Trojan Attacks on Variational Quantum Circuits and Countermeasures

Subrata Das
Dept. of Electrical Engineering
Pennsylvania State University
University Park, PA
sjd6366@psu.edu

Swaroop Ghosh
Dept. of Electrical Engineering
Pennsylvania State University
University Park, PA
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/ISQED60706.2024.10528776

***Abstract*—Quantum computing holds tremendous potential for various applications, but its security remains a crucial concern. Quantum circuits need high-quality compilers to optimize the depth and gate count to boost the success probability on current noisy quantum computers. There is a rise of efficient but unreliable/untrusted compilers; however, they present a risk of tampering, such as malicious insertion of Trojans, which can degrade circuit performance and reliability. This work focuses on analyzing the impact of Trojans in Quantum Approximate Optimization Algorithm (QAOA) circuits, which are widely used for solving combinatorial optimization problems. We propose a methodology to reveal vulnerable locations and adversarial gate types for Trojan insertion that maximizes the negative impact on QAOA's approximation ratio in solving Max-Cut problem. By disrupting critical paths and altering qubit states, the strategic insertion of additional gates degrade the approximation ratio by up to 50% based on evaluations on benchmark graphs. These insights on plausible attack mechanisms advance the understanding of optimization-oriented Trojan vulnerabilities specific to quantum computing. Additionally, a Convolutional Neural Network (CNN) model, referred to as QTrojanNet, is presented to detect the presence of Trojans in compiled QAOA circuits by learning inherent features that indicate malicious modifications. Experimental results showcase an average accuracy of 98.80% and an average F1-score of 98.53% in effectively detecting and classifying Trojan-inserted QAOA circuits.**

***Index Terms*—Quantum security, Hardware Trojan, QAOA**

I. Introduction

Quantum computing has emerged as a transformative technology, offering significant advancements in fields such as optimization, cryptography, and material science [1]. The Quantum Approximate Optimization Algorithm (QAOA) is central to utilizing quantum computers for combinatorial optimization problems, employing variational techniques to discover near-optimal solutions [2], [3]. This versatility and scalability make QAOA circuits vital in quantum computing research and applications, often incorporating proprietary algorithms or domain-specific knowledge, such as in financial portfolio optimization. In leveraging quantum computing's advantages, prioritizing security and privacy is essential [4], [5]. Quantum circuits, notably those using QAOA, depend on advanced compilers for optimal performance [6]–[8]. These compilers are vital for minimizing circuit depth and gate count, enhancing success on error-prone quantum computers. Recent advancements, however, have introduced efficient third-party compilers with claimed superior optimization for complex circuits. Despite their efficiency, the reliability and security compliance of these compilers are uncertain, potentially compromising quantum computing systems' integrity and trustworthiness.

One of the main risks associated with relying on unreliable compilers is the potential for tampering, specifically the insertion of Trojan gates. In conventional digital circuits, Hardware Trojans can be detected post-manufacturing by applying test patterns and validating the outputs against expected patterns [9]. Test vectors can be cleverly crafted to trigger rare Trojan activation, revealing incorrect outputs that imply the presence of Trojan. Conversely, the Trojan should be designed to bypass such detection methodologies. However, testing based approaches are ineffective for quantum circuits as the user lacks an oracle to verify results. This also eases the Trojan design process since it does not need to be activated rarely anymore. Though the outputs are deterministic, the user does not know the correct values beforehand for practical sized quantum circuits (since they cannot be simulated in classical computers). Our threat model involves an untrusted compiler inserting Trojans into user circuits. When the user receives back these compiled Trojan circuits, they have no means to validate the behavior or outputs. Running the circuits on actual quantum hardware also does not reveal Trojans, since the user is unaware of the Trojan-free outputs. Therefore, the traditional test vector based Trojan detection methods are ineffective for quantum circuits. Specialized techniques that can analyze the circuit structure itself are required to identify the subtle signs of Trojans.

Proposed Idea: In this work, we propose a vulnerability analysis technique for Trojans inserted in variational quantum circuits. The goal is to enhance the security of quantum circuits by providing insights into the impact of Trojan attacks. We focus on QAOA circuits which are crucial for combinatorial optimization but lack comprehensive threat analysis. Firstly, we strategically insert Trojans, considering different gate types and locations to maximize their negative impact on the QAOA optimization process. We select small 3-5 node graphs and generate optimized QAOA circuits for solving the graph Max-Cut problem using Qiskit simulations. These

Supported by the National Science Foundation

Trojan-free circuits are converted to directed acyclic graphs to identify critical paths. Trojans in the form of additional X, H, R_x, R_Z, CX and SWAP gates are inserted at the front, middle, and back of critical and non-critical paths. The Trojan-inserted circuits are compiled and executed with the same parameters as their Trojan-free counterparts. By comparing the approximation ratios, the most impactful insertion strategies are identified. Later, we generate diverse datasets of compiled Trojan-free and Trojan-inserted QAOA circuits to facilitate further analysis. Finally, we develop a Convolutional Neural Network (CNN) model to detect the presence of Trojans in compiled QAOA circuits by identifying disruptions in inherent patterns.

