Noise Resilient Compilation Policies for Quantum Approximate Optimization Algorithm

Invited Talk

Mahabubul Alam[†], Abdullah Ash-Saki[†], Junde Li[†], Anupam Chattopadhyay[*], Swaroop Ghosh[†]
[†]School of Electrical Engineering and Computer Science, Pennsylvania State University, University Park, PA, USA
[*]School of Computer Science and Engineering, Nanyang Technological University, Singapore
{mxa890,axs1251,jul1512}@psu.edu,anupam@ntu.edu.sg,szg212@psu.edu

ABSTRACT

Quantum approximate optimization algorithm (QAOA) is a promising quantum-classical hybrid algorithm to solve hard combinatorial optimization problems using noisy quantum devices. The multi-qubit CPHASE gates used in the quantum circuit for QAOA are commutative i.e., the order of the gates can be altered without changing the output state. This re-ordering leads to the execution of more gates in parallel and a smaller number of additional SWAP gates to compile the QAOA circuit resulting in lower circuit-depth and gate-count. A less number of gates generally indicates a lower accumulation of gate-errors, and a reduced circuit-depth means less decoherence time for the qubits. However, near-term quantum devices exhibit significant variations in the gate success probabilities. Variation-aware compilation policies (i.e. putting most gate operations on qubits with higher gate success probabilities) can enhance the probability of successful program execution on the hardware. The greater flexibility of QAOA-circuits offer better scope of optimization with QAOA-tailored compilation policies. This paper presents an argument for compilation policies to exploit the unique characteristics of QAOA-circuits alongside the variation-awareness of the noisy devices. We present two procedures - variation-aware qubit placement (VQP) and variation-aware iterative mapping (VIM) that can improve the circuit success probability quite significantly (≈8.408X on average) for a set of QAOA-MaxCut problems on ibmq_16_melbourne.

ACM Reference Format:
Mahabubul Alam[†], Abdullah Ash-Saki[†], Junde Li[†], Anupam Chattopadhyay[*], Swaroop Ghosh[†]. 2020. Noise Resilient Compilation Policies for Quantum Approximate Optimization Algorithm: *Invited Talk*. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '20), November 2–5, 2020, Virtual Event, USA*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3400302.3415745

1 INTRODUCTION

Quantum computing is getting traction with the newly demonstrated quantum supremacy by Google [6]. Prototypical quantum computers with 5-128 qubits are available or proposed [20–22, 29] in the past few years from industry vendors like IBM, Google, Rigetti, etc. However, the interest in quantum computing can wane and the progress can halt without a quantum application of significant economic potential in the near-term [28]. Quantum Approximate Optimization Algorithm (QAOA) [15–17] has been touted as a prime candidate for early demonstration of quantum advantage [18]. It is particularly useful to solve NP-hard combinatorial optimization problems using noisy quantum devices without error-correction [2, 11, 36]. However, the perceived quantum advantage through QAOA may be lost due to the accumulation of gate errors and decoherence [4]. An optimized circuit in terms of gate-count/depth/success probability can show greater resilience to noise, and enhance the probability of generating the correct quantum state [7, 9, 25, 32]. *This makes QAOA circuit optimization an important problem in the NISQ era [28].*

The detailed theoretic discussion on QAOA can be found in other literature [11, 15–18, 36]. QAOA involves parameter optimization of a multi-level parameterized quantum circuit (PQC) that runs in a quantum-classical hybrid optimization loop to minimize (or maximize) the expectation value of a classical cost function. QAOA performance improves with added levels in the PQC. The total number of levels is referred to as 'p'. However, each level adds additional two parameters (γ, β) to the PQC which may affect the convergence and the speed of the algorithm [2, 36].

The PQC to solve the maximum cut (MaxCut) problem of a 4-node 3-regular graph (Figure 1(a)) with QAOA is shown in Figure 1(b) ('p' = 1). Note that, the PQC has an associated CPHASE operation in every level of the circuit for every edge in the problem graph for the MaxCut problem [8, 34]. CPHASE is a two-qubit unitary parametric quantum gate operating between a control and a target qubit. Also, note that two consecutive gates can be executed concurrently if they operate on a different set of qubits. For example, the first two CPHASE operations in the circuit in Figure 1(b) can not be executed concurrently as they share a logical qubit (q2).

The CPHASE operation in a QAOA circuit are commutative [11, 34] i.e., the order of these CPHASE gates can be interchanged without affecting the output state of the quantum circuit. We can use this knowledge to maximize concurrent gate operations by choosing an optimal order of the gates. Figure 1(b) shows the QAOA-MaxCut circuit instance with randomly ordered CPHASE operations for the problem graph in Figure 1(a) (circ-1). Figure 1(c) shows a gate re-ordered circuit (circ-2). Note that, if these circuits are executed

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICCAD '20, November 2–5, 2020, Virtual Event, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8026-3/20/11…$15.00
https://doi.org/10.1145/3400302.3415745

