Quantum Optimization for Regression Testing: How far are we?

Anonymous Author(s)

ABSTRACT

In the dynamic landscape of software development, maintaining software quality assurance is crucial as systems undergo continuous modifications. Regression testing, a pivotal component of software testing, ensures that software functions as expected after changes are implemented. However, re-executing all test cases for every modification is often impractical and costly, particularly for large systems.

This paper addresses the challenges faced by traditional test suite optimization techniques, which are computationally intensive and may not be practical in resource-constrained scenarios. To overcome these limitations, we explore the potential of quantum computing in the realm of test case selection. Our focus is on Quantum Simulated Annealing, a quantum optimization algorithm, as a novel approach to enhance the efficiency of regression test case selection. By harnessing the power of quantum computing, we aim to address the computational challenges that have long plagued traditional methods.

CCS CONCEPTS

• **Theory of computation → Quantum computation theory**; • **Software and its engineering → Search-based software engineering**; **Maintaining software**; • **Computing methodologies → Search methodologies**.

KEYWORDS

Quantum Computing, Quantum Software Engineering, QSE Challenges

ACM Reference Format:

Anonymous Author(s). 2024. Quantum Optimization for Regression Testing: How far are we?. In *Proceedings of 5th International Workshop on Quantum Software Engineering (Q-SE 2024).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In the ever-evolving landscape of software development, software quality assurance is of fundamental importance. As software systems undergo continuous modifications and enhancements, it becomes imperative to ensure that these changes do not introduce unintended side effects or defects. In response to this need, the practice of regression testing has emerged [37], a crucial component of software testing that verifies whether previously developed and tested software still works as expected after a change is performed.

Ideal regression testing, would consist of re-running all the available test cases of a given software system. However, in addition to being potentially very costly, this could even be impractical in some cases [22, 25], especially for large systems and when the re-execution of entire test suites occurs for every single modification. Several approaches have been suggested to streamline the regression testing process, such as selecting a potentially minimal subset of test cases from the test suite based on specific testing criteria [1, 21, 31, 33, 35, 36, 38]. Alternatively, strategies include prioritizing the execution of test cases, aiming to first run those anticipated to uncover faults earlier in the testing cycle [2, 7, 10, 19].

Regression test case selection [9, 37] is one of the most widely investigated techniques in this scenario. Test case selection is the process of choosing a subset of test cases from a pool of possibilities (i.e., the test suite), ensuring comprehensive coverage and efficient use of testing resources.

While the significance of test case selection is undeniable, traditional techniques face serious challenges, primarily related to computing costs. Conventional approaches, often rooted in optimization algorithms, such as greedy algorithms [35] and search-based approaches [21], demand extensive computational resources to deliver optimal results. The computational burden not only reduces the efficiency of these methods but also poses practical limitations, especially in scenarios where resource-intensive testing is not viable.

To overcome the limitations of traditional techniques, one could think about relying on quantum computation. Quantum computing harnesses the principles of quantum mechanics to process information in ways fundamentally different from classical computers. The intrinsic parallelism and exponential computational capacity of quantum systems offer a potential breakthrough for overcoming the resource constraints associated with traditional test case selection techniques [15, 18].

The work described in this paper is the first to propose the potential of quantum computing in the realm of test case selection. The quantum environment chosen for the realization of this work is the D-Wave environment (https://dwavesys.com), which is able to solve NP-hard combinatorial problems through an approach called "*Quantum Simulated Annealing*" [8, 11]. The choice of relying on this approach was driven by the fact that the size of the test case selection problem is still too large to be solved by other quantum strategies, such as those based on the gate model. Leveraging the power of Quantum Simulated Annealing [16, 30], a quantum optimization algorithm, we delve into a novel approach that holds the promise of addressing the computational challenges that have long plagued traditional methods. As quantum computing continues to emerge as a driving force in optimization tasks, our focus on Quantum Simulated Annealing for Test Case Selection signifies a significant stride toward advancing the efficiency and effectiveness of regression testing in contemporary software development paradigms.

Q-SE 2024, April 16, 2024, Lisbon, Portugal

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

2 BACKGROUND & RELATED WORK

2.1 Test Case Selection

Test case selection is centered around choosing a subset from an initial test suite to assess software modifications, ensuring that unaltered segments of a program continue to function correctly following changes in other parts [24]. Various techniques, such as Integer Programming [12], symbolic execution [34], data flow analysis [23], dependence graph-based methods [5], and flow graph-based approaches [24], can be employed to identify the modified portions of the software. Once test cases covering the unchanged program segments are pinpointed using a specific technique, an optimization algorithm (e.g., greedy) can be applied to select a minimal set of these test cases based on certain testing criteria (e.g., branch coverage). The ultimate aim is to reduce the expenses associated with regression testing.

