Obfuscating Quantum Hybrid-Classical Algorithms for Security and Privacy

Suryansh Upadhyay
The Pennsylvania State University
University Park, PA, USA
sju5079@psu.edu

Swaroop Ghosh
The Pennsylvania State University
University Park, PA, USA
szg212@psu.edu

2024 25th International Symposium on Quality Electronic Design (ISQED) | 979-8-3503-0927-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISQED60706.2024.10528769

***Abstract*—As quantum computing gains popularity, it's crucial to tackle security and privacy issues upfront. One major concern is the involvement of third-party tools and hardware. With more quantum computing services available, even from less reputable sources, users might be drawn in by lower costs and easier access. However, usage of untrusted hardware could present the risk of intellectual property (IP) theft. For instance, popular algorithms like Quantum Approximate Optimization Algorithm (QAOA) encode graph properties in parameterized quantum circuits, opening the door to potential risks. For mission critical applications like power grid optimization, the graph structure can reveal the power grid and their connectivity (an IP that should be protected). To mitigate this risk, we propose an edge pruning obfuscation method for QAOA along with a split iteration methodology. The basic idea is to, (i) create two flavors of QAOA circuit each with few distinct edges eliminated from the problem graph for obfuscation, (ii) iterate the circuits alternately during optimization process to uphold the optimization quality, and (iii) send the circuits to two different untrusted hardware provider so that the adversary has access to partial graph protecting the IP. We demonstrate that combining edge pruning obfuscation with split iteration on two different hardware secures the IP and increases the difficulty of reconstruction while limiting performance degradation to a maximum of 10% ($\approx 5\%$ on average) and maintaining low overhead costs (less than 0.5X for QAOA with single layer implementation).**

***Index Terms*—Quantum Computing, Quantum Security, IP protection, Trustworthy Computing, QAOA**

I. Introduction

Quantum computing (QC) has the potential to solve many combinatorial problems exponentially faster than classical counterparts by utilizing the quantum-mechanical properties such as, superposition and entanglement. It can be used in various fields such as, machine learning [1], security [2], drug discovery, and optimization [6]. However, QC also faces technical challenges like qubit decoherence, measurement error, gate errors, and temporal variation which can lead to errors in the output of a quantum circuit. While quantum error correction codes (QEC) can provide reliable operations, they currently require a large number of physical qubits per logical qubit, making them impractical for widespread use in the near future. Noisy Intermediate-Scale Quantum (NISQ) computers, which have a limited number of qubits and operate in the presence of noise, offer a potential solution to important problems such as discrete optimization and quantum chemical simulations. To make the most of the limited capabilities of these computers,

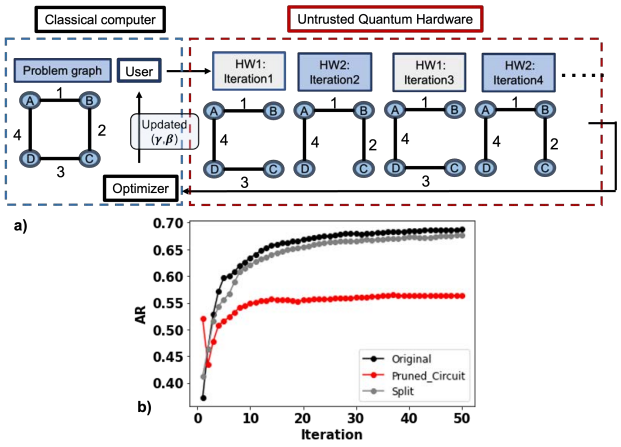

Fig. 1. a) Two versions of pruned graphs obtained from the problem graph and run on two different untrusted hardware as partial circuits for alternative iteration. b) Sample 4-node maxcut using QAOA, simulated on (Fake_montreal) for 50 iterations. The performance (Approximation Ratio) of the obfuscated circuit is significantly degraded with the edge removed (pruned), however we report performance recovery using the proposed split iteration heuristic.

various hybrid algorithms have been proposed, such as the Quantum Approximate Optimization Algorithm (QAOA) [7] and Variational Quantum Eigensolver (VQE) [3], in which a classical computer iteratively adjusts the parameters of a quantum circuit to guide it to the best solution for a given problem.

Motivation: When using variational algorithms such as, QAOA to design problem-specific parametric quantum circuits to solve certain problems, the topology of the problem is embedded in the circuit and can be considered as an asset or intellectual property (IP) (refer to Fig. 3). For example, in applications such as, power grid or other critical infrastructure optimization, the client may want to keep the problem information confidential. These IPs may not present a risk for small scale quantum circuits that can be compiled on trusted vendors such as, IBM and Rigetti. However, with the emergence of third-party service providers offering potentially higher performance, and the scaling trend of current Noisy Intermediate Scale Quantum (NISQ) computers there is an increased risk of IP infringement. Third-party compilers like Orquestra [4] and tKet [5] that support hardware from multiple vendors are becoming available. Additionally, companies such as Baidu

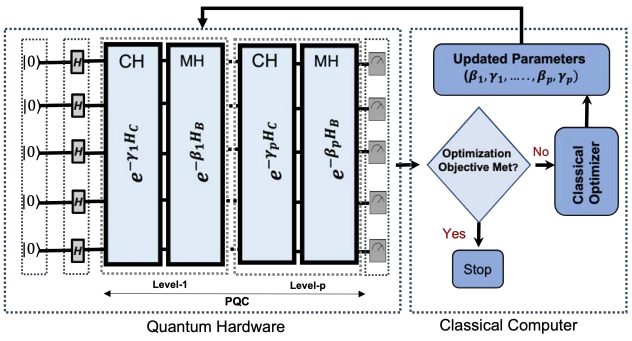

Fig. 2. Schematic of a p-level quantum-classical hybrid algorithm QAOA. A quantum circuit takes input qubit states and alternately applies Cost Hamiltonian (CH) which encodes the problem and Mixing Hamiltonian (MH) 'p' times and the final state is measured to obtain expectation value with respect to the objective function. This is fed to a (classical) optimizer to find the best parameters (γ, β) that maximizes or minimizes the cost.

