

Quantum Approximate Optimization Algorithm for Test Case Optimization

Xinyi Wang, Shaukat Ali, Tao Yue, and Paolo Arcaini

Abstract—Test case optimization (TCO) reduces the software testing cost while preserving its effectiveness. However, to solve TCO problems for large-scale and complex software systems, substantial computational resources are required. Quantum approximate optimization algorithms (QAOAs) are promising combinatorial optimization algorithms that rely on quantum computational resources, with the potential to offer increased efficiency compared to classical approaches. Several proof-of-concept applications of QAOAs for solving combinatorial problems, such as portfolio optimization, energy optimization in power systems, and job scheduling, have been proposed. Given the lack of investigation into QAOA's application for TCO problems, and motivated by the computational challenges of TCO problems and the potential of QAOAs, we present IGDec-QAOA to formulate a TCO problem as a QAOA problem and solve it on both ideal and noisy quantum computer simulators, as well as on a real quantum computer. To solve bigger TCO problems that require many qubits, which are unavailable these days, we integrate a problem decomposition strategy with the QAOA. We performed an empirical evaluation with five TCO problems and four publicly available industrial datasets from ABB, Google, and Orona to compare various configurations of IGDec-QAOA, assess its decomposition strategy of handling large datasets, and compare its performance with classical algorithms (i.e., Genetic Algorithm (GA) and Random Search). Based on the evaluation results achieved on an ideal simulator, we recommend the best configuration of our approach for TCO problems. Also, we demonstrate that our approach can reach the same effectiveness as GA and outperform GA in two out of five test case optimization problems we conducted. In addition, we observe that, on the noisy simulator, IGDec-QAOA achieved similar performance to that from the ideal simulator. Finally, we also demonstrate the feasibility of IGDec-QAOA on a real quantum computer in the presence of noise.

Index Terms—QAOA, test case optimization, quantum computing, search-based software engineering.

1 INTRODUCTION

TO ensure the quality and reliability of software systems, software testing is extremely vital in the software development life cycle. However, with the growing size and complexity of software systems, traditional testing approaches, such as unit testing and integration testing, face challenges regarding effectiveness and efficiency since a large number of test cases need to be executed. Indeed, such test case execution is always limited by available time and resource budgets. This leads to the development of test case optimization (TCO) techniques, which aim to reduce the number of test cases to execute to minimize the testing cost (e.g., execution time) while preserving its effectiveness (e.g., code coverage). TCO problems are combinatorial optimization problems and are NP-hard [1], [2]. Thus, with the increasing software scale, even larger classical computational resources cannot explore the significant part of the vast search space. Consequently, traditional TCO approaches face computational challenges when optimizing test cases for classical software systems.

Quantum Computing (QC) is a novel computational paradigm that can potentially tackle these computational challenges. QC has been expanding in recent years, with the potential to perform certain complex calculations faster than classical computers by leveraging quantum mechanics prin-

ciples. QC's applications in various domains (e.g., finance, complex physics and chemistry simulations, and drug discovery [3], [4], [5]) have emerged. Among all these potential applications, optimization applications are expected to demonstrate a quantum advantage in the near future, especially for solving combinatorial optimization problems. Quantum Approximate Optimization Algorithm (QAOA) is one of the promising algorithms [6], [7] for solving such problems. It is a hybrid algorithm that contains a variational quantum circuit and a classical optimizer, to explore large search space efficiently, and approximate the optimal solution of a combinatorial optimization problem. QAOA applications have increasingly emerged and demonstrated the capability of solving optimization problems, such as scheduling [8], portfolio optimization [9], and optimization in power systems [10].

To deal with the computational challenges of TCO problems and demonstrate the potential of QAOA as a possible solution, we present the first application of QAOAs for solving TCO problems, i.e., test case selection (TCS) and test case minimization (TCM) problems in the context of testing classical software systems. In particular, we propose an approach named IGDec-QAOA, based on QAOAs, to solve TCO problems, by making the following contributions. First, we demonstrate how to formulate optimization objectives for TCO problems as a generic function in the Ising formulation [11] such that QAOAs can be applied to solve them. Second, to solve problems with a large number of test cases that cannot be directly solved with current quantum computers having a limited number of qubits, we apply an impact-guided decomposition strategy adapted

- Xinyi Wang is with Simula Research Laboratory, Oslo, Norway and University of Oslo, Oslo, Norway.
 - Shaukat Ali is with Simula Research Laboratory, Oslo, Norway and Oslo Metropolitan University, Oslo, Norway.
 - Tao Yue is with Simula Research Laboratory, Oslo, Norway.
 - Paolo Arcaini is with National Institute of Informatics, Tokyo, Japan.
- Manuscript received April 19, 2005; revised August 26, 2015.

from D-Wave [12] to break down a complex TCO problem into a set of smaller and more manageable sub-problems so that they can be solved with current quantum computers.

Third, we perform an empirical evaluation with five publicly available industrial case studies from ABB, Google, and Orona to demonstrate IGDec-QAOA's capability of finding approximate optimal solutions for TCO problems. In particular, we study: (a) Which configurations of IGDec-QAOA are the best for solving TCO problems? (b) Does the impact-guided decomposition of IGDec-QAOA contribute to its performance? (c) How is the performance of IGDec-QAOA as compared to classical algorithms, i.e., Genetic Algorithm (GA) and Random Search (RS)? (d) How is the performance of IGDec-QAOA on a noisy quantum computer? Given that current quantum computers inherently suffer from noise in their computations, we also compare the results of IGDec-QAOA on the ideal simulator with that of the noisy simulator to evaluate the impact of noise. Finally, we assess the feasibility of IGDec-QAOA on IBM's quantum computer by solving a small-scale TCM problem in the presence of noise.

Based on the results of our empirical evaluation on the ideal simulator, we select the best configurations of IGDec-QAOA to solve the TCO problems. Our results show that the impact-guided decomposition strategy of IGDec-QAOA indeed contributes to its performance in finding optimal solutions. In addition, the effectiveness of our approach is on par with GA, with 2 (out of 5) TCO problems outperforming GA. This demonstrates the benefit of using IGDec-QAOA for solving TCO problems with currently available small-scale quantum computers. In addition, the results of the experiment comparing IGDec-QAOA on ideal and noisy simulator showed that the performance of IGDec-QAOA on noisy simulator is comparable to that on the ideal simulator. Regarding the experiment on IBM's quantum computer for solving a small-scale TCM problem, IGDec-QAOA also reached the optimal solution obtained by the ideal simulator, thereby providing evidence about the feasibility of running IGDec-QAOA on real quantum computers. These findings also demonstrate that the impact of noise on IGDec-QAOA is insignificant. We envision a further improvement of IGDec-QAOA's performance in the future when large-scale quantum computers will be available. We provide detailed experiment results, code, and data in the repository: <https://doi.org/10.5281/zenodo.13911651>.

Remark 1. This paper does not intend to demonstrate the quantum advantage of solving TCO problems with QAOA over classical approaches. This is practically impossible, since only small-scale noisy quantum computers are currently available. Rather, we demonstrate how TCO problems can be formulated and solved with QAOA by achieving comparable performance as classical approaches with manageable execution costs, which should be considered as a cornerstone for future quantum computing applications in solving software engineering optimization problems.

2 BACKGROUND

2.1 Test Case Optimization

Delivering software on time while ensuring its reliability is a major challenge in the software industry. It is necessary

TABLE 1: Descriptions of gates used in QAOA

Gate	Description
<i>Hadamard (H)</i>	It puts a qubit into an equal superposition, i.e., giving an equal chance (i.e., 50%) of being in state $ 0\rangle$ or $ 1\rangle$, when measured. Applying an H gate on a qubit with $ 0\rangle$ state creates a state of $\frac{1}{\sqrt{2}} 0\rangle + \frac{1}{\sqrt{2}} 1\rangle$, which can be denoted as $ +\rangle$ state.
<i>Pauli-z</i>	It rotates a qubit around the z-axis with π radians represented as σ^z .
<i>Pauli-x</i>	It rotates a qubit around the x-axis with π radians. It can be represented as σ^x .
<i>Rz(θ)</i>	It rotates a qubit around the z-axis with θ radians. It is represented as $e^{-i\theta/2\sigma^z}$.
<i>Rx(θ)</i>	It rotates a qubit around the x-axis with θ radians. It can be represent as $e^{-i\theta/2\sigma^x}$.
<i>Controlled-NOT (C-NOT)</i>	It is a two-qubit gate consisting of a control qubit and a target qubit. If the control qubit is in state $ 1\rangle$, the target qubit rotates around the x-axis with π radians.

to conduct software testing efficiently while keeping high effectiveness; to this aim, test case selection (TCS) and test case minimization (TCM) are commonly used techniques that choose a test suite from a test case pool of software under test (SUT). The main goal of TCS is to ensure that the most relevant and important tests are run, optimizing resource use while maintaining test effectiveness. TCM focuses on eliminating redundant or non-essential test cases while maintaining essential functionality and coverage. Both techniques aim to solve combinatorial optimization problems, seeking to find tradeoff solutions among several conflicting objectives, such as execution time, code coverage, and failure rate.

2.2 Quantum Computing

Quantum Computing (QC) utilizes quantum bits (*qubits*) for performing computations. Qubits are different from classical bits since a classical bit can be in a state of either 0 or 1 at a time, while each qubit can exist in a superposition of both states until measurement, which then collapses the qubit to a definite state of either 0 or 1, resulting in a classical bit. A one-qubit quantum state can be represented in the Dirac notation as: $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, where α_0 and α_1 represents the *amplitude* of the quantum state and is a complex number. $|\alpha_0|^2$ and $|\alpha_1|^2$ determine the probabilities of being in states $|0\rangle$ and $|1\rangle$ respectively when measured. The sum of $|\alpha_0|^2$ and $|\alpha_1|^2$ shall be equal to 1. Analogously to classical logic gates used in classical computers, we use quantum gates operated on qubits to process quantum information. These quantum gates are represented by unitary matrices, also called unitary operators. We provide quantum gate descriptions in Table 1 that are relevant to building QAOA quantum circuits.

2.3 Quantum Approximate Optimization Algorithm (QAOA)

QAOA is a promising hybrid quantum-classical algorithm proposed to tackle combinatorial optimization problems [6], [7]. It is a variational quantum algorithm that aims to find the approximate optimal solution to an optimization

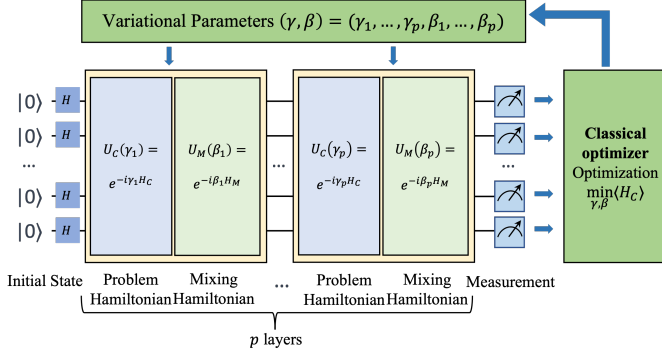


Fig. 1: A typical configuration of QAOA (p is the depth)

problem. A parameterized quantum circuit is at the core of QAOA, whose parameters are optimized with a classical optimizer. The objective of the parameterized circuit is to approximate the adiabatic evolution of a quantum system, starting with an initial Hamiltonian¹ and reaching the ground state that represents the solution, i.e., the final Hamiltonian. Fig. 1 shows a typical configuration of QAOA.

