connectivity constraints
 1: function ApplyZZGate($O$, $c$, t)
 2:     Apply ZZ($\gamma[O[c], O[t]]$) on ($c$, t)
 3: end function
 4: function ApplyZZSWAPGate($O$, $c$, t)
 5:     Apply ZZ($\gamma[O[c], O[t]]$) on ($c$, $t$)
 6:     Apply SWAP on ($c$, $t$)
 7:     $O[c] \leftrightarrow O[t]$                ▷ Exchange
 8: end function
 9: Initialize the qubit order $O$ $\{0, 1, ..., n-1\}$
10: Prepare the initial state $|0\rangle^{\otimes n}$
11: Apply $H^{\otimes n}$
12: while $p > 0$ do
13:     $s \leftarrow 0$
14:     while $s < n$ do
15:         for $q := 0$ to $n-1$ step 2 do
16:             if $s == 0$ or $s == n-1$ then
17:                 ApplyZZGate($O$, $q$, $q+1$)
18:             else
19:                 ApplyZZSWAPGate($O$, $q$, $q+1$)
20:             end if
21:         end for
22:         $s \leftarrow s+1$
23:         if $s < n$ then
24:             for $q := 1$ to $n-1$ step 2 do
25:                 if $s == 0$ or $s == n-1$ then
26:                     ApplyZZGate($O$, $q$, $q+1$)
27:                 else
28:                     ApplyZZSWAPGate($O$, $q$, $q+1$)
29:                 end if
30:             end for
31:         end if
32:         $s \leftarrow s+1$
33:     end while
34:     for $k := 0$ to $n$ do
35:         Apply RZ($\alpha[O[k]]$) on $k$
36:         Apply RX($\beta[O[k]]$) on $k$
37:     end for
38:     $p \leftarrow p-1$
39: end while
40: Measure the qubits ($[0, ..., n-1] \rightarrow [O[0], ..., O[n-1]]$)
```

SWAP gates on each odd layer are fixed, whereas the first SWAP gate on each even layer changes, i.e., SWAP gates on (d, a) and (c, f) in Fig. 7 (c) alternate. The SWAP layer cycle on a T-shaped subtopology consists of four layers, while that on a linear subtopology consists of two layers. Algorithm 2 describes the procedure to generate n SWAP layers for n-qubit QAOA on a T-shaped subtopology, where qubit 1 is connected to qubits 0, 2, and 3, as shown in Fig. 7 (a).

Algorithm 2 SWAP Layers on T-shaped Subtopology

```
Input: Number of qubits $n$
Output: List of SWAP layers $S$
 1: $n_{\text{even}} \leftarrow (n-1) - (n-1)\%2$
 2: $n_{\text{odd}} \leftarrow (n-1) - 1 + (n-1)\%2$
 3: $S \leftarrow$ empty List
 4: $j \leftarrow 0$
 5: while $j < n$ do
 6:     $S[j] \leftarrow S[j] \cup [(1, 3), (4, 5), ..., (n_{\text{odd}} - 1, n_{\text{odd}})]$
 7:     if $++j \geq n$ then break
 8:     end if
 9:     $S[j] \leftarrow S[j] \cup [(1, 0), (3, 4), ..., (n_{\text{even}} - 1, n_{\text{even}})]$
10:     if $++j \geq n$ then break
11:     end if
12:     $S[j] \leftarrow S[j] \cup [(1, 3), (4, 5), ..., (n_{\text{odd}} - 1, n_{\text{odd}})]$
13:     if $++j \geq n$ then break
14:     end if
15:     $S[j] \leftarrow S[j] \cup [(1, 2), (3, 4), ..., (n_{\text{even}} - 1, n_{\text{even}})]$
16:     if $++j \geq n$ then break
17:     end if
18: end while
```

Algorithm 3 shows the pseudocode of AlgAw for QAOA on a T-shaped subtopology. Considering all the required ZZ gates in QAOA, some of them are executed once the connectivity constraints are satisfied. Each SWAP layer introduces a new qubit order and the gates satisfying the connectivity constraints in the remaining ZZ gates are then executed until all ZZ gates have been implemented. Compared to the solution on a linear subtopology that has a determined number of SWAP layers, i.e., $(n-2) \times p$ for n-qubit QAOA with depth p, the solution on T-shaped subtopology requires at least $(n-2) \times p$ and at most $n \times p$ SWAP layers. For $p = 1$, if there are still some remaining ZZ gates after $n-2$ SWAP layers, then the SWAP gates in the penultimate and/or the last layer are required. Note that if the SWAP gate is at the end of all two-qubit gates on each qubit pair, it can be removed since there are no remaining two-qubit gates that need to satisfy the connectivity constraints by introducing a new qubit order.

We have shown that for the QAOA on dense portfolio optimization problems, there are $n(n-1)/2$ two-qubit gates, and the SWAP gate inserted to satisfy the connectivity constraints can be optimized with CX gate cancellation if it is immediately adjacent to a ZZ gate. For other applications of QAOA, it may not be necessary that every qubit has to interact with all other qubits, which leads to the absence of some ZZ gates in Fig. 5 and Fig. 7. However, these SWAP layers still allow the circuit to satisfy the connectivity constraints but just without the full benefits of combining with ZZ gates.