[14], the Chinese internet giant, have recently announced all-platform quantum hardware-software integration solutions, such as Liang Xi, that provide access to various quantum chips via mobile app, PC, and cloud, and connect to other third-party quantum computers. These trends not only lead to reliance on untrustworthy third-party compilers and hardware suites instead of trusted counterparts, but also to reliance on third-party service providers, which can pose a significant risk to IP protection as these hardware providers/compilers can steal sensitive intellectual property (IP) and problem properties.

To mitigate these risks, this paper proposes an obfuscation approach to protect IP while ensuring correct compilation and functionality. *To the best of our knowledge, this is the first effort to identify a new security and privacy threat space for hybrid-classical quantum algorithm QAOA and develop countermeasures*.

Proposed Idea: QAOA can be used to solve complex optimization problems, such as the graph maxcut. The QAOA circuit is composed of two main parts: the mixing unitary and the problem unitary. The problem unitary is responsible for encoding the problem into the quantum circuit, while the mixing unitary is used to mix the different states of the circuit together. In our case, we consider the maxcut problem, used for finding the maxcut of a graph. To protect the IP of the QAOA circuit, one can remove or add information (e.g., graph edges) in the original graph. For example, considering a 4-node graph where nodes A, B, C, and D are connected in a circular fashion (Fig. 1a). For this case, if an edge (A, B) is removed the adversary will have 3 possible edges to check to determine the original graph, (A,C), (A,D), and (B,D), which would take $2^3 = 8$ trials in the worst-case scenario. Furthermore, there is a lack of oracle model (the quantum chip implements functionality using microwave/laser pulses that are not publicly accessible) to validate the adversarial guess. Therefore, adversarial effort to RE the obfuscated design is high. Any attempt to reuse the circuit without adding the removed edge will result in corrupted or severely degraded performance. As an example, Fig. 1b illustrates the performance degradation when only the pruned circuit is used. An incomplete circuit will not be able to generate all possible solutions, leading to suboptimal solutions. Furthermore, using an incomplete circuit for QAOA can result in poor performance as it may get stuck in a local optima. Our approach eliminates the information from the problem graph without degrading the solution quality. This is achieved by, (i) developing two variations of QAOA circuit by removing certain distinct edges from the problem graph to conceal the IP , (ii) iterating the circuits alternately during optimization process to uphold the optimization quality, (iii) sending the circuits to two different untrusted hardware provider so that the adversary has access to partial graph protecting the IP. The proposed heuristic ensures that each hardware device only receives a portion of the circuit, making it challenging for an attacker to reverse engineer the complete circuit. At the same time, the optimization performance is retained. To further illustrate the idea , we consider a simple example of a 4-node graph with only adjacent edges connected, specifically, with 4 edges (E0, E1, E2, E3). We apply the QAOA algorithm to three different cases: the original circuit, an obfuscated circuit with one edge (E0) removed, and an obfuscated circuit with the proposed hardware split heuristics, each for a total of 50 iterations Fig. 1. We use the Approximation Ratio (AR)(Section 2.6.2) as a metric to evaluate performance. The performance of the obfuscated circuit is significantly degraded when an edge (E0 out of 4) is removed, while leaving the adversary with limited information to determine the original circuit. However, when we run the first circuit (missing E0) on hardware 1 and the second circuit (missing E1) on hardware 2 with alternating iterations, we are able to recover performance while ensuring that each hardware device only has access to a partial circuit, making it challenging for an attacker to reverse engineer the full circuit.

Contributions: We, (a) propose a novel threat model, (b) propose hardware split heuristic along with edge pruning obfuscation technique to counteract the threat, (c) perform exhaustive experiments on graphs of varying complexity, (d) present analysis and validation of the proposed heuristic.

Paper organization: In the remaining paper, Section II provide quantum computing background. The proposed threat model is described in Section III. Section IV describes the proposed obfuscation procedure and presents the simulations, results and analysis. The discussions are presented in Section V. Section VI concludes the paper.

II. Background

A. Qubits and Quantum gates

Qubits are analogous to classical bits that store data as various internal states (i.e., $|0\rangle$ and $|1\rangle$). A qubit state is represented as φ = a $|0\rangle$ + b $|1\rangle$ where a and b are complex probability amplitudes of states $|0\rangle$ and $|1\rangle$ respectively. In quantum systems, computation is performed by manipulating the qubit states. Mathematically, quantum gates are represented using unitary matrices (a matrix U is unitary if $UU^{\dagger}$ = I, where $U^{\dagger}$ is the adjoint of matrix U and I is the identity matrix).