Contributions: Firstly, we propose a strategy for inserting Trojans or extra gates in QAOA circuits, considering different gate types and insertion locations. This comprehensive exploration allows us to identify the configurations that have the most significant impact on the optimization process. By quantitatively assessing the optimization results, we provide insights into the most critical areas of vulnerability and potential attack vectors. Secondly, we generate a comprehensive dataset comprising original Trojan-free QAOA circuits and their corresponding Trojan-inserted counterparts. Thirdly, we introduce QTrojanNet to detect and classify Trojan-inserted QAOA circuits. QTrojanNet leverages the power of deep learning to capture subtle patterns and features within the circuits, enabling accurate identification of Trojans. Finally, we conduct extensive experiments to evaluate the efficacy of QTrojanNet in accurately detecting Trojans in QAOA circuits.

II. Background

A. Quantum Computation Preliminaries

1) Qubits: Quantum computing utilizes quantum mechanics principles, employing qubits that exist in a superposition of 0 and 1 states, allowing simultaneous representation of multiple states [10]. A qubit's state is defined in a two-dimensional Hilbert space as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, with α and β as complex amplitudes where $|\alpha|^2 + |\beta|^2 = 1$.

2) Quantum Entanglement: Entanglement, a key quantum feature, enables qubits to be correlated across distances, requiring a collective description. An entangled state of two qubits is expressed as $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$, with normalization $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$.

3) Quantum Gates: Quantum gates manipulate qubit states through unitary matrices. Essential gates include Pauli gates (X, Y, Z), Hadamard (H), rotation gates (R_x, R_y, R_z), SWAP, and CNOT. These gates enable complex calculations beyond classical computing capabilities.

4) Quantum Superposition and Parallelism: Superposition allows qubits to process multiple states simultaneously, underpinning quantum parallelism. This enables quantum algorithms to simultaneously explore solutions, potentially outperforming classical algorithms for certain problems.

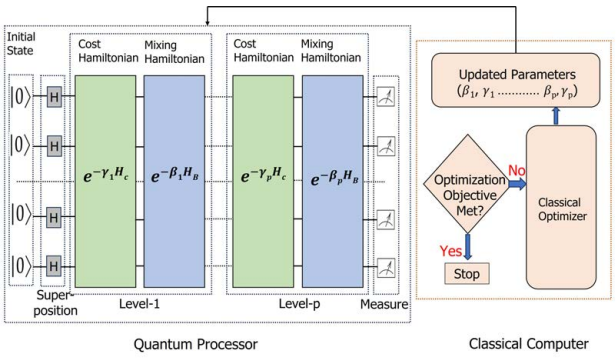

Fig. 1. Schematic of QAOA showing p layers of CH and MH applications. Classical optimization refines parameters (γ, β) for cost minimization.

B. Compilation of Quantum Circuits

Quantum circuit compilation transforms abstract circuits into executable instructions. Key steps include virtual optimization to minimize gates, decomposition into hardware-compatible gates, qubit mapping considering hardware constraints, and physical optimization to further reduce execution time [11].

C. Quantum Algorithm

1) Quantum Approximate Optimization Algorithm (QAOA): QAOA, a hybrid algorithm, addresses combinatorial optimization by applying a sequence of quantum gates in alternating layers of Cost (H_C) and Mixing (H_M) Hamiltonians [2], as depicted in Fig. 1. It starts with a uniform superposition, followed by layers of $e^{-i\gamma_l H_C}$ and $e^{-i\beta_l H_M}$ operations, exploring configurations and promoting state mixing. The process aims to optimize gate parameters to minimize the objective function, with the approximation ratio (AR) evaluating performance.

AR quantifies QAOA's solution quality, aiming for AR close to 1 for optimal performance:

$$\text{AR} = \frac{E(\theta)}{E_{\text{opt}}},$$

where E_{opt} is the optimal value.

2) Graph Max-Cut Problem: The Max-Cut problem, seeking to maximize edges between two node sets, is NP-hard for approximation ratios $\geq \frac{16}{17}$ [12]. It's represented by maximizing $f(x) = \sum_{(i,j)\in E} w_{ij} \cdot (1 - x_i \cdot x_j)$, with x_i indicating node i's set.

3) QAOA for Solving Graph Max-Cut Problem: For the Max-Cut, QAOA encodes graph structure into qubits, optimizing partitioning to maximize edge cuts. The objective $C(z) = \sum_{\alpha=1}^{m} C_\alpha(z)$ relates to maximizing edge separations, with the QAOA circuit applying problem and mixing Hamiltonians in layers to find optimal partitions:

$$U(\gamma, \beta) = \prod_{l=1}^{p} e^{-i\beta_l H_M} e^{-i\gamma_l H_C},$$

optimizing parameters to guide towards the solution.

D. Related Work and Their Limitations

Several studies have addressed the challenges and vulnerabilities associated with quantum computing. One area of research focuses on the security threats posed by untrusted compilers in quantum circuits [5], [13]. They point out potential IP (Intellectual Property) theft issues introduced by unreliable compilers during the optimization process. Another work [14] presents a split compilation methodology to address the same concern. By splitting the quantum circuit into multiple parts and sending them to a single compiler at different times or to multiple compilers, this methodology provides partial information to adversaries and introduces factorial time reconstruction complexity. Another relevant work [4] focuses on obfuscating quantum hybrid-classical algorithms, specifically QAOA, to protect sensitive information encoded in the circuit parameters from untrusted quantum hardware. The proposed edge pruning obfuscation method and split iteration methodology secure the IP while maintaining low overhead costs. Further, [15] proposes equal distribution of computations among hardware options and an adaptive heuristic to identify tampered hardware. While previous works have explored quantum security from various angles such as untrusted compiler, the specific issue of Trojan insertion, insertion methods, and detection mechanisms in quantum circuits has not been studied yet to the best of our knowledge.