Algorithm 3 AlgAw Qubit Mapping for QAOA on T-shaped Subtopology

```
Input: Number of qubits $n$, QAOA-depth $p$, Parameters $\gamma[i,j]$, $\alpha[k]$, and $\beta[k]$ of gates $Z_iZ_j$, $RZ_k$, and $RX_k$, respectively, where $i,j,k \in \{0,...,n-1\}$ and $i<j$
Output: Circuit satisfying connectivity constraints
function ApplyZZGate($O$, $i$, $j$)
    Apply ZZ($\gamma[i,j]$) on $(O.index(i), O.index(j))$
end function
function ApplySWAPGate($O$, $i$, $j$)
    Apply SWAP on ($i$, $j$)
    $O[i] \leftrightarrow O[j]$                                   ▷ Exchange
end function
$E \leftarrow$ List of connected edges on T-shaped subtopology
$L \leftarrow$ List of all ZZ gates
$S \leftarrow$ List of SWAP layers                                ▷ Algorithm 2
Initialize the qubit order $O$ $\{0,1,...,n-1\}$
Prepare the initial state $|0\rangle^{\otimes n}$
Apply $H^{\otimes n}$
while $p>0$ do
    for $k := 0$ to $n-2$ do
        if $L$ is empty then continue
        end if
        $E_O \leftarrow [(O[s], O[t])$ for $(s,t) \in E]$
        $S_O \leftarrow [(O[s], O[t])$ for $(s,t) \in S[k]]$
        for gate $Z_iZ_j \in L$ do
            if $(i,j) \in E_O$ and $(i,j) \notin S_O$ then
                ApplyZZGate($O$, $i$, $j$)
                $L$.remove($Z_iZ_j$)
            end if
        end for
        for $(i,j) \in S[k]$ do                                   ▷ SWAP on $(i,j)$
            if $Z_{O[i]}Z_{O[j]} \in L$ then
                Apply ZZ($\gamma[O[i], O[j]]$) on $(i,j)$
                $L$.remove($Z_{O[i]}Z_{O[j]}$)
                if $L$ is empty then continue
                end if
                ApplySWAPGate($O$, $i$, $j$)
            end if
        end for
    end for
```

Algorithm 3 (Continued) AlgAw Qubit Mapping for QAOA on T-shaped Subtopology

```
    for gate $Z_iZ_j \in L$ do
        $E_O \leftarrow [(O[s], O[t])$ for $(s,t) \in E]$
        if $(i,j) \in E_O$ then
            ApplyZZGate($O$, $i$, $j$)
            $L$.remove($Z_iZ_j$)
        end if
    end for
    for $k := n-2$ to $n$ do
        if $L$ is empty then continue
        end if
        for $Z_iZ_j \in L$ do
            $E_O \leftarrow [(O[s], O[t])$ for $(s,t) \in E]$
            $S_O \leftarrow [(O[s], O[t])$ for $(s,t) \in S[k]]$
            if $(i,j) \in E_O$ and $(i,j) \notin S_O$ then
                ApplyZZGate($O$, $i$, $j$)
                $L$.remove($Z_iZ_j$)
            end if
        end for
        for $(i,j) \in S[k]$ do
            if $Z_{O[i]}Z_{O[j]} \in L$ then
                Apply ZZ($\gamma[O[i], O[j]]$) on $(i,j)$
                $L$.remove($Z_{O[i]}Z_{O[j]}$)
            end if
            if $L$ is empty then continue
            end if
            ApplySWAPGate($O$, $i$, $j$)
        end for
    end for
    if SWAP is located on the end then
        Remove SWAP
    end if
    for $k := 0$ to $n$ do
        Apply RZ($\alpha[O[k]]$) on $k$
        Apply RX($\beta[O[k]]$) on $k$
    end for
    $p \leftarrow p-1$
end while
Measure the qubits ($[0,...,n-1] \rightarrow [O[0],...,O[n-1]]$)
```

C. COMPARISON

Compared to the SWAP-Network in [16], AlgAw on the linear subtopology saves $(n-1) \times p$ SWAP gates for n-qubit QAOA with depth p. The total number of SWAP gates C using AlgAw is $C_0 \times p$, which is a linear function of p, where C_0 is the number of SWAP gates when $p=1$ and satisfies $C_0 = (n-1)(n-2)/2$. Compared to the solutions found by the exact approach in Table 1, AlgAw requires the same number of SWAP gates for $p=1$ and a smaller number of SWAP gates for higher p, which yields better results than the exact method.

Fig. 8 (a) shows the number of SWAP gates for AlgAw on a linear subtopology (AlgAw-L, blue), AlgAw on a T-shaped subtopology (AlgAw-T, orange), and SWAP-Network (SWAP-Nk, green). Compared to the SWAP-Network, AlgAw-L leads to a reduction of SWAP gates by 66.7% for 3Q- and 20% for 10Q-QAOA mapping, as shown in Fig. 8 (b), whereas AlgAw-T reduces SWAP gates by 66.7% for 4Q- and 28.9% for 10Q-QAOA mapping.

Although a T-shaped subtopology provides more connectivity for qubits and requires a lower number of SWAP gates than a linear subtopology, the topology of a QPU contains more linear subtopologies than T-shaped, implying that a larger set of qubits can be used to implement the circuit based on a linear subtopology. In benchmarking below, we compare AlgAw-L with other methods.

IV. BENCHMARKING EXPERIMENTS

In this section, we report the benchmarking results of the QAOA on portfolio optimization problems with different numbers of qubits and depths using AlgAw-L, Qiskit [24], Tket [25], and SWAP-Network [16] on ibmq_ehningen, ibm_auckland, and ibm_hanoi. The three IBM QPUs contain 27 qubits and have the same topology, as shown in Fig. 4. For the experiments, we set the number of shots in Qiskit to 50000 for 3Q-, 4Q-, and 5Q-QAOA, and 60000 for 6Q-QAOA. For AlgAw-L, Tket, and SWAP-Network, we use Mapomatic [26] to find the qubits on which the circuit has the maximum fidelity, whereas for Qiskit transpile we use the default settings.

A. METRICS

To evaluate the quality of qubit mapping, we consider circuits mapped by various methods and use Qiskit transpile to decompose their gates into the basis gate set of the QPU on which the circuit is executed. We use as metrics circuit properties including CX gate count, circuit depth, and the total number of gates of these mapped and decomposed circuits to estimate the quality of different approaches.