To encode a combinatorial optimization problem for QAOA, an objective function should be defined in the form of an Ising formulation [11]. Such formulation is expressed as a quadratic formulation with N binary variables, denoted by z_i , which takes values of either +1 or -1.

$$C(\vec{z}) = \sum_{i=1}^N h_i z_i + \sum_{i=1}^N \sum_{j=i+1}^N J_{i,j} z_i z_j, \quad z_i \in \{+1, -1\} \quad (1)$$

In Eq. 1, $C(\vec{z})$ is the objective function, and $\vec{z}$ represents the bit string. h_i is the linear coefficient and $J_{i,j}$ represents the quadratic coefficient. With this objective function, a classical problem can be converted into a problem Hamiltonian that can be solved by QAOA.

As depicted in Fig. 1, QAOA has the following three main components.

Initial state. It is the starting point for the evolution of a quantum system represented by a quantum circuit, where each qubit represents a variable in Eq. 1. Commonly, we initialize the quantum system into an equal superposition of all its qubits by applying Hadamard gates (detailed description is shown in Table 1) on all qubits in $|0\rangle$ state, by which we put the quantum system in an equal superposition of all possible solutions to the problem.

Problem Hamiltonian. In this quantum system, each qubit represents a variable of the optimization problem. The problem Hamiltonian encodes the objective function of Eq. 1 by mapping each variable z_k to a corresponding quantum spin σ_k^z , which is the *Pauli-z* gate shown in Table 1. We use H_C to represent the problem Hamiltonian, and it is known as an Ising Hamiltonian, as formulated below:

$$H_C = C(\sigma_1^z, \sigma_2^z, \dots, \sigma_N^z) \quad (2)$$

H_C is a matrix whose diagonal elements correspond to all the possible values of $C(\vec{z})$. The ground state of H_C

1. Hamiltonian is a Hermitian operator that describes the quantum state evolution in a quantum system [13].

represents the solution to the optimization problem. To simulate the problem Hamiltonian with a quantum circuit, we need to leverage a unitary operator to be applied on the circuit, which is defined as:

$$U_C(\gamma) = e^{-i\gamma H_C} \quad (3)$$

In this equation, e is the natural logarithm, i is the imaginary unit, and γ represents the variational parameter of the QAOA algorithm, which determines the rotation amount of each qubit around the z-axis.

Mixing Hamiltonian. The mixing Hamiltonian enables the algorithm to explore different parts of the search space and avoids getting stuck in local optima. It is typically chosen as:

$$H_M = \sum_{k=1}^N \sigma_k^x \quad (4)$$

σ_k^x is the *Pauli-x* gate (shown in Table 1). A corresponding unitary operator is defined as:

$$U_M(\beta) = e^{-i\beta H_M} \quad (5)$$

where β is another variational parameter of the QAOA algorithm and determines the rotation amount of qubits around the x-axis.

The QAOA process is shown in Fig. 1. The quantum circuit is first initialized with the Hadamard (H) gates. According to Table 1, each qubit will be in $|+\rangle$ state. Thus, we represent the whole circuit as in $|+\rangle^{\otimes N}$ state.² We then apply the problem Hamiltonian and mixing Hamiltonian alternately p times, which determines the depth of the QAOA algorithm (i.e., Fig. 1 shows a QAOA circuit of p layers). Increasing the value of p can theoretically improve the approximation quality and increase the required computational resources. With the unitary operators defined in Eq. 3 and Eq. 5, we calculate the quantum state as follows:

$$\begin{aligned} |\psi_p(\vec{\gamma}, \vec{\beta})\rangle &= U_C(\beta_p)U_C(\gamma_p), \dots, U_C(\beta_1)U_C(\gamma_1) |+\rangle^{\otimes N} \\ &= e^{-i\gamma_p H_C} e^{-i\beta_p H_M}, \dots, e^{-i\gamma_1 H_C} e^{-i\beta_1 H_M} |+\rangle^{\otimes N} \end{aligned} \quad (6)$$

Since the values of γ and β vary across different layers, the equation is defined with $2p$ parameters ($\vec{\beta} = \beta_1, \dots, \beta_p$ and $\vec{\gamma} = \gamma_1, \dots, \gamma_p$). The QAOA algorithm needs to find the optimal values of $\vec{\beta}$ and $\vec{\gamma}$ in order to find the minimum expectation value of H_C , which corresponds to the optimal solution $\vec{z}_{opt}$. The expectation value is calculated in the following:

$$E_C(\vec{\gamma}, \vec{\beta}) = \langle \psi_p(\vec{\gamma}, \vec{\beta}) | H_C | \psi_p(\vec{\gamma}, \vec{\beta}) \rangle \quad (7)$$

The expectation value $E_C(\vec{\gamma}, \vec{\beta})$ determines the results we measure from the quantum circuit. With such results, we can extract the solutions for $C(\vec{z})$. Specifically, the observed value of each qubit corresponds to the value of the variable z_k that it represents.³

In the QAOA algorithm, a classical optimizer is used to search for the optimal $\vec{\gamma}$ and $\vec{\beta}$ values. According to Fig. 1,

2. For a quantum circuit with multiple qubits, we use the Kronecker product to describe the collective states.

3. Taking the z-axis as the computational basis, if a qubit is observed as 0, the corresponding variable value is -1. Otherwise, the related variable value is +1.

in each iteration, after the measurement, the classical optimizer updates the quantum circuit with the newly achieved parameters. The whole process repeats until a stopping condition is met.

3 METHODOLOGY

This section defines a generic Ising formulation for TCO problems (Sect. 3.1), illustrated with a running example followed by QAOA circuit (Sect. 3.2), and the process of IGDec-QAOA (Sect. 3.3).

3.1 Ising formulation for test case optimization

We propose an Ising formulation for test case selection (TCS) and test case minimization (TCM) problems to be solved with QAOA.

Definition 1 (Test case selection (TCS)). Given a test suite $T = \{t_0, \dots, t_{n-1}\}$ with a set of testing attributes for the software under test and the corresponding testing objectives $F = \{f_0, \dots, f_{m-1}\}$, TCS aims to select a subset $T' \subset T$ to satisfy all objectives as much as possible. Each objective is associated with one of the attributes of the test cases.

Definition 2 (Test case minimization (TCM)). Given a test suite $T = \{t_0, \dots, t_{n-1}\}$ with a set of testing attributes for the software under test and the corresponding testing objectives, TCM aims to select minimum subset of test cases $T' \subset T$ that satisfy all testing objectives as much as possible. The testing objectives $F = \{f_0, \dots, f_{m-1}\}$ include ones associated with attributes of the test cases and the one specific objective aiming to minimize the size of T' , which makes TCM different from TCS.

To construct the objective function of the TCO problem as an Ising formulation, we first represent z_i as the decision variable for selecting the test case t_i . $z_i = -1$ denotes that t_i is **selected** and $z_i = +1$ that is **not selected**. Thus, the selection of all test cases is represented with a bit string $\vec{z} \in \{-1, +1\}^n$.

TCO problems maximize the effectiveness of the selected test cases in test suite T' , while minimizing the cost when executing the selected test cases. We divide all testing objectives into two groups according to their corresponding testing attributes: *effectiveness objectives* and *cost objectives*. Specially, for TCM problems, the objective of minimizing the size of selected test suite T' belongs to the *cost objectives* group. We minimize a cost-effective overall objective function for the TCO problem to seek the optimal solution $\vec{z}^*$, which finds an optimal trade-off between these two attribute groups to find the approximate optimal solution that minimizes the corresponding Ising formulation.

We represent the objective function for TCO problems as $\mathcal{O}(f_0, f_1, \dots, f_{m-1})$. Since different attribute values are obtained with various metrics, we normalize all objectives ranging from 0 to 1 to ensure comparability.

Given the k -th objective in the *effectiveness objectives* group, we represent the corresponding attribute values of all test cases as $\mathbf{v}_k = \{v_0^k, \dots, v_{n-1}^k\}$ and build the objective as follows:

$$f_k(\vec{z}) = \frac{1}{2} \left(1 + \frac{\sum_{i=0}^{n-1} v_i^k z_i}{\sum_{i=0}^{n-1} v_i^k} \right), \quad z_i \in \{+1, -1\} \quad (8)$$

We normalize the range of the objective between 0 and 1. If a test case z_i is selected (i.e., $z_i = -1$), the corresponding attribute value v_i^k decreases, thus reducing the objective value. In contrast, if a test case is not selected (i.e., $z_i = +1$), the attribute value is added to the objective value as a penalty.

Given the k -th objective in the *cost objectives* group, the corresponding attribute values of all test cases are represented as $\mathbf{c}_k = \{c_0^k, \dots, c_{n-1}^k\}$. For TCM problems, with respect to the objective of minimizing the size of the selected test suite, the attribute value for each test case is assigned as 1. We build the corresponding objective in the following:

$$f_k(\vec{z}) = \frac{1}{2} \left(1 - \frac{\sum_{i=0}^{n-1} c_i^k z_i}{\sum_{i=0}^{n-1} c_i^k} \right), \quad z_i \in \{+1, -1\} \quad (9)$$

The value of the objective is still in the range of 0 and 1. If a test case z_i is selected ($z_i = -1$), the corresponding attribute value c_i^k will be added to the objective value as a penalty since a higher number of selected test cases will increase the cost; otherwise, the attribute value will contribute towards decreasing the objective value.

For each objective, to emphasize the larger deviation between the obtained value and the theoretical optimal value (i.e., 0), we square all objectives when combined to amplify the impact and penalize the improper selection. For TCO problems, we formulate the overall objective function $\mathcal{O}(f_0, f_1, \dots, f_{m-1})$ as:

$$\min \mathcal{O}_{TCO}(\vec{z}) = w_0(f_0)^2 + \dots + w_{m-1}(f_{m-1})^2 \quad (10)$$

$w_0, w_1, \dots, w_{m-1}$ are weights assigned to various objectives, indicating the degrees of importance of each objective, based on user requirements in practice. These weights, which sum up to 1, $\sum_{i=0}^m w_i = 1$, can be freely chosen based on user preferences, and in some cases, domain experts can provide the weight values. While the selection of weights changes the optimal solution of the problem, the performance of IGDec-QAOA is not dependent on the weights selected.

Suppose there are s objectives in the *effectiveness objectives* group and q objectives in the *cost objectives* group ($s + q = m$). Since $z_i \in \{-1, +1\}$, it holds $z_i^2 = 1$. We expand the objective function as:

$$\begin{aligned} \mathcal{O}_{TCS}(\vec{z}) &= \sum_{\langle i,j \rangle} X_{ij} z_i z_j + \sum_i Y_i z_i + \text{Const} \quad (11) \\ X_{ij} &= \sum_{k=0}^{s-1} w_k \frac{v_i^k v_j^k}{\left(\sum_{l=0}^{n-1} v_l^k \right)^2} - \sum_{k=0}^{q-1} w_{k+s} \frac{c_i^k c_j^k}{\left(\sum_{l=0}^{n-1} c_l^k \right)^2} \\ Y_i &= \sum_{k=0}^{s-1} w_k \frac{v_i^k}{\sum_{j=0}^{n-1} v_j^k} - \sum_{k=0}^{q-1} w_{k+p} \frac{c_i^k}{\sum_{j=0}^{n-1} c_j^k} \end{aligned}$$

3.2 Realizing TCO in a QAOA Circuit

From the objective function in Eq. 11, we first remove *Const* because it does not affect the optimization process and only changes the absolute value of the fitness value. Then, we replace each variable z_j with σ_j^z to create the corresponding problem Hamiltonian.

$$H_C = \sum_{\langle j,k \rangle} X_{j,k} \sigma_j^z \sigma_k^z + \sum_j Y_j \sigma_j^z \quad (12)$$

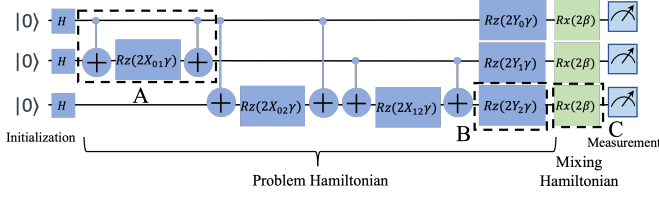


Fig. 2: QAOA example circuit. A typical 1-layer circuit from the literature is used for TCO problems with 3 test cases. Each test case corresponds to one qubit

When creating the quantum circuit, we represent each test case with one qubit. σ_j^z corresponds to placing a *Pauli-z* gate on the j -th qubit, which is explained in Sect. 2. To simulate the problem Hamiltonian in a quantum circuit, we need to create the unitary operator:

$$U_C(\gamma) = e^{-i\gamma H_C} = e^{-i\gamma \sum_{(j,k)} X_{j,k} \sigma_j^z \sigma_k^z} e^{-i\gamma \sum_j Y_j \sigma_j^z} \quad (13)$$

According to [14], to create the unitary operator $e^{-j\gamma Y_j \sigma_j^z}$ related to the j -th test case (i.e., the j -th qubit), we use the $Rz(\theta)$ defined in Table 1 in Sect. 2. In a quantum circuit, we can apply the gate on the j -th qubit with rotation $\theta = 2Y_j\gamma$ around the z -axis where Y_j is the corresponding coefficient of the linear term in the objective function, and γ is the parameter of problem Hamiltonian to optimize (introduced in Sect. 2). For each qubit, if the corresponding linear coefficient Y_j is different from 0, we apply gates inside the box labeled B in Fig. 2 on the j -th qubit.