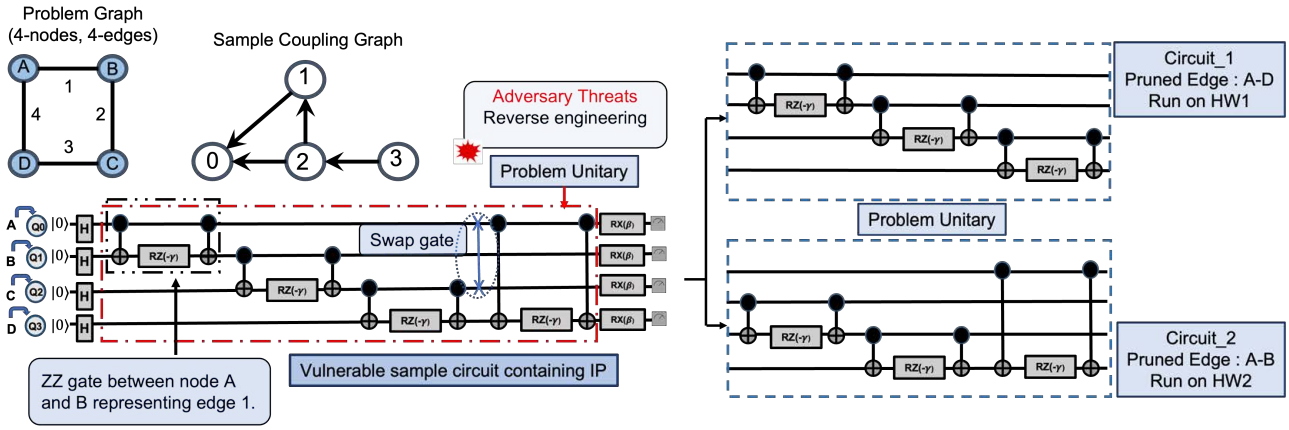

Fig. 3. Threat model proposed in this paper. QAOA is used to solve max-cut for a sample 4 node graph, the problem-specific information is encoded in the problem unitary. The pruned circuits are sent to the different hardware providers eliminating one edge (edge specific gates from the problem unitary) for hardware 1 (Edge: A-D) and including that edge but removing another for hardware 2 (Edge: A-B), ensuring each hardware device only gets a partial circuit, making it difficult for an attacker to reverse engineer the full circuit.

B. QAOA

QAOA is a hybrid quantum-classical variational algorithm designed to solve combinatorial optimization problems. The quantum state in QAOA is created by a p-level variational circuit with 2p variational parameters. Even at the smallest circuit depth (p = 1), QAOA delivers non-trivial verifiable performance guarantees, and the performance is anticipated to get better as the p-value increases [6], however the study [8] demonstrates that noise sources place a limit on those claims and the anticipated improvement. Fig.2 shows an overview of QAOA to solve a combinatorial problem. Recent developments in finding effective parameters for QAOA have been developed [6], [7].

In QAOA, a qubit is used to represent each of the binary variables in the target cost function C(z). In each of the p levels of the QAOA circuit, the classical objective function C(z) is transformed into a quantum problem Hamiltonian (Fig.2). The output of the QAOA instance is sampled many times with optimal control parameter values, and the classical cost function is evaluated with each of these samples. The solution is determined by the sample measurement with the highest cost [9]. In a quantum classical optimization procedure, the expectation value of H_C is determined in the variational quantum state $E_p(\gamma,\beta) = \varphi_p(\gamma,\beta)|H_C|\varphi_p(\gamma,\beta)$. A classical optimizer iteratively updates these variables (γ,β) so as to maximize $E_p(\gamma,\beta)$. A figure of merit (FOM) for benchmarking the performance of QAOA is the approximation ratio (AR) and is given as [6]

$$AR = E_p(\gamma,\beta)/Cmax \tag{1}$$

where $Cmax = MaxSat(C(z))$.

C. Relation to prior work

In [12], the authors proposed a method for protecting IP from an untrusted compiler by adding dummy gates in quantum circuits for obfuscation. This makes it difficult for an adversary to extract the original circuit without removing the dummy gates, which is a computationally hard problem. The authors developed a heuristic to find an optimal location for the dummy gate insertion that causes significant degradation in the output. Another work, [13] proposes split compilation to address the same issue. The idea is to split the quantum circuit into multiple parts that are sent to a single compiler at different times or to multiple compilers. In contrast, we propose a new approach for protecting IP embedded in hybrid-classical algorithms from untrusted quantum hardware. The works in [16] [17] focus on trojan insertion in reversible circuits before fabrication, which is not applicable to quantum circuits since they are not physically fabricated. Another work [18] assumes an untrusted foundry that can locate ancillary and garbage lines in a reversible circuit and extract the circuit functionality. Dummy ancillary and garbage lines are added to the circuit which increases the ancillary and garbage lines post-synthesis. However, this approach is only applicable for oracle-type or pure Boolean logic based quantum circuits and not for general quantum computing.

Authors in [2] assume that a malicious adversary in the form of untrusted hardware provider will report incorrect measurement results to the user, thereby tampering with the results. They model and simulate adversarial tampering of input parameters and measurement outcomes on QAOA. Whereas our adversarial model considers the hardware provider to be untrusted with the objective to steal the IP embedded within the quantum circuit. Therefore, our proposed approach aims to provide a more robust solution for protecting IP in hybrid-classical algorithms such as, QAOA.

III. Threat Model

A. Adversary capabilities

We assume that adversary, (a) has access to the user's circuit. This is likely if the quantum computing cloud provider is rogue, (b) has the expertise on quantum computing principles e.g., construction of QAOA circuit for a given problem and knows the details of various qubit technologies, (c) can reverse

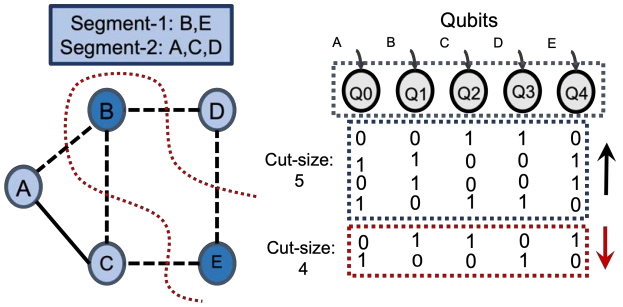

Fig. 4. QAOA illustration for solving the maximum-cut (MaxCut) problem. The MaxCut size for the represented 5 node graph is 5 (cut in red). The probabilities of basis state measurements that represent larger cut-sizes for the problem graph are increased iteratively by QAOA. After the QAOA is completed, the states in cut size 5 will have higher probabilities than the states in cut size 4.

engineer the problem properties such as, the graph from the QAOA problem circuit, (d) does not manipulate the quantum circuit, as the aim of the adversary is to steal the IP.