While initially, test case selection was considered as a single-objective optimization problem, advancements in technology revealed that optimizing only one objective is insufficient for testing, as many tests often need to satisfy multiple criteria simultaneously. Therefore, Yoo et al. [35] introduced the idea of using Pareto sets to address the case selection problem. This paper explores test case selection through an experimental study, employing the greedy algorithm, NSGA2 algorithm, and a variant of NSGA2 called the vNSGA algorithm. The findings indicate that the greedy algorithm is more suitable for single-objective test case selection, while NSGA2 and vNSGA2 algorithms demonstrate better performance in multi-objective test case selection.

Later on, Panichella et al. [21], introduced DIV-GA, an algorithm that improves the NSGA2 algorithm by injecting diversity into the genetic algorithm, reducing the genetic drift phenomenon in NSGA2 and enhancing the efficiency of test case selection.

In the present days, techniques like machine learning, particularly reinforcement learning, have gained widespread attention. Spieker et al. [28] pioneered the use of reinforcement learning for test case prioritization, proving that it can enhance testing efficiency through the design of reward functions and agent selection. Test case selection techniques often leverage unsupervised clustering algorithms to structurally cluster test cases and then select some for testing. Olsthoorn et al. [20] integrated unsupervised clustering algorithms, especially the hierarchical clustering algorithm, with NSGA2. This integration formed a chain structure, which was used as part of the crossover operator in genetic algorithms, enhancing diversity and efficiency in the resulting test case set.

2.2 Simulated Annealing

Simulated annealing, a meta-heuristic optimization technique introduced by Kirkpatrick *et al.* [17], is used to find approximate solutions to optimization problems, such as the traveling salesman problem. It simulates the metallurgical process of annealing, starting at high temperature to explore a wide range of solutions, then gradually cooling to refine towards a global minimum. The algorithm involves an initial solution represented by a bit string, which is improved through random perturbations accepted with decreasing probability over time. The objective function, or *energetic function*, corresponds to the Hamiltonian defined earlier. Perturbations generate new solutions slightly different from the current one, ensuring a comprehensive search. The acceptance criterion accepts better solutions or worse ones with a probability dependent on the temperature, cooling factor α, energy variation ΔE, and iteration count t.

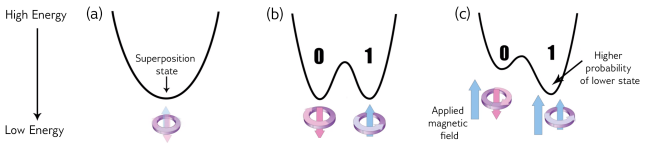

Figure 1: Quantum Annealing process as reported by Dwave documentation

2.3 Quantum Optimization and Quantum Annealing

In quantum optimization, quantum computing offers more efficient solutions to complex computational problems, thanks to qubits' ability to embody multiple states simultaneously due to superposition. Quantum optimization algorithms, like Grover's Algorithm [14], use quantum oracles to conduct searches in unknown spaces with linear complexity. Quantum environments, such as D-Wave2's Quantum Leap, optimize NP-hard problems through adiabatic quantum optimization, defining the system as a Hamiltonian and finding the lowest energy solution.

Quantum annealing, a type of quantum computation, differs from the gate model approach in its problem-solving methodology. While the gate model divides problems into a sequence of operations on individual qubits, quantum annealing translates the problem into a quantum system of qubits to find the minimum energy configuration through a gradual quantum state transition. This method is suitable for problems where an approximate solution is acceptable. The preference for quantum annealing in this context is due to the limited practical development of gate-based quantum computers and the demonstrated effectiveness of its classical counterpart, Simulated Annealing, in solving similar optimization problems, as shown by Shi and Dworak [27] and Zhang et al. [32].

Quantum annealing surpasses classical search algorithms in solving NP-complete optimization problems, potentially in polynomial time [6]. It involves a journey through an energy landscape, starting from a broad valley representing a superposition state and gradually shifting to a double-well potential state. This transition signifies the convergence to the lowest energy solution, representing the optimal solution. Quantum annealing's ability to explore a vast array of potential solutions efficiently improves the efficiency of solving complex optimization problems beyond classical computing's scope. The quantum annealing process is depicted in Figure 1.

Simulated and quantum annealing are similar in their inspiration from metallurgical annealing and objective of finding the lowest-energy state. However, quantum annealing's use of quantum mechanics allows it to evaluate multiple states simultaneously and find optimal solutions more efficiently, especially in complex landscapes where classical methods may struggle [30].

3 RESEARCH METHOD

In Section 2, has been described that simulated annealing obtained good results resolving problems of test suite optimization. However, the simulated-annealing-based algorithms can have long execution times, especially for large codebases. Quantum annealing shows promise for solving this problem in constant time, making it a convenient solution for industrial-level software production. It has also been shown to produce high-quality solutions[26]. Hence, in this work, with the *purpose* of assessing whether the simulated annealing performance for test case selection can be improved, our *goal* is to evaluate the efficiency and effectiveness of a quantum-annealing-based test case selection algorithm. The perspective is of both researchers and practitioners: while the former are interested in improving state-of-the-art and classical computing techniques, the latter are interested in having a practically exploitable solution to their testing problems.