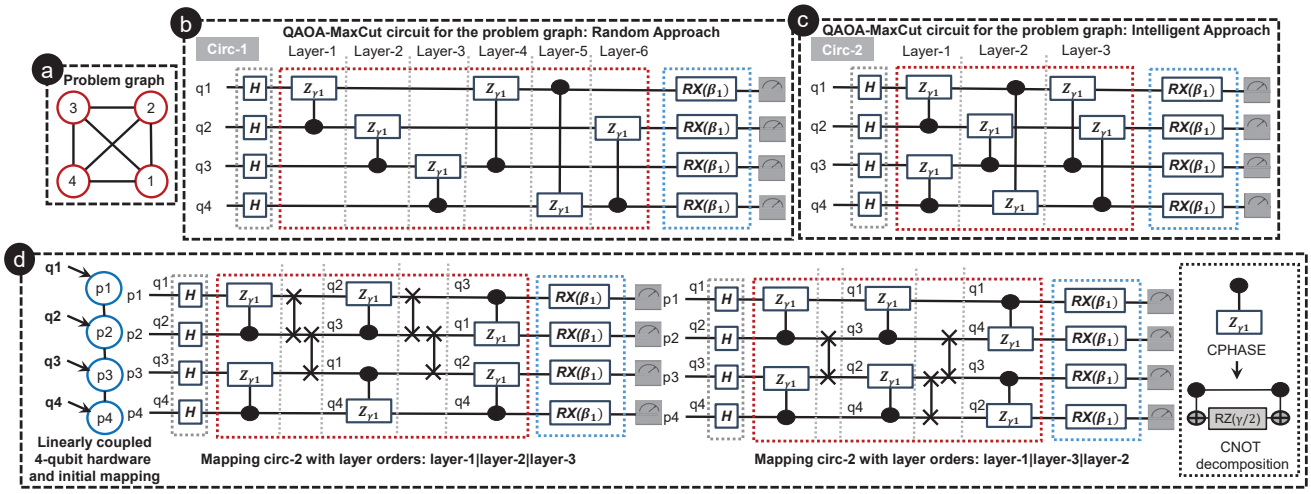


Figure 1: (a) A 4-node 3-Regular graph, (b) a randomly constructed QAOA-MaxCut instance (circ-1) of the 4-node graph with p = 1, (c) an optimized circuit (circ-2) for the problem with reduced number of layers, (d) SWAP addition during circuit compilation for a target hardware with different layer orders.

in quantum hardware with full qubit-to-qubit connectivity supporting the following basis gates: H, RX, and CPHASE, circ-1 will require 9 time steps while circ-2 will take 6 time steps (including the measurement operations). Therefore, circ-2 will be 50% faster and will experience less decoherence. Re-ordering these layers of CPHASE gates (e.g. interchanging layer-2 and layer-3 in circ-2) will not reduce the circuit cumulative execution time.

However, if we consider target hardware with limited connectivity such as the 4 linearly coupled physical qubits (p1, p2, p3, p4) in Figure 1(d), there will be further scope of optimization in circ-2. For such architectures, SWAP gates are added between two layers to meet the hardware constraints [12, 37]. For the initial logical-to-physical qubit assignment (choice of hardware qubits to execute the quantum program) shown in Figure 1(d), interchanging the CPHASE layer 2 and 3 (in circ-2) will reduce the additional SWAP operations from 4 to 3. Therefore, the CPHASE gates that are picked for different layers will affect the quality of the compiled circuit for such target architectures.

The work in [5, 34] utilized these flexibilities in QAOA-circuits to compile quantum circuits with reduced gate-count and depth. In [34], the authors formulated the QAOA-circuit compilation problem as a planning problem and used off-the-shelf temporal planners for compilation with an objective to minimize the circuit makespan or depth. The work in [5] proposed two sub-routines namely min-layer formation and iterative compilation that re-compiles the QAOA-circuit with re-ordered gates to generate a circuit with reduced gate-count or depth. The iterative compilation procedure is guided by a branch and bound optimization heuristic [23]. However, none of these works did take qubit-to-qubit variability into consideration which can have significant impact on the probability of successful execution of the program on the hardware [1, 32]. Additionally, they did not consider optimization of the initial logical-to-physical qubit placement which can also influence the quality of the compiled circuits [3].

To elucidate on the fact, let us consider a hypothetical scenario where the CNOT gate success probabilities between physical qubit pairs p1-p2, p2-p3, and p3-p4 in Figure 1(d) are 0.98, 0.96, and 0.92 respectively. A SWAP gate can be decomposed to three consecutive CNOT operations [37] and every CPHASE operation can be implemented with two CNOT operations [11]. The success probabilities of circ-2 with layer-1|layer-2|layer-3 and circ-2 with layer-1|layer-3|layer-2 become $0.98^{12} * 0.96^6 * 0.92^6$ or 0.3724 and $0.98^6 * 0.96^6 * 0.92^9$ or 0.3274 respectively (considering no error in single-qubit operations). Note that, despite having larger number of gates, circ-2 with layer-1|layer-2|layer-3 can be executed more reliably than circ-2 with layer-1|layer-3|layer-2. A different initial qubit mapping for circ-2 with layer-1|layer-3|layer-2 (q1→p4, q2→p3, q3→p2, q4→p1) can increase the probability to $0.98^9 * 0.96^6 * 0.92^6$ or 0.3957. Therefore, in the NISQ-era where the available quantum devices are plagued with various noise sources, it is more appropriate to enhance the circuit reliability through qubit-to-qubit variation-aware compilation policies rather than blindly minimizing circuit cumulative gate-count or depth [7, 25, 32].

In this article, we make the following contributions: we, (i) develop variation-aware compilation policies that are tailored for QAOA-circuits to exploit its unique characteristics, (ii) present a variation-aware qubit placement procedure (VQP) for QAOA-circuits, and, (iii) present a branch and bound optimization heuristic to enhance QAOA-circuit successful execution probability through a variation-aware iterative mapping (VIM) procedure.