B. Reverse engineering attack

A quantum circuit may contain sensitive IPs such as the problem being solved, financial analysis, and proprietary algorithms that must be safeguarded. One potential threat to this IP is the use of untrusted third-party hardware providers. The adversary in the proposed scenario takes the form of a less reliable/untrusted quantum service provider who may pose as a reliable or trusted hardware provider. These attacks can be carried out by a variety of actors, including nation-states, cybercriminals, and competitors in various industries. The goal of a reverse engineering attack on QAOA is to gain information about the original optimization problem. For example: In the field of portfolio management, an attacker could use this information to learn about the portfolio of the user; In power grid optimization, the attacker could learn about the number and connectivity of power nodes which could be used to fine-tune or craft a follow up attack on the power grid.

Reverse engineering the graph from the QAOA: In QAOA circuit for solving graph maxcut problem, each node in the graph corresponds to a qubit and each edge corresponds to a ZZ gate applied between the qubits represented by the nodes (Fig. 3). Therefore, recovery of the graph from the QAOA circuit could be trivial. A potential complexity in the RE process could be due to swap gates that are added by the compiler to meet the coupling constraints of the target hardware. This is specifically true for superconducting qubit technology. For example, consider a coupling map as shown in Fig. 3, where the qubits are arranged in a two-dimensional grid and can interact with their nearest neighbors. To apply a ZZ gate between qubit Q_0 and qubit Q_3, a swap operation would be needed to first bring the two qubits together. This can be achieved by swapping the positions of Q_2 and Q_3. An adversary may infer the swap gates (since they contain sequence of 3 CNOT gates) in the QAOA circuit, remap the physical to logical qubit and still reconstruct the original graph.

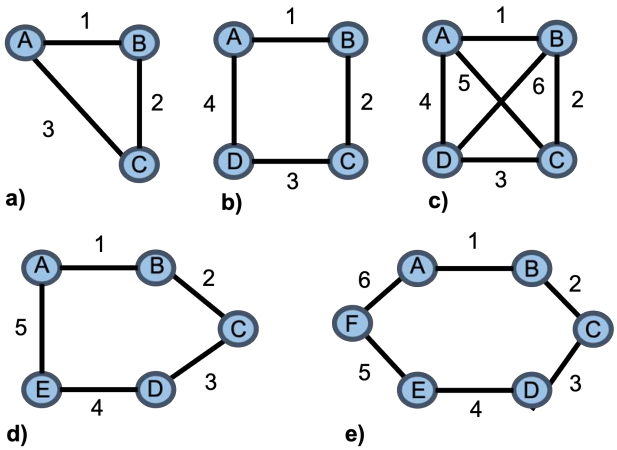

Fig. 5. Various type of graphs used for simulation as benchmarks.

C. Feasibility

The proposed attack model is a feasible threat since, a) the cost of quantum computing is still high and many customers rely on cloud-based services to access these resources. Our research on providers like AWS Braket, IBM, and Google Cloud revealed prices ranging from $0.35 to $1.60 per second for qubit counts of 8 to 40. With the entry of more vendors into the quantum service market, some may be untrusted, offering cheaper cloud access to quantum hardware. This risk increases if vendors operate offshore, where costs are lower, b) accessing the quantum computers requires long wait time. When a user submits a job to a quantum system, it enters a scheduler where it is queued. For IBM Quantum systems [15], reports indicate that only about 20% of total circuits have ideal queuing times of less than a minute, with an average wait time of about 60 minutes. Furthermore, more than 30% of the jobs have queuing times of more than 2 hours, and 10% of the jobs are queued for as long as a day or longer. This can be a significant barrier for certain applications, such as quantum machine learning, where quick access to the hardware is vital to lower the training and inference time. Third-party vendors may provide access to quantum hardware with little or no wait time.

IV. Proposed Approach and Results

A. Proposed Obfuscation Procedure

We propose an obfuscation procedure for hiding the true functionality of a quantum circuit from untrusted hardware providers. Our approach involves pruning edges, or edge-specific gates, from the problem unitary of the circuit's problem graph. We demonstrate the effectiveness of this technique for QAOA to solve the maxcut problem. To protect the IP of the QAOA circuit, we eliminate one edge from the problem unitary for hardware 1 and include that edge but remove another for hardware 2. This creates two partial circuits that are run on different hardware with alternative iterations, making it challenging for either hardware provider to reverse engineer the full circuit. Fig. 4 illustrates the proposed heuristic for a