To fulfill our goal, we aim to answer the following research questions:

RQ$_1$ Is the proposed quantum solution more effective than the simulated annealing?

RQ$_2$ Is the proposed quantum solution more efficient then the simulated annealing?

Effectiveness in our first research question refers to the capability of the quantum annealing technique to detect failures at various execution cost levels, compared its classical counterpart. It is measured using the hypervolume metric, as employed by previous works about the test case selection problem su as the one by Panichella et al.[21] that considers NSGA2, DIV-GA and greedy algorithms for comparisons. We have chosen to use the same metric in order to be able to make comparisons also with the classical techniques that currently represent the state of the art of algorithmic solutions to the problem of test case selection. This metric assesses the quality of solutions (test suites) generated by optimization algorithms. Specifically, it calculates the volume enclosed between a Pareto front $P = \{p_1, ..., p_h\}$ and an ideal front R[3][39], with lower hypervolume values indicating higher quality solutions.

Quantum annealing (such as simulated annealing) combines objectives into a single function. Let us consider $P = \{p_1, ..., p_h\}$ as a set of solutions, a set of test suites. Now, let us consider $f(p_i)$ as the percentage of failures detected by solution $p_i \in P$ and $cost(p_i)$ the corresponding execution cost. Let $R = \{r_1, ..., r_h\}$ be the ideal corresponding set of solutions, that is, the set of test suites capable of detecting all failures at different execution cost levels; in particular, it is required that $cost(p_i) = cost(r_i)$ and that $f(r_i) = 1$ for each r_i. The hypervolume contained between these points is calculated as the sum of the rectangles of width $[cost(p_{i+1}) - cost(p_i)]$ and height $[f(r_i) - f(p_i)]$, so that:

$$I_H(P) = \text{cost}(p_1) + \sum_{i=1}^{h} [\text{cost}(p_{i+1}) - \text{cost}(p_i)] \cdot [1 - f(p_i)] \quad (1)$$

Table 1: Algorithm Definition

#	Definition
i	unique index for identifying a test case
k	unique index representing a statement
T_k	list of all test cases running the statement number k
x_i	binary variable that specifies if in the solution of the algorithm the i-th test case has been inserted
$cost(\tau_i)$	execution cost of the i-th test
e_i	binary variable that indicates whether the i-th test case has detected errors in the past or not
α	weight factor whose purpose is to modulate the contrast ratio between the objectives of the problem
P	penalty coefficient whose purpose is to regulate the importance of constraints in the final equation

Further, the effectiveness is quantified as the percentage of the area below the ideal front R:

$$I_{CE}(P) = \frac{I_H(P)}{cost(p_h)} \quad (2)$$

Here, $cost(p_h)$ is the cost of the last point in P, and the hypervolume metric is calculated during the quantum sampler's execution.

To answer our second research question, we define efficiency by considering the execution time of the quantum annealing algorithm. The sampler used in this project was a D-Wave Hybrid Binary Quadratic Model (Version 2), and the execution time was measured using the sampler quantum processor provided by D-Wave. To accurately estimate the real mean execution time of the Quantum Annealing process for each program, we employed the bootstrap technique to compute the 95% confidence interval of the execution time. This technique provides a reliable range of values within which the true population parameter is likely to lie, enabling us to make informed decisions and interpretations in statistical analysis. The methodology involves iterative resampling from the original dataset, executed with replacement, and identifying the 2.5th and 97.5th percentiles within the distribution of computed statistics to establish the confidence interval's bounds. This approach is widely accepted as a robust and precise method for determining the precision and reliability of an estimate, especially in the presence of variability or uncertainty. The same approach has been used to measure the simulated execution time. The algorithm has been tested on a machine with a 2.30GHz Intel Core i5 processor and 8GB RAM.

3.1 Algorithm Definitions

In order to fully understand how the problem to be solved has been modeled, definitions are given that specify the meaning of variables and functions necessary for the formulation of the algorithm. Table 1 contains all the information.

3.2 Algorithm Approach

The implementation of the test case selection problem as a QUBO problem involves a series of steps that seamlessly integrate into a coherent framework. Initially, the primary objectives are set out, focusing on minimizing the execution cost of the test suite while maximizing its efficacy in detecting failures. Each test case in the suite is characterized by its execution cost, history in detecting failures, and the specific program statements it covers. The problem is akin to the *Minimum Cover* problem, where the goal is to find the smallest subset of test cases that collectively cover all necessary program statements, referred to as set S. Each test case is essentially a subset of S, and the challenge is to identify the minimal collection of these subsets that efficiently covers the entire set S.