2 QUANTUM COMPUTING PRELIMINARIES

Qubits and Quantum gates: Qubit is analogous to classical bits however, a qubit can be in a superposition state i.e., a combination of 0 and 1 at the same time. Quantum gates such as single qubit (e.g., Pauli-X (σ_x) gate) or multiple qubit (e.g., 2-qubit CNOT gate) gates modulate the state of qubits and thus perform computations. In superconducting transmon qubits, any gate operation is executed in the actual hardware using pre-compiled microwave pulses [8].
Gate Error, Decoherence and Crosstalk: Quantum gates are error-prone. Besides, the qubits suffer from decoherence i.e., the qubits spontaneously interact with the environment and lose states. Therefore, the output of a quantum circuit can be erroneous. The deeper quantum circuit needs more time to execute and gets affected

more by decoherence. More gates in the circuit also increase the accumulation of gate errors. Thus, lower depth and number of gates in the circuit improves noise resiliency. Parallel gate operations on different qubits can reduce the circuit execution time. However, such operations may affect each others performance which is referred to as crosstalk [27].

Success Probability: The success probability of a gate is the conjugate of the error-rate ($1 - error$). The success probability of a circuit (referred to as circuit-success-probability or CSP throughout this paper) is defined as the product of the success probabilities of individual gates [32].

Basis Gates and Coupling Constraints: A practical quantum computer supports a limited number of single and multi-qubit gates known as basis (or native) gates of the hardware. IBM quantum computers offer single-qubit {U1, U2, U3, ID} and two-qubit CNOT gate as basis gates. However, the quantum circuit may contain non-native gates to the target hardware e.g., the CPHASE gate that need to be decomposed into the basis gates before execution [31]. A CNOT decomposition of a CPHASE gate is shown in Figure 1(d). The native two-qubit gate may or may not be permitted between all the two-qubit pairs in the target hardware. These limitations are also known as coupling constraints. Conventional compilers add necessary SWAP gates to meet these constraints [26, 35, 37].

3 PROPOSED METHODOLOGIES

We present two noise resilient compilation policies - VQP and VIM - which can be incorporated in any conventional quantum circuit compiler to improve the noise resilience of QAOA-circuits. VQP is an intelligent qubit allocation and initial mapping approach which applies to any circuit compilation. VIM manipulates the flexibility of QAOA-circuits to improve circuit noise resilience using any standard backend quantum circuit compiler (i.e. Qiskit [12]). The workflow to incorporate VQP and VIM in QAOA-circuit compilation is shown in Figure 2. Starting with a QAOA problem instance, target hardware coupling graph and hardware calibration data, VQP generates an initial logical-to-physical qubit mapping that is passed to VIM which in turn, uses a backend compiler to generate the hardware compliant circuit. The details of the individual procedures are discussed in this section.

3.1 Variation-aware Qubit Placement (VQP)

Variation-awareness in qubit placement can improve circuit noise resilience [10, 25, 32]. Qiskit uses a heuristic called greedyE* [25] where program CNOTs, and their control and target qubits are placed in a heaviest edge first order (maximum CNOT operations between two logical qubits). Note that, each qubit may interact with another qubit only once within a level (either 1 CPHASE or no CPHASE) in QAOA circuits. The number of CNOT operations between any two-interacting qubits is a constant. Therefore, GreedyE* which prioritizes qubit pair placements with maximum interactions may not offer much performance benefits for QAOA.

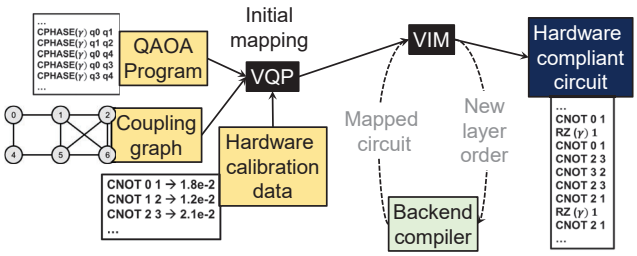


Figure 2: A generic workflow incorporating the proposed compilation methodologies on top of a traditional compiler backend.

Qubit placement procedure for QAOA circuits should consider, (i) placing logical qubits with higher number of CPHASE operations to the physical qubits with higher density of reliable two-qubit links (considering two-qubit gate errors as the dominating error source in the NISQ devices [32]) so that more operations can be executed with higher reliability, and (ii) minimizing initial placement distances (distance is the shortest path length between two hardware qubits in the coupling graph) between the logically neighboring qubits (program qubits that interact with each other) so that the need for qubit movement through SWAP insertion is reduced. The proposed variation-aware qubit placement (VQP) aims to achieve these two objectives. VQP utilizes a hardware and a program profiling statistics. We first discuss the profiling methods and later, we discuss the steps in the VQP procedure.

Hardware Profiling: Hardware profiling is done based on the connectivity strength of the physical qubits [32]. A physical qubit connectivity strength is defined as the summation of the success rate of all the two-qubit links with its neighbors. The success rate of all the CNOT links in ibmq_16_melbourne after calibration on a certain day is shown in Figure 3(a). The qubit connectivity strengths are shown in Figure 3(b). For example, qubit-0 has two CNOT links with success rates of 0.976 and 0.95242. Therefore, the connectivity strength of qubit-0 is 1.92842.

Program Profiling: The program profile used in VQP is similar to the one used in [25]. For any input QAOA-circuit, we calculate the number of CPHASE operations per logical qubit to create the program profile. A demonstrative example is shown in Figure 3(c).