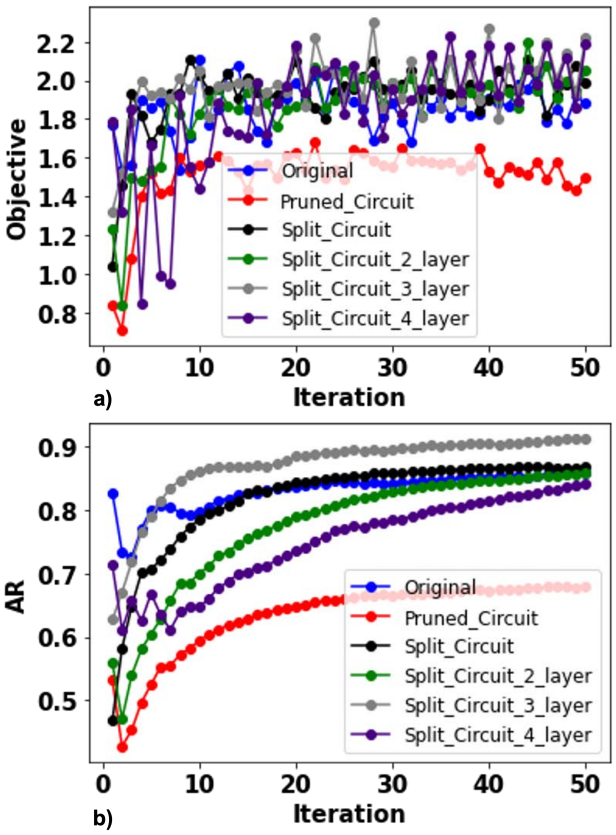

Fig. 6. a) Objective and b) Approximation ratio (r) variation for a 3 node graph: (1) original circuit run on a single hardware (HW1), (2) circuit with one edge pruned run on a single hardware (HW1), (3) pruned circuits run on 2 different hardware, HW1 (25 iterations) and HW2 (25 iterations) in alternative fashion.

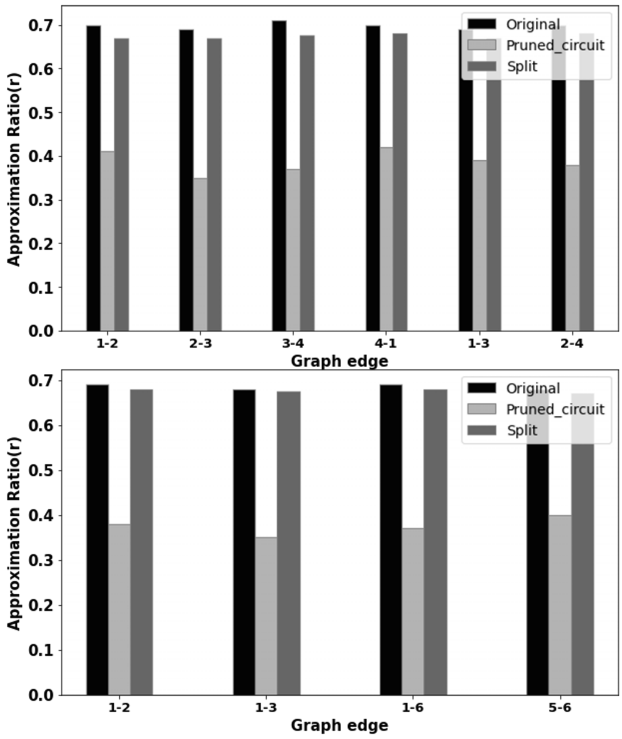

Fig. 7. a) AR comparison for the 4 node graph (from Fig. 5b) between the original circuit run on a single hardware (HW1) vs a pruned circuit run on a single hardware (HW1) vs the pruned circuits run on 2 different hardware, HW1 (25 iterations) and HW2 (25 iterations) in alternative fashion. b) Comparing AR using various combinations of single edge pruned for a fully connected 4 node graph (from Fig. 5c).

sample 4-node graph. To hardware 1, we send Circuit_1 with the edge A-D removed and to hardware 2, we send Circuit_2 with edge A-B removed making it challenging for either of the hardware provider to reverse engineer the full circuit. We use approximation ratio (AR) (Eq.1) for benchmarking the performance of QAOA for the proposed obfuscation heuristic.

B. Results and Analysis

1) Benchmark and Simulator: We use the open-source quantum software development kit from IBM (Qiskit) [11] for simulations. We implement QAOA [6] the iterative algorithm to solve the combinatorial optimization problem MaxCut [10]. The MaxCut problem involves identification of a subset S∈V such that the number of edges between S and it's complementary subset is maximized for a given graph G = (V, E) with nodes V and edges E. Though MaxCut is an NP-hard problem [10], there are efficient classical algorithms that can approximate the solution within a certain factor of optimality. Using a p-level QAOA, an N-qubit quantum system is evolved with H_C and H_B p-times to find a MaxCut solution of an N-node graph Fig. 4. QAOA-MaxCut iteratively increases the probabilities of basis state measurements that represent larger cut-size for the problem graph. Qubits measured as 0's and 1's are in two different segments of the cut Fig. 4. Various sparse and dense graphs used for benchmarking are depicted in Fig. 5. We use Qiskit's fake provider module for noisy simulation in our benchmarks, mimicking real IBM Quantum systems using system snapshots containing vital information such as coupling maps, basis gates, and qubit parameters.

2) Impact of edge selection on AR: We used fake backends, HW1:fake_montreal and HW2:fake_mumbai, to run QAOA for the different graphs shown in Fig. 5, each for 50 iterations. We first ran the original representative circuit and compared it's performance to the proposed edge pruning and split technique. We exhaustively tried various combinations of pruning a single edge to obtain circuit_1 run on HW1 and circuit_2 run on HW2 for the graphs presented. Fig. 6 showcases the objective and approximation ratio (r) variation for a 3 node graph. Here, the possible combinations are A-B: 1-2, 2-3, 3-1, where A represents the edge removed for circuit_1 and B represents the edge removed from the original circuit to obtain circuit_2. Fig. 7a) shows the comparison for the 4 node graph (from fig. 5b)) between the original circuit run on a single hardware (HW1) vs the pruned circuits run on 2 different hardware, HW1 (25 iterations) and HW2 (25 iterations) in alternative fashion, keeping the overall iterations to 50. We observed that picking any combination had no significant effect on the final AR performance for the 4 node graph. We verified this result further by repeating the same for the 4 node fully connected graph from Fig. 5c). Using the

TABLE I
PERFORMANCE TRENDS FOR VARIOUS GRAPHS (50 ITERATIONS)