In terms of Hardware Trojan (HT) detection in integrated circuits (ICs), several studies using machine learning techniques have been proposed [16]. For instance, [17] proposed a golden reference-free HT detection method using Graph Neural Networks (GNNs) for both Register Transfer Level (RTL) and gate-level netlists. The authors leverage a Data Flow Graph (DFG) representation of the hardware design to train the GNN model, enabling the detection of unknown HTs with high recall rates. Another study [18] explored the use of Artificial Immune Systems (AIS) for detecting RTL Trojans by leveraging the behavior classification of high-level hardware descriptions. Furthermore, [19] focused on machine learning-based HT detection at the RTL level, employing the gradient boosting algorithm and a server-client mechanism for timely updates. While previous research has made significant contributions to Trojan detection in traditional ICs, there is a noticeable lack of studies focusing on Trojan detection in quantum circuits, such as QAOA. Although existing machine learning-based detection tools can be trained to detect HTs in quantum circuits, it will require a new training dataset. To bridge this gap, we develop a dataset specific to quantum circuits. We also utilize the insights from existing machine learning-based research on HT detection and apply them to the unique characteristics and vulnerabilities of QAOA circuits, enabling effective Trojan detection in the quantum computing domain.

III. Threat model and adversary capability

A. Threat Model

We assume that the user designs a QAOA circuit to solve the Max-Cut problem and employs untrusted or less-trusted third-party compilers to optimize the depth and gate count. This choice may arise due to the limited availability of trusted compilers that keep pace with the latest optimization advancements. The QAOA circuits sent to untrusted compilers can be subjected to impactful Trojan insertion and/or tampering. By introducing Trojans or extra gates in specific locations, adversaries can introduce biases or perturbations that disrupt the optimization process. As a result, the quality and reliability of the obtained solutions are compromised. Once adversaries have strategically inserted Trojans or extra gates into the QAOA circuits, they compile the modified circuits and transmit them to the user. The process of compilation often transforms the original circuit into a form that is not easily inspectable or discernible, making it more challenging to identify the presence of Trojans. This inherent difficulty in detecting Trojans within the compiled circuit amplifies the risks associated with compromised optimization outcomes. Post-compilation, the user will execute the QAOA circuit in quantum hardware, obtain cost and optimize the parameters. After running several iterations, the QAOA algorithm will converge. At that point, the user will get the result of Max-Cut, which may not be optimal (without his knowledge, as the result cannot be validated). Therefore, the objective of the adversary will be fulfilled.

B. Adversary Objectives Behind Trojan Insertion

Trojans pose a pertinent threat to quantum circuits as malicious adversaries can leverage them to undermine the outcomes in various ways. Here, we will discuss several plausible attack goals applicable to the quantum domain. Firstly, the attacker can aim to increase the number of shots needed to obtain the correct output. By reducing the probability of measuring the right state, the Trojan forces the user to expend more shots to extract the correct result. Secondly, the adversary can corrupt the core functionality of the algorithm itself. For instance, a Trojan inserted into a QAOA circuit optimized for portfolio optimization could alter the asset allocation to intentionally provide a poor investment strategy. Another objective could be degrading the reliability of outcomes. If the original circuit produces accurate solutions with a high success probability, the Trojan can diminish this reliability. For instance, a QAOA circuit for combinatorial optimization may originally find quality solutions with a 95% chance, but after Trojan insertion, the success rate may drop to 50%. This significantly lowers the utility for users. This is the objective we consider in this work and evaluate the quality degradation using approximation ratio for the Max-Cut problem.

C. Adversary Capability

We assume that adversaries possess expertise in the structure and functioning of QAOA circuits. They have a comprehensive understanding of the circuit's components, including the sequence of gates, qubit connectivity, and measurement operations. This knowledge enables them to identify suitable insertion points for extra gates and determine the optimal timing to achieve maximum impact on the circuit's behavior.

Moreover, adversaries have access to significant computational resources, allowing them to analyze and manipulate the circuits effectively. With the computational power, they can perform detailed analyses of the circuit's properties, such as entanglement patterns and optimization landscape. This enables them to strategically insert extra gates that introduce biases or modify the entanglement structure, ultimately leading to the generation of suboptimal solutions during the optimization process.

IV. Trojan Vulnerability Analysis for QAOA Circuits

This section outlines our approach to assess vulnerabilities in QAOA circuits, focusing on identifying critical Trojan insertion points and their impact on optimization performance.

A. Overview of the Methodology

Our analysis begins with small graphs (3 to 5 nodes) to explore the Max-Cut problem using QAOA. We optimize the QAOA circuit parameters over a set number of iterations to achieve an optimal approximation ratio (AR), reflecting the solution's quality. The optimized circuit is then represented as a Directed Acyclic Graph (DAG), allowing us to pinpoint the circuit's critical path. By inserting extra gates at various positions along the critical and non-critical paths and compiling the modified circuits with IBM Qiskit [11], we compare the ARs of circuits with and without Trojans. The position yielding the most significant AR reduction is deemed the most vulnerable. Additionally, we experiment with different types of gates inserted at this critical location to ascertain which gate type most adversely affects the QAOA performance.

B. Case Study: A 3-Node Graph

In this subsection, we present a case study to demonstrate the application of the proposed methodology using a 3-node benchmark graph, as depicted in Fig. 2(a). All edges of the graph are assigned equal weight. Initially, we employ the QAOA algorithm to solve the Max-Cut problem for this graph and optimize the parameters of the QAOA circuit. We set the number of layers in the QAOA circuit to 1 and utilize the Qiskit for simulations. The optimized QAOA circuit is shown in Fig. 2(b).