VQP Procedure: Starting with the list of CPHASE operations in a QAOA-circuit, target hardware, and program profiling statistics, VQP adopts following steps:

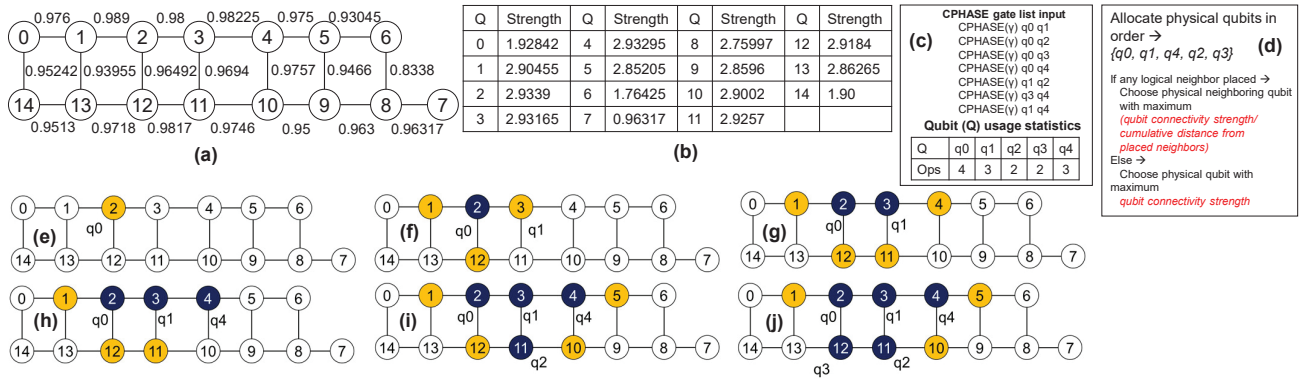
Step-1: The logical qubits are sorted (descending order) in a list based on the number of CPHASE operations per qubit. Physical qubits are allocated to the logical qubits in this order.

Step-2: The first logical qubit is assigned to the physical qubit with the highest connectivity strength. After the assignment, the qubit is removed from the list.

Step-3: For the next logical qubit in the list, we check if any of its logical neighbors has been already placed. If none of them has been placed, we pick the unallocated physical qubit with the highest connectivity strength for allocation. If some of its logical neighbors are placed, we find the unallocated physical neighbors of these placed qubits. We pick a qubit from these neighbors maximizing the cost metric - qubit connectivity strength/cumulative distance from the placed neighbors. Distances between physical qubits can be measured once (using Floyd-Warshall algorithm [24]) and accessed from memory during VQP. After the assignment, we remove the logical qubit from the list.

Step-4: We repeat *Step–3* until the list is empty.

Example 1: The VQP procedure for the QAOA-circuit in Figure 3(c) is shown in Figure 3(d) and (e)-(j). Logical qubit 'q0' is



Q	Strength	Q	Strength	Q	Strength	Q	Strength
0	1.92842	4	2.93295	8	2.75997	12	2.9184
1	2.90455	5	2.85205	9	2.8596	13	2.86265
2	2.9339	6	1.76425	10	2.9002	14	1.90
3	2.93165	7	0.96317	11	2.9257		

Q	q0	q1	q2	q3	q4
Ops	4	3	2	2	3

Figure 3: (a) Coupling graph of a 15-qubit quantum computer from IBM (ibmq_16_melbourne), (b) connectivity strength metrics of different qubits in ibmq_16_melbourne, (c) a toy QAOA cost Hamiltonian circuit with qubit activity profiles, (d) VQP decision metric, and (e)-(j) qubit allocation and initial mapping for the toy example on using VQP.

placed to hardware qubit-2 as it has the highest qubit connectivity strength of 2.9339 (Figure 3(e)). Logical qubit 'q1' has 3 possible candidates (as it is a logical neighbor of 'q0'), all at a distance of 1 from 'q0' (Figure 3(f)). Physical qubit-3 has the highest connectivity strength/cumulative distance from the placed neighbors ('q0'). Therefore, qubit-3 is chosen for 'q1'. The other qubits 'q4','q2','q3' are placed to qubit-4, qubit-11, and qubit-12 respectively.

3.2 Variation-aware Iterative Mapping (VIM)

VIM follows an iterative compilation procedure similar to the work in [5] . It utilizes a backend compiler to compile the circuit with re-ordered gates. However, instead of trying to minimize the depth or cumulative gate-count of the compiled circuit as presented in [5], VIM tries to maximize the circuit success probability (CSP).

VIM has a pre-processing step (Instruction Parallelization) to parallelize the CPHASE operations of any given QAOA-circuit. Previously, similar pre-processing step was presented in [5]. The work in [5] formulated the problem as an instance of a bin-packing problem and used the first-fit heuristic to find the solution [14]. In this work, we replace the first-fit heuristic with first-fit decreasing heuristic as it generally provides more effective solutions [13]. The details of the procedures are discussed below.

Instruction Parallelization: The logical qubits in the given QAOA-circuit are sorted based on the number of CPHASE operations that involves a certain qubit (decreasing order). Next, a sorted list of CPHASE operations is created by putting the CPHASE operations involving the highest ranked qubit first order. The rest of the procedure is identical to the layer-formation routine presented in [5].