For the unitary operator $e^{-i\gamma X_{j,k} \sigma_j^z \sigma_k^z}$ related to the j -th and k -th test cases (i.e., the j -th and k -th qubits), according to [14], we can use the *CNOT* gate first to entangle two qubits, then apply the $Rz(\theta)$ gate followed by another *CNOT* gate, which creates the unitary gate $e^{-i\sigma_j^z \sigma_k^z \theta/2}$. We depict this gate combination inside the box labeled A in Fig. 2. The angle parameter of $Rz(\theta)$ gate is $\theta = 2X_{jk}\gamma$, where X_{jk} is the coefficient of the corresponding interaction term. For every pair of qubits, if the corresponding coefficient $X_{jk} \neq 0$, we apply the set of gates to the two qubits.

Similarly, for the mixing Hamiltonian,

$$H_M = \sum_{j=1}^N \sigma_j^x, \quad U_C(\beta) = e^{-i\beta H_M} = e^{-i\beta \sum_{j=1}^N \sigma_j^x} \quad (14)$$

Since $Rx(\theta)$ generates unitary operator of $e^{-i\theta/2\sigma_j^x}$, which rotates θ angle around the x -axis (see Sect. 2), the angle parameter should be $\theta = 2\beta$ for each qubit to create the unitary operator $e^{-i\beta\sigma_j^x}$, where β is the parameter of the mixing Hamiltonian to be optimized (introduced in Sect. 2). Thus, for the mixing Hamiltonian, we apply a gate of the box labeled C on each qubit as shown in Fig. 2.

Example 1. We demonstrate a QAOA circuit with depth $p = 1$ using a small-scale running example for a TCM problem to construct an objective function. Assume we have three test cases t_0 , t_1 , and t_2 , with their failure rate values of 50%, 70%, and 80%, respectively, and execution time of 3, 6, and 1 second, respectively. The corresponding objective of the failure rate is an *effectiveness objective*, and the objective for execution time is a *cost objective*. Together with the objective of minimizing the number of selected

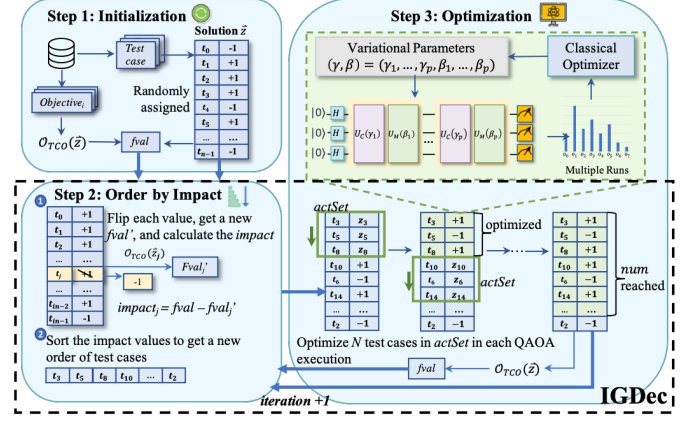


Fig. 3: Process of IGDec-QAOA. It has three steps: initialization, test case ordering by impact, and optimization

test cases, we assign equal weights (i.e., 1/3) as an example to the three objectives. Note that any other weights can be selected depending on the user's preference. We build the objective function below according to Eq. 11:

$$\mathcal{O}_{TCM}(\mathbf{z}) = X_{01}z_0z_1 + X_{02}z_0z_2 + X_{12}z_1z_2 + Y_0z_0 + Y_1z_1 + Y_2z_2 + Const \quad (15)$$

$$\begin{aligned} X_{01} &= 0.063, X_{02} = 0.040, X_{12} = 0.052, \\ Y_0 &= -0.064, Y_1 = -0.097, Y_2 = -0.0069, \end{aligned} \quad (16)$$

The quantum circuit is shown in Fig. 2. Three test cases are mapped to 3 qubits. We first initialize all qubits in $|0\rangle$ state with the Hadamard gate. Next, we create the circuit for the problem Hamiltonian. Because the coefficients of all interaction terms are not 0, we need to entangle all three qubits with each other using the gates set in box A . Next, regarding all linear terms, we apply the Rz gate on each qubit. Then, the mixing Hamiltonian is created with the Rx gate on each qubit. Since $p = 1$, we directly apply measurements on each qubit to read the problem's possible solution. If $p > 1$, the problem Hamiltonian and mixing Hamiltonian should be applied p times alternately before the measurement.

3.3 Process of IGDec-QAOA

Currently, available quantum computers with limited qubits cannot directly solve TCO problems with many test cases with QAOA. Thus, we combine an impact-guided decomposition (IGDec) strategy adapted from *qbsolv* [12] by D-Wave with QAOA to solve TCO problems as a classical-quantum hybrid approach with the Ising formulation and QAOA circuit described before. In the IGDec strategy, the decomposition of the problem is determined by the *impact* value of each test case. Fig. 3 shows the overview of IGDec-QAOA that has the three main steps described below, and the IGDec strategy is composed of Step 2 and Step 3.

Step 1. Initialization. Given a test suite containing n test cases, we initially assign random values (i.e., -1 or $+1$) to the decision variables associated with each test case. We represent the bit string as $\bar{z}$. With the objective function of Eq. 11, we get an initial fitness value $fval$.

Step 2. Ordering test cases by *impact*. This step constitutes the ordering phase of the IGDec strategy. We first calculate the *impact* value of each test case, and then sort the test cases according to this value. This impact value indicates how much a test case contributes to reducing the fitness value of the TCO problem when its decision variable is altered. We use the calculation of the impact value of t_k as an example. Taking the bit string of the solution $\vec{z}$ and the associated fitness value $fval$, for a test case t_k , we flip the value of its decision variable z_k (i.e., -1 to $+1$ or $+1$ to -1) to get a new bit string and the corresponding new fitness value $fval'_k$. Next, we calculate the difference between $fval$ and $fval'_k$, which is the *impact* of t_k (i.e., $impact_k = fval'_k - fval$). In this way, we calculate the impact values of each test case and sort them based on the values. Since we are solving a minimization problem, we arrange the test cases in ascending order of impact values. The test cases with lower impact values are positioned towards the front because they significantly decrease the fitness value when flipped.

Step 3. Optimization. This step constitutes the decomposition phase of the IGDec strategy. With the rearranged test cases, we decompose the problem and run QAOA iteratively to get the approximate optimal solution. We first determine the sub-problem size N that a quantum computer can afford. During the optimization process, we always optimize over a set of N variables in each QAOA execution, which belong to the *active* variable set $actSet$. For other variables, the values are fixed, maintaining the current values. Thus, with the overall objective function from Eq. 11, irrespective of the constant term, we get the objective function:

$$\mathcal{O}_{TCO}(\vec{z}) = \sum_{i,j \in actSet} X_{ij} z_i z_j + \sum_{i \in actSet} (Y_i + \sum_{j \notin actSet} X_{ij} z_j^*) z_i \quad (17)$$

z_j^* refers to the fixed values of variables not in set $actSet$. We optimize the variables (i.e., test cases) in order. In the initial execution, we select the first N variables as the *active* variables. Then, we run the QAOA algorithm on the objective function to get optimized values of test cases in the *active* variables. Each variable in $actSet$ corresponds to one qubit. Next, we optimize the next N variables, with the first N variables keeping the current optimized values. We continuously run the QAOA algorithm until *num* test case variables are optimized. With the new bit string $\vec{z}$, we get the new fitness value $fval$. If the stopping condition is not met, we return to **Step 2** and start a new iteration with the new bit string $\vec{z}$ and $fval$, then arrange the test cases again.

Throughout the process of IGDec-QAOA, we identify and mark the best $fval$ (i.e., the lowest fitness value) and the corresponding bit string $\vec{z}$ (i.e., the solution). We repeat **Step 2** and **Step 3** until the stopping condition is achieved. Upon reaching this condition, the marked $\vec{z}$ is recognized as the optimal solution to the problem.

4 EXPERIMENT DESIGN

4.1 Research Questions

RQ1 How do the number of layers and the sub-problem size affect the performance of IGDec-QAOA?

RQ2 How is the performance of IGDec-QAOA compared with that of the baseline QAOA approach?

RQ3 How is the performance of IGDec-QAOA compared with that of classical approaches?

RQ4 How is the performance of IGDec-QAOA on a noisy quantum computer?

RQ1 studies how the two key parameters of IGDec-QAOA affect its performance. The first parameter is the depth of the QAOA circuit (i.e., the number of layers p). A higher value of p incurs higher cost in terms of execution. It is, therefore, important to study which number of layers is optimal for TCO problems. The second parameter is the sub-problem size (i.e., N) to be solved on QAOA and how a particular sub-problem size affects the performance of IGDec-QAOA. This aspect helps us study which sub-problem size is optimal for TCO. In addition, we study whether a higher number of N incurs more execution time, i.e., studying the cost associated with N .

RQ2 studies whether IGDec-QAOA's test case ordering by the IGDec strategy helps the QAOA to find approximately optimal solutions. To this end, we replace IGDec-QAOA's IGDec strategy with a random strategy while keeping the QAOA part the same.

RQ3 compares IGDec-QAOA with two classical search algorithms: Random Search (RS) and a Genetic Algorithm (GA). We assess whether we can obtain optimal solutions close to, or even outperform, the classical search algorithm, i.e., GA, whereas comparison with RS ensures that we solve a complex problem that cannot be solved randomly.

RQ4 analyzes the effect of noise on IGDec-QAOA. We run QAOA on a noisy simulator and compare the results with those obtained by IGDec-QAOA on an ideal simulator, as well as those obtained by GA. Moreover, we run a small case study on IBM's quantum computer, which aims to demonstrate the feasibility of our approach on a real quantum computer in the presence of noise.

4.2 Experiment Setup

This section discusses datasets we used for experiments, baselines, parameter settings, and execution environment.

4.2.1 Datasets

We employed four publicly available industrial datasets.