Subsequently, we convert the optimized QAOA circuit into a DAG as illustrated in Fig. 2(c). By analyzing the DAG, we identify the critical paths within the circuit, namely the paths starting from qubit 0 and qubit 1. To determine the most vulnerable insertion locations, we insert a Trojan X gate at the front, middle, and back positions of both the critical and non-critical paths, as shown in Fig. 2(d). The Trojan-inserted QAOA circuits are then compiled using Qiskit to ensure compatibility with IBM's $qasm\_simulator$ backend. Using the Trojan-inserted QAOA circuit, we address the same Max-Cut problem for the benchmark graph, employing the same maximum number of iterations (2500) as the Trojan-free circuit. It should be noted that in all the experiments, QAOA circuit parameter optimization is performed by running the algorithm for a fixed number of iterations. This aims to enable controlled vulnerability assessment across circuits under consistent conditions. While running QAOA optimization for more number of iterations could improve results, it was challenging due to resource constraints.

The resulting loss in AR, depicted in the bar chart of Fig. 3(a), indicates the impact of Trojan insertion. Interestingly, we observe that the insertion of a Trojan at the front of the critical path leads to a higher loss in AR compared to the non-critical path. Furthermore, we investigate the effect of different gate types inserted at the front of the QAOA circuits, including single-qubit gates (X, H, R_x with rotation angle 2.52, R_z with rotation angle 6.91) and two-qubit gates (CX and SWAP). We deliberately chose these rotation angles for the inserted gates to match those of the original R_x and R_z gates in the circuits, thereby making the detection of the Trojan more challenging. Strikingly, we find that the loss in AR is notably higher when an X gate is inserted (Fig. 3(b)).

Our analysis reveals that the most vulnerable insertion location for Trojans in QAOA circuits is at the front of the critical path. Also, the X gate shows a significant detrimental effect on the circuit's performance. This can be attributed to the disruption caused by the X gate altering the initial state of the qubits. The QAOA algorithm relies on an initial superposition state, typically achieved through H gates, which allows for the exploration of a larger solution space. However, the insertion of an X gate at the beginning disrupts the superposition, limiting the circuit's ability to explore and find optimal solutions. Consequently, the optimization process is hindered, resulting in a higher loss in AR compared to other gate types.

C. Evaluation of Trojan-Insertion Strategy on Benchmark QAOA Circuits

In this subsection, we assess the effectiveness of our identified most impactful Trojan-insertion strategy on a set of five benchmark QAOA circuits. These circuits are specifically optimized for solving the Max-Cut problem of five benchmark graphs consisting of four and five nodes. Two representative benchmark graphs are illustrated in Fig. 4(a). To begin, we utilize the QAOA algorithm and Qiskit to simulate the optimization process on a local machine with an AMD Ryzen 7 4800U CPU operating at 1.80 GHz and 16 GB RAM, running Windows 11 Pro. The number of layers in the optimized QAOA circuits is set to 1. After optimizing the QAOA circuits, we identify the critical path within each circuit by converting them into DAG. Subsequently, we insert a Trojan X gate at the front of the critical path. The Trojan-inserted circuits are then compiled using Qiskit's $qasm\_simulator$ to ensure compatibility and accurate evaluation. Next, we employ the compiled Trojan-inserted circuits to solve the respective graph's Max-Cut problem, using the same number of iterations as the Trojan-free circuits. The resulting ARs for both the Trojan-free and Trojan-inserted benchmark circuits are presented in Fig. 4(b). It demonstrates that the proposed Trojan-insertion strategy can degrade the optimization performance by