The Approximation Ratio (AR) and Success Probability (SP) are used to evaluate the performance of QAOA on QPUs. Portfolio optimization aims at maximizing the overall return while minimizing the overall risk. Generally, a defined number of assets needs to be selected from n assets (n qubits), which is called the budget constraint. Considering the cost function F of the portfolio optimization problems, if the solution is infeasible, i.e., the solution does not satisfy the budget constraint, the value of AR is defined as 0, otherwise it has the following definition:

$$AR = \frac{F - F_{max}}{F_{opt} - F_{max}}, \tag{1}$$

where F and F_{opt} are the average value found by QAOA and the optimal/minimum value, respectively, whereas F_{max} is the worst-case/maximum value. The SP is defined as the probability of obtaining an optimal solution. A higher value of AR or SP implies a better performance of QAOA, whereas a lower value of circuit depth or CX gate count indicates a higher quality of the qubit mapping solution.

B. CIRCUIT PROPERTIES

The number of CX gates, circuit depth, and total number of gates for QAOA with different numbers of qubits and depths on ibmq_ehningen, ibm_auckland, and ibm_hanoi are summarized in Table 3.

For 3Q-QAOA, AlgAw-L has the least number of CX gates and the lowest circuit depth on three IBM QPUs. In contrast, Qiskit introduces the largest number of CX gates, whereas Tket produces the highest circuit depth for higher p. As we have discussed, the AlgAw-L saves $(n-1)\times p$ SWAP gates than SWAP-Network for n-qubit QAOA with depth p, corresponding to $(n-1)\times p$ CX gates because of the CX gate cancellation between ZZ and SWAP gates. In addition, AlgAw-L, Tket, and SWAP-Network yield a constant number of CX gates, circuit depth, and total gates on three IBM QPUs, whereas Qiskit produces different results for each mapping of the same circuit, implying that AlgAw-L, Tket, and SWAP-Network provide more stable results than Qiskit.

4Q-QAOA using AlgAw-L has also the least number of CX and total gates, and the lowest circuit depth for each p on both IBM QPUs, whereas Qiskit introduces the largest number of CX and total gates and produces the highest circuit depth. SWAP-Network yields better results than Tket.

As with the 3Q- and 4Q-QAOA results, for 5Q-QAOA, AlgAw-L yields the least CX and total gate count and the lowest circuit depth compared to others. In comparison, Qiskit has more than twice the number of CX gates and circuit depth than AlgAw-L.

For 6Q-QAOA, AlgAw-L provides also the best results on both IBM QPUs. Compared to AlgAw-L and SWAP-Network, Qiskit and Tket introduce more CX gates leading to higher circuit depths. On ibmq_ehningen, Qiskit delivers comparable results to Tket. However, it produces poor results on ibm_auckland.

The data from circuit properties of QAOA on three IBM QPUs show that the AlgAw-L produces the least number of CX gates and the lowest circuit depth than other methods. SWAP-Network yields better results than Qiskit and Tket.

C. APPROXIMATION RATIO (AR) AND SUCCESS PROBABILITY (SP)

We investigate the performance of various approaches for qubit mapping on IBM QPUs. The average percental CX gate error rates provided by IBM for the experiments of QAOA using AlgAw-L, Qiskit, Tket, and SWAP-Network are summarized in Table 4. The error rates range between 0.41% and 1%.

The AR and SP of 3Q-QAOA on ibmq_ehningen, ibm_auckland, and ibm_hanoi are shown in Fig. 9 (a), (b), and (c), respectively. AlgAw-L yields the largest average values of AR and SP for p from 1 to 7 on all three IBM QPUs, which corresponds to the highest quality of solutions produced by the AlgAw qubit mapping. SWAP-Network provides the same robust results as AlgAw-L, but with lower AR and SP values. In comparison, Qiskit and Tket produce inconsistent results. On ibmq_ehningen, Qiskit gives better results than Tket, whereas on ibm_auckland, Tket performs better, and on ibm_hanoi, they produce comparable results.

The AR and SP of 4Q-QAOA with different values of p on ibmq_ehningen and ibm_auckland are shown in Fig. 9 (d) and (e), respectively. On both IBM QPUs, AlgAw-L produces higher average values of AR and SP than others. On ibmq_ehningen, SWAP-Network performs better than Tket, whereas Qiskit generates the lowest AR values. On ibm_auckland, Tket with more CX gates and higher circuit depth still outperforms SWAP-Network. A possible explanation is that the crosstalk effect occurring on the ZZ-SWAP gates in SWAP-Network is more pronounced than that of the two-qubit gates in Tket.

TABLE 3. Circuit Properties of n-qubit QAOA with p from 1 to 7