Graph	Pruned edge (HW1-HW2)	Simulator	Approximation ratio(r)					
			Original	Pruned circuit	Split (1-Layer)	Split (2-Layers)	Split (3-Layers)	Split (4-Layers)
3(Fig.5a)	1-2	Ideal	.81	.62	.78	.89	.89	.9
		Fake_backend	.79	.59	.77	.84	.84	.83
4(Fig.5b)	1-2	Ideal	.74	.53	.71	.84	.86	.9
		Fake_backend	.7	.42	.67	.73	.75	.72
	(1,4)-(2,3)	Ideal	.73	.45	.62	.71	.72	.74
		Fake_backend	.68	.4	.62	.66	.66	.65
4(Fig.5c)	1-2	Ideal	.72	.45	.71	.73	.72	.74
		Fake_backend	.69	.4	.68	.7	.69	.67
	(1,4)-(2,3)	Ideal	.7	.4	.61	.69	.7	.71
		Fake_backend	.68	.32	.62	.66	.67	.66
	(1,4)-(5,6)	Ideal	.71	.41	.65	.7	.71	.71
		Fake_backend	.69	.33	.64	.67	.68	.67
5(Fig.5d)	1-2	Ideal	.7	.52	.69	.72	.73	.73
		Fake_backend	.68	.48	.67	.69	.68	.66
	(1,2)-(3,4)	Ideal	.71	.5	.61	.69	.7	.71
		Fake_backend	.69	.46	.64	.68	.68	.66
	(1,2,3)-(4,5)	Ideal	.72	.49	.6	.68	.68	.7
		Fake_backend	.68	.45	.63	.66	.64	.64
6(Fig.5e)	1-2	Ideal	.71	.55	.69	.69	.72	.73
		Fake_backend	.68	.38	.66	.68	.67	.65
	(1,2)-(3,4)	Ideal	.72	.49	.66	.73	.72	.72
		Fake_backend	.69	.38	.64	.64	.65	.64
	(1,2,3)-(4,5)	Ideal	.71	.48	.64	.66	.68	.67
		Fake_backend	.68	.36	.59	.63	.63	.61
	(1,2,3)-(4,5,6)	Ideal	.71	.41	.61	.68	.67	.67
		Fake_backend	.68	.37	.59	.64	.64	.64
	(1,2,3,4)-(5,6)	Ideal	.71	.38	.6	.62	.64	.66
		Fake_backend	.67	.36	.55	.61	.62	.64

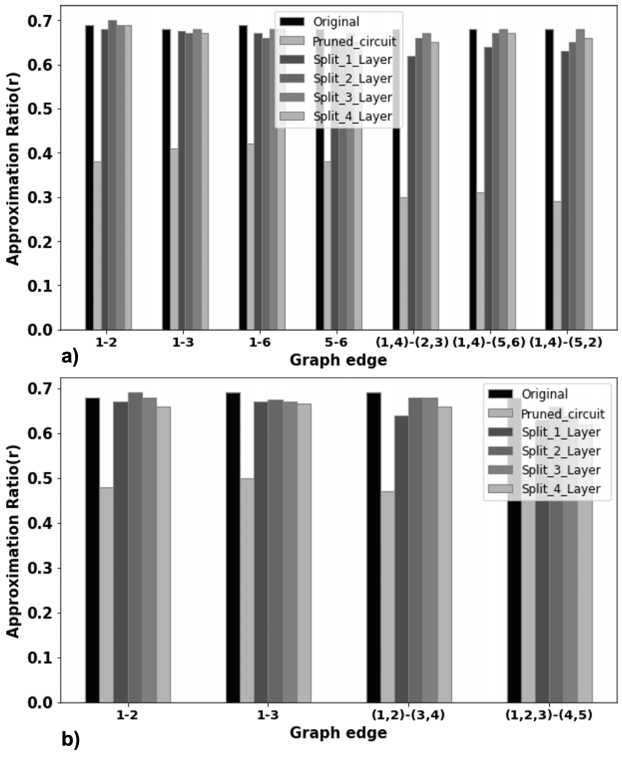

Fig. 8. Performance variation with the number of layers used in the obfuscated and split cases. a) AR variation of the split circuit with different numbers of layers for a 4-node fully connected graph (from Fig. 5c) with various combinations of pruned edges. b) AR variation for a 5-node graph(from Fig. 5d) with various combinations of pruned edges.

results from the previous graph, we just compared one case from the distinct types of combinations possible such as a) one with an adjacent edge pruned [1-2,2-3,3-4,1-4], b) one with an opposite edge pruned [1-3,2-4], c) one regular edge and one edge from the diagonal [1-5,2-5,3-5,4-5,1-6,2-6,3-6,4-6] and d) both diameter edges 5-6. **We observe (Fig. 7b) that pruning any edge, irrespective of the connectivity, resulted in similar AR performance.**

3) Impact of pruning more than one edge at a time: Pruning multiple edges simultaneously can hinder an adversary's ability to reverse engineer the circuit by limiting the amount of information available to them. However, AR degradation is much more significant. We observed to regain AR, additional layers can be added to the circuit, which will bring the AR performance closer to the original circuit (average recovery $\approx 90\%$), but this also increases the execution time overhead. Table I shows the AR variation with a few different combinations of multiple edge pruned for various node graphs. **For simplicity we only pruned adjacent edges and compared performance with varying number of layers.**

4) Performance comparison with varying QAOA layers: Fig. 8 represents the performance variation with the number of layers used in the obfuscated and split cases. We compare the performance with a single layer of the original circuit as the baseline. Fig. 8a) showcases the performance variation of the split circuit with different numbers of layers for a 4-node fully connected graph with various combinations of pruned edges, while Fig. 8b) depicts the same for a 5-node graph. **We**

observe that for just one edge pruned in either case, the AR degradation when compared to the original is minimal, and increasing the number of layers may slightly improve AR but comes with a cost of computation time and resources overhead. For removal of 2 or more edges, the AR degrades significantly, which can be recovered by increasing the number of layers. However, increasing the number of layers past 2 leads to degraded performance, bounded by the qubit quality metrics (noise, short lifetime, and imperfect gate operations) of the target hardware. As per [8], the optimal number of stages (p-value) for any QAOA instance is limited by the noise characteristics (gate error, coherence time, etc.) of the target hardware, as opposed to the current perception that higher-depth QAOA will provide monotonically better performance for a given problem compared to low-depth implementations.