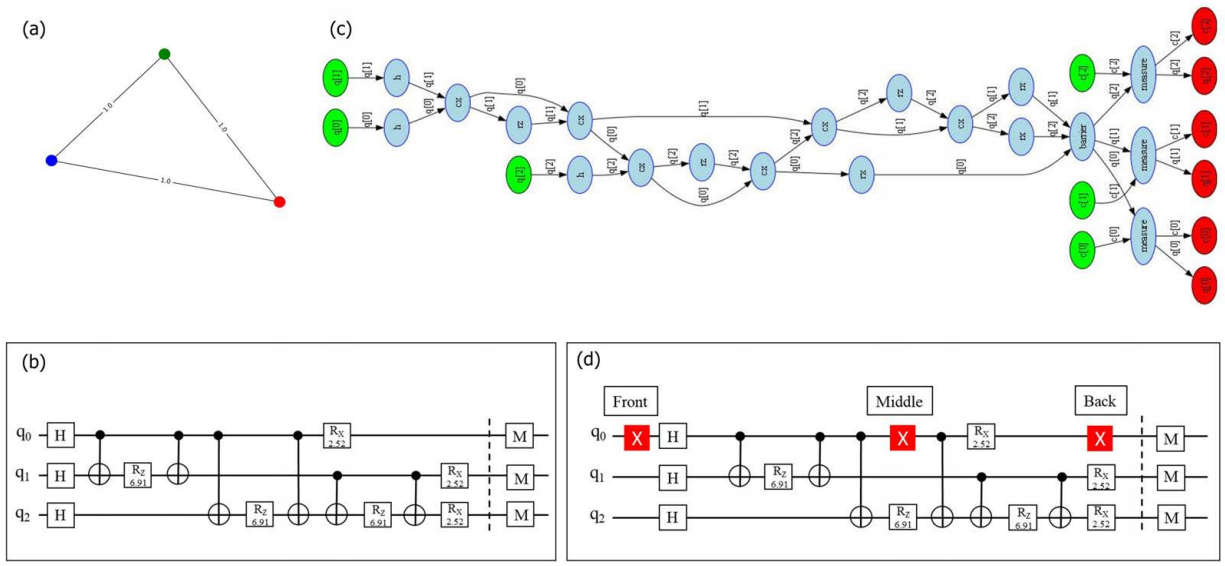

Fig. 2. Illustration of the case study using a 3-node benchmark graph. (a) The 3-node graph used for the MAx-Cut problem. (b) The optimized QAOA circuit obtained for solving the MAx-Cut problem. (c) The Directed Acyclic Graph (DAG) representation of the optimized QAOA circuit. (d) Insertion of a Trojan X gate (red gates) at the front, middle, and back positions of the critical path.

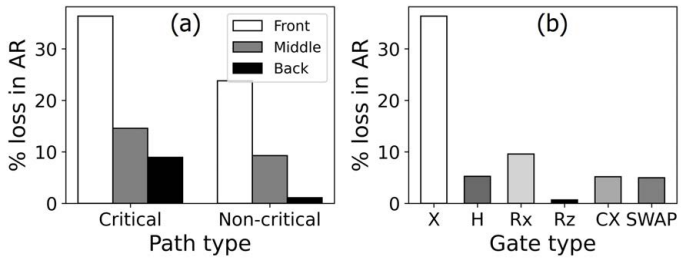

Fig. 3. Impact of Trojan insertion on the AR for the 3-node benchmark graph. (a) Bar chart showing the %loss in AR due to Trojan insertion at different locations. (b) Comparison of the %loss in AR for different gate types inserted at the front of the QAOA circuits, including single-qubit gates (X, H, R_x, R_z) and two-qubit gates (CX and SWAP).

as much as 50%. This evaluation provides valuable insights into the vulnerability of QAOA circuits to Trojan insertion and underscores the importance of developing robust defenses against potential attacks.

V. Design and Performance Analysis of QTrojanNet

A. Overview

In this section, we delve into the design and performance analysis of QTrojanNet. We discuss the dataset generation process, the architecture of QTrojanNet, the training and evaluation procedures, and present the performance results and evaluation metrics. To facilitate further research and ensure the reproducibility of our results, the CNN model developed in this study, along with the dataset used, is hosted on GitHub [20].

QTrojanNet is specifically tailored to capture the subtle patterns and features in QAOA circuits that indicate the presence of Trojans. For instance, a typical QAOA circuit for the Max-Cut problem follows a pattern of alternating

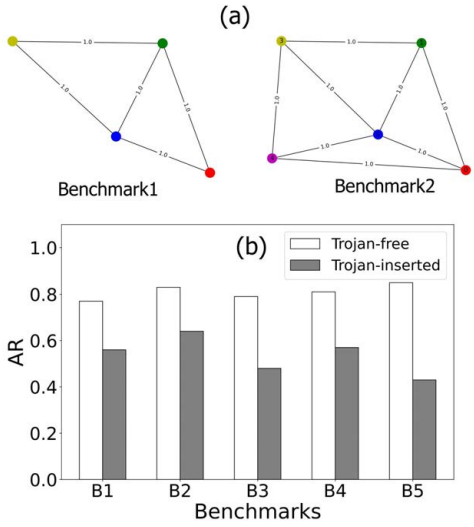

Fig. 4. Evaluation of Trojan-insertion strategy on benchmark QAOA circuits. (a) Sample benchmark graphs for the Max-Cut problem. (b) Approximation ratio comparison between Trojan-free and Trojan-inserted QAOA circuits, showcasing up to 50% degradation in optimization.

mixing and cost operator layers. The mixing layer consists of R_x rotation gates, while the cost layer includes Controlled-X (CX) gates and R_z rotation gates. Trojans disrupt this pattern by inserting extra gates and modifying the circuit structure, thereby breaking the regularity that QAOA circuits exhibit. QTrojanNet can effectively detect Trojans by identifying these disruptions in the circuit's pattern. Fig. 5(a) illustrates this idea further.

B. Dataset Generation

To begin with, we generate a total of 813 unique graphs with 3, 4, and 5 nodes using Qiskit. These graphs are designed to meet the requirements for the Max-Cut problem, ensuring that