			# CX Gates							Depth							# Gates						
n	QPU	Method	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
3	*ibmq_ehningen*	AlgAw-L	7	14	21	28	35	42	49	19	34	49	64	79	94	109	37	62	87	112	137	162	187
		Qiskit	10	27	35	51	54	68	85	22	47	63	87	98	120	145	40	75	101	135	156	188	223
		Tket	9	18	30	39	51	60	72	26	43	68	85	110	127	152	39	66	96	123	153	180	210
		SWAP-Nk	9	18	27	36	45	54	63	21	38	55	72	89	106	123	39	66	93	120	147	174	201
	ibm_auckland	AlgAw-L	7	14	21	28	35	42	49	19	34	49	64	79	94	109	37	62	87	112	137	162	187
		Qiskit	12	22	34	44	51	66	80	24	42	62	80	95	118	140	42	70	100	128	153	186	218
		Tket	9	18	30	39	51	60	72	26	43	68	85	110	127	152	39	66	96	123	153	180	210
		SWAP-Nk	9	18	27	36	45	54	63	21	38	55	72	89	106	123	39	66	93	120	147	174	201
	ibm_hanoi	AlgAw-L	7	14	21	28	35	42	49	19	34	49	64	79	94	109	37	62	87	112	137	162	187
		Qiskit	10	22	31	47	61	75	76	22	42	59	83	105	127	136	40	70	97	131	163	195	214
		Tket	9	18	30	39	51	60	72	26	43	68	85	110	127	152	39	66	96	123	153	180	210
		SWAP-Nk	9	18	27	36	45	54	63	21	38	55	72	89	106	123	39	66	93	120	147	174	201
4	*ibmq_ehningen*	AlgAw-L	15	30	45	60	75	90	105	23	42	61	80	99	118	137	57	98	139	180	221	262	303
		Qiskit	25	53	81	109	137	165	193	31	61	91	121	151	181	211	67	121	175	229	283	337	391
		Tket	19	41	63	85	107	129	151	25	49	73	97	121	145	169	61	109	157	205	253	301	349
		SWAP-Nk	18	36	54	72	90	108	126	25	46	67	88	109	130	151	60	104	148	192	236	280	324
	ibm_auckland	AlgAw-L	15	30	45	60	75	90	105	23	42	61	80	99	118	137	57	98	139	180	221	262	303
		Qiskit	25	53	81	109	137	165	193	31	61	91	121	151	181	211	67	121	175	229	283	337	391
		Tket	19	41	63	85	107	129	151	25	49	73	97	121	145	169	61	109	157	205	253	301	349
		SWAP-Nk	18	36	54	72	90	108	126	25	46	67	88	109	130	151	60	104	148	192	236	280	324
5	*ibmq_ehningen*	AlgAw-L	26	52	78	104	130	156	182	27	50	73	96	119	142	165	81	142	203	264	325	386	447
		Qiskit	50	104	168	218	277	353	385	55	110	171	221	280	349	386	105	195	293	379	472	584	650
		Tket	28	64	100	136	172	208	244	34	68	102	136	170	204	238	83	154	225	296	367	438	509
		SWAP-Nk	30	60	90	120	150	180	210	29	54	79	104	129	154	179	85	150	215	280	345	410	475
	ibm_auckland	AlgAw-L	26	52	78	104	130	156	182	27	50	73	96	119	142	165	81	142	203	264	325	386	447
		Qiskit	63	119	162	250	295	330	392	67	125	168	249	297	330	394	119	210	287	411	491	560	658
		Tket	28	64	100	136	172	208	244	34	68	102	136	170	204	238	83	154	225	296	367	438	509
		SWAP-Nk	30	60	90	120	150	180	210	29	54	79	104	129	154	179	85	150	215	280	345	410	475
6	*ibmq_ehningen*	AlgAw-L	40	80	120	160	200	240	280	31	58	85	112	139	166	193	109	194	279	364	449	534	619
		Qiskit	66	140	218	282	360	432	509	55	105	171	223	260	323	382	135	254	377	486	609	726	848
		Tket	68	134	203	277	343	412	486	60	101	151	214	255	305	368	137	248	362	481	592	706	825
		SWAP-Nk	45	90	135	180	225	270	315	33	62	91	120	149	178	207	114	204	294	384	474	564	654

TABLE 4. Average Percental Error Rates of CX Gates in n-qubit QAOA Reported by the Calibration Data from IBM

n	QPU	AlgAw-L	Qiskit	Tket	SWAP-Nk
3	*ibmq_ehningen*	0.49	0.49	0.49	0.49
	ibm_auckland	0.46	0.46	0.53	0.49
	ibm_hanoi	0.41	0.49	0.46	0.49
4	*ibmq_ehningen*	0.49	0.49	0.49	0.49
	ibm_auckland	0.55	0.55	0.55	0.55
5	*ibmq_ehningen*	0.65	0.64	0.55	0.63
	ibm_auckland	1.00	0.87	0.78	0.51
6	*ibmq_ehningen*	0.77	0.66	0.57	0.62

Fig. 9 (f) and (g) show the benchmarking results of 5Q-QAOA on ibmq_ehningen and ibm_auckland, respectively. The AR values of 5Q-QAOA using AlgAw-L are the highest compared to other methods. Although SWAP-Network produces fewer CX gates and lower circuit depth, Tket performs better.

The results of 6Q-QAOA on ibmq_ehningen is shown in Fig. 9 (h). AlgAw-L produces significantly higher values of AR and SP than the others. Tket has a better performance than Qiskit and SWAP-Network.

The data show that the AlgAw-L produces the least number of CX gates and the lowest circuit depth resulting in the highest average values of AR and SP. Compared to SWAP-Network, AlgAw-L performs significantly better and more consistently on all three IBM QPUs. On the one hand, AlgAw-L has $(n-1) \times p$ fewer SWAP gates than SWAP-Network. On the other hand, the circuit elements in the AlgAw-L are changed compared to the SWAP-Network which contains only ZZ-SWAP gates, whereas AlgAw-L has both ZZ and ZZ-SWAP gates. Compared to the ZZ-SWAP gate, the ZZ gate introduces less noise in terms of the CX gate's error rate and the crosstalk error.

V. OTHER APPLICATIONS OF ALGAW

We have discussed the application of AlgAw in QAOA. In this section, we demonstrate another application and provide the possibility of applying AlgAw to other algorithms.