In the QUBO framework, this problem is expressed using binary variables (0 or 1) to represent the inclusion or exclusion of a test case in the final suite. The QUBO problem is then described by a Hamiltonian function or a Binary Quadratic Model (BQM), which incorporates both the linear impacts of each test case and the quadratic terms representing interactions between different test cases, such as overlapping coverage of program statements. Following the methodologies proposed by Glover *et al.* [13], we transform the linear objectives of the test case selection problem into a quadratic form. This step involves creating a Hamiltonian that encapsulates the individual contributions of each test case and their interrelations. Finally, the formulated Hamiltonian is processed using quantum annealing, specifically the D-Wave system, designed for solving QUBO problems. This approach allows for efficient exploration of the solution space to find the optimal subset of test cases. Leveraging the capabilities of quantum computing, this method promises more efficient solutions compared to classical algorithms, particularly for extensive and complex test suites.

The first goal is written as a BQM expression as follows:

$$\alpha \sum_{i=1}^{|\Gamma|} [x_i \cdot cost(\tau_i)] \tag{3}$$

The second objective is converted into a problem to be minimized for simplification. This is translated below into a BQM expression:

$$-(1-\alpha) \sum_{i=1}^{|\Gamma|} (e_i \cdot x_i) \tag{4}$$

Note the presence of the α coefficient within the two target functions. It is a weight factor ($0 < \alpha < 1$) whose purpose is to allow to establish preference towards one goal rather than the other possibly. This is a weighted sum approach [29], when $\alpha = 0.5$, the importance of the two objectives is the same. When α takes on other values, one of the two objectives will be more relevant in the search for the final result. Therefore, the final function to be minimized consists of two parts, that is, *composite* and is formulated as follows:

$$H = \alpha \sum_{i=1}^{|\Gamma|} [x_i \cdot cost(\tau_i)] - (1-\alpha) \sum_{i=1}^{|\Gamma|} (e_i \cdot x_i) \tag{5}$$

To ensure the final test suite maintains the same coverage level as the initial one, a critical constraint is introduced: each program statement executed in the original suite must be covered by at least one test case in the final selection. This constraint is essential to ensure that, despite reducing the number of test cases, the final suite still comprehensively covers all necessary program statements. This approach maintains the integrity and effectiveness of the test coverage. The constraint shall be expressed as follows:

$$\sum_{i \in T_k} (x_i) \geq 1 \tag{6}$$

The constraint states that at least one test case that performs the k statement must be selected. The list of test cases that run the k-th statement is T_k. Since a program has several statements, we apply the constraint for each statement to be covered. The obtained constraints are thus expressed as follows:

$$\sum_{k} \Big(\sum_{i \in T_k} (x_i - 1)^2 \Big) \tag{7}$$

To transform the linear test case selection problem into a QUBO problem, the Hamiltonian expression is constructed, integrating both objectives and the newly defined constraints. This requires the addition of a penalty constant (P), as per [13], which balances the importance of constraints within the Hamiltonian. Empirically, P is set to be slightly higher than the maximum value of the objective function, ensuring that the penalty aligns with the application domain and significantly influences the solution process [4]. Hence, we have:

$$H = \alpha \sum_{i=1}^{|\Gamma|} [x_i \cdot cost(\tau_i)] - (1-\alpha) \sum_{i=1}^{|\Gamma|} (e_i \cdot x_i) + P \cdot \sum_{k} \Big(\sum_{i \in T_k} (x_i - 1)^2 \Big) \tag{8}$$

The part of the equation representing the constraint, including weight, can be simplified, as follows:

$$P \cdot \sum_{k} \Big(\sum_{i \in T_k} (x_i - 1)^2 \Big) = P \cdot \sum_{k} \Big(\sum_{i,j \in T_k} (x_i^2 + 1^2 + 2x_i x_j - 2x_j) \Big) \tag{9}$$

Which can be simplified as follows:

$$\sum_{k} \Big(\sum_{i,j \in T_k} (-Px_i + 2Px_i x_j) \Big) \tag{10}$$

Thus, the final expression BQM (in the specific case QUBO) of Hamiltonian is depicted as follows:

$$H = \alpha \sum_{i=1}^{|\Gamma|} x_i \cdot cost(\tau_i) - (1-\alpha) \sum_{i=1}^{|\Gamma|} (e_i \cdot x_i) + \sum_k \sum_{i,j \in T_k} (-Px_i + 2Px_ix_j) \quad (11)$$

3.3 Data Collection from SIR Programs

The SIR is a repository that provides open-source industrial programs to collect the necessary information through the use of specific tools. In this work, the SIR programs chosen for the experiments were: flex, grep, gzip, and sed. To solve the problem, we defined three distinct criteria: *statement coverage*, *cost of execution of the test cases*, and *history of detected failures*.