Example-2: A demonstrative example is shown in Figure 4. The example problem QAOA instance has 6 CPHASE gates between the logical qubit pairs {(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)}. Here, all the logical qubits are involved in exactly the same number of CPHASE operations (3 each). Such a tie can be broken by random selection. In this toy example, we sort the logical qubits in the following order: {1, 2, 3, 4}. Next, we place the CPHASE operations involving qubit 1 first. Then, we place the remaining operations involving qubit 2, 3, and 4 - in that order. A total of 3-layers are constructed through the procedure (Figure 4(b)). Each iteration in the procedure is shown in Figure 4(a). The constructed circuit after the layer formation procedure is shown in Figure 1(c). A randomly generated circuit for the same problem required 6 layers (Figure 1(b)).

Note that, the minimum number of layers where we can fit all the CPHASE operations is the highest number of CPHASE operations involving a logical qubit (3 in the toy example). The first-fit decreasing heuristic does not guarantee the optimal solution.

VIM Procedure: VIM starts with the layer-order produced during the pre-processing step (we call it the *root* order) and update the order iteratively. In each iteration, VIM considers exchanging the order of two-layers at a time for a given QAOA circuit instance. Each of these exchanges will produce a distinct order of the layers. For a given problem with 'n' layers, we will have 'n(n-1)/2' possibilities to explore in each iteration which is far smaller than 'n!' (total number of possible layer orders) for larger values of 'n'. VIM compiles the circuit for these layer orders in the current iteration and picks the order that provides the best improvement in CSP. We set the picked order as the new *root* order and move to the next iteration. We terminate the procedure until there is no further improvement in CSP between consecutive iterations.

Example-3: An example of the VIM procedure is shown in Figure 5. Starting with a hypothetical QAOA instance with 4 CPHASE layers (L1||L2||L3||L4), we compile the circuit and find the CSP to be 0.80. In the current iteration, we consider all possible two-layer interchanges (a total of 6) and compile the circuit for the generated layer orders. Interchanging L1 with L4 results in a maximum improvement in CSP i.e., increased to 0.84 from 0.8. Hence, we pick this layer order as the *root* for the next iteration. In iteration-2, none of the layer orders produced by the two-layer interchange approach gives any gain. Hence, the procedure terminates.

4 EVALUATION OF THE PROPOSED METHODOLOGIES

In this section, we first discuss the metrics and framework that is used for performance evaluation. Later, we present the comparative results and discuss the pros and cons of the proposed solutions.

Evaluation Metrics: We use circuit depth, cumulative gate-count, compilation time, and circuit success probability (CSP) metrics to compare various compilation strategies [5, 24, 37]. A lower depth and gate-count can be helpful to mitigate the impact of decoherence and gate-errors. Compilation time is the time taken by the compiler

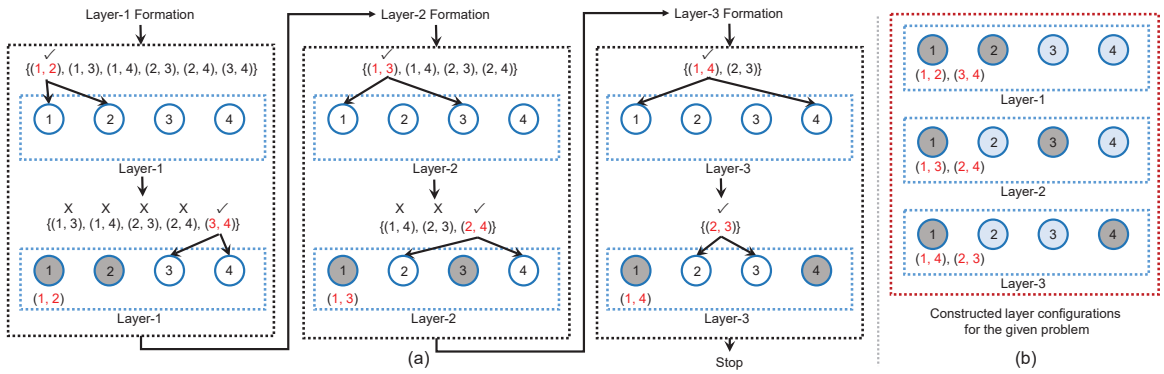


Figure 4: (a) The intermediate steps of the proposed layer formation procedure for an example problem (a pair (1,2) denotes a required CPHASE operation between qubit-1 and 2 for the given QAOA problem instance), (b) the constructed layers.

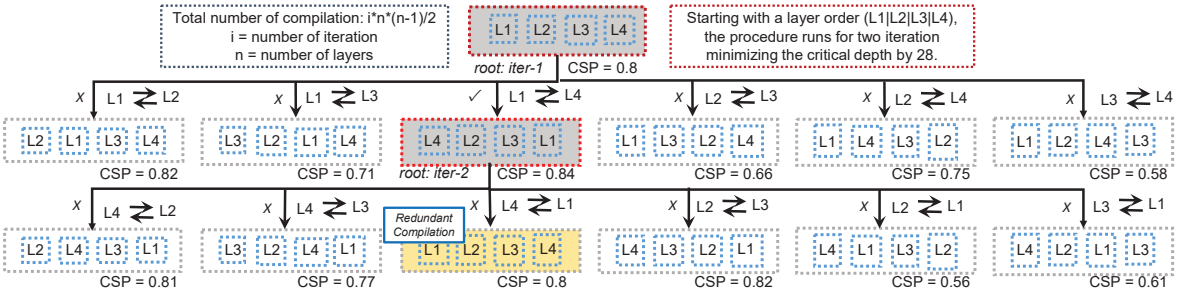


Figure 5: Hypothetical iteration steps of a VIM procedure to maximize the circuit success probability (CSP) of a QAOA-MaxCut circuit instance with 4 layers of CPHASE operations.

to generate the hardware compliant circuit. A faster compilation is desired for scalability. The CSP metric is useful to quantify performance benefits with variation-aware compilation policies [32].