The *Paint Control* dataset contains 90 test cases, and the *IOF/ROL* dataset has 1941 test cases. Both of them are from ABB Robotics Norway [15]. The *GSDTSR* dataset⁴ from Google has 5555 test cases. Each test case contains two attributes: "execution time" and "failure rate". The first attribute contains a value telling how much time it was taken on average by a test case to execute, whereas the failure rate determines the possibility of a test case failing based on the historical ratio of failures to the total number of test case executions. We run experiments on these datasets for the TCM problems. To do so, we encode the objective of minimizing the number of selected test cases. The overall fitness function objectives are to minimize the test execution cost and the number of selected test cases while maximizing the fault detection rate of selected test cases.

4. <https://code.google.com/archive/p/google-shared-dataset-of-test-suite-results/wikis/DataFields.wiki>

The *ELEVATOR* dataset is from Orona⁵ [16], which was produced when testing a dispatcher, a software component in elevators responsible for efficient elevator scheduling while maximizing service quality (e.g., reducing passengers' waiting time) in the software in the loop setup. The dataset has a total of 1925 test cases, and we selected two fitness functions to form the two TCS problems: 1) *ELEVATOR*_{o2}: Minimizing the overall test case execution of selected test cases (i.e., cost reduction) while maximizing the test case diversity (called "input diversity"). 2) *ELEVATOR*_{o3}: Minimizing the overall test case execution of selected test cases (i.e., cost reduction) while maximizing passenger counts and travel distance in the elevator system.

4.2.2 Baselines

We selected the following baselines for comparison with IGDec-QAOA.

DIV-QAOA. We implemented as a baseline approach a version of IGDec-QAOA without the IGDec strategy. The aim is to assess whether the IGDec strategy contributes to the overall performance of IGDec-QAOA in solving TCO problems. Specifically, we start with **Step 1** of QAOA from Sect. 3.3 to randomly initialize variable values for each test case with -1 or +1. Next, we randomly select test cases as *active variables* into the *actSet*, whose size is fixed. Similarly to *Step 3* in Sect. 3.3, with one *actSet*, we use Eq. 17 to optimize sub-problems. We randomly select the second *actSet* with the newly optimized variable values. We repeatedly select and run sub-problems until a determined number of QAOA runs has been reached. During the optimizing process, we ensure each variable is selected as the *active variables* and optimized at least once. We record the fitness values achieved after each run. Finally, the solution after the last sub-problem is solved is considered the final solution.

Genetic Algorithm. We used the implementation of the Genetic Algorithm (GA) provided by the jMetalPy 1.6.0 [17] framework. We represented each individual of the TCO problems as a bit string. We used the default parameter settings from jMetalPy, i.e., bit flip mutation operation being equal to the reciprocal of the size of the individual, the SPX crossover (the crossover rate=1.0). For each dataset, we experimented with population size *popSize* = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}. We ran 400,000 evaluations for each experiment to ensure that GA had sufficient time to converge. Note that in the literature, the default parameter settings of GA have shown good results [18].

Random Search. We used the existing random search (RS) implemented by jMetalPy 1.6.0 [17]. It randomly selects binary values for each test case and calculates the fitness value repeatedly until a determined number of iterations has been reached, and we pick the lowest fitness value to compare with other approaches.

4.2.3 Parameters and Execution Environment

In **Step 2** of IGDec-QAOA, we set $num = 0.15 \times n$ (see [12]) for larger datasets (i.e., *IOF/ROL*, *GS DTSR*, and *ELEVATOR*). For the smaller dataset (i.e., *Paint Control*), if *num* is smaller than the sub-problem size *N*, we set $num = N$ to ensure that at least one execution of QAOA in each iteration takes

place. IGDec-QAOA has the termination condition of both **Step 2** and **Step 3** (as detailed in Sect. 3.3), i.e., no reduction in *fval* for three consecutive iterations. The maximum generation is set as 30. We run IGDec-QAOA with $N = \{7, 8, 10, 12, 14, 16\}$ considering the execution time limitation. For all experiments, including IGDec-QAOA and the three baseline approaches, we repeat them 30 times to account for the randomness inherent in the approaches. We use *aer simulator* in Qiskit to run the QAOA algorithm. For the noisy quantum computer, we implement QAOA on IBM's quantum computer "ibm_brisbane" and a simulator equipped with a noise model based on this quantum computer. As a classical optimizer, we select the *Constrained Optimization By Linear Approximation* optimizer (COBYLA) [19], with 100 iterations, which is commonly used in literature for the QAOA algorithm. We run QAOA with $p = \{1, 2, 4, 8, 16\}$. The experiment was run on one CPU node in the eX3⁶ cluster with AMD EPYC 7601 32-Core Processor.

4.3 Evaluation metrics and statistical tests

4.3.1 Evaluation metrics

Approximation ratio (*ar*) is a common evaluation metric for assessing QAOA's effectiveness in finding approximate optimal solutions [10]. We calculate *ar* as:

$$ar = \frac{fval(\vec{z})}{fval_{min}} \quad (18)$$

where $fval_{min}$ is the optimal solution of the problem and $fval(\vec{z})$ is the final solution obtained by QAOA. We use the lowest fitness value achieved in the whole experiment as the optimal fitness value for each case study. For *ar*, a lower value is indicative of a better performance. The optimal *ar* is 1.0 for all case studies.

To evaluate the time efficiency, we calculate the execution time *exTime*. For IGDec-QAOA, it contains both the QAOA execution time and classical computation time, including the time for the IGDec strategy.

In addition, we define fitness evaluations *numEva* to help analyze the performance of each approach. For IGDec-QAOA, we calculate the number of QAOA runs (i.e., the sub-problems solved by QAOA) in all iterations until the stopping condition is satisfied as *numEva*. Similarly, for DIV-QAOA, the number of randomly selected sub-problems is considered as *numEva*. For GA, *numEva* refers to the number of evaluations through all generations. For RS, each run (i.e., iteration) represents one evaluation, and the determined number of iterations represents the *numEva* value (each iteration contains one run).

4.3.2 Statistical tests

Following the guideline from [20], we apply statistical tests to evaluate the performance (i.e., *ar*) of IGDec-QAOA with different layers (i.e., *p*) and sub-problem sizes (i.e., *N*), and the baseline approaches. We first utilize a non-parametric Kruskal-Wallis H-test to check whether there is a statistical difference among all groups of *ar* values obtained by IGDec-QAOA with different configurations or baseline approaches.

6. eX3 is a national, experimental, heterogeneous computational cluster for researchers conducting experiments.

5. <https://www.orona-group.com/int-en/>

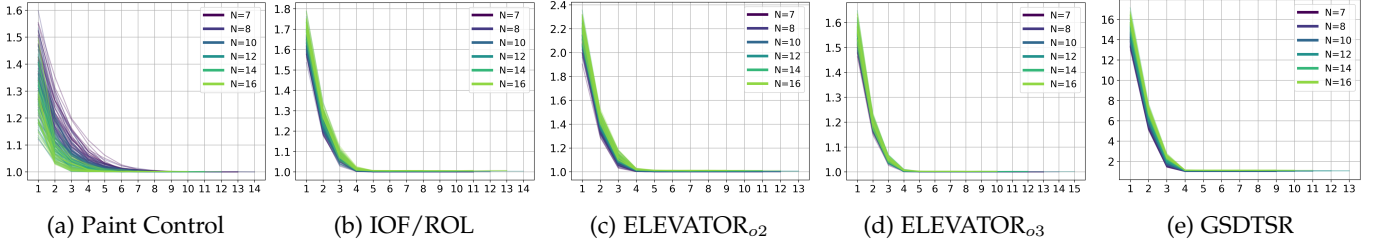


Fig. 4: RQ1 – Trends of approximation ratios (ar) along the iterations of IGDec-QAOA. The x-axis shows the iteration number, whereas the y-axis shows the approximation ratio. For each sub-problem size (i.e., $N = 7, 8, 10, 12, 14$, or 16), we show 30 trend lines corresponding to 30 runs.

We set the significance level as 0.05. A p-value smaller than 0.05 indicates that at least two groups of ar values are significantly different.

Next, we apply the Mann-Whitney U test and Vargha and Delaney's $\hat{A}_{12}$ effect size to compare all possible pairs of IGDec-QAOA with different configurations, and we also compare IGDec-QAOA with the other baseline approaches respectively. We set the significance level as 0.05. A p-value less than 0.05 indicates that there is a significant difference between the ar values of the two selected groups. In this case, we utilize Vargha and Delaney's $\hat{A}_{12}$ statistic as the effect size measure to quantify the magnitude of the difference between the two groups. If $\hat{A}_{12}$ is 0.5, the result is achieved by chance. If $\hat{A}_{12}$ is smaller than 0.5, the first approach performs better than the second one; otherwise, the second one is better.

5 RESULTS AND ANALYSES

This section presents the results and analyses for each RQ, followed by the threats to validity.

5.1 RQ1 – Selecting Optimal Depth and Sub-problem Size

As discussed in Sect. 4.1, we study two key parameters of IGDec-QAOA, i.e., depth p and sub-problem size N .

Results for Optimal Depth. To study whether the depth of a QAOA circuit affects the performance of IGDec-QAOA, we compared the approximate values for 30 runs of each p value (i.e., 1, 2, 4, 8, and 16, Sect. 4.2.3) for all sub-problem sizes (i.e., 7, 8, 10, 12, 14, and 16 in Sect. 4.2.3) for each case study. As a result, for each p value, when combining all sub-problem results, we obtain 180 approximation ratio values (i.e., ar). To assess the overall differences among all the p values, we applied the Kruskal-Wallis H-test. We obtained p-values greater than 0.05, indicating no significant differences among the number of layers chosen for QAOA solving the test optimization problems for all the chosen case studies. Using $p = 1$ in our test optimization problem makes sense since it is the least costly. Thus, to study the effect of sub-problem size in this RQ, we chose $p = 1$.

Results for Optimal Sub-Problem Size. We also study whether there is an optimal sub-problem size for IGDec-QAOA solving the test optimization problems. We present the results for all five case studies in Figs. 4a-4e. Each figure shows the trend of approximation ratio for 30 runs of each sub-problem size along the increasing iterations, i.e., we see

30 trend lines for each sub-problem size. One observation is that the ar values decrease dramatically in the first three iterations, consistently for all case studies, and stabilize to near 1.0 (the best ar). Note that for all our case studies, except *Paint Control*, IGDec-QAOA with smaller N (darker purple lines) tends to achieve lower ar in each iteration, which indicates a faster decline rate. However, the results are a bit different for *Paint Control*, for which the values of ar achieved by IGDec-QAOA with larger sub-problem sizes tend to decline more quickly. As the dataset size of *Paint Control* is small (i.e., only 90 test cases) and the number of optimized test cases cannot exceed $0.15 \times n$ (Sect. 4), for smaller N (i.e., 7, 8, 10, and 12), QAOA can only run once in each iteration. However, for larger N (i.e., 14 and 16), we also need to ensure at least one run of QAOA is executed in each iteration. This indicates that num for larger N is greater than that for small N . Thus, IGDec-QAOA with larger N can optimize more test cases in each iteration such that the optimization process is accelerated for *Paint Control*. In summary, within a case study for IGDec-QAOA, with the same num , smaller N tends to result in faster convergence than larger ones.

In Figs. 5a-5e, we show the violin plots corresponding to the 30 runs of IGDec-QAOA with each N for all five case studies. Note that we pick the best ar of each run (i.e., the lowest value) across all the iterations. From the figure, we can observe that for *Paint Control*, IGDec-QAOA with smaller N (i.e., 7 and 8) found the optimal solution 1.0. A similar pattern is observed for the other case studies when IGDec-QAOA takes N of 7. Moreover, for *Paint Control*, IGDec-QAOA with N of 10, 12, 14, and 16, we can easily observe higher ar values with a certain level of variances. For the other four case studies, ar values obtained by IGDec-QAOA with N over 8 are higher than the optimal solutions. Note that the differences among the optimal solutions obtained by IGDec-QAOA with different N are not high. For instance, for IOF/ROL, IGDec-QAOA with $N = 16$ achieved, on average, a value of ar 0.00517 higher than the optimal value 1.0.