3.3.1 Statement Coverage. To collect information about the regression testing phase of a program, we need to describe which test cases from the starting suite have performed each statement. To do this, we use the tool called *gcov* which is a compiler for C GNU (gcc). Gcov can track the statements performed by each test case, allowing us to obtain the necessary information.

3.3.2 Execution cost. To calculate the cost of executing individual test cases, it was decided not to rely on their execution time as it could be influenced by external factors. Instead, the cost is calculated by counting the elementary instructions that are actually executed. This approach is consistent with previous work cited in [21]. To determine the execution frequency of each basic block (i.e. a linear section of code that is not branched and has only one entry and exit point) that makes up each line of code, *gcov* is used. It is preferred to use block count rather than line count, as one line may contain multiple branches or function calls.

3.3.3 Past Fault History. The SIR makes available versions of the programs that contain within them fault *innested*, specifying, through the use of the so-called *fault matrix*, if a given test case can detect errors or not. This information can then be translated into a binary value to associate with the corresponding test cases.

3.4 Algorithm Implementation

Following the Hamiltonian definition in Section Section 3.2, the Python code of Listing 1 has been developed, in which the QUBO matrix to send to the quantum sampler for the annealing process is filled. The algorithm makes a triangular matrix, which defines the QUBO matrix for the Binary Quadratic Model (BQM) of the previous Hamiltonian.

```
def create_adjvector_bqm(sir_program, alpha, P):
    qubo = dimod.AdjVectorBQM(dimod.BINARY)

    for i in range(sir_programs_tests_number[sir_program
    ]):
        cost = alpha * test_cases_costs[sir_program][i] -
     (1 - alpha) * faults_dictionary[sir_program][i]
        qubo.set_linear(i, cost)

    QUBO matrix
    for k in coverage[sir_program].keys():
        test_cases = coverage[sir_program][k]
        for i in test_cases:
            for j in test_cases:
                if i < j:
                    qubo.set_linear(i,qubo.linear[i]-P)
                    qubo.set_linear(j,qubo.linear[j]-P)
                    try:
                        qubo.set_quadratic(i, j,qubo.
    quadratic[i,j] + 2 * P)
                    except:
                        qubo.set_quadratic(i, j, 2 * P)

    return qubo
```

Listing 1: Matrix generation encoding objectives and constraints of the problem

Table 2: Average Hypervolume Values (QA is quantum annealing, SA is simulated annealing)

Program	QA	SA
flex	**0.73**	0.80
grep	**0.51**	0.54
gzip	0.51	**0.50**
sed	0.52	**0.48**

The final matrix obtained must be sent to a quantum solver made available by D-Wave. In the case of this work, 30 independent experiments were performed for each program under examination to take into account the random nature of quantum computation.

4 RESULTS

4.1 RQ_1: Is the proposed quantum solution more effective than the simulated annealing?

Table 2 shows the mean values of the hypervolume metric (I_{CE}) relative to the sets of solutions built by quantum annealing and simulated annealing. The values reported correspond to the average of the values of I_{CE} obtained after 30 independent executions of each algorithm.

In the study, there isn't a clear winner in terms of effectiveness between the quantum annealing and the simulated annealing algorithms. The two techniques produced solution similar in terms of quality, this means that on average they build test suite that, with similar execution costs, cover a similar amount of faults.

4.2 RQ_2: Is the proposed quantum solution more efficient than the simulated annealing?

Table 3 shows the average execution times of the two algorithm tested, with quantum annealing performing consistently well. It is worth noting that the tests were carried out on different machines. Quantum annealing was run on the D-Wave Hybrid Binary Quadratic Model (Version2) sampler, while the simulated annealing was tested on a machine with a 2.30GHz Intel Core i5 processor and 8GB RAM.

The results obtained from Table 3 seem to confirm our expectations. Quantum annealing is significantly more efficient than

Table 3: Average execution times of QA and SA

Program	Quantum Annealing	Simulated Annealing
flex	2.9s	15min 31sec
grep	2.9s	15min 28sec
gzip	2.9s	25s
sed	2.9s	4min 10sec

simulated annealing, obtaining a constant execution time. Simulated annealing, on the other hand, has very low performance in terms of execution time due to the magnitude of the problem to be solved and the large number of parameters to tune, as expected. Table Table 4 supports our findings by showing the confidence intervals of the execution times of quantum annealing for each program. The very narrow interval indicates that the real execution times when applying a Quantum Annealing solution will largely surpass any possible classical result.

Table 4: 95% confidence intervals of the average execution time of Quantum Annealing for each considered program.

Program	Lower Limit	Upper Limit
flex	2.9937s	2.9966s
grep	2.9939s	2.9965s
gzip	2.9908s	2.9937s
sed	2.9890s	2.9924s

5 DISCUSSION

With the aim of providing additional insights to our study's findings, we performed an additional comparison of our implemented technique with a wider set of state-of-the-art techniques. In the following the main results and insights are reported.