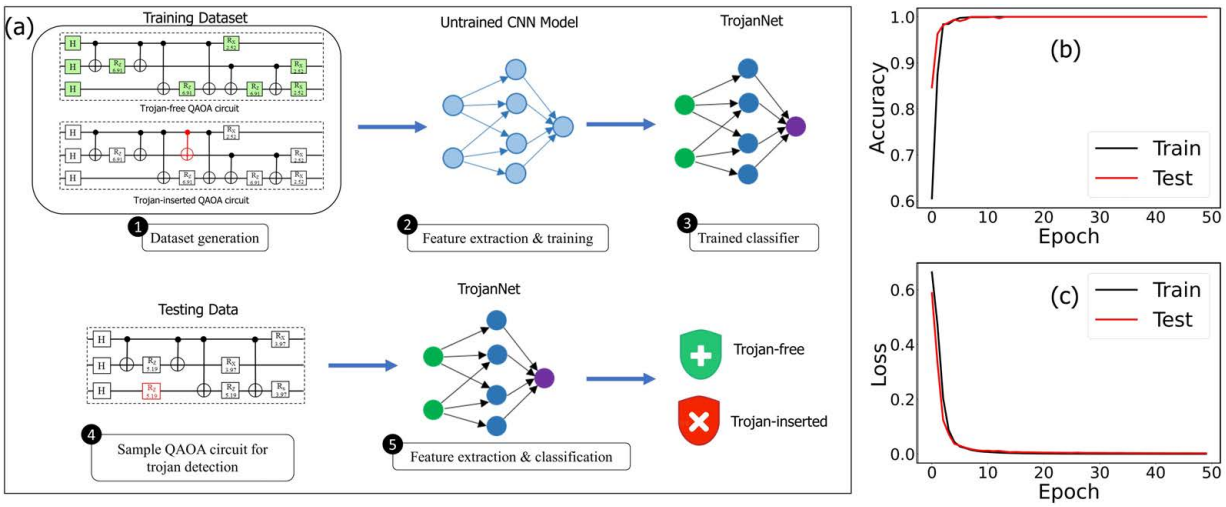

Fig. 5. (a) Overview of QTrojanNet for Trojan detection. (b) Accuracy and (c) Loss vs. Epoch plots demonstrating QTrojanNet's training progress and performance.

each node is connected to at least one edge. Using the QAOA algorithm, we solve the Max-Cut problem for each generated graph and optimize the parameters of the QAOA circuit. The optimized circuits, referred to as 'Trojan-free QAOA circuits', are then compiled using Qiskit's $qasm\_simulator$ backend. Next, we insert a Trojan X gate at the front of the critical path within each non-compiled Trojan-free QAOA circuit by applying the strategy that led to the highest AR loss. These Trojan-inserted circuits are then compiled using the $qasm\_simulator$. Thus, our dataset comprises 813 Trojan-free and 813 Trojan-inserted compiled QAOA circuits.

We further create 11 additional datasets that incorporate variations in Trojan gate types (X/ H/ R_x/ CNOT), number of gates (1 or 2), insertion location (front/ middle) and compiler backend ($qasm\_simulator$/ $FakeManilaV2$). While not all of these Trojan insertion strategies show a significant loss in AR, this approach allows us to assess the efficiency and effectiveness of QTrojanNet across diverse scenarios.

To avoid potential confusion, it should be emphasized that the graphs are solely used for generating the QAOA circuits for the Max-Cut solution. The actual inputs to the QTrojanNet CNN model are the compiled QAOA circuits represented as 2D unitary matrices. These unitary matrices capture the quantum gate sequence and structure of the QAOA circuits optimized for the Max-Cut problems. No explicit conversion or transformation of the graphs themselves is performed for the CNN model input.

C. Architecture of QTrojanNet

QTrojanNet is implemented as a CNN model using the TensorFlow framework, specifically the Keras API [21]. It is specifically designed to effectively differentiate between Trojan-free and Trojan-inserted compiled QAOA circuits by leveraging the inherent characteristics of the circuit representations. The CNN model consists of multiple layers that operate on the 2D representations of the QAOA circuits obtained by converting the Quantum Assembly Language (QASM) files to unitary matrices using Qiskit's *Operator* method. These unitary matrices represent the quantum gates in the circuit, which serve as the input data. The unitary matrices are padded or truncated to a common size to create viable CNN input. While this could potentially impact model accuracy, the information loss appears minimal based on high-accuracy results.

QTrojanNet employs a Convolutional layer with 32 filters (3x3 kernel) for initial feature extraction, using ReLU activation for non-linearity. Followed by a 2x2 MaxPooling layer to reduce dimensionality while capturing key features, it simplifies the data to a 1D vector for processing in a Dense layer of 64 units, again applying ReLU. The final layer, with sigmoid activation, classifies circuits as either Trojan-free or Trojan-inserted, outputting class probabilities. This architecture, optimized through validation, balances feature detection and computational efficiency.

D. Training and Evaluation

The dataset is split 80%-20% into training-testing set for model evaluation. The training methodology involves optimizing the model's weights and biases through an iterative process. The Adam optimizer is employed, which adaptively adjusts the learning rate and performs efficient parameter updates. During training, the QTrojanNet model learns to extract meaningful features and patterns from the circuit representations. The training dataset is split into training and validation sets, enabling the model to generalize well to unseen data. The model is evaluated on the validation dataset to monitor its performance and prevent overfitting during training. The binary cross-entropy loss function is used to measure the discrepancy between the predicted and actual labels. The model aims to minimize this loss by adjusting its parameters.

To assess the performance of the QTrojanNet model, various evaluation metrics are utilized. While accuracy is a commonly used metric, it may not capture the model's robustness against

TABLE I
PERFORMANCE OF QTROJANNET FOR DIFFERENT DATASETS

Backend	Location	Gate Type	# of Gate	Accuracy	Precision	Recall	F1-Score
Qasm	Front	X	1	100.00%	100.00%	100.00%	100.00%
	Front	H	1	100.00%	100.00%	100.00%	100.00%
	Front	R_x	1	98.77%	98.11%	98.11%	98.11%
	Front	CX	1	100.00%	100.00%	99.37%	99.68%
	Middle	R_x	1	98.47%	98.70%	95.60%	97.12%
	Middle	R_x	2	98.77%	98.14%	99.37%	98.75%
FakeManilaV2	Front	X	1	99.03%	98.72%	99.35%	99.04%
	Front	H	1	99.01%	98.69%	99.23%	99.05%
	Front	R_x	1	97.49%	97.30%	97.30%	97.30%
	Front	CX	1	99.35%	99.35%	98.89%	99.03%
	Middle	R_x	1	97.10%	97.23%	94.64%	96.45%
	Middle	R_x	2	97.56%	97.93%	98.49%	97.86%

different scenarios. Therefore, additional metrics, such as precision, recall, and F1-score, are considered, providing insights into the model's performance in correctly identifying both Trojan-free and Trojan-inserted circuits.