We also conduct statistical tests to compare the performance of IGDec-QAOA, in terms of ar , when configured with different N for each case study. The Kruskal-Wallis H-test results show that all p-values are smaller than 0.05 for each case study. Then, for each pair of N , we use the Mann-Whitney U test and Vargha and Delaney's $\hat{A}_{12}$ effect size. Consistently, for all case studies, IGDec-QAOA with $N = 7$ statistically outperforms the other N (except for

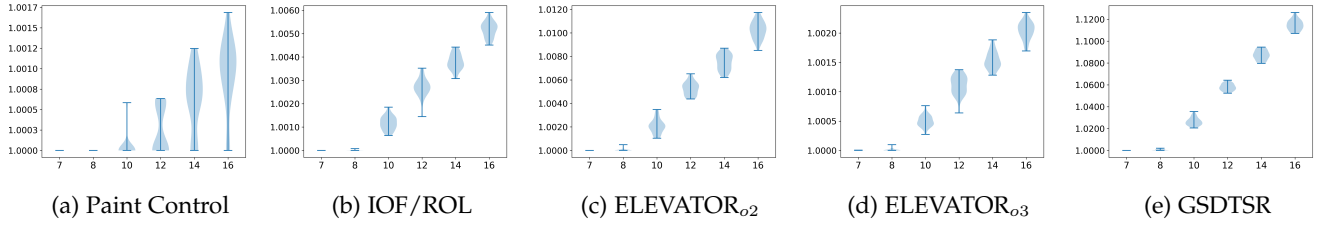


Fig. 5: Violin plots of ar across different N of IGDec-QAOA-RQ1. The x-axis shows the sub-problem sizes (i.e., N of 7, 8, 10, 12, 14, and 16), whereas the y-axis shows the approximation ratio ar

TABLE 2: Execution time of IGDec-QAOA with different N -RQ1. The total execution time (i.e., $exTime$ in seconds) is shown in the format of *average $\pm$ standard deviation*; QAOA% shows the percentage of execution time out of the total execution time spent on executing QAOA

Case Study	$N = 7$		$N = 8$		$N = 10$		$N = 12$		$N = 14$		$N = 16$	
	$exTime$ (s)	QAOA%	$exTime$ (s)	QAOA%	$exTime$ (s)	QAOA%	$exTime$ (s)	QAOA%	$exTime$ (s)	QAOA%	$exTime$ (s)	QAOA%
Paint Control	7.5 $\pm$ 0.7	96.2%	9.8 $\pm$ 1.0	97.2%	19.1 $\pm$ 2.6	98.3%	28.5 $\pm$ 3.7	98.7%	54.4 $\pm$ 8.5	99.2%	56.3 $\pm$ 9.3	99.2%
IOF/ROL	260.0 $\pm$ 26.5	92.5%	311.5 $\pm$ 39.5	94.4%	463.4 $\pm$ 67.1	96.8%	668.3 $\pm$ 118.8	97.8%	1078.5 $\pm$ 170.6	98.8%	1160.4 $\pm$ 272.1	98.7%
ELEVATOR _{o2}	264.9 $\pm$ 21.3	93.0%	343.9 $\pm$ 108.3	94.8%	531.8 $\pm$ 333.8	97.0%	733.3 $\pm$ 350.3	97.9%	1274.0 $\pm$ 645.1	98.8%	15309.4 $\pm$ 19040.3	99.9%
ELEVATOR _{o3}	302.3 $\pm$ 34.1	92.4%	349.0 $\pm$ 48.0	94.3%	473.7 $\pm$ 74.9	96.8%	681.1 $\pm$ 136.4	97.8%	1204.5 $\pm$ 306.7	98.7%	1594.9 $\pm$ 2823.1	98.7%
GSDTSR	540.0 $\pm$ 39.2	83.1%	673.8 $\pm$ 66.5	87.5%	1066.8 $\pm$ 199.2	92.9%	1568.4 $\pm$ 861.9	94.7%	1580.5 $\pm$ 427.7	96.0%	2055.3 $\pm$ 2741.2	95.8%

Paint Control for which IGDec-QAOA with $N = 7, 8$ shares the same effectiveness). Also, IGDec-QAOA with a lower N is consistently more effective than a larger N , and the difference is statistically significant. Thus, we conclude that with lower sub-problem sizes (e.g., 7 and 8), IGDec-QAOA tends to find more optimal solutions.

Finally, we study the execution cost of IGDec-QAOA with different N . Its total execution time consists of QAOA execution time and other computation time spent, mainly on calculating the impact order. Results are summarized in Table 2. There are two key observations. First, for all the case studies, IGDec-QAOA with higher N incurs higher execution cost, even though IGDec-QAOA with smaller N needs more QAOA runs. This is because executing large problems with QAOA on the simulator takes much longer than solving smaller problems. Second, QAOA takes the most execution time (e.g., 96.2% for Paint Control when $N = 7$), which also increases when N increases. This is because the other portion of the execution time (mainly about applying the IGDec strategy) does not change much when N changes. It is also important to mention that the current time performance of IGDec-QAOA is based on executing QAOA on the simulator. When they are executed on real quantum computers, it is expected to be significantly more efficient.

Conclusions for RQ1: There is no significant difference between different numbers of layers of QAOA in terms of the performance of IGDec-QAOA for test case optimization. Thus, we suggest using $p=1$, which incurs minimum cost. IGDec-QAOA with smaller N tends to achieve better effectiveness. Thus, we suggest using lower N for test optimization with IGDec-QAOA.

5.2 RQ2 – Comparing IGDec-QAOA with Baseline

RQ2 explores the benefits of employing the IGDec strategy of IGDec-QAOA. To this end, we choose the best parameter settings of IGDec-QAOA, i.e., $p = 1$ and $N = 7$. We use the results of 30 runs of IGDec-QAOA and DIV-QAOA and compare them in terms of ar .

Fig. 6 (a)-(e) compare the two approaches across all five case studies. Note that both approaches use the same number of QAOA executions (the total number of sub-problems being solved by QAOA in all iterations, i.e., $numEva$) to ensure a fair comparison. Regarding the overall trends, one can observe that the ar values of the two approaches all decrease rapidly during the initial half of the QAOA executions and then stabilize. However, IGDec-QAOA exhibits a faster decrease for all case studies because of its impact ordering strategy; IGDec-QAOA orders test cases in each iteration, contributing to the reduction of fitness values. Conversely, in DIV-QAOA, although each variable (i.e., test case) is optimized at least once into the $actSet$, the sequence of optimization is determined randomly, lacking an efficient order for optimization. Thus, the absence of the IGDec strategy results in a slower decline of fitness values and, consequently, of the ar values as well.

Fig. 7 shows that IGDec-QAOA always manages to reach the optimal value 1.0 throughout all 30 runs for all five case studies. In comparison, DIV-QAOA consistently performs worse than IGDec-QAOA. In addition, results of the statistical test we conducted tell that IGDec-QAOA significantly outperforms DIV-QAOA for all case studies. These results indicate that the random selection of test cases for each sub-problem in each execution is ineffective for optimization, thereby requiring the IGDec strategy in IGDec-QAOA.

Additionally, as depicted in Sect. 5.1, the time consumed by IGDec-QAOA, excluding QAOA execution, is relatively insignificant compared to the time required for QAOA execution. In summary, IGDec-QAOA outperforms DIV-QAOA both in terms of effectiveness and time cost.

Conclusions for RQ2: IGDec-QAOA significantly outperforms DIV-QAOA in terms of finding approximate optimal solutions, thereby suggesting that combining the impact ordering strategy with QAOA is promising in solving test optimization problems.

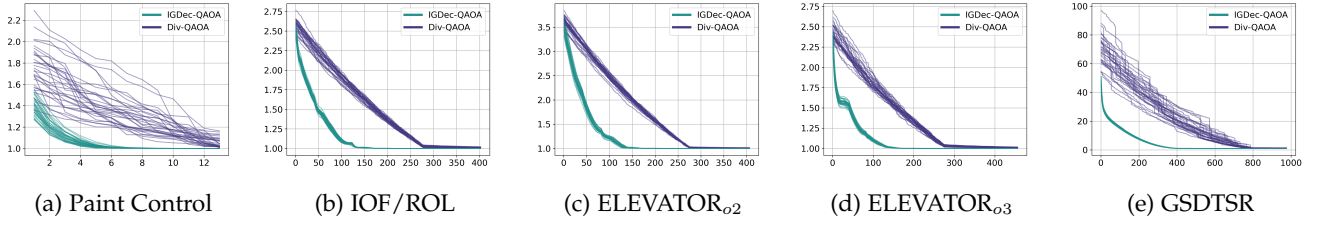


Fig. 6: Approximation ratio (ar) trends of IGDDec-QAOA and DIV-QAOA along with the QAOA executions ($numEva$) for five case studies – RQ2

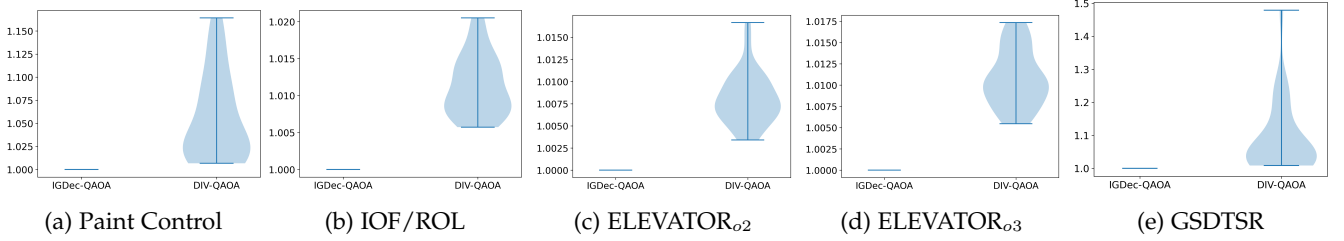


Fig. 7: Violin plots of the final approximation ratio (ar) of IGDDec-QAOA and DIV-QAOA for five case studies – RQ2

TABLE 3: Results of the performance of IGDDec-QAOA and GA – RQ3. Column $\text{avg}(ar) - 1.0$ show the difference between the average ar across 30 runs and the optimal (i.e., 1.0); $\text{std}(ar)$ show standard deviations of 30 runs; $popSize$ refers to the optimal population size of GA; $numEva$ refers to the average values; **fail** lists the number of runs that fail to achieve the optimal ar 1.0; **Result** presents comparison between IGDDec-QAOA and GA.

Case	IGDDec-QAOA				RS				GA					Result
	$\text{avg}(ar) - 1.0$	$\text{std}(ar)$	fail	$numEva$	$\text{avg}(ar) - 1.0$	$\text{std}(ar)$	fail	$numEva$	$\text{avg}(ar) - 1.0$	$\text{std}(ar)$	fail	$popSize$	$numEva$	
Paint Control	0	0	0	11.6	0.473	0.079	30	12	0	0	0	10	2,530	≡
IOF/ROL	0	0	0	400.4	1.387	0.024	30	401	6.223×10^{-7}	9.495×10^{-7}	16	30	28,670	✓
ELEVATOR _{o2}	0	0	0	412.7	2.348	0.030	30	413	2.067×10^{-6}	3.860×10^{-6}	19	80	135,600	✓
ELEVATOR _{o3}	1.355×10^{-6}	1.916×10^{-6}	10	463.3	1.187	0.023	30	464	1.608×10^{-6}	2.439×10^{-6}	10	90	145,962	≡
GSDTSR	0	0	0	959.9	44.881	1.472	30	960	0	0	0	50	17,744	≡

In the **Result** column: $\equiv$ denotes no statistically significant difference between IGDDec-QAOA and GA, while $\checkmark$ denotes IGDDec-QAOA performs significantly better than GA.