5.1 Does quantum annealing outperform actual state-of-the-art solutions?

5.1.1 Effectiveness. Quantum annealing has also been compared with the most commonly used algorithmic solutions for solving the test case selection problem. The algorithms examined are vNSGA2, DIV-GA and greedy, which were discussed in the Section 2.1.

Comparisons between the values of the hypervolume and execution time metrics have been possible thanks to the use of the data reported by the previous work of Panichella et al.[21].

The results obtained in terms of hypervolume are therefore shown in the Table 5.

As can be seen for 3 out of 4 programs, the quantum annealing algorithm produced higher values of the hypervolume metric compared to other algorithms. However, for the program "gzip", the quantum annealing algorithm was only able to achieve a metric value equal to the effectiveness of the DIV-GA algorithm. Similarly,

Table 5: Average Hypervolume Values of QA, DIV-GA, vNSGA2 and Greedy

Program	QA	DIV-GA	vNSGA-II	Greedy
flex	0.73	**0.05**	0.20	0.15
grep	0.51	**0.27**	0.28	0.51
gzip	**0.51**	**0.51**	0.54	0.52
sed	0.52	**0.10**	**0.10**	0.20

Table 6: Average execution times for QA, DIV-GA, vNSGA2 and greedy algorithms

Program	QA	DIV-GA	vNSGA-II	Greedy
flex	2.9s	2min 51sec	5min 40sec	1min 7sec
grep	2.9s	3min 45sec	5min 57sec	2min 4sec
gzip	2.9s	33s	1min 22sec	4s
sed	2.9s	1min 33sec	2min 25sec	14s

the simulated annealing algorithm also produced higher hypervolume metric values for the same 3 out of 4 programs compared to other algorithms.

This suggests that classical MOGA and greedy strategies are more effective compared to quantum and simulated annealing, as they manage to build higher quality solutions. These results indicate that the problem with the quality of solutions obtained by quantum annealing may not be due to its quantum nature, but rather by the characteristics of the annealing strategy itself.

5.1.2 Efficiency. Despite the good results in terms of effectiveness, MOGA and greedy techniques for test case selection have long execution times. DIV-GA is better than vNSGA-II and greedy techniques, but it adds expensive computation of SVD. Adiabatic optimization algorithms use quantum superposition to achieve results similar to processing on multiple threads with a linear or constant time.

Table 6 compares the average execution times of the quantum annealing with those of the state-of-the-art MOGA and greedy algorithms. Again, the values related to DIV-GA, vNSGA2 and greedy have been extrapolated from the work of Panichella et al.[21], in which it is specified that the execution of these algorithms took place on a machine equipped with a 2.40GHz Intel Core i7 processor and an 8GB RAM.

Our results show that quantum annealing dominates over other algorithms.

5.2 The Problem Of Annealing's Strategy

The quantum annealing algorithm did not produce solutions of the same quality as the other algorithms examined. Simulated annealing produced similar results, indicating that the issue with quantum annealing is not related to its quantum nature, but rather to the annealing strategy employed.

Table 5 indicates that the annealing strategy is not the most effective approach for selecting test cases.

One possible explanation for this result is the high complexity of the research area. The problem of selecting test cases involves a dense network of interconnections between different variables of a test suite, which makes it difficult for both simulated annealing and quantum annealing to converge toward an optimal solution. Moreover, quantum annealing is not suitable for *high precision* problems (that require QUBO models) due to the noise that is still challenging to manage by quantum machines. Additionally, the problems that show good results with quantum annealing usually have constraints where the number and connectivity of qubits required for the problem do not increase significantly with the size of the problem, which is not the case in this study.

The observations about the results in terms of efficiency and effectiveness of quantum annealing as opposed to those of state of the art strategies therefore open a trade-off issue between efficiency and effectiveness. Quantum annealing, that present incredible results in terms of execution time, is preferred, for example, in situations where time is particularly limited, in scenarios where scalable and effective suite of very large tests have to be managed, or in the presence of small computational resources that should be handled optimally. But there are also many contexts in which effectiveness would be preferable to efficiency. For example, to ensure a high quality of the regression suites, in contexts where there is the need to execute tests that ensure security and robustness of a software system; or for example in contest in which it is needed to work with small test suites, that can be managed by the classical solutions in execution times comparable to those of quantum annealing.

6 THREATS TO VALIDITY

This section discusses all aspects that could threaten the validity of the study. Threats to be investigated can be divided into micacce to validity: construct, internal, external and conclusion.

6.1 Construct Validity

The main threat in this regard concerns the correctness of the measures used as criteria for the selection of tests: coverage, history of failures, and cost of execution. In order to limit this issue, code coverage information has been collected using open-source profiling and compilation tools (i.e., GNU gcc and gcov).