Compiler Backend, Target Hardware, and Problem Sets: We use Qiskit as the backend circuit compiler (run on an Intel Core i-7 processor at 3.41 GHz frequency). We use a Python-based implementation of VQP and VIM procedures which has been integrated with the Qiskit backend. We choose 15-qubit ibmq_16_melbourne as the target hardware. Randomly chosen 15-nodes erdos-renyi random graphs (with varied edge probabilities) are used for the validation purpose inspired from recent works on QAOA [11, 36]. An edge probability of 0.5 means an edge between any two nodes in the graph is 50% likely to be included in a random sample. The edge probabilities are varied between 0.3, 0.4, 0.5, 0.6 and 0.7. We randomly generated 40 graph instances with each of these edge probabilities. A total of 200 graph MaxCut problems have been used for the evaluation purpose. Graphs with higher edge probabilities are dense graphs that require many CPHASE gates in their corresponding QAOA-MaxCut circuits. All the QAOA-circuits are compiled with the minimum QAOA-level ($p = 1$).

VQP vs. GreedyE*: To compare VQP with GreedyE*, we have compiled a set of QAOA-MaxCut circuits for erdos-renyi random graphs with these two initial placement methods where the CPHASE gate sequence is chosen randomly for each of the QAOA-circuit instances. The mean circuit cumulative gate-count, depth, and CSP ratios between VQP and GreedyE* for different edge probabilities are shown in Figure 6(a), (b), and (c) respectively.

Note that, the mean value of circuit gate-count and depth ratios - both were found to be higher for VQP compared to GreedyE* as evident from Figure 6(a), and (b). VQP produced circuits with 5.69% larger depth and 8.85% larger gate-count compared to GreedyE*. However, VQP generated circuits with significantly higher CSP compared to GreedyE* as evident from Figure 6(c). On average, VQP generated circuits with 2.38X larger CSP than GreedyE*. The performance improvement was less pronounced for sparser graphs. For instance, for the problems with 0.3 and 0.4 edge probabilities, VQP was found to be 1.575X better. For denser graphs (i.e. edge probabilities of 0.5 and 0.6), VQP was 3.585X better. However, for even denser graphs (i.e. edge probability of 0.7), the improvement was smaller (1.58X). The reason is the higher number of logical neighbors of a logical qubit for dense graphs than the number of physical neighbors of any hardware qubit. For any qubit placement, some of the logical neighbors will remain far from a logical qubit placed in a physical qubit. Therefore, we do not note large performance improvement with VQP. Note that, VQP incurred a small (≈16.52%) compilation time penalty over the GreedyE* heuristic for the 200 chosen MaxCut instances.

VIM+VQP vs. GreedyE*: Later, we compare the performance between VIM+VQP with the GreedyE* heuristic (+random CPHASE sequence). The mean compiled circuit gate-count, depth, and CSP ratios between these two approaches are shown in Figure 7(a), (b), and (c) respectively. The mean cumulative gate-count was found to be quite similar. The average value of the gate-count ratio was found to be 0.999 for all 200 graphs. The value remained close to ≈1 for different edge probabilities (Figure 7(a)). VIM+VQP performed better than GreedyE* for sparse graphs and worse for dense graphs.

The depth was found to be significantly smaller in VIM+VQP (≈20.38% on average). Note that, the pre-processing step used in VIM+VQP increased parallel gate operation which is reflected in the circuit depth.

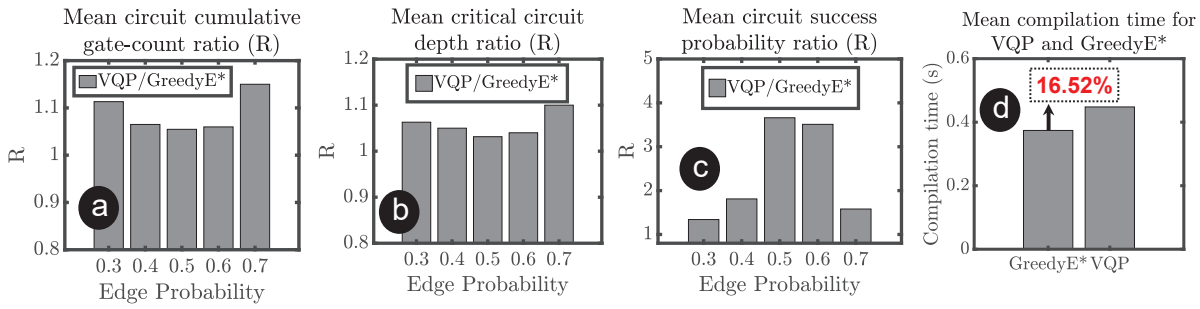


Figure 6: The ratio between mean compiled circuit (a) gate-count, (b) depth, (c) circuit success probability, and (d) compilation time of the proposed VQP and GreedyE* (+random CPHASE sequence) with qiskit compiler backend for a set of erdos-renyi random graphs with varying edge probabilities (40 random instances of 15-node QAOA-MaxCut circuits used for each bars).