E. Results and Analysis

The trained QTrojanNet model is evaluated using the generated datasets. Performance results for a specific dataset, created with the most impactful Trojan insertion strategy (X gate at the front, $qasm\_simulator$ backend), are shown in Fig. 5(b) and (c). The model was trained for 50 epochs, with accuracy and loss monitored. Training and validation accuracies gradually increased, reaching 100% validation accuracy at epoch 10 (Fig. 5(b)). The model loss decreased with increasing epochs, indicating convergence (Fig. 5(c)). Additional evaluation metrics, including precision, recall, and F1-score, were calculated. For this dataset, precision, recall, and F1-score all achieved a perfect score of 100%, demonstrating the model's ability to accurately detect Trojan-inserted circuits.

Table I further presents the evaluation results of QTrojanNet on the remaining datasets. As can be seen, the accuracy of QTrojanNet in detecting and classifying Trojan-inserted QAOA circuits ranges from 97.10% to 100.00%, with an average accuracy of 98.80%. Noticeably, when using the $FakeManilaV2$ backend, the accuracy is slightly lower compared to the $qasm\_simulator$ backend. This discrepancy can be attributed to the increased number of gates in the compiled circuits due to qubit mapping and swap operations, which introduce additional complexity. However, overall performance remains impressive, as indicated by the high precision, recall, and F1-score values across all datasets.

F. Comparison with Classical Hardware Trojan Detection Technique

Table II compares the performance of QTrojanNet with existing ML-based techniques in terms of precision and recall metrics. While we acknowledge that the hardware Trojan detection models compared in Table II were developed for conventional digital circuits [17]–[19], their training details and datasets are not publicly available. Therefore, a direct apples-to-apples comparison through retraining of these models using our dataset is not feasible. However, these works represent the current state-of-the-art in hardware Trojan detection using machine learning techniques. To provide a performance benchmark, we present the published precision and recall metrics of these models on classical circuit Trojan detection tasks. This comparison offers valuable insights into the capabilities of the QTrojanNet, demonstrating its ability to achieve competitive accuracy despite operating on quantum circuits with fundamentally different properties and structures compared to classical digital circuits.

VI. DISCUSSIONS

Bypassing Detection by QTrojanNet: This work focuses on simple Trojan insertion to degrade optimization, but knowledgeable attackers may attempt more advanced, stealthy Trojans that avoid detection. These advanced Trojans will require examining QTrojanNet weaknesses and QAOA properties to counter threats. Techniques like adversarial training, exposing QTrojanNet to diverse Trojans during training, could improve model robustness against subtle Trojan designs.

Diverse Trojan Classes: This study demonstrates QTrojanNet's effectiveness in detecting basic extra gate insertions [9]. However, several classical Trojan categories may not directly apply, given the unique characteristics of quantum computing. Defining a comprehensive quantum-specific Trojan taxonomy and evaluating detection across these Trojan classes remains an open challenge as quantum hardware security matures.

Effects Beyond Approximation Ratio: This analysis focuses on the impact of Trojans on approximation ratio, but examining other effects like timing, reliability etc. would enable a more holistic study. For example, extra gates could increase circuit execution time and degrade outcome quality due to extra decoherence. Further studies on the proposed Trojan model could analyze effects like increased heat generation, which hurts the service provider.

Scalability to Larger Circuits: While this work utilizes small graphs with 3-5 nodes for initial assessment, the complexity may increase for larger quantum circuits. Evaluating on more complex, realistic QAOA implementations can demonstrate better scalability, and exploring efficient CNN implementations can improve feasibility.

Limitations of Classical Hardware Trojan Prevention Techniques for Quantum Circuits: Classical strategies to thwart hardware Trojans, such as logic encryption, involve embedding key gates or modifying circuit elements to obscure the design from attackers, ensuring functionality only with the correct key [24], [25]. Techniques like leveraging hybrid CMOS and emerging devices (e.g., SiNW FETs) for logic encryption [26], or using evolutionary optimization to place encryption gates [27], highlight the adaptability of these strategies. However, the direct application of these key-based encryption methods faces challenges in quantum computing. The necessity to embed key values within quantum circuits exposes them to adversaries, especially when using untrusted

TABLE II
COMPARING THE PERFORMANCE OF QTROJANNET WITH EXISTING ML-BASED HARDWARE TROJAN DETECTION TECHNIQUE

Type of Trojan	Technique Category- Method	Precision	Recall	Reference
Hardware Trojan in conventional circuits	ML - Graph neural network on RTL graph	92%	97%	[17]
	ML - Graph neural network on netlist graph	91%	84%	[17]
	ML - Artificial immune system	87%	85%	[18]
	ML - Gradient boosting algorithm	NA	100%	[19]
	ML - Multi-layer neural networks	NA	90%	[22]
	CA - Socio-network analysis	98%	98%	[23]
Trojan in quantum circuit	ML-Convolutional Neural Network on QASM files	98.68%	98.36%	This work

compilers, illustrating the complexity of securing quantum circuits against hardware Trojans with classical methods.