5) Performance comparison in ideal simulator vs fake backends: The performance of QAOA on different node graphs is compared on ideal and fake backend simulators (Table-I). The results show that the average relative (AR) for each case is degraded by 10% for the fake backend when compared to the ideal counterpart. **As the number of layers in QAOA increases, there is an observed improvement in performance for all simulated graphs in the ideal simulator. However, when using the fake backend, after a certain point (depending on the number of nodes pruned), there is no increase or even some performance degradation with increasing the number of layers**. This can be attributed to the hardware-induced errors that can alter the parameter and solution landscape of QAOA, limiting the performance gain achievable with higher numbers of layers. Furthermore, as the number of layers in QAOA increases, the circuit execution time also increases, which may exceed the qubit decoherence time and introduce more errors. In contrast, QAOA instances on ideal hardware have small variations between them.

6) Adversarial reverse engineering effort: In the context of circuit obfuscation, an adversary's goal is to identify and add the removed edge(s) to retrieve the original graph/circuit. The number of trials required to determine the original graph depends on the properties of the original graph and the specific edge(s) that were removed. For example, consider an n-node complete graph, where every node is connected to every other node. In this case, any edge that is removed would be a bridge edge, and it would split the graph into two connected components, each with $(n-1)$ nodes. This means that the disconnected components would have a fixed number of edges, and it would be relatively easy to determine the original graph, as there is only one possibility. However, for an n-node cycle graph, where every node is connected to exactly two other nodes, forming a cycle if an edge is removed, the graph would become a tree, which is a connected graph with $(n-1)$ edges and $(n-1)$ nodes. In this case, the adversary would have $(n-1)(n-2)/2$ possible edges to check, and it would take $2^{(n-1)(n-2)/2}$ trials to determine the original graph in the worst-case scenario. Furthermore, the adversary has no way to validate the guess, and any attempt to reuse the circuit without adding the removed edge will result in corrupted or severely degraded performance. Adversary also lacks insights on the number of deleted edges which can add another layer of complexity. Therefore, the adversarial effort to reverse engineer the obfuscated design is high. For example, considering a 10-node graph. There can be $(10*9)/2 = 45$ possible edges that can be present or absent. For a complete graph with one deleted edge, the minimum number of guesses required is just 1. For a cyclic graph, the adversary would have (9*8)/2 = 36 possible edges to check which would take $2^{36} = 6.9X10^{10}$ trials to determine the original graph in worst-case scenario. In case the original graph is an arbitrary graph, the average number of guesses required would be the average of all possible graphs, which would be $2^{(45/2)} = 1.5X10^7$. The actual number of guesses required may vary depending on the specific graph and the specific edge that was removed.

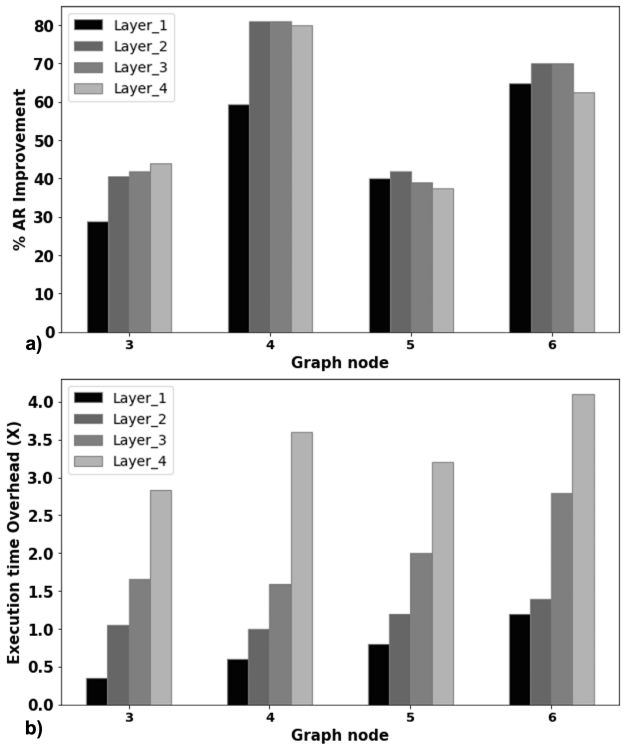

Fig. 9. a) Improvement in AR with the number of layers, with one edge pruned for graphs (Fig. 5a,b,d,e). The baseline for comparison is a QAOA circuit with single pruned edge run on one untrusted hardware. b) Execution time for different layers compared to the baseline of a single edge pruned circuit.

C. Overhead analysis

Fig. 9 compares the AR improvement versus execution time overhead with respect to the case where a single edge is pruned from a single layer QAOA circuit and is executed on a single untrusted hardware. We observe an average improvement of 5% in performance and AR recovery upto 90% by increasing from layer 1 to layer 2 with an average time overhead increase of 0.4X. However, subsequent increases in the number of layers resulted in marginal performance improvement (0.1%

on average) at the cost of significant increase in time overhead (2x on average). If the original QAOA circuit contains multiple layers, the proposed addition of extra layers may lead to only minimal improvement (average 5%). In such cases, the proposed pruning and split heuristic may lead to execution time overhead ($\approx 2X$ on average) to provide security guarantees. Additionally, the use of the pruning and split heuristic requires an additional quantum hardware which can negate some of the cost savings from the use of cheaper quantum hardware. However, each hardware employs only half of the original number of iterations which can recover the cost due less computation time per hardware.