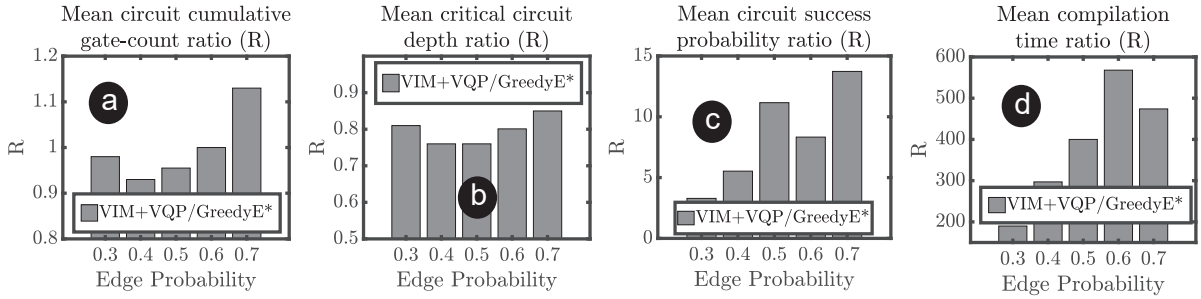


Figure 7: The ratio between mean compiled circuit (a) gate-count, (b) depth, (c) circuit success probability, and (d) compilation time of the proposed VIM+VQP and GreedyE* (+random CPHASE sequence) methods with qiskit compiler backend for a set of erdos-renyi random graphs (40 random instances of 15-node QAOA-MaxCut circuits used for each bars).

The major benefit is observed in CSP. VIM+VQP generated circuits with 8.408X CSP compared to GreedyE*. The performance improvement was more pronounced for denser graphs as evident from Figure 7(c). For instance, VIM+VQP generated circuits with 4.41X higher CSP with edge probabilities 0.3 and 0.4 on average. For even higher edge probabilities, the CSP was 11.07X higher.

However, the benefits in CSP comes at a cost of high compilation time penalty as evident from Figure 7(d). This is expected as VIM re-compiles the complete circuit many times with re-ordered gates. The VIM+VQP compilation time was found to be 190X-568X to GreedyE* (≈385.8X on average). The overhead was higher for denser graphs. For denser graphs, the corresponding QAOA-MaxCut circuit has large number of CPHASE operations. Therefore, the number of circuit layers is also higher for denser graphs. With large number of layers, the backend compiler takes more time for each circuit compilation. Moreover, VIM needs to compile more number of circuits in each iteration. The branch and bound heuristic also runs longer which translates to high compilation time.

5 DISCUSSION

Applicability beyond QAOA-MaxCut: The proposed compilation methodologies can be applied to other classes of QAOA instances. VQP can be useful for arbitrary quantum circuits to a varied extent. VIM can be useful for quantum circuits with large number of commuting operators such as the UCCSD ansatz for VQE [19, 30].
Compiling higher-depth ($p > 1$) QAOA-circuits: Note that, each level of a multi-level QAOA-circuit has the same set of CPHASE operations with separate rotation parameters [2, 4, 5, 11, 15, 36]. A larger circuit generally takes higher compilation time [5] which can be avoided by compiling QAOA-circuit with the lowest depth ($p = 1$) and repeating the compiled circuit with different rotation parameters to extend it to a multi-level QAOA-circuit.
VIM solution quality: Note that, the branch and bound heuristic used in VIM does not offer any provable performance guarantee. More rigorous optimization algorithm (i.e. simulated annealing [33]) can replace the proposed heuristic to generate better quality solutions utilizing the same concept of layer re-ordering.

6 CONCLUSION

We proposed QAOA-tailored circuit compilation policies that exploit the unique characteristics of QAOA-circuits alongside the hardware variation-awareness to maximize the performance from noisy quantum devices. We present two variation-aware compilation procedures - VQP and VIM that can be integrated in any conventional quantum circuit compiler. We demonstrate ≈8.408X performance improvement on average over the state-of-the-art compilation approach for a set of QAOA-MaxCut problems. We only considered variations in two-qubit gate success probabilities. A more comprehensive solution should consider all the error sources together for optimal compilation.

ACKNOWLEDGEMENTS

The work is supported in parts by National Science Foundation (NSF) (CNS- 1722557, CCF-1718474, DGE-1723687 and DGE-1821766) and seed grants from Penn State Institute for Computational and Data Sciences and Penn State Huck Institute of the Life Sciences.

REFERENCES

[1] Mahabubul Alam, Abdullah Ash-Saki, and Swaroop Ghosh. 2019. Addressing Temporal Variations in Qubit Quality Metrics for Parameterized Quantum Circuits. In *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 1–6.

[2] M. Alam, A. Ash-Saki, and S. Ghosh. 2020. Accelerating Quantum Approximate Optimization Algorithm using Machine Learning. In *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*. 686–689.

[3] Mahabubul Alam, Abdullah Ash-Saki, and Swaroop Ghosh. 2020. Circuit Compilation Methodologies for Quantum Approximate Optimization Algorithm. *IEEE/ACM International Symposium on Microarchitecture* (2020).

[4] M. Alam, A. Ash-Saki, and S. Ghosh. 2020. Design-Space Exploration of Quantum Approximate Optimization Algorithm under Noise. In *2020 IEEE Custom Integrated Circuits Conference (CICC)*. 1–4.