A. VARIATIONAL-QUANTUM-EIGENSOLVER (VQE)

The Variational-Quantum-Eigensolver (VQE) [27]–[29] belongs to one of the VQAs and has many applications in quantum chemistry [30], condensed matter physics [31], and combinatorial optimization [32], etc. The VQE is a hybrid algorithm designed to find the ground state energy or eigenvalue of a Hamiltonian. Let H be the Hamiltonian of a given quantum system, and let $|\psi\rangle$ be a trial wavefunction. The Rayleigh-Ritz quotient is bounded below by the ground state energy E_0:

$$E_0 \leq \frac{\langle\psi| H |\psi\rangle}{\langle\psi|\psi\rangle}. \tag{2}$$

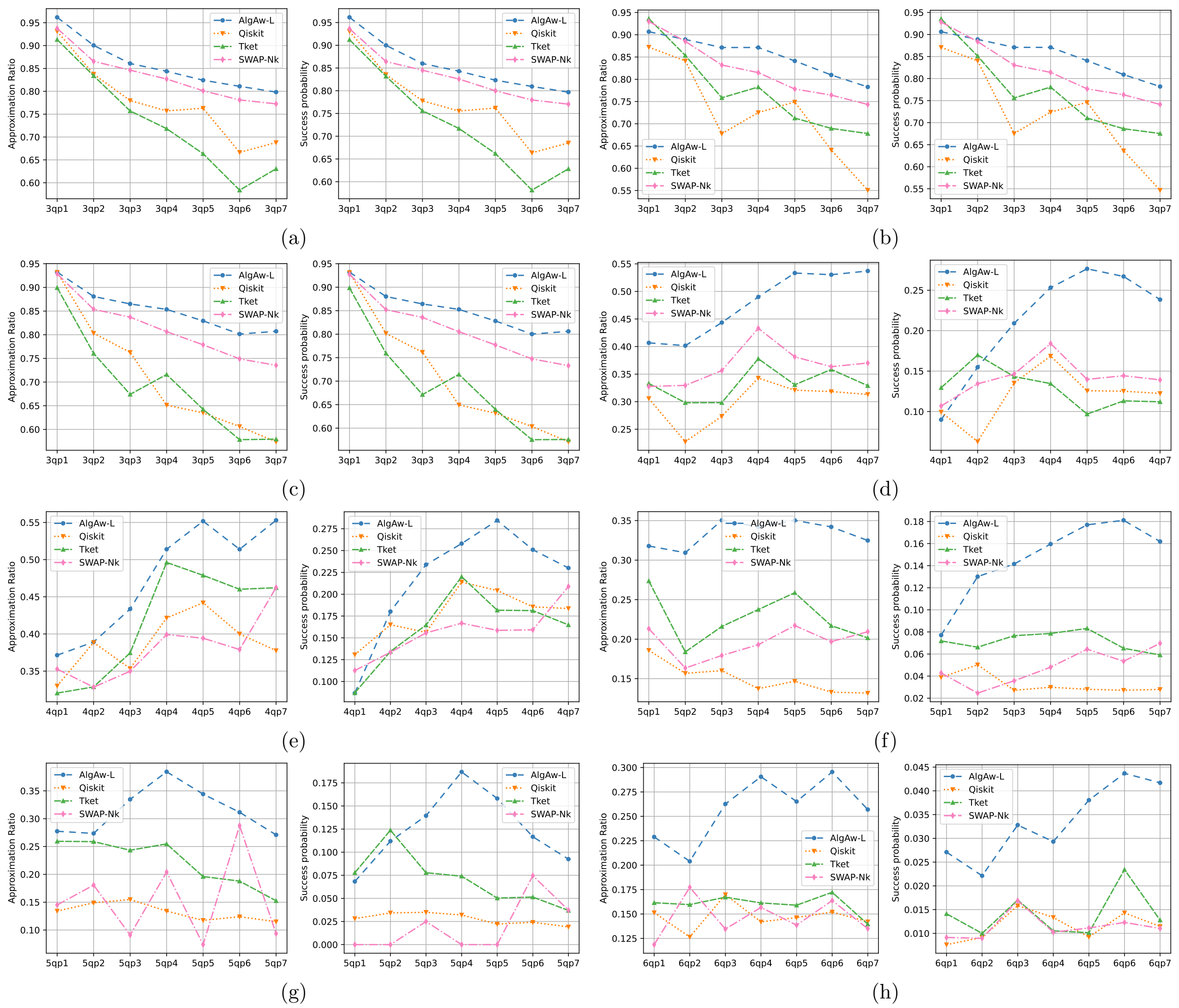

FIGURE 9. Approximation ratio and success probability of QAOA for 3 qubits on (a) *ibmq_ehningen*, (b) *ibm_auckland*, and (c) *ibm_hanoi*; for 4 qubits on (d) *ibmq_ehningen* and (e) *ibm_auckland*; for 5 qubits on (f) *ibmq_ehningen* and (g) *ibm_auckland*; and for 6 qubits on (h) *ibmq_ehningen*.

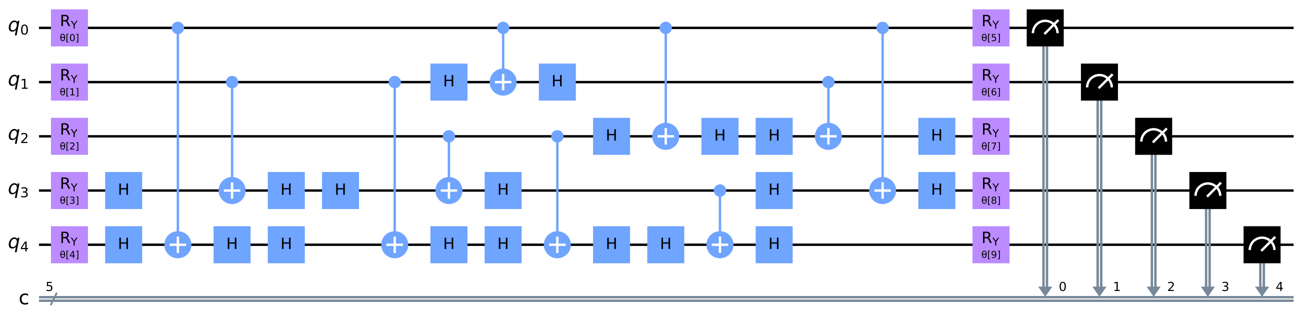

FIGURE 10. 5Q-VQE-HEA circuit with depth $p = 1$. The first layer is generated by parameterized RY gates. Then CZ gates are performed on all qubit pairs, followed by another set of parameterized RY gates and the measurement. For higher depth p, the CZ gates and the second set of RY gates are repeated p times.

The objective of VQE is to find a quantum state by searching through a parametrized ansatz state $|\psi(\theta)\rangle = U(\theta)|0\rangle$ such that the expectation value of the Hamiltonian is minimized, where $U(\theta)$, typically referred to as variational form or ansatz, is a parameterized unitary that can be implemented by, e.g., a quantum circuit, θ is a vector of parameters, and $|0\rangle$ is the initial state. The Hamiltonian is problem-specific and can be rewritten into quantum operators, usually Pauli operators, that are directly measurable on the QPU.