V. Discussion

A. Implementation on real hardware

Fake backends can run on classical computers and are useful for testing and evaluating algorithms before they are run on actual hardware. Since fake backends are calibrated with real hardware on daily basis, the results obtained from fake backends are very realistic. Furthermore, simulation on fake backend is quick since it avoids the long wait queue associated with the real hardware. Running QAOA using the proposed pruning-split heuristic on actual hardware, such as a superconducting qubit or trapped ion quantum computer, can provide more accurate results however, the conclusions will remain the same.

B. Consideration for adding an extra edge for obfuscation

Adding an extra edge can also be one potential way for obfuscation in QAOA that could provide a level of protection for IP by making it more difficult for unauthorized parties to reverse engineer the algorithm and access sensitive information. However, this approach will increase the complexity and increased computational resources due to processing of an extra edge. Furthermore, the quality degradation due to cutting of extra edge/s could be difficult to recover in contrast to pruning of edge/s where recovery involves iteration with another circuit that has a different edge pruned.

C. Consideration for using more than 2 flavors of pruned circuit and multiple hardware

Using multiple "flavors" of pruned circuits, as well as multiple hardware providers, is one way to further mitigate potential risks. In practice, this could be achieved by creating multiple versions of a circuit, each with a distinct set of edges eliminated, and alternating between these versions during the optimization process. Extra layers may be added to recover optimization quality. Though it may lead to performance decline and added costs, it enhances security and privacy.

VI. Conclusion

QAOA circuit can reveal the problem information to untrusted or unreliable hardware and pose a risk for IP theft. Our solution involves an edge pruning obfuscation method paired with a split iteration approach. This heuristic enhances IP security by distributing circuit components with partial information across various hardware, complicating adversaries' attempts to access complete data. Additionally, we suggest expanding the number of layers in QAOA circuits to offset any quality reduction resulting from the obfuscation. Our analysis demonstrates that the pruning-split method increases the adversary's reconstruction complexity significantly while maintaining solution quality with minimal additional overhead.

VII. Acknowledgment

This work is supported in parts by NSF (CNS-1722557, CNS-2129675, CCF-2210963, CCF-1718474, OIA-2040667, DGE-1723687, DGE-1821766, and DGE-2113839), Intel's gift and seed grants from Penn State ICDS and Huck Institute of the Life Sciences.

References

[1] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," Nature Physics, 2019. [Online]. Available: https://doi.org/10.1038/s41567- 019- 0648- 8.

[2] Upadhyay, Suryansh, and Swaroop Ghosh. "Robust and Secure Hybrid Quantum-Classical Computation on Untrusted Cloud-Based Quantum Hardware." arXiv preprint arXiv:2209.11872 (2022).

[3] A. Kandala,et al "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," Nature, vol. 549, Sep 2017.

[4] Z. Computing, "Orquestra," May 2021. [Online]. Available: https://www.zapatacomputing.com/orquestra/

[5] C. Q. Computing, "pytket," May 2021. [Online]. Available: https://cqcl.github.io/pytket/build/html/index.html

[6] L. Zhou et al., "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," arXiv preprint arXiv:1812.01041, 2018.

[7] G. Guerreschi and A. Matsuura, "Qaoa for max-cut requires hundreds of qubits for quantum speed-up," arXiv preprint arXiv:1812.07589, 2018.

[8] Alam, Maha Anis et al. "Analysis of Quantum Approximate Optimization Algorithm under Realistic Noise in Superconducting Qubits." ArXiv abs/1907.09631 (2019): n. pag.

[9] F. G. Brandao et al., "For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances," arXiv preprint arXiv:1812.04170, 2018.

[10] R. M. Karp, "Reducibility among combinatorial problems," in Complexity of computer computations. Springer, 1972, pp. 85–103.

[11] A. Cross, "The IBM q experience and qiskit open-source quantum computing software," in APS Meeting Abstracts, 2018.

[12] Suresh, Aakarshitha and Saki, Abdullah Ash and Alam, Mahabubul and Topalaglu, Rasit o and Ghosh, Dr. Swaroop. A Quantum Circuit Obfuscation Methodology for Security and Privacy, 10.48550/ARXIV.2104.05943. arXiv 2021.

[13] A. A. Saki, A. Suresh, R. O. Topaloglu and S. Ghosh, "Split Compilation for Security of Quantum Circuits," 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2021, pp. 1-7, doi: 10.1109/ICCAD51958.2021.9643478.

[14] https://www.insidequantumtechnology.com/news-archive/chinas-baidu-rolls-beijing-based-quantum-computer-and-access-platform/

[15] Ravi, G. S., Smith, K. N., Gokhale, P., and Chong, F. T. (2021, November). Quantum Computing in the Cloud: Analyzing job and machine characteristics. In 2021 IEEE International Symposium on Workload Characterization (IISWC) (pp. 39-50). IEEE.

[16] Cui, X., Saeed, S.M., Zulehner, A., Wille, R., Wu, K., Drechsler, R. and Karri, R., 2018. On the difficulty of inserting trojans in reversible computing architectures. IEEE Transactions on Emerging Topics in Computing, 8(4), pp.960-972.

[17] Limaye, Nimisha, Muhammad Yasin, and Ozgur Sinanoglu. "Revisiting logic locking for reversible computing." In 2019 IEEE European Test Symposium (ETS), pp. 1-6. IEEE, 2019.

[18] Saeed, S.M., Zulehner, A., Wille, R., Drechsler, R. and Karri, R., 2019. Reversible circuits: Ic/ip piracy attacks and countermeasures. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 27(11), pp.2523-2535.