[5] Mahabubul Alam, Abdullah Ash-Saki, and Swaroop Ghosh. 2020. An Efficient Circuit Compilation Flow for Quantum Approximate Optimization Algorithm. *IEEE/ACM Design Automation Conference* (2020).

[6] Frank Arute, Kunal Arya, Ryan Babbush, Bacon, et al. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510.

[7] Abdullah Ash-Saki, Mahabubul Alam, and Swaroop Ghosh. 2019. QURE: Qubit Re-allocation in Noisy Intermediate-Scale Quantum Computers. In *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 141.

[8] George S Barron, FA Calderon-Vargas, Junling Long, David P Pappas, and Sophia E Economou. 2020. Microwave-based arbitrary cphase gates for transmon qubits. *Physical Review B* 101, 5 (2020), 054508.

[9] Debjyoti Bhattacharjee and Anupam Chattopadhyay. 2017. Depth-optimal quantum circuit placement for arbitrary topologies. *arXiv preprint arXiv:1703.08540* (2017).

[10] Debjyoti Bhattacharjee, Abdullah Ash Saki, Mahabubul Alam, Anupam Chattopadhyay, and Swaroop Ghosh. 2019. MUQUT: Multi-constraint quantum circuit mapping on NISQ computers. In *38th IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2019*. Institute of Electrical and Electronics Engineers Inc., 8942132.

[11] Gavin E Crooks. 2018. Performance of the quantum approximate optimization algorithm on the maximum cut problem. *arXiv preprint arXiv:1811.08419* (2018).

[12] Andrew Cross. 2018. The IBM Q experience and QISKit open-source quantum computing software. In *APS Meeting Abstracts*.

[13] György Dósa. 2007. The tight bound of first fit decreasing bin-packing algorithm is FFD (I) 11/9OPT (I)+ 6/9. In *International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Springer, 1–11.

[14] György Dósa and Jirí Sgall. 2013. First Fit bin packing: A tight analysis. In *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[15] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028* (2014).

[16] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Hartmut Neven. 2017. Quantum algorithms for fixed qubit architectures. *arXiv preprint arXiv:1703.06199* (2017).

[17] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Leo Zhou. 2019. The Quantum Approximate Optimization Algorithm and the Sherrington-Kirkpatrick Model at Infinite Size. *arXiv preprint arXiv:1910.08187* (2019).

[18] Edward Farhi and Aram W Harrow. [n.d.]. Quantum supremacy through the quantum approximate optimization algorithm. *arXiv preprint arXiv:1602.07674* ([n. d.]).

[19] Harper R Grimsley, Daniel Claudino, Sophia E Economou, Edwin Barnes, and Nicholas J Mayhall. 2019. Is the Trotterized UCCSD Ansatz Chemically Well-Defined? *Journal of Chemical Theory and Computation* (2019).

[20] Jeremy Hsu. 2018. CES 2018: Intel's 49-qubit chip shoots for quantum supremacy. *IEEE Spectrum Tech Talk* (2018).

[21] Julian Kelly. 2018. A preview of Bristlecone, Google's new quantum processor. *Google Research Blog* 5 (2018).

[22] Will Knight. 2018. IBM Raises the Bar with a 50-Qubit Quantum Computer, News.

[23] Eugene L Lawler and David E Wood. 1966. Branch-and-bound methods: A survey. *Operations research* 14, 4 (1966), 699–719.

[24] Gushu Li, Yufei Ding, and Yuan Xie. 2019. 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*. 1001–1014.

[25] Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. 2019. 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*. 1015–1029.

[26] Prakash Murali, Norbert Matthias Linke, Margaret Martonosi, Ali Javadi Abhari, Nhung Hong Nguyen, and Cinthia Huerta Alderete. 2019. Full-stack, real-system quantum computer studies: architectural comparisons and design insights. In *Proceedings of the 46th International Symposium on Computer Architecture*. 527–540.

[27] Prakash Murali, David C McKay, Margaret Martonosi, and Ali Javadi-Abhari. 2020. Software mitigation of crosstalk on noisy intermediate-scale quantum computers. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. 1001–1016.

[28] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (2018), 79.

[29] Chad Rigetti. 2018. The Rigetti 128-qubit chip and what it means for quantum. *Medium* (2018).

[30] Jonathan Romero, Ryan Babbush, Jarrod R McClean, Cornelius Hempel, Peter J Love, and Alán Aspuru-Guzik. 2018. Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz. *Quantum Science and Technology* 4, 1 (2018), 014008.

[31] Vivek V Shende, Stephen S Bullock, and Igor L Markov. 2006. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25, 6 (2006), 1000–1010.

[32] Swamit S Tannu and Moinuddin K Qureshi. 2019. Not all qubits are created equal: a case for variability-aware policies for NISQ-era quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 987–999.

[33] Peter JM Van Laarhoven and Emile HL Aarts. 1987. Simulated annealing. In *Simulated annealing: Theory and applications*. Springer, 7–15.

[34] Davide Venturelli, Minh Do, Eleanor Rieffel, and Jeremy Frank. 2018. Compiling quantum circuits to realistic hardware architectures using temporal planners. *Quantum Science and Technology* 3, 2 (2018), 025004.

[35] Robert Wille, Lukas Burgholzer, and Alwin Zulehner. 2019. Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations. In *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 142.

[36] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. 2020. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices. *Physical Review X* 10, 2 (2020), 021067.

[37] Alwin Zulehner, Alexandru Paler, and Robert Wille. 2018. An efficient methodology for mapping quantum circuits to the IBM QX architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2018).