The choice of ansatz circuits for the state preparation in VQE is crucial to its performance. Three common categories of ansatz circuits are chemically inspired [33], Hardware-Efficient Ansatz (HEA) [34], and Hamiltonian variational [35]. In this paper, we investigate the VQE using HEA (VQE-HEA) with full entanglement (e.g., [36]). The ansatz circuit on five qubits is illustrated in Fig. 10. The first layer consists of parameterized RY gates. We use controlled-Z (CZ) gates as entangling gates. Unlike the CX gate, which distinguishes between control and target qubits, the CZ gate belongs to undirected gates. After that, another set of parameterized RY gates is performed, followed by the measurement. For higher p, the subcircuit between the first layer and the measurement is repeated p times. The circuit with n qubits and depth p contains $(p+1)\times n$ parameters that need to be optimized by a classical optimizer. For the full entanglement in the considered circuit, there are $n(n-1)/2$ CZ gates, each consisting of one CX gate and two Hadamard gates.

By analyzing the ansatz circuit, we observe that it has the same structure as the QAOA on dense portfolio optimization problems. The solution to qubit mapping of all two-qubit gates in QAOA can be similarly extended here. Unlike the ZZ gate in QAOA, which is bordered by the CX gates, the CZ gate in the VQE-HEA circuit is bordered by the Hadamard gates. Inserting a SWAP gate directly behind the CZ gate loses the advantage of the CX gate cancellation between the CX and SWAP gates. Therefore, we introduce the SWAP gate behind the CX gate and adjust the position of the second Hadamard gate accordingly.

The Algorithm 4 describes the procedure of AlgAw for the VQE-HEA with full entanglement on a linear subtopology. First, we implement the CZ gate using CX and Hadamard gates. c and t denote control and target qubits, respectively. The Hadamard gates act on the physical qubit representing the target qubit of the CX gate. We then realize the CZ-SWAP gate by inserting a SWAP gate after the CX gate and performing the second Hadamard gate on the physical qubit representing the control qubit of the CX gate, rather than on the target qubit like the first Hadamard gate, because the introduced SWAP gate changes the qubit order. Analogously, we construct $n-2$ SWAP layers for n qubits on the linear subtopology, meaning that CZ gates are performed on the first and last layer, while the constructed CZ-SWAP gates are implemented on the remaining layers. Finally, the RY gates are assigned correspondingly, followed by the measurement. The solution to qubit mapping of the VQE-HEA circuit on a T-shaped subtopology can be obtained in a similar manner.

Algorithm 4 AlgAw Qubit Mapping for VQE-HEA on Linear Subtopology

```
Input: Number of qubits $n$, VQE-depth $p$, Vector of parameters $\theta$ with dimension $(p+1)\times n$
Output: Circuit satisfying connectivity constraints
 1: function ApplyCZGate($c$, $t$)
 2:     Apply H on $t$
 3:     Apply CX on ($c$, $t$)
 4:     Apply H on $t$
 5: end function
 6: function ApplyCZSWAPGate($O$, $c$, $t$)
 7:     Apply H on $t$
 8:     Apply CX on ($c$, $t$)
 9:     Apply SWAP on ($c$, $t$)
10:     $O[c] \leftrightarrow O[t]$                         ▷ Exchange
11:     Apply H on $c$
12: end function
13: Initialize the qubit order $O$ $\{0, 1, ..., n-1\}$
14: Prepare the initial state $|0\rangle^{\otimes n}$
15: for $i := 0$ to $n$ do
16:     Apply RY($\theta[i]$) on $i$
17: end for
18: $p_{max} \leftarrow p$
19: while $p > 0$ do
20:     $s \leftarrow 0$
21:     while $s < n$ do
22:         for $q := 0$ to $n-1$ step 2 do
23:             if $s == 0$ or $s == n-1$ then
24:                 ApplyCZGate($q$, $q+1$)
25:             else
26:                 ApplyCZSWAPGate($O$, $q$, $q+1$)
27:             end if
28:         end for
29:         $s \leftarrow s+1$
30:         if $s < n$ then
31:             for $q := 1$ to $n-1$ step 2 do
32:                 if $s == 0$ or $s == n-1$ then
33:                     ApplyCZGate($q$, $q+1$)
34:                 else
35:                     ApplyCZSWAPGate($O$, $q$, $q+1$)
36:                 end if
37:             end for
38:         end if
39:         $s \leftarrow s+1$
40:     end while
41:     for $i := 0$ to $n$ do
42:         Apply RY($\theta[i + n * (p_{max} - p + 1)]$) on $O[i]$
43:     end for
44:     $p \leftarrow p-1$
45: end while
46: Measure the qubits $([0, ..., n-1]) \to [O[0], ..., O[n-1]]$
```

The mapped circuit with depth $p = 1$ on five qubits is represented in Fig. 11. Each SWAP gate introduces a new qubit order and adds an additional CX gate, resulting in a

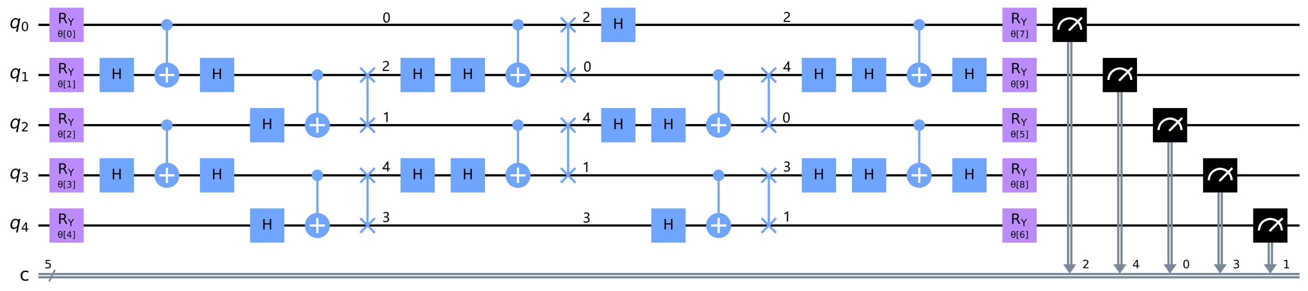

FIGURE 11. Mapped 5Q-VQE-HEA circuit with $p = 1$ on a linear subtopology. The SWAP gate is inserted behind the CX gate to take advantage of the CX gate cancellation. Hadamard gates, the second set of RY gates, and measurement operators are assigned according to the current qubit order.