5.3 RQ3 – Comparing IGDDec-QAOA with classical Algorithms

In RQ3, we compare IGDDec-QAOA (configured with $p = 1$ and $N = 7$) with RS and GA. For each problem, we run RS with the average number of QAOA executions $numEva$ of IGDDec-QAOA as the iteration count, and we use the lowest fitness values obtained as the solution to calculate ar . Results are shown in Table 3.

RS fails to achieve the optimal ar value 1.0 in all runs for all case studies. Especially for the largest case study GSDTSR, the ar value achieved by RS is as high as 44.881. The results of the Mann-Whitney U test revealed all p -values less than 0.05, signifying that RS is significantly worse than IGDDec-QAOA for all case studies. Moreover, $\hat{A}_{12}$ values are less than 0.01; indicating strong differences. This result suggests that the TCO problems are difficult to solve, justifying the need for optimization algorithms.

As described in Sect. 4.2.2, we run GA with various population sizes $popSize$, and for each experiment, we run 400,000 evaluations, sufficient to observe when GA converges. For each dataset, we calculate the average lowest fitness value among 30 runs achieved by each $popSize$ and also report the average number of evaluations $numEva$ GA takes to obtain the lowest fitness values in that run. We select the

optimal $popSize$ that achieved the lowest fitness value with the smallest $numEva$ for each dataset. The information of the optimal $popSize$ values and their corresponding $numEva$ are reported in Table 3. The average ar and standard deviation values of those 30 runs are also reported, together with the number of runs that failed to obtain the optimal ar value (i.e., 1.0). Besides, we conduct statistical tests to compare ar of 30 runs achieved by our approach and GA with optimal $popSize$ for each dataset and provide the results in Table 3.

As shown in Table 3, for IOF/ROL and ELEVATOR_{o2}, IGDDec-QAOA significantly outperforms GA and achieves 1.0 consistently for all 30 runs, which is, however, not the case for GA. When looking at $numEva$, those runs in which GA fails to obtain ar 1.0, achieve the lowest fitness values with a relatively small $numEva$ (e.g., 28,670 for IOF/ROL) as compared to 400,000 in total. Although the discrepancy between the achieved ar and the optimal 1.0 is minor, with both the difference and standard deviation values in the magnitude of 10^{-7} and 10^{-6} , GA is trapped in local optima for hundreds of thousands of evaluations without further optimization. This slight deviation from the optimal ar is probably due to the different selection of a small number of test cases for the final solution. IGDDec-QAOA, however, does not have this limitation. It is probably because, in the

TABLE 4: RQ4 – Results of the performance of IGDec-QAOA on noisy simulator.

Case	IGDec-QAOA with noise				Result
	avg(ar) - 1.0	std(ar)	fail	numEva	
Paint Control	0	0	0	12.1	≡
IOF/ROL	0	0	0	397.7	✓
ELEVATOR _{o2}	0	0	0	404.5	✓
ELEVATOR _{o3}	1.6256×10^{-6}	1.991×10^{-6}	12	451.0	≡
GSDTSR	0	0	0	971.8	≡

In the **Result** column: ≡ denotes no statistically significant difference between IGDec-QAOA and GA, while ✓ denotes IGDec-QAOA performs significantly better than GA.

IGDec strategy, each time a current solution is updated by a sub-solution, it will jump out of a local optimum. Also, the mixing Hamiltonian part of the QAOA circuit helps explore the search space.

In 4 out of 5 case studies, IGDec-QAOA manages to achieve the optimal ar value 1.0 consistently (i.e., Paint Control, IOF/ROL, ELEVATOR_{o2}, and GSDTSR) while GA achieves 1.0 for all 30 runs only in Paint Control and GSDTSR. Statistically, the effectiveness of the two approaches is equivalent in Paint Control and GSDTSR. In Paint Control, as the dataset size is relatively small, both approaches can reach 1.0 in all 30 runs. In GSDTSR, although the number of test cases is large (i.e., 5555), the “failure rate” values of 94.8% test cases (see Sect. 4.2) are 0%. Thus, since one objective is to maximize the “failure rate”, the problem will be simple for both IGDec-QAOA and GA to solve after they filter those test cases during the optimization process within the early stages of generations; hence, both consistently achieve the optimal ar , i.e., 1.0.

For ELEVATOR_{o3}, both approaches cannot achieve ar 1.0 in all 30 runs with a slight difference, which is probably because the search problem is complex, and both approaches are stuck in local optima in some runs. The number of failing runs in both approaches is 10 of 30 runs. Also, according to the statistical test, there is no significant difference between the two approaches.

Conclusions for RQ3: RS fails to achieve the optimal ar value for all cases, suggesting the need for search algorithms to solve TCO problems. IGDec-QAOA can outperform or perform comparably with GA, and it is less prone to be trapped in local optima than GA.

5.4 RQ4 – Performance of IGDec-QAOA on Noisy Quantum Computer

Performance of IGDec-QAOA on noisy simulator: We ran IGDec-QAOA with the best setting (i.e., $p = 1$ and $N = 7$) on a noisy simulator to analyze the results. We implement the QAOA algorithm on a noisy quantum simulator based on the quantum computer “ibm_brisbane”, which closely mimics gates that will be executed on a real device and noise properties on the quantum computer. The experiment is repeated 30 times. Table 4 shows the performance of IGDec-QAOA on the noisy simulator. All case studies except ELEVATOR_{o3} achieve the optimal solutions in all 30 runs, which is similar to the performance of IGDec-QAOA on the ideal

simulator. We conduct statistical tests on the ar of IGDec-QAOA and GA: results show that we achieve performance on the noisy simulator that is equivalent to and in two cases better than, GA.

We show the trends of ar for all 30 runs on both ideal and noisy simulators in Fig. 8 along the increasing iterations. The purple line shows the trends of the ar values achieved by IGDec-QAOA on the ideal simulator, while the yellow lines represent the trends on the noisy simulator. Our observation is that the two trends overlap a lot in all five case studies, indicating that the decline rates ar values achieved by the two simulators are quite similar, as they all decrease dramatically in the first three iterations and stabilize to near 1.0 in the next few iterations. We employ the Mann-Whitney U test to compare $numEva$ achieved by IGDec-QAOA on the two simulators. It shows that all p-values obtained in all case studies are larger than 0.05, demonstrating no significant difference between $numEva$ values achieved by the two simulators. Furthermore, when comparing the solutions achieved, IGDec-QAOA implemented on both simulators consistently finds the optimal solution in all case studies except for ELEVATOR_{o3}. According to Sect. 5.3, IGDec-QAOA with the ideal simulator fails to find the optimal solution in 10 runs, while IGDec-QAOA with the noisy simulator fails in 12 runs. We also conduct the statistical test in terms of ar achieved by two simulators. Mann-Whitney U test indicates that the p-value is greater than 0.05. All these observations demonstrate that, since QAOA is resilient to noise [6], the effect of noise on IGDec-QAOA is not significant.

Performance of IGDec-QAOA on quantum computer Due to the limitation of quantum resources, we perform the TCM problem on a small case study extracted from dataset *Paint Control*. Since we use the best setting of IGDec-QAOA in the paper ($N = 7$ and $p = 1$), we select 46 test cases to ensure one optimized sub-problem covers 15% (see in Sect. 4.2.3) of the case study. We use the K-means clustering algorithm to select representative test cases from the dataset. We implement our experiment on the “ibm_brisbane” quantum computer in the IBM platform. We first use the ideal simulator to solve TCM and consider the lowest fitness value achieved as the optimal. Next, we run IGDec-QAOA on the quantum computer. We observe that the ar value keeps decreasing from the initial 2.46 to the optimal value of 1.0 in five iterations, as IGDec-QAOA finds the optimal solution on the 5th iteration, which demonstrates the feasibility of implementing IGDec-QAOA on the real quantum computer in the presence of noise.

Conclusions for RQ4: Since QAOA is a noise-resilient quantum algorithm, the effect of noise on IGDec-QAOA is not significant when implemented on a noisy quantum simulator. In addition, it is also feasible to implement IGDec-QAOA on the quantum computer in the presence of noise.

5.5 Concluding Remarks

In this section, we discuss the effectiveness (Sect. 5.5.1) and efficiency (Sect. 5.5.2) of IGDec-QAOA over datasets of different sizes, and how it could be possibly adapted to other software engineering problems (Sect. 5.5.3).

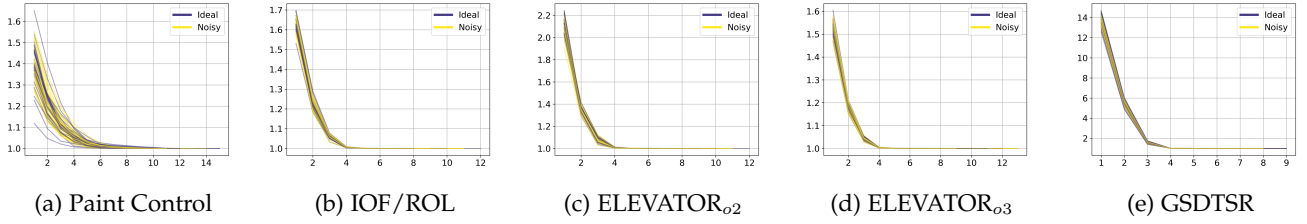


Fig. 8: RQ4 – Trends of approximation ratios (ar) along the iterations of IGDec-QAOA on both ideal and noisy simulators

TABLE 5: Execution time of GA

Case	Paint Control	IOF/ROL	ELEVATOR _{o2}	ELEVATOR _{o3}	GSDTSR
exTime (s)	0.4	146.7	232.1	279.1	1081.5

5.5.1 Effectiveness of IGDec-QAOA

This paper investigates using QAOA, a quantum-classical algorithm, for solving test case minimization and selection problems for classical software systems. Our empirical evaluation based on four industrial datasets of five case studies reveals that even at low-cost configurations (i.e., the shortest depth and smallest sub-problem size), IGDec-QAOA obtained equivalent or better approximate optimal solutions compared to GA. Our results showed that IGDec-QAOA achieved compatible performance compared to the classical approach GA, and in two cases (i.e., IOF/ROL and ELEVATOR_{o2}) outperformed them in terms of effectiveness. The sizes of datasets used in IGDec-QAOA range from 90 to 5555 data points, demonstrating its capability to handle both small and large datasets effectively. This versatility in performance across diverse dataset sizes indicates the robustness of our approach, suggesting that it maintains effectiveness regardless of the dataset scale, and our approach is likely to generalize well beyond the specific datasets used in this study. Nonetheless, additional experiments are needed in the future to explore the stability of IGDec-QAOA.

5.5.2 Efficiency of IGDec-QAOA

Recall, from RQ1, that the execution time (i.e., $exTime$) of IGDec-QAOA remains manageable for large-scale industrial problems. However, for GA, as shown in Table 5, the average optimization time of GA with the optimal $popSize$ reaching the lowest fitness values ranges from 0.4 to 1081.5 seconds. The results show that the execution time of GA is of a similar order of magnitude to that of IGDec-QAOA, with both being in the order of hundreds of seconds (except for Paint Control, for which GA is more efficient). For the largest case study GSDTSR, the average $exTime$ taken by GA is 1081.5 seconds, higher than that of IGDec-QAOA (540.0 seconds). In general, the time performance of GA and IGDec-QAOA are comparable. However, with a real quantum computer, the time performance of IGDec-QAOA is expected to improve significantly.