6.2 Internal Validity

One of these threats is undoubtedly the random nature of simulated and quantum annealing algorithms. For this reason, the experiments were repeated 30 times for each program under examination and then considered a confidence interval regarding the execution time of the quantum case. The *tuning* of the P penalty parameter is also a factor that could undermine the internal validity of this job. For this reason, a method used in the literature [4] was used when choosing the value to be entrusted to P. Other parameters whose tuning could undermine the internal validity of this work correspond to those used for the implementation of simulated annealing. The parameter α was chosen, as well as on the basis of repeated tests, using a method already widespread in the literature. As for the other parameters, they were all validated following repeated trials.

6.3 External Validity

There may be other algorithmic solutions that might have not been considered in relation to the solution proposed in this work. However, at the time of writing, other possible solutions have been neither developed nor discovered yet. Moreover, no better solution for the considered problem was developed at the time of writing. Hence, this work compared the performances of quantum annealing to the solutions currently used as vNSGA-II.

6.4 Conclusion Validity

Regarding the conclusion validity, the results are supported by appropriate metrics already used in previous studies[21], calculated on the basis of 30 independent executions to obtain statistically reliable results. In particular, the hypervolume metric has been used to obtain a quantifiable value of the effectiveness of an algorithmic strategy; finally, a confidence interval has been calculated to support the results regarding the average execution time of the quantum annealing.

7 CONCLUSION

This work has developed and analyzed an approach that makes use of Quantum Simulated Annealing for regression test case selection, with the aim of proposing a valid alternative to the currently available algorithmic solutions. As a result from a series of experiments conducted on 4 different programs extracted from the SIR, we concluded that the proposed quantum strategy, despite undoubtedly domains its traditional counterpart in terms of performance, builds solutions that on average are less effective than those obtained by the other state-of-the-art techniques.

As future work, we plan to explore the implementation and experimentation of alternative quantum-based versions of well-established and effective algorithms for Test Case Selection (TCS), such as the Greedy algorithm.

The aim of these further experiments is precisely to propose quantum strategies that not only far exceed the classical alternatives in terms of performance, but also and above all in terms of effectiveness.

REFERENCES

[1] Aitor Arrieta, Shuai Wang, Urtzi Markiegi, Ainhoa Arruabarrena, Leire Etxeberria, and Goiuria Sagardui. 2019. Pareto efficient multi-objective black-box test case selection for simulation-based testing. *Information and Software Technology* 114 (2019), 137–154.

[2] Wesley Klewerton Guez Assunção, Thelma Elita Colanzi, Silvia Regina Vergilio, and Aurora Pozo. 2014. A multi-objective optimization approach for the integration and test order problem. *Information Sciences* 267 (2014), 119–139.

[3] Anne Auger, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler. 2009. Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point. (2009), 87–102.

[4] Mayowa Ayodele. 2022. Penalty Weights in QUBO Formulations: Permutation Problems. (2022).

[5] Samuel Bates and Susan Horwitz. 1993. Incremental program testing using program dependence graphs. In *Proceedings of the 20th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 384–396.

[6] Vladimír Černỳ. 1993. Quantum computers and intractable (NP-complete) computing problems. *Physical Review A* 48, 1 (1993), 116.

[7] Dario Di Nucci, Annibale Panichella, Andy Zaidman, and Andrea De Lucia. 2018. A test case prioritization genetic algorithm guided by the hypervolume indicator. *IEEE Transactions on Software Engineering* 46, 6 (2018), 674–696.

[8] A. Das e B. K. Chakrabart. 2008. Colloquium: Quantum annealing e analog quantum computation. *Reviews of Modern Physics, vol. 80, no. 3, p. 1061* (2008).

[9] Emelie Engström, Per Runeson, and Mats Skoglund. 2010. A systematic review on regression test selection techniques. *Information and Software Technology* 52, 1 (2010), 14–30.

[10] Michael G Epitropakis, Shin Yoo, Mark Harman, and Edmund K Burke. 2015. Empirical evaluation of pareto efficient multi-objective regression test case prioritisation. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. 234–245.

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

[12] Kurt F Fischer. 1977. A test case selection method for the validation of software maintenance modifications. (1977).

[13] Rick Hennig e Yu Du Fred Glover, Gary Kochenberger. 2022. Quantum Bridge Analytics I: A Tutorial on Formulating and Using QUBO Models. *Ann Oper Res 314, 141–183* (2022).

[14] Lov K Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 212–219.

[15] Tony Hoare and Robin Milner. 2005. Grand challenges for computing research. *Comput. J.* 48, 1 (2005), 49–52.

[16] Tadashi Kadowaki and Hidetoshi Nishimori. 1998. Quantum annealing in the transverse Ising model. *Physical Review E* 58, 5 (1998), 5355.