total of $p \times (n-1)^2$ CX gates. The introduced SWAP gates ensure that all the required CZ gates can be executed. The RY and measurement gates are assigned according to the current qubit order.

The application of AlgAw in VQE shows that the solution of qubit mapping can be obtained by analyzing the features of the ansatz circuit. Moreover, exploring the insertion location of the SWAP gate can reduce additional CX gates. Designing the solution at the algorithm level allows for optimal and scalable results.

B. OTHERS

The previous results demonstrate that the SWAP layers can be constructed not only directly behind each element in the subcircuit, but also flexibly inside this element, such as constructing SWAP layers behind the CX gate in the CZ gate to take advantage of the CX gate cancellation, thus providing applications in other algorithms with a regular structure.

In addition to mapping the fully connected two-qubit gates, the AlgAw qubit mapping can also be used to study other types of connectivity of two-qubit gates on specific subtopologies. Furthermore, the study of AlgAw qubit mapping helps to design the algorithm with its scalability in mind. On the one hand, designing a specific algorithm to adapt the architecture can improve the performance of the algorithm since the present NISQ computers only support deterministic connectivity. On the other hand, studying the specific structure of the algorithm can provide experience for developing the architecture of the QPUs.

VI. CONCLUSIONS AND FUTURE WORK

VQAs with regular circuit structures have the opportunity to obtain the solutions to qubit mapping by analyzing the solutions for small-scale circuits computed with exact methods. The mapping can be performed efficiently by analyzing the features of the algorithm.

AlgAw qubit mapping provides optimal and scalable solutions for the QAOA on dense portfolio optimization problems with arbitrary numbers of qubits and depths on linear and T-shaped subtopologies. The optimality comes from using the exact method for small-scale circuits, while the scalability results from analyzing these solutions. The benchmarking results of QAOA on three IBM QPUs show that AlgAw provides the least number of CX gates and the lowest circuit depth resulting in better performance than Qiskit, Tket, and SWAP-Network.

In the further, we are interested in developing the application of AlgAw further to other algorithms with regular structures. In addition, different types of subtopologies can also be explored and evaluated.

ACKNOWLEDGMENT

The authors would like to thank Kathrin F. Koenig for her constructive criticism of the manuscript. The authors would also like to thank Thomas Wellens, Joris Kattemölle, Sebastian Brandhofer, and Andreas Ketterer for their useful discussions and exchanges. The authors acknowledge the financial support of the project QORA by Ministerium für Wirtschaft, Arbeit und Wohnungbau Baden-Württemberg in the frame of the Competence Center Quantum Computing Baden-Württemberg.

REFERENCES

[1] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio et al., "Variational quantum algorithms," Nature Reviews Physics, vol. 3, no. 9, pp. 625–644, 2021. [Online]. Available: https://doi.org/10.1038/s42254-021-00348-9

[2] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke et al., "Noisy intermediate-scale quantum algorithms," Reviews of Modern Physics, vol. 94, no. 1, p. 015004, 2022. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.94.015004

[3] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for nisq-era quantum devices," in Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA, April 13-17, 2019, I. Bahar, M. Herlihy, E. Witchel, and A. R. Lebeck, Eds. ACM, 2019, pp. 1001–1014. [Online]. Available: https://doi.org/10.1145/3297858.3304023

[4] A. Cowtan, S. Dilkes, R. Duncan, A. Krajenbrink, W. Simmons, and S. Sivarajah, "On the qubit routing problem," in 14th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2019, June 3-5, 2019, University of Maryland, College Park, Maryland, USA, ser. LIPIcs, W. van Dam and L. Mancinska, Eds., vol. 135. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 5:1–5:32. [Online]. Available: https://doi.org/10.4230/LIPIcs.TQC.2019.5

[5] D. Bhattacharjee and A. Chattopadhyay, "Depth-optimal quantum circuit placement for arbitrary topologies," CoRR, vol. abs/1703.08540, 2017. [Online]. Available: http://arxiv.org/abs/1703.08540

[6] A. Shafaei, M. Saeedi, and M. Pedram, "Qubit placement to minimize communication overhead in 2D quantum architectures," in 19th Asia and South Pacific Design Automation Conference, ASP-DAC 2014, Singapore, January 20-23, 2014. IEEE, 2014, pp. 495–500. [Online]. Available: https://doi.org/10.1109/ASPDAC.2014.6742940

[7] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems, 2019, pp. 1015–1029. [Online]. Available: https://doi.org/10.1145/3297858.3304075

[8] M. Y. Siraichi, V. F. d. Santos, C. Collange, and F. M. Q. Pereira, "Qubit allocation as a combination of subgraph isomorphism and token swapping," Proceedings of the ACM on Programming Languages, vol. 3, no. OOPSLA, pp. 1–29, 2019. [Online]. Available: https://doi.org/10.1145/3360546

[9] B. Tan and J. Cong, "Optimal layout synthesis for quantum computing," in IEEE/ACM International Conference On Computer Aided Design, ICCAD 2020, San Diego, CA, USA, November 2-5, 2020. IEEE, 2020, pp. 137:1–137:9. [Online]. Available: https://doi.org/10.1145/3400302.3415620

[10] M. Y. Siraichi, V. F. d. Santos, C. Collange, and F. M. Q. Pereira, "Qubit allocation," in Proceedings of the 2018 International Symposium on Code Generation and Optimization, ser. CGO 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 113–125. [Online]. Available: https://doi.org/10.1145/3168822

[11] Y. Ji, S. Brandhofer, and I. Polian, "Calibration-aware transpilation for variational quantum optimization," in IEEE International Conference on Quantum Computing and Engineering, QCE 2022, Broomfield, CO, USA, September 18-23, 2022. IEEE, 2022, pp. 204–214. [Online]. Available: https://doi.org/10.1109/QCE53715.2022.00040