The QAOA is theoretically considered to be a promising algorithm in terms of both effectiveness and efficiency [21]. Generally, the performance of QAOA improves with the increment of depth p (which results in higher cost). However, with the smallest depth $p = 1$, QAOA is proved to reach a guaranteed minimum approximation ratio and cannot be

efficiently simulated by any classical computer [6], [22]. Here we conduct the theoretical analysis of the efficiency of IGDec-QAOA. IGDec-QAOA contains an initialization (i.e., **Step 1** in Sect. 3.3) and an iterative decomposition strategy IGDec (i.e., **Step 2** and **Step 3** in Sect. 3.3). Let n represent the total number of test cases available. **Step 1** randomly assigns values to all decision variables related to test cases to get the initial fitness value, whose time complexity is $\mathcal{O}(n)$. In **Step 2**, we flip each test case's variable value and calculate its impact on the objective function while keeping others constant. The time complexity is $\mathcal{O}(n)$. Then, we implement a quick sort for all test cases to order the impact, whose time complexity is $\mathcal{O}(n \log n)$. In **Step 3**, assume that the sub-problem size is predefined as N . Since the number of test cases that we optimize with QAOA is $num = 0.15n$ (see Section. 4), the number of QAOA executions we implement in one iteration is $0.15n/N$. The execution time of QAOA depends on sub-problem size N and the depth p selected. We represent the time complexity of QAOA as $\mathcal{O}(f(N, p))$. Thus, the time complexity of **Step 3** is $\mathcal{O}(nf(N, p)/N)$. Assuming the number of iterations of IGDec as r , the total time complexity of IGDec-QAOA is $\mathcal{O}(n) + \mathcal{O}(rn) + \mathcal{O}(rn \log n) + \mathcal{O}(rn f(N, p)/N)$.

In our experiment, for the iterative steps, **Step 2** and **Step 3**, since we empirically limit the maximum number of iterations for *BootQA* to 30 (see Section. 4), we treat the number of iterations r as a constant. We also observe that both the sub-problem size N and the number of layers p are independent of n , as we achieve comparable results of GA in all case studies with setting a fixed configuration $N = 7$ and $p = 1$ respectively. As a result, $\mathcal{O}(f(N, p))$ remains constant and the time complexity **Step 3** is $\mathcal{O}(n)$. Thus, the overall time complexity of *BootQA* with our configuration is $\mathcal{O}(n \log n)$, which is considered to be efficient.

5.5.3 Possible Further Applications of IGDec-QAOA

These results encourage further investigation for other TCO problems (such as test case prioritization) and even other software engineering optimization problems, e.g., optimally allocating requirements for reviewing, finding optimal configurations for highly configurable systems, and searching for optimal software refactoring methods [23]. The way of creating the Ising formulations for such problems will be very similar to what we present in this paper for TCO problems. For example, for resource allocation problems, assigning a resource to a specific task can be represented as a decision variable, and the suitability of that resource to the task can be considered the corresponding attribute value. We optimize the assignment of various resources to all tasks

to maximize the total suitability [24]. In the future, we foresee the need to recommend optimal Ising formulations for each kind of software engineering problem and identify suitable settings for QAOA and the impact-guided decomposition strategy of IGDDec-QAOA, via empirical studies.

5.6 Threats to Validity

Generalizing results to datasets other than the ones used in this paper is a common threat to the validity of our experiments. However, the sizes of the five industrial TCO problems range from 90 to 5555, allowing for an evaluation of approaches across diverse problem scales. It exhibited the potential of applicability of IGDDec-QAOA on other datasets with similar characteristics as ours. In addition, the Ising formulation proposed in our work is general to be applied to other similar search-based software engineering problems (mentioned in Sect. 5.5), which indicates a potential direction for further exploration. Still, additional experiments are warranted with more diverse datasets of varied sizes while also considering the limited number of qubits we could simulate on classical and real quantum computers currently. As quantum computing technology evolves, future research could explore the scalability and applicability of our methods to larger and more varied datasets.

For the baseline algorithms, we used the default parameter settings of GA and RS from jMetalPy. Default parameters have been demonstrated to give good results [18]. We empirically chose the optimal population size of GA for each case study. For IGDDec-QAOA, we had to configure some parameters, i.e., number of layers p , sub-problem size N , and num . We chose p based on our experiment results and num based on *qbsolv* from D-Wave [12]. For N , we chose it to be 7, 8, 10, 12, 14, and 16, considering the time budget for the quantum simulator execution and the existing qubit ranges of small-scale near-term quantum computers.

To deal with randomness, we repeated each experiment 30 times. Following, we also employed statistical tests, i.e., the Kruskal-Wallis test to study differences among more than two samples, whereas the Mann Whitney U Test and $\hat{A}_{12}$ for comparing two samples based on existing guide [20].

Another threat is the choice of the termination criterion, which is predefined and can incorporate human biases. To alleviate this problem, we choose the termination criterion empirically for each algorithm to achieve its best performance. We set the maximum iteration number to 30, and within these 30 runs, we stop IGDDec-QAOA, when there is no improvement in fitness for three consecutive generations. Note that the number of iterations needed to find the optimal solution varies depending on the specific optimization of the problem and is not necessarily achieved within 30 iterations. However, since all our experiments concluded within 30 iterations, we have set this as the maximum limit. For GA, we observed the evolution trend in each case study and ran 400,000 evaluations in total to ensure that GA converges to the optimal solution. For RS, we generate solutions randomly until the number of solutions reaches $numEva$ of IGDDec-QAOA with $p = 1$ and $N = 7$.

6 RELATED WORK

Classical Test Case Optimization: Given the scarcity of software testing resources, it is common to apply test optimization techniques to optimize testing resources while at the same time not compromising testing effectiveness. Some surveys summarize research on test case optimization techniques, e.g., test case minimization (TCM), test case selection (TCS), and test case prioritization (TCP) [25], [26]. Among those works, search-based approaches are commonly applied, such as genetic algorithm [27], simulated annealing [28], and NSGA-II [29]. Given the popularity of machine learning (ML), several approaches for test case optimization have been published [30] (e.g., using supervised learning [31], [32], [33], unsupervised learning [34], [35], reinforcement learning [31], [36], and natural language processing [35], [37]). However, as the scale of software increases, large classical computational resources become necessary to explore the significant part of the search space of possible test case optimization solutions, posing computational challenges for classical algorithms. Thus, we explore integrating quantum computational resources to tackle this challenge. In particular, we investigate using a quantum optimization algorithm, i.e., QAOA, to solve TCO problems on quantum computers.

Quantum Algorithms for Solving SE problems: With the continuous QC development, its applications in various domains are increasing, such as chemistry, finance, and cryptography [38]. In recent years, several quantum algorithms have been proposed to solve software engineering (SE) problems. Miranskyy *et al.* [39] examined the potential of applying eight groups of quantum algorithms (e.g., quantum machine learning, combinatorial optimization) to SE problems throughout the software development life-cycle (e.g., code quality estimation and release problems). Miranskyy *et al.* [40] also used Grover's algorithm and quantum counting algorithm to speed up the process of dynamic testing on classical programs. Jhaveri *et al.* [41] proposed an approach to explore quantum annealing to code clone detection problems. Especially, there has been work on TCO problems. For example, Hussein [42] proposed a quantum algorithm to solve TCM problems. This approach contains two stages: first, preparing an incomplete superposition of the search space, and second, finding the solution using an updated version of Arima's algorithm [43]. Results show that the algorithm can find a solution with high probability in $O(\sqrt{2^n})$ where n represents the number of test cases. This paper's TCM problem is formulated as an SAT problem and is demonstrated theoretically using a 4-test case example. In contrast, IGDDec-QAOA is designed to solve large-scale, realistic TCO problems. Bajaj *et al.* [44] proposed a quantum-behaved particle swarm optimization combined with GA to solve the TCO problem. This algorithm is inspired by the quantum behavior of particles and QC is not involved. Wang *et al.* [45] propose a hybrid approach to solve TCM problems with random sampling and quantum annealing, which uses different problem formulations and different technology.

Applications of QAOA: The computations performed by current quantum computers are noisy, thereby limiting their successful applications. To this end, hybrid algorithms are

being advocated to maximize the use of scarce quantum resources and combine them with classical computations in near-term quantum computing applications. QAOA is one such hybrid algorithm that focuses on solving combinatorial optimization problems [6]. It owns the potential of computational advantages [21], [22], [46]. It has been proved to be applicable to finding approximated solutions to many well-known optimization problems, such as MaxCut problems, maximum independent set problems, Knapsack problems, job shop scheduling problems, and portfolio optimization problems [21]. Some works have also proved the speedup potential of QAOA. For instance, Zhang *et al.* [47] apply QAOA to solve the minimum vertex cover problems in polynomial time with a high probability, which realizes exponential acceleration. Jiang *et al.* [48] obtain a quadratic speedup over classical algorithms, which is the same as Grover search when solving an unstructured search problem with QAOA. They demonstrate the quantum advantage even with multiple layers (i.e., p) in QAOA. For solving quantum linear system problems, An *et al.* [49] find that QAOA with an optimal control protocol outperforms the best classical baseline algorithm in terms of runtime performance with exponential advantage. It also outperforms several quantum algorithms such as HHL algorithm [50]. Real-world applications have also been coming out recently. For instance, Vikstål *et al.* [51] applies QAOA to an airline scheduling problem, tail-assignment problems, with small-scale real-world data. Results indicate that QAOA can solve the problems with high success probabilities even with low layers (i.e., p). In the computer vision domain, Li *et al.* [52] implement QAOA in the partially occluded object detection problem. They achieve execution speedup by choosing a proper classical optimizer, exploiting parameter symmetry, and rescheduling gate operation. Niroula *et al.* [53] construct extractive summarization as an optimization problem and formulate the objective as a quadratic binary optimization problem (QUBO). They implement a QAOA variant, XY-QAOA, on a trapped-ion quantum computer and emphasize the necessity of encoding the constraints into the quantum circuit. In the financial industry, Baker *et al.* [54] benchmark the performance of QAOA with various quantum backends on financial assets portfolio optimization problems. Date *et al.* [55] use QAOA for accelerating the process of training machine learning models. They formulate the training problems of three machine learning models as QUBOs, and they show that both the time and space complexities of their formulations are better than those of their classical counterparts. Despite so much investigation into QAOA, there is a lack of investigation into using QAOA to solve software testing problems. Our work combines the IGDc strategy and QAOA to target a large-scale realistic TCO problem. Such investigation is novel.

7 CONCLUSION AND FUTURE WORK

Quantum Computing (QC) holds great potential to solve many important and challenging problems in the long term. In the near future, their applications to solve optimization problems seem realistic. To this end, Quantum Approximate Optimization Algorithms (QAOAs) are emerging as promising algorithms with several applications demonstrated in

areas such as finance and scheduling. Along this line of investigation, we studied QAOAs in the context of test optimization for software systems. We contributed by providing a generic formulation of test optimization to solve it with QAOAs. Moreover, we integrated an existing test impact order strategy with QAOAs such that they can solve larger problems that cannot be solved with today's limited number of available qubits. We performed an empirical study with five case studies and four industrial datasets. First, we empirically evaluated IGDc-QAOA on an ideal simulator. The results demonstrated the benefit of using IGDc-QAOA for solving test optimization problems, which managed to reach similar or even better effectiveness compared to a classical genetic algorithm. Second, our experiments on a noisy simulator demonstrated that the impact of noise on IGDc-QAOA is insignificant, thus making it a viable solution for real quantum computers. In the future, we want to investigate solving other software engineering optimization problems with QAOAs.