VII. CONCLUSION

We investigate the vulnerability of QAOA circuits to Trojan insertion during compilation by untrusted third parties and develop QTrojanNet for detection and classification. We identified the most vulnerable insertion location for Trojans in QAOA circuits to be at the front of the critical path, with an X gate having the most detrimental effect. The evaluation of our Trojan-insertion strategy on benchmark QAOA circuits revealed up to a 50% loss in the approximation ratio. We also proposed QTrojanNet for detecting such Trojans which demonstrated remarkable accuracy, reaching 98.80% in accurately differentiating between Trojan-free and Trojan-inserted QAOA circuits. The dataset and the QTrojanNet tool will be released to the public to further the research on quantum security. This work marks an important early step on Trojan vulnerability in quantum computing. However extensive future research is needed to establish comprehensive analysis and defense mechanisms tailored for quantum computing as it continues to mature.

ACKNOWLEDGEMENTS

This work is supported by NSF (CNS-1722557, CNS-2129675, CCF-2210963, CCF-1718474, OIA-2040667, DGE-1723687, DGE-1821766 and DGE-2113839) and Intel's gift.

REFERENCES

[1] National Academies of Sciences, Engineering, and Medicine and others, "Quantum computing: progress and prospects," 2019.

[2] 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.

[3] M. Alam, A. Ash-Saki, and S. Ghosh, "Accelerating quantum approximate optimization algorithm using machine learning," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 686–689.

[4] S. Upadhyay and S. Ghosh, "Obfuscating Quantum Hybrid-Classical Algorithms for Security and Privacy," *arXiv preprint arXiv:2305.02379*, 2023.

[5] S. Das and S. Ghosh, "Randomized Reversible Gate-Based Obfuscation for Secured Compilation of Quantum Circuit," *arXiv preprint arXiv:2305.01133*, 2023.

[6] R. Shaydulin, H. Ushijima-Mwesigwa, C. F. Negre, I. Safro, S. M. Mniszewski, and Y. Alexeev, "A hybrid approach for solving optimization problems on small quantum computers," *Computer*, vol. 52, no. 6, pp. 18–26, 2019.

[7] Z. Computing, "Orquestra," Apr 2023. [Online]. Available: https://www.zapatacomputing.com/orquestra-platform/

[8] C. Q. Computing, "Pytket," Apr 2023. [Online]. Available: https://cqcl.github.io/tket/pytket/api/index.html

[9] Trust-Hub. (2023) Trust-hub. Accessed: 04 Sept. 2023. [Online]. Available: https://www.trust-hub.org/

[10] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.

[11] M. S. Anis, H. Abraham, R. A. AduOffei, G. Agliardi, M. Aharoni, I. Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, M. Amy, S. Anagolum *et al.*, "Qiskit: An open-source framework for quantum computing," *Qiskit/qiskit*, 2021.

[12] J. Håstad, "Some optimal inapproximability results," *Journal of the ACM (JACM)*, vol. 48, no. 4, pp. 798–859, 2001.

[13] A. Suresh, A. A. Saki, M. Alam, D. S. Ghosh *et al.*, "A quantum circuit obfuscation methodology for security and privacy," *arXiv preprint arXiv:2104.05943*, 2021.

[14] A. A. Saki, A. Suresh, R. O. Topaloglu, and S. Ghosh, "Split Compilation for Security of Quantum Circuits," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–7.

[15] S. Upadhyay, R. O. Topaloglu, and S. Ghosh, "Trustworthy Computing using Untrusted Cloud-Based Quantum Hardware," *arXiv preprint arXiv:2305.01826*, 2023.

[16] K. G. Liakos, G. K. Georgakilas, S. Moustakidis, P. Karlsson, and F. C. Plessas, "Machine learning for hardware Trojan detection: A review," in *2019 Panhellenic Conference on Electronics & Telecommunications (PACET)*. IEEE, 2019, pp. 1–6.

[17] R. Yasaei, L. Chen, S.-Y. Yu, and M. A. Al Faruque, "Hardware Trojan detection using graph neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[18] F. Zareen and R. Karam, "Detecting RTL trojans using artificial immune systems and high level behavior classification," in *2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, 2018, pp. 68–73.

[19] T. Han, Y. Wang, and P. Liu, "Hardware Trojans detection at register transfer level based on machine learning," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, pp. 1–5.

[20] S. Das and S. Ghosh, "Quantum-TrojanNet," https://github.com/das-subrata/Quantum-TrojanNet/tree/main, 2024, accessed: 2024-02-16.

[21] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[22] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists using multi-layer neural networks," in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2017, pp. 227–232.

[23] S. A. Islam, F. I. Mime, S. Asaduzzaman, and F. Islam, "Socio-network analysis of RTL designs for hardware trojan localization," in *2019 22nd International Conference on Computer and Information Technology (ICCIT)*. IEEE, 2019, pp. 1–6.

[24] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Transactions on computers*, vol. 64, no. 2, pp. 410–424, 2013.

[25] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Proceedings of the conference on Design, automation and test in Europe*, 2008, pp. 1069–1074.

[26] Q. Alasad, J.-S. Yuan, and Y. Bi, "Logic locking using hybrid CMOS and emerging SiNW FETs," *Electronics*, vol. 6, no. 3, p. 69, 2017.

[27] A. Marcelli, M. Restifo, E. Sanchez, and G. Squillero, "An evolutionary approach to hardware encryption and trojan-horse mitigation," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 1593–1598.