[17] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science* 220, 4598 (1983), 671–680.

[18] Will Knight. 2018. Serious quantum computers are finally here. What are we going to do with them. *MIT Technology Review. Retrieved on October* 30 (2018), 2018.

[19] Zheng Li, Mark Harman, and Robert M Hierons. 2007. Search algorithms for regression test case prioritization. *IEEE Transactions on software engineering* 33, 4 (2007), 225–237.

[20] Mitchell Olsthoorn and Annibale Panichella. 2021. Multi-objective test case selection through linkage learning-based crossover. In *Search-Based Software Engineering: 13th International Symposium, SSBSE 2021, Bari, Italy, October 11–12, 2021, Proceedings 13*. Springer, 87–102.

[21] Annibale Panichella, Rocco Oliveto, Massimiliano Di Penta, and Andrea De Lucia. 2014. Improving multi-objective test case selection by injecting diversity in genetic algorithms. *IEEE Transactions on Software Engineering* 41, 4 (2014), 358–383.

[22] Gilles Perrouin, Sebastian Oster, Sagar Sen, Jacques Klein, Benoit Baudry, and Yves Le Traon. 2012. Pairwise testing for software product lines: comparison of two approaches. *Software Quality Journal* 20 (2012), 605–643.

[23] Gregg Rothermel and Mary Jean Harrold. 1993. A safe, efficient algorithm for regression test selection. In *1993 Conference on Software Maintenance*. IEEE, 358–367.

[24] Gregg Rothermel and Mary Jean Harrold. 1996. Analyzing regression test selection techniques. *IEEE Transactions on software engineering* 22, 8 (1996), 529–551.

[25] Gregg Rothermel and Mary Jean Harrold. 1998. Empirical studies of a safe regression test selection technique. *IEEE Transactions on Software Engineering* 24, 6 (1998), 401–419.

[26] Manuel A Serrano, Luis E Sánchez, Antonio Santos-Olmo, David García-Rosado, Carlos Blanco, Vita Santa Barletta, Danilo Caivano, and Eduardo Fernández-Medina. 2023. Minimizing incident response time in real-world scenarios using quantum computing. *Software Quality Journal* (2023), 1–30.

[27] Yiwen Shi and Jennifer Dworak. 2013. A Simulated Annealing Inspired Test Optimization Method for Enhanced Detection of Highly Critical Faults and Defects. *Journal of Electronic Testing* (2013).

[28] Helge Spieker, Arnaud Gotlieb, Dusica Marijan, and Morten Mossige. 2017. 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*. 12–22.

[29] R. E. Steuer. 1986. Multi-Criteria Optimization: Theory, Computation, and Application. *John Wiley, New York* (1986).

[30] Sei Suzuki. 2009. A comparison of classical and quantum annealing dynamics. In *Journal of Physics: Conference Series*, Vol. 143. IOP Publishing, 012002.

[31] Shuai Wang, Shaukat Ali, and Arnaud Gotlieb. 2013. Minimizing test suites in software product lines using weight-based genetic algorithms. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. 1493–1500.

[32] Rui Dong Weixiang Zhang, Bo Wei, Huiying Zhang, Sihong Wang, and Fengju Liu. 2022. Test Case Prioritization Based on Simulation Annealing Algorithm. *8th International Conference on Computing and Artificial Intelligence (ICCAI '22), Tianjin, China* (2022).

[33] Yinxing Xue and Yan-Fu Li. 2020. Multi-objective integer programming approaches for solving the multi-criteria test-suite minimization problem: Towards sound and complete solutions of a particular search-based software-engineering problem. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 29, 3 (2020), 1–50.

[34] Stephen S Yau and Zenichi Kishimoto. 1987. METHOD FOR REVALIDATING MODIFIED PROGRAMS IN THE MAINTENANCE PHASE.. In *Proceedings-IEEE Computer Society's International Computer Software & Applications Conference*. IEEE, 272–277.

[35] Shin Yoo and Mark Harman. 2007. Pareto efficient multi-objective test case selection. In *Proceedings of the 2007 international symposium on Software testing and analysis*. 140–150.

[36] Shin Yoo and Mark Harman. 2010. Using hybrid algorithm for pareto efficient multi-objective test suite minimisation. *Journal of Systems and Software* 83, 4 (2010), 689–701.

[37] Shin Yoo and Mark Harman. 2012. Regression testing minimization, selection and prioritization: a survey. *Software testing, verification and reliability* 22, 2 (2012), 67–120.

[38] Shin Yoo, Mark Harman, and Shmuel Ur. 2011. Highly scalable multi objective test suite minimisation using graphics cards. In *Search Based Software Engineering: Third International Symposium, SSBSE 2011, Szeged, Hungary, September 10-12, 2011. Proceedings 3*. Springer, 219–236.

[39] Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. 2007. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. (2007), 862–876.