[12] A. Zulehner, A. Paler, and R. Wille, "An efficient methodology for mapping quantum circuits to the IBM QX architectures," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 7, pp. 1226–1236, 2018. [Online]. Available: https://doi.org/10.1109/TCAD.2018.2846658

[13] A. M. Childs, E. Schoute, and C. M. Unsal, "Circuit transformations for quantum architectures," CoRR, vol. abs/1902.09102, 2019. [Online]. Available: http://arxiv.org/abs/1902.09102

[14] Z.-T. Li, F.-X. Meng, Z.-C. Zhang, and X.-T. Yu, "Qubits' mapping and routing for NISQ on variability of quantum gates," Quantum Information Processing, vol. 19, no. 10, pp. 1–25, 2020. [Online]. Available: https://doi.org/10.1007/s11128-020-02873-5

[15] S. Niu, A. Suau, G. Staffelbach, and A. Todri-Sanial, "A hardware-aware heuristic for the qubit mapping problem in the NISQ era," IEEE Transactions on Quantum Engineering, vol. 1, pp. 1–14, 2020. [Online]. Available: https://doi.org/10.1109/TQE.2020.3026544

[16] M. P. Harrigan, K. J. Sung, M. Neeley, K. J. Satzinger, F. Arute, K. Arya, J. Atalaya, J. C. Bardin, R. Barends, S. Boixo et al., "Quantum approximate optimization of non-planar graph problems on a planar superconducting processor," Nature Physics, vol. 17, no. 3, pp. 332–336, 2021. [Online]. Available: https://doi.org/10.1038/s41567-020-01105-y

[17] I. D. Kivlichan, J. McClean, N. Wiebe, C. Gidney, A. Aspuru-Guzik, G. K.-L. Chan, and R. Babbush, "Quantum simulation of electronic structure with linear depth and connectivity," Physical review letters, vol. 120, no. 11, p. 110501, 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.120.110501

[18] B. O'Gorman, W. J. Huggins, E. G. Rieffel, and K. B. Whaley, "Generalized swap networks for near-term quantum computing," arXiv preprint arXiv:1905.05118, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1905.05118

[19] J. Weidenfeller, L. C. Valor, J. Gacon, C. Tornow, L. Bello, S. Woerner, and D. J. Egger, "Scaling of the quantum approximate optimization algorithm on superconducting qubit based hardware," arXiv preprint arXiv:2202.03459, 2022. [Online]. Available: https://doi.org/10.22331/q-2022-12-07-870

[20] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," arXiv preprint arXiv:1411.4028, 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1411.4028

[21] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," Physical Review X, vol. 10, no. 2, p. 021067, 2020. [Online]. Available: https://doi.org/10.1103/PhysRevX.10.021067

[22] M. Willsch, D. Willsch, F. Jin, H. De Raedt, and K. Michielsen, "Benchmarking the quantum approximate optimization algorithm," Quantum Information Processing, vol. 19, no. 7, pp. 1–24, 2020. [Online]. Available: https://doi.org/10.1007/s11128-020-02692-8

[23] S. Brandhofer, D. Braun, V. Dehn, G. Hellstern, M. Hüls, Y. Ji, I. Polian, A. S. Bhatia, and T. Wellens, "Benchmarking the performance of portfolio optimization with qaoa," Quantum Information Processing, vol. 22, no. 1, pp. 1–27, 2023. [Online]. Available: https://doi.org/10.1007/s11128-022-03766-5

[24] IBM. 2022., "IBM Qiskit," https://qiskit.org/, Accessed: 2022-10-31.

[25] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "t|ket⟩: a retargetable compiler for nisq devices," Quantum Science and Technology, 2020. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/ab8e92

[26] mapomatic, https://github.com/Qiskit-Partners/mapomatic, Accessed: 2022-10-31.

[27] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, "A variational eigenvalue solver on a photonic quantum processor," Nature communications, vol. 5, no. 1, pp. 1–7, 2014. [Online]. Available: https://doi.org/10.1038/ncomms5213

[28] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth et al., "The variational quantum eigensolver: a review of methods and best practices," Physics Reports, vol. 986, pp. 1–128, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0370157322003118

[29] D. A. Fedorov, B. Peng, N. Govind, and Y. Alexeev, "Vqe method: A short survey and recent developments," Materials Theory, vol. 6, no. 1, pp. 1–21, 2022. [Online]. Available: https://doi.org/10.1186/s41313-021-00032-6

[30] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. Sawaya et al., "Quantum chemistry in the age of quantum computing," Chemical reviews, vol. 119, no. 19, pp. 10 856–10 915, 2019. [Online]. Available: https://doi.org/10.1021/acs.chemrev.8b00803

[31] B. Bauer, S. Bravyi, M. Motta, and G. K.-L. Chan, "Quantum algorithms for quantum chemistry and quantum materials science," Chemical Reviews, vol. 120, no. 22, pp. 12 685–12 717, 2020. [Online]. Available: https://doi.org/10.1021/acs.chemrev.9b00829

[32] D. Amaro, C. Modica, M. Rosenkranz, M. Fiorentini, M. Benedetti, and M. Lubasch, "Filtering variational quantum algorithms for combinatorial optimization," Quantum Science and Technology, vol. 7, no. 1, p. 015021, 2022. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/ac3e54

[33] J. Romero, R. Babbush, J. R. McClean, C. Hempel, P. J. Love, and A. Aspuru-Guzik, "Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz," Quantum Science and Technology, vol. 4, no. 1, p. 014008, 2018. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/aad3e4

[34] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," Nature, vol. 549, no. 7671, pp. 242–246, 2017. [Online]. Available: https://doi.org/10.1038/nature23879

[35] D. Wecker, M. B. Hastings, and M. Troyer, "Progress towards practical quantum variational algorithms," Physical Review A, vol. 92, no. 4, p. 042303, 2015. [Online]. Available: https://doi.org/10.1103/PhysRevA.92.042303

[36] G. Nannicini, "Performance of hybrid quantum-classical variational heuristics for combinatorial optimization," Phys. Rev. E, vol. 99, p. 013304, Jan 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.99.013304