ACKNOWLEDGMENTS

This work is supported by the Qu-Test project (#299827) funded by the Research Council of Norway. X. Wang is also supported by Simula's internal strategic project on quantum software engineering. S. Ali also acknowledges the support from the *Quantum Hub initiative* (OsloMet). P. Arcaini is supported by Engineerable AI Techniques for Practical Applications of High-Quality Machine Learning-based Systems Project (Grant Number JPMJMI20B8), JST-Mirai; and by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST, Funding Reference number: 10.13039/501100009024 ERATO.

REFERENCES

- [1] A. Bajaj and O. P. Sangwan, "A systematic literature review of test case prioritization using genetic algorithms," *IEEE Access*, vol. 7, pp. 126 355–126 375, 2019.
- [2] R. Pan, T. A. Ghaleb, and L. Briand, "ATM: Black-box test case minimization based on test code similarity and evolutionary search," in *Proceedings of the 45th International Conference on Software Engineering*, ser. ICSE '23. IEEE Press, 2023, p. 1700–1711.
- [3] M. Motta and J. E. Rice, "Emerging quantum computing algorithms for quantum chemistry," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 12, no. 3, p. e1580, 2022.
- [4] M. Zinner, F. Dahlhausen, P. Boehme, J. Ehlers, L. Bieske, and L. Fehring, "Quantum computing's potential for drug discovery: Early stage industry dynamics," *Drug Discovery Today*, vol. 26, no. 7, pp. 1680–1688, 2021.
- [5] D. J. Egger, C. Gambella, J. Marecek, S. McFaddin, M. Mevissen, R. Raymond, A. Simonetto, S. Woerner, and E. Yndurain, "Quantum computing for finance: State-of-the-art and future prospects," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–24, 2020.
- [6] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.
- [7] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme, "Quantum optimization using variational algorithms on near-term quantum devices," *Quantum Science and Technology*, vol. 3, no. 3, p. 030503, jun 2018.
- [8] K. Kurowski, T. Pecyna, M. Słysz, R. Różycki, G. Waligóra, and J. Weglarz, "Application of quantum approximate optimization algorithm to job shop scheduling problem," *European Journal of Operational Research*, vol. 310, no. 2, pp. 518–528, 2023.

- [9] S. Brandhofer, D. Braun, V. Dehn, G. Hellstern, M. Hüls, Y. Ji, I. Polian, A. Bhatia, and T. Wellens, "Benchmarking the performance of portfolio optimization with qaoa," *Quantum Information Processing*, vol. 22, no. 12, 2022.
- [10] H. Jing, Y. Wang, and Y. Li, "Data-driven quantum approximate optimization algorithm for power systems," *Communications Engineering*, vol. 2, no. 1, 3 2023.
- [11] A. Lucas, "Ising formulations of many NP problems," *Frontiers in physics*, vol. 2, p. 5, 2014.
- [12] A. R. Michael Booth, Steven P. Reinhardt, "Partitioning optimization problems for hybrid classical/quantum execution," D-Wave, Tech. Rep., 2017.
- [13] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.
- [14] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik, "Simulation of electronic structure hamiltonians using quantum computers," *Molecular Physics*, vol. 109, no. 5, pp. 735–750, 2011.
- [15] H. Spieker, A. Gotlieb, D. Marijan, and M. Mossige, "Reinforcement learning for automatic test case prioritization and selection in continuous integration," in *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISTA 2017. New York, NY, USA: Association for Computing Machinery, 2017, pp. 12–22.
- [16] J. Ayerdi, A. Arrieta, E. B. Pobee, and M. Arratibel, "Multi-objective metamorphic test case selection: an industrial case study (practical experience report)," in *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, 2022, pp. 541–552.
- [17] A. Benítez-Hidalgo, A. J. Nebro, J. García-Nieto, I. Oregi, and J. Del Ser, "jMetalPy: A python framework for multi-objective optimization with metaheuristics," *Swarm and Evolutionary Computation*, vol. 51, p. 100598, 2019.
- [18] A. Arcuri and G. Fraser, "On parameter tuning in search based software engineering," in *Search Based Software Engineering*, M. B. Cohen and M. Ó Cinnéide, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 33–47.
- [19] M. J. Powell, "A view of algorithms for optimization without derivatives," *Mathematics Today-Bulletin of the Institute of Mathematics and its Applications*, vol. 43, no. 5, pp. 170–174, 2007.
- [20] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 1–10.
- [21] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, "A review on quantum approximate optimization algorithm and its variants," *Physics Reports*, vol. 1068, pp. 1–66, 2024.
- [22] E. Farhi and A. W. Harrow, "Quantum supremacy through the quantum approximate optimization algorithm," *arXiv preprint arXiv:1602.07674*, 2016.
- [23] A. Ramirez, J. R. Romero, and S. Ventura, "A survey of many-objective optimisation in search-based software engineering," *Journal of Systems and Software*, vol. 149, pp. 382–395, 2019.
- [24] L. D. Otero, G. Centeno, A. J. Ruiz-Torres, and C. E. Otero, "A systematic approach for resource allocation in software projects," *Computers & Industrial Engineering*, vol. 56, no. 4, pp. 1333–1339, 2009.
- [25] R. Mukherjee and K. S. Patnaik, "A survey on different approaches for software test case prioritization," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 33, no. 9, pp. 1041–1054, nov 2021.
- [26] Z. Xiao and L. Xiao, "A systematic literature review on test case prioritization and regression test selection," in *2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA)*. IEEE, 2023, pp. 235–242.
- [27] S. Wang, S. Ali, and A. Gotlieb, "Cost-effective test suite minimization in product lines using search techniques," *J. Syst. Softw.*, vol. 103, no. C, pp. 370–391, may 2015.
- [28] T. Nithya and S. Chitra, "Soft computing-based semi-automated test case selection using gradient-based techniques," *Soft Computing*, vol. 24, no. 17, pp. 12981–12987, 2020.
- [29] A. Arrieta, S. Wang, U. Markiegi, A. Arruabarrena, L. Etxeberria, and G. Sagardui, "Pareto efficient multi-objective black-box test case selection for simulation-based testing," *Information and Software Technology*, vol. 114, pp. 137–154, 2019.
- [30] R. Pan, M. Bagherzadeh, T. A. Ghaleb, and L. Briand, "Test case selection and prioritization using machine learning: A systematic literature review," *Empirical Softw. Engg.*, vol. 27, no. 2, mar 2022.
- [31] A. Bertolino, A. Guerriero, B. Miranda, R. Pietrantuono, and S. Russo, "Learning-to-rank vs ranking-to-learn: Strategies for regression testing in continuous integration," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 1–12.
- [32] J. Chen, Y. Lou, L. Zhang, J. Zhou, X. Wang, D. Hao, and L. Zhang, "Optimizing test prioritization via test distribution analysis," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 656–667.
- [33] M. Mahdiah, S.-H. Mirian-Hosseinabadi, K. Etemadi, A. Nosrati, and S. Jalali, "Incorporating fault-proneness estimations into coverage-based test case prioritization methods," *Information and Software Technology*, vol. 121, p. 106269, 2020.
- [34] Z. Khalid and U. Qamar, "Weight and cluster based test case prioritization technique," in *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2019, pp. 1013–1022.
- [35] P. Kandil, S. Moussa, and N. Badr, "Cluster-based test cases prioritization and selection technique for agile regression testing," *Journal of Software: Evolution and Process*, vol. 29, no. 6, p. e1794, 2017.
- [36] M. Bagherzadeh, N. Kahani, and L. Briand, "Reinforcement learning for test case prioritization," *IEEE Transactions on Software Engineering*, vol. 48, no. 8, pp. 2836–2856, 2021.
- [37] N. Medhat, S. M. Moussa, N. L. Badr, and M. F. Tolba, "A framework for continuous regression and integration testing in iot systems based on deep learning and search-based techniques," *IEEE Access*, vol. 8, pp. 215716–215726, 2020.
- [38] V. Hassija, V. Chamola, A. Goyal, S. S. Kanhere, and N. Guizani, "Forthcoming applications of quantum computing: peeking into the future," *IET Quantum Communication*, vol. 1, no. 2, pp. 35–41, 2020.
- [39] A. Miranskyy, M. Khan, J. P. L. Faye, and U. C. Mendes, "Quantum computing for software engineering: Prospects," in *Proceedings of the 1st International Workshop on Quantum Programming for Software Engineering*, 2022, pp. 22–25.
- [40] A. Miranskyy, "Using quantum computers to speed up dynamic testing of software," in *Proceedings of the 1st International Workshop on Quantum Programming for Software Engineering*, 2022, pp. 26–31.
- [41] S. Jhaveri, A. Krone-Martins, and C. V. Lopes, "Cloning and beyond: A quantum solution to duplicate code," in *Proceedings of the 2023 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, 2023, pp. 32–49.
- [42] H. Hussein, A. Younes, and W. Abdelmoez, "Quantum algorithm for solving the test suite minimization problem," *Cogent Engineering*, vol. 8, no. 1, p. 1882116, 2021.
- [43] K. Arima, N. Shigei, and H. Miyajima, "A proposal of a quantum search algorithm," in *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*. IEEE, 2009, pp. 1559–1564.
- [44] A. Bajaj and A. Abraham, "Test case prioritization and reduction using hybrid quantum-behaved particle swarm optimization," in *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2022, pp. 1–8.
- [45] X. Wang, A. Muqet, T. Yue, S. Ali, and P. Arcaini, "Test case minimization with quantum annealers," *arXiv preprint arXiv:2308.05505*, 2023.
- [46] G. G. Guerreschi and A. Y. Matsuura, "QAOA for max-cut requires hundreds of qubits for quantum speed-up," *Scientific reports*, vol. 9, no. 1, p. 6903, 2019.
- [47] Y. Zhang, X. Mu, X.-W. Liu, X. Wang, X. Zhang, K. Li, T. Wu, D. Zhao, and C. Dong, "Applying the quantum approximate optimization algorithm to the minimum vertex cover problem," *Applied Soft Computing*, vol. 118, p. 108554, 2022.
- [48] Z. Jiang, E. G. Rieffel, and Z. Wang, "Near-optimal quantum circuit for grover's unstructured search using a transverse field," *Physical Review A*, vol. 95, no. 6, p. 062317, 2017.
- [49] D. An and L. Lin, "Quantum linear system solver based on time-optimal adiabatic quantum computing and quantum approximate optimization algorithm," *ACM Transactions on Quantum Computing*, vol. 3, no. 2, pp. 1–28, 2022.

- [50] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Physical review letters*, vol. 103, no. 15, p. 150502, 2009.
- [51] P. Vikstål, M. Grönkvist, M. Svensson, M. Andersson, G. Johansson, and G. Ferrini, "Applying the quantum approximate optimization algorithm to the tail-assignment problem," *Physical Review Applied*, vol. 14, no. 3, p. 034009, 2020.
- [52] J. Li, M. Alam, A. A. Saki, and S. Ghosh, "Hierarchical improvement of quantum approximate optimization algorithm for object detection," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2020, pp. 335–340.
- [53] P. Niroula, R. Shaydulin, R. Yalovetzky, P. Minssen, D. Herman, S. Hu, and M. Pistoia, "Constrained quantum optimization for extractive summarization on a trapped-ion quantum computer," *Scientific Reports*, vol. 12, no. 1, p. 17171, 2022.
- [54] J. S. Baker and S. K. Radha, "Wasserstein solution quality and the quantum approximate optimization algorithm: a portfolio optimization case study," *arXiv preprint arXiv:2202.06782*, 2022.
- [55] P. Date, D. Arthur, and L. Pusey-Nazzaro, "QUBO formulations for training machine learning models," *Scientific reports*, vol. 11, no. 1, p. 10029